An Evaluation of Predictive Models for Individual-Level Employee Voluntary Turnover

Student Name:	Robbin Stigter
ANR:	602969
Student nr. :	1247025

Abstract

Human capital is becoming a more important part of organizations, and employee voluntary turnover has been identified as a key issue. To contribute, this study's focus is on the prediction of individual-level voluntary employee turnover. Accurate predictions enable organizations to act for retention or succession planning of employees. To find the most appropriate predictive model for voluntary leave; sixteen models were evaluated based on their ROC curves. It was found that voluntary employee turnover can be linearly separated, and the most appropriate model is the *Support Vector Machine* (SVM). To evaluate the predictive performance, and since the data is not in balance (only 16% of the cases is labeled as leaving the organization); altered cutoff values, and the sampling methods up-sampling, and synthetic minority over-sampling technique (SMOTE), are applied and the resulting models evaluated based on the F1 score and balanced accuracy. Sampling was found to not provide significantly better models. The best performing model is a SVM with a F1 score and balanced accuracy of respectively, 0.86 and 0.77. Further, based on employee voluntary turnover literature, eight hypotheses were formed and tested based on logistic regression. And last, the relationships between the most important and significant predictors and voluntary turnover are identified. The top 10 most important and significant predictors found are: *Overtime, Environment Satisfaction, Number of Companies Worked for, Job Satisfaction, Business Travel, Job Involvement, Years Since Last Promotion, Distance from Home, Age*, and relationship status *Single*. The results are based on a publicly available dataset provided by IBM (McKinley, 2015). In accordance with (Rubenstein, Eberly, Lee, & Mitchell, 2017), the author wants to highlight the context-sensitive nature of individual-level voluntary leave; and advises practitioners to analyze their own data to find, and be able to react to, the predictors that are important and significant in their organization.

Keywords: turnover prediction; voluntary turnover; employee turnover; attrition; machine learning; class imbalance; sampling; Human Resource Analytics;

Table of Contents

1. Introduction

The corporate landscape has changed dramatically over the past few decades. These changes are due to globalization, information availability, and the requirements of a high-tech economy. One specific change that has been noticed relates to the assets held by a company. Whereas in the 1970s, more than 95% of a company's assets could be attributed to tangible holdings, by the early 2000s that number had reduced to less than 30%(King, 2016; Mcclure, 2003). This means that more than 70% of a firm's total worth is due to intangible assets, including human capital.

Human resource management (HRM) is the main department within organizations that manages human capital. The function of HRM is to motivate employees and enhance workforce effectiveness. According to (King, 2016), integrating information technologies and HRM will provide smarter work. Turnover can be considered as a subgroup of HRM, and is the focus of the current study. This paper is a machine learning approach for prediction of quit among staff.

Researchers and HRM have focused on employee turnover for decades because it negatively affects organizations' performance (Glebbeek et al., 2014; Hancock, Allen, & Bosco, 2013; Shaw, 2011). Employee retention is one of the main challenges in organizations, especially for those with a long lead time to hire a new employee. Also, organizations are complex and dynamic environments. (Akkermans, 2014) found that a shorter hiring and capacity delay have positive effects for new product development introduction projects, and can even avoid tipping point behavior for product success/failure. In accordance with Akkermans, (Kacmar, Andrews, & System, 2006) found that employee turnover is both costly and disruptive to the organizational function.

Turnover causes many different types of costs for organizations. These costs can be divided between direct and indirect costs (Ongori, 2010). Direct costs activities such as advertising the position, replacement, recruitment and selection, temporary staff, and management time. Indirect costs are morale related costs, pressure on remaining staff, costs of learning, product/service quality, and organizational memory. Research suggests that 15-30 percent of turnover costs are direct and about 70-85 percent of turnover costs are hidden costs such as lost productivity and opportunity (Boles, Dudley, Onyemah, Rouziès, & Weeks, 2012). (Sagie, Birati, & Tziner, 2002) found that a high-tech firm lost 2.8 million US dollars or 16.5% of before-tax annual income because of employee turnover. Since voluntary turnover is expensive, companies do not want their employees to voluntarily leave (Allen, Bryant, & Vardaman, 2010). Among the reasons for termination, voluntary turnover is one of the major ones, accounting for 26% (X. Zhu, 2016). And compared to the other types of turnover, voluntary turnover is harder for companies to control. Understanding and forecasting turnover at the firm and departmental levels is essential for reducing it (Kacmar et al., 2006), as well as for effectively planning, budgeting, and recruiting in the human resource field.

Fortunately for companies, due to fast pace of developments in artificial intelligence (AI) and the decreasing prices of storage and computing power; machine learning (ML) capabilities have become increasingly more accessible (Shmueli, Patel, & Bruce,

2010; Witten, Frank, Hall, & Pal, 2016). Besides the increase in computing power and storage, also the volume of data that companies collect and is freely available has drastically increased (Goodfellow, Bengio, & Courville, 2016; Shmueli et al., 2010). It has been estimated that the amount of data stored in the world's databases doubles every 20 months (Witten et al., 2016). Machine learning algorithms feed on this data. It is what they use to learn, figure out patterns, and spot trends.

Every year Gardner proposes a top 10 of strategic technology trends. They indicated that AI and ML have reached a critical tipping point. According to them, AI and ML will increasingly augment and extend virtually every technology enabled service, thing or application. Creating intelligent systems that learn, adapt and potentially act autonomously rather than simply execute predefined instructions is primary battleground for technology vendors through at least 2020. (Panetta, 2016)

Analytics in human resource management has been around for years. For example, the notion of measurement in human resources can be traced back to the early 1900s (Kaufman, 2014). Even though it has been discussed for many years, with only 16% of organizations reporting adoption, HR Analytics represents a new innovation according to (Marler & Boudreau, 2017).

So, since HR Analytics represents a new innovation, human capital is becoming an increasingly more important part of organizations (King, 2016; Mcclure, 2003), employee turnover is an important part of HRM and expensive for organizations (Glebbeek et al., 2014; Hancock et al., 2013; Sagie et al., 2002; Shaw, 2011), and the machine learning opportunity of the percent (Panetta, 2016); this study's objective is to examine the machine learning opportunities present today and apply these on an employee attrition (i.e. employees who voluntarily quit their job) dataset. This leads to the following four key questions:

1. What machine learning algorithm is most appropriate for predicting employee voluntary leave?
2. Does sampling of the dataset increase the predictive performance of the models?
3. What results can be expected from predictive modeling of employee voluntary turnover?
4. What are the significant predictors for determining employee voluntary leave?

For this study the open-source program R is used for analysis. The appendix includes all R code and corresponding outcomes. A random set seed of 1247 was used to make the results reproducible.

To answer the key questions, first, chapter 2 will provide the reader with a thorough description and analysis of the publicly available dataset used in this study (McKinley, 2015). Chapter 3 will provide the reader with the context to place this study in, as well as forming some hypotheses based on individual-level voluntary turnover literature. In chapter 4 the process of creating the datasets is described. This is related to chapter 5, which provides the reader with a concise description of the machine learning algorithms used, in that not one dataset is appropriate for all methods. Next, in chapter 6 and 7, the model validation and evaluation techniques and metrics get described.

Some remedies for severe class imbalances get discussed in chapter 8. This brings us to part one of the results, chapter 9, where the results related to the predictive models are represented. More specifically, the 16 different models (described in chapter 5) are compared on effectiveness of modelling voluntary turnover; and the performance of the predictive models, after applying the methods to counter the severe class imbalance, are compared. Chapter 10 can be considered as part two of the results, this chapter will present the results related to predictor importance and significance. In chapter 11 the current study is compared to similar studies conducted in this area. This to provide the reader with better context of the contribution of the current study. The paper will end with a discussion of the results, a conclusion, and identification of areas of further research (Chapter 12).

2. Data description

Employee turnover analytics is an understudied subject in research. One likely explanation for the lack of employee turnover research is the difficulty associated with obtaining data. This data is hard to obtain, because once it becomes public it is prone to legal/privacy issues, and can be bad for an organization's reputation. In 2015 IBM uploaded an employee attrition dataset which was originally intended to be used in combination with their product Watson. This publicly available dataset is used in this paper and can be found at: https://www.ibm.com/communities/analytics/watson-analytics-blog/hr-employee-attrition/ (McKinley, 2015).

The dataset contains 1470 records, 34 predictor variables, and one outcome variable, namely whether employee attrition (i.e. voluntary leave) took place. The predictor variables include information about employee's demographic, satisfaction and performance, and some financial measures. The dataset does not include a full description of all the predictors. Therefore, based on the knowledge of an HR professional from Accenture, definitions for each of the predictors were created. The list of predictors and definitions used in this paper are included in the Table 1.

The use of this open dataset has some negative side effects. First, since it is an open dataset the company background is unknown and the author is unable to speak to subject matter experts (SME) within the client company. This is a common an important step in obtaining the underlying structure and relations between predictors and outcome. Second, since it is unknown how the dataset is created the external validity and construct validity of the results might be in jeopardy. Third, the dataset has only 1470 records which can limit the effectiveness of some of the machine learning algorithms used. It is probable that a dataset provided by a real client contains more records. However, datasets of similar previous studies contain even less records, namely 731 (X. Zhu, 2016) and 881 (Chang, 2009). On the other hand, using an open dataset has positive side effects as well. First, since the dataset is anonymized, financial data related to the employees are included; often these are excluded from the dataset. Second, since the dataset is open and available to everyone the internal validity of this study is greatly enhanced.

Table 1 Predictors with corresponding definition and metric

Predictor	Definition	Metric
Demographics		
EmployeeNumber	Unique identifier for individual	#
Age	Chronological number of years an individual has lived. (Rubenstein et al., 2017)	#
Gender	Biologically based categories. (Rubenstein et al., 2017)	Male/ Female
MaritalStatus	Relationship status of employee.	Single/ Married/ Divorced
NumCompaniesWorked	Number of companies an individual has worked for.	#
EmployeeCount	Count of employee	#
Over18	Indication whether an individual is over 18 years of age	Y/N
Internal measurements		
TotalWorkingYears	Time employed since end of one's study (measured in years).	#
YearsAtCompany (i.e. Tenure)	Time employed with one's current organization (measured in years). (Rubenstein et al., 2017)	#
YearsInCurrentRole	Number of years an individual practices the same role.	#
YearsSinceLastPromotion	Number of years since an individual got last promoted.	#
YearsWithCurrManager	Number of years under last manager.	#
TrainingTimesLastYear	Number of trainings an individual took last year.	#
Financial related measures		
DailyRate	Income of one day hire out of individual.	#
HourlyRate	Income of one hour hire out of individual.	#
MonthlyRate	Income of one month hire out of individual.	#
MonthlyIncome (i.e. Pay)	Amount of money an individual receives for the job each month. (Rubenstein et al., 2017)	#
PercentSalaryHike	Percentage of salary increase coming year.	#
StandardHours	Number of standard hours for an individual.	#
StockOptionLevel	Number indicating amount of monthly income goes to buying stock at a discount.	0/ 1/ 2/ 3, assumed 3 indicating higher monthly amount

OverTime	Indication of whether an individual gets compensated for extra hours.	Yes/No
Individual attributes		
Education	An individual's maximum level of education attained. (Rubenstein et al., 2017)	1 'Below College', 2 'College', 3 'Bachelor', 4 'Master', and 5 'Doctor'
EducationField	Individual's main education field	6 unique classes
Department	The department the individual works for.	Human Resources/ Research & Development/ Sales
JobLevel	Individual's job level	1-5, 5 being the higher job level
BusinessTravel	Amount of travel the individual does for the organization	Non-Travel/ Travel_Rarely/ Travel_Frequently
DistanceFromHome	Travel distance from individual's home to work location, measured in Km.	#
Individual ratings		
PerformanceRating	Grade given as indication of individual's performance	1-4, 4 indicating outstanding
JobInvolvement	Degree to which an individual identifies with his or her job. (Rubenstein et al., 2017)	1-4, 4 indicating very high
EnvironmentSatisfaction	Degree to which an individual likes his or her work environment.	1-4, 4 indicating very high
JobSatisfaction	Degree to which an individual likes his or her job. (Rubenstein et al., 2017)	1-4, 4 indicating very high
RelationshipSatisfaction	Degree to which an individual likes his or her job related relationships.	1-4, 4 indicating very high

2.1 Data exploratory analysis

Within this dataset only 16%, 237 out of 1470, was labeled as having left the organization. The rate of interest is thus under represented. Even the basic rule 'each employee is predicted to stays with the organization' will result in an expected accuracy of 84% (100%-16%). This could negatively affect the performance of some of the predictive models. However, some methods exist to counter this, these will be discussed in chapter 8 (p.26). The age lies between 18 and 60 with an average age of 36,92 years old. Within this dataset 60% is male and 40% female. The total working years range from 0 to 40 with an average of 11.28 years. Monthly income range from 1009 to 19999 with an average income of 6503 a month. The performance rating is a grade given between 1 and 4, the former indicating bad performance and the latter good. This dataset only contains the performance ratings of 3 and 4, indicating that only the good performers are included in this dataset. Further, three departments are included in this dataset, namely, *Human Resources*, *Research and Development*, and *Sales*, where the bulk of employees is in the Research and Development department. Also, there are six job roles included in this dataset, namely, *Sales Executive*, *Research Scientist*, *Laboratory Technician*, *Manufacturing Director*, *Healthcare Representative*, and *Manager*, as well as a category '*Other*'. Given these departments and roles, an educated guess of the industry this dataset was gathered from would be the pharmaceutical industry.

2.2 Data validity check

To counter the second negative side effect discussed above, namely external validity and construct validity, the author did some analysis on the dataset. To increase the external validity of this study, the author first conducted some basic tests. The six basic checks that got tested are*:*

- *TotalWorkingYears >= YearsAtCompany,*
- *YearsAtCompany >= YearsInCurrentRole,*
- *YearsAtCompany >= YearsSinceLastPromotion,*
- *YearsAtCompany >= YearsWithCurrManager,*
- *MonthlyRate >= DailyRate,*
- and *DailyRate >= HourlyRate*.

No indication for low internal dataset validity were found.

Next, the author searched the forum of IBM to get a statement about the external validity of the dataset. A similar question was asked, namely *"Is the sample HR employee attrition dataset completely fabricated or is it an anonymized dataset of actual employee information?".* The response of the Watson Analytics Support: *"The HR Employee Attrition data set is based of real data with all personal identifiers removed. The data was also tweaked so that it performs better in telling a story about attrition in the HR department.".* (Watson Analytics Support, 2015)

According to the HR professional from Accenture, almost all predictors are commonly measured within organizations. For the rates, however, it was thought to be more common for organizations to use daily rate and calculated the other rates based on this variable. So, hourly rate is daily rate divided by eight, and monthly rate is equal to

daily rate times 20. Based on this, one would expect to find a strong correlation between these variables. However, this was not the case in this dataset.

Another questionable variable was *NumCompaniesWorked*. The HR professional indicated that in general organizations do not measure this variable. However, we agreed it could be informative for employee attrition, and the statistic can be easily acquired based on the employee's CV. Therefore, it was decided to leave this predictor in the dataset. If this variable is found to be significant and important in the predictive models it will be an indication to start acquiring this statistic.

In conclusion, external validity establishes how generalizable the relationship is. Construct validity investigates whether measurement of the key constructs is sufficient to adequately assess the relationship (Marler & Boudreau, 2017). Based on some basic internal checks, information from IBM Watson Analytics Support, and an internal meeting with an HR professional; the author concludes the dataset to be representative to actual organizational data. The external validity of this study is therefore high. The construct validity is harder to assess, but there are no indications that this validity is in jeopardy.

3. Literature review

Human Resource Analytics (HRA) is a relatively new term. (Marler & Boudreau, 2017) conducted an evidence-based review of HR analytics. According to their research of major databases, HR analytics first appeared in the HR published literature in 2003–2004.

By bringing various definitions used throughout literature together, (Marler & Boudreau, 2017) defined HR Analytics as: "*A HR practice enabled by information technology that uses descriptive, visual, and statistical analyses of data related to HR processes, human capital, organizational performance, and external economic benchmarks to establish business impact and enable data-driven decision-making*". According to (Marler & Boudreau, 2017) HR analytics have five characteristics. First, HR Analytics is not HR Metrics. It involves more sophisticated analysis of HR-related data. Second, HR Analytics does not focus exclusively on HR functional data, and involves integrating data from different internal functions and data external to the firm. Third, HR Analytics involves using information technology to collect, manipulate, and report data. Fourth, HR Analytics is about supporting people related decisions. Finally, HR Analytics is about linking HR decisions to business outcomes and organizational performance. Based on this definition and these characteristics, the current study can be labeled as HR Analytics research.

In conducting their review of the literature on HR Analytics, and despite evidence of a growing interest in this innovation, (Marler & Boudreau, 2017) found very little and limited scientific evidence to aid decision-making concerning whether to adopt HR Analytics. They further state that there are two notable paradoxes. First is that despite the popularity of HR Analytics there is very limited high-quality scientific evidence-based research on this topic. The second paradox is the apparently limited adoption of HR Analytics when the available research seems frequently to suggest that it is associated with positive organizational outcomes. Also, much of the literature on the

topic of HR analytics has been focused on normative questions of what should be done rather than analytical questions of how it can be done, in what contexts, and with what results (Angrave, Charlwood, Kirkpatrick, & Stuart, 2016). With this article it is intended to contribute to the high quality scientific evidence-based research in HRA, and is focused on a more applied direction of HRA, as suggested by (King, 2016).

3.1 Individual-level employee voluntary turnover

Employee turnover is a general term referring to the loss of employees resulting from a wide range of causes, such as retirement, death, quitting, termination, promotion, and reassignment. Each of these turnover modes has different foundational causes and may be more or less prevalent during different points in one's career (X. Zhu, 2016). In this study, in accordance with (Sikaroudi, Ghousi, & EsmaieeliSikaroudi, 2015), we consider turnover as the rate of employees' leave and replacement in a predefined period of time. Turnover has various forms. It can be voluntary or involuntary, functional or dysfunctional, avoidable or unavoidable.

In this study, we focus on employee voluntary turnover, also known as attrition. Employee voluntary turnover is part of the overall employee turnover, and is accounted for when an employee voluntarily leaves the organization. Among the reasons for termination, voluntary turnover is one of the major ones accounting for 26% of the total turnover (X. Zhu, 2016). Compared to the other types, employee voluntary turnover is more problematic for companies to control. Employee voluntary turnover is often dysfunctional and can be avoidable.

Many researchers have attempted to identify turnover factors to prevent and reduce turnover. This study attributes by testing some of the suggested turnover factors. Nearly two decades ago (Griffeth, Hom, & Gaertner, 2000) conducted the broadest meta-analysis of the turnover literature. Since then, however, a sizable number of primary studies have been published, also seeking to understand this phenomenon. This year (Rubenstein et al., 2017) present an update and holistic picture of how the studied constructs operate within the turnover literature. Similar to this study, their focus is on individual voluntary turnover. In their study they include 57 predictors across 1800 effect sizes, which is a 27% increase in constructs and a 114% increase in effects compared to (Griffeth et al., 2000).

The author formed hypotheses based on the findings of (Rubenstein et al., 2017) and the predictors in the current dataset. A total of 8 predictors were found to be similar as the ones studied in the past. These are *age, education, marital status, sex, tenure, pay, job involvement,* and *job satisfaction*. However, although marital status is measured in this study, the measurement is somewhat unconventual as it indicates the three categories married, divorced, and single, while the conventual measure would be, unmarried, married. Therefore, marital status in this study was found to be unfit to compare with previous studies.

The relationship between age and voluntary turnover has been investigated in many different studies (k =121). Among individual attributes, age (ρ = -0.21) falls in the group that demonstrate the strongest effects. Here ρ indicates the sample size weighted average correlation corrected for measurement error in the predictors. The effect can

be interpreted as older workers are less likely to quit, which leads to the following hypothesis:

Hypothesis 1: Age is negatively related to voluntary turnover.

In this study the education level of the employee is indicated by a number between 1 and 5, 1 indicating below college, and 5 indicating doctor. In their meta-analysis 51 studies investigated the relationship between education and voluntary turnover. The ρ is equal to 0.04, indicating only a moderate effect. Which leads to the following hypothesis:

Hypothesis 2: Education is a significant predictor of voluntary turnover.

Sex is a predictor found to be significant in predicting voluntary turnover. With a ρ = -0.01, which can be interpreted as males are less likely to voluntary leave, although it's effect on voluntary turnover is only moderate. Sex is in the current study denoted as gender. This leads to the following hypothesis:

Hypothesis 3: Sex is a significant predictor of voluntary turnover.

Among the individual attributes, tenure was found to have the strongest effect with a ρ = -2.7, which can be interpreted as employees that have been working for the current organization for a longer time are less likely to leave. In the current study tenure is denoted as 'YearsAtCompany'. This leads to the following hypothesis:

Hypothesis 4: Tenure is negatively related to voluntary turnover.

Pay has been found to significantly affect voluntary turnover with an effect of ρ = -0.17, indicating that employees which receive more monetary compensation are less likely to leave. In the current study pay is denoted as MonthlyIncome. This leads to the following hypothesis:

Hypothesis 5: Pay is negatively related to voluntary turnover.

Job involvement was found to have an even greater effect when including more current studies compared to the analysis done by (Griffeth et al., 2000). It now has a ρ = –0.19 which indicates that more job involvement will lead to less voluntary turnover. This leads to the following hypothesis:

Hypothesis 6: Job involvement is negatively related to voluntary turnover.

As job involvement, job satisfaction has also been found to have even greater effects compared to the older meta-analysis study. With a ρ = –0.28 it has a strong effect on voluntary turnover. This leads to the following hypothesis:

Hypothesis 7: Job satisfaction is negatively related to voluntary turnover.

And lastly, Employee performance is found to be significantly negatively correlated to employee voluntary turnover, with a ρ = –0.21. In this study the performance is rated on a 4 point scale, and only performance ratings of 3 and 4 are present in the dataset. This leads to the following hypothesis:

Hypothesis 8: Employee performance is negatively related to voluntary turnover.

The hypotheses are tested by fitting a logistic regression model to the data. The results get discussed in chapter 10 (p. 35).

4. Dataset creation

One of the first steps in the model building process is to transform, or encode, the original data structure into a form that is most informative for the model (i.e. feature engineering). This encoding process is critical and must be done with foresight into the analyses that will be performed so that appropriate predictors can be elucidated from the original data. A consequence of not appropriately formatting the predictors is the development of ineffective predictive models.

During the initial check of the data, first all predictors were transformed to their correct data type. Next, the data got checked for missing values, and predictors with near zero variances (i.e. containing close to only 1 unique value). No missing values were found, and three predictors got identified having low variances, namely Over18, EmployeeCount, and StandardHours, and got removed.

Some machine learning (ML) algorithms only work with numeric data; therefore, two dummy sets were created. During this process the categories are reencoded into smaller bits of information called “dummy variables”. Each category gets its own dummy variable that is a zero/one indicator for that group. A variable with four categories can be transformed into only 3 new binary variables, since the fourth can be inferred. However, for interpretation purposes it can be useful to include all dummy variables. Some ML algorithms, such as simple linear regression, would have numerical issues if each dummy variable was included in the model. The reason is that, for each sample, these variables all add up to one and this would provide the same information as the intercept. Other ML algorithms are unaffected by this (e.g. tree based models). That is why two dummy sets were created, one containing all dummy categories (HRdataDummy with 52 columns), and one with # of categories -1 (HRdataDummyFullRank with 46 columns).

A correlation matrix can only handle numeric variables. Some ML algorithms have a significant performance decrease if the data is highly correlated, as indicated in the paragraph above. To check for correlations and the structure within this dataset a correlation matrix was created. It was created based on the dataset HRdataDummyFullRank, and is shown in Figure 1. A few strong correlations can be spotted. Next, the predictors with a minimum correlation of 0.75 got identified and removed. Five were found namely, *JobLevel, YearsAtCompany, Department.Research & Development, Department.Sales,* and *BusinessTravel.Travel_Rarely*. The author created a new dataset excluding these five variables, called HRdataDummyFullRankLowCorr with 41 columns.

During the first impression analysis, the predictor EmployeeNumber stood out. The author decided that there is no causal link between EmployeeNumber and attrition. And since uninformative predictors can decrease the performance of some models (Kuhn & Johnson, 2016), this variable is deleted from the datasets.

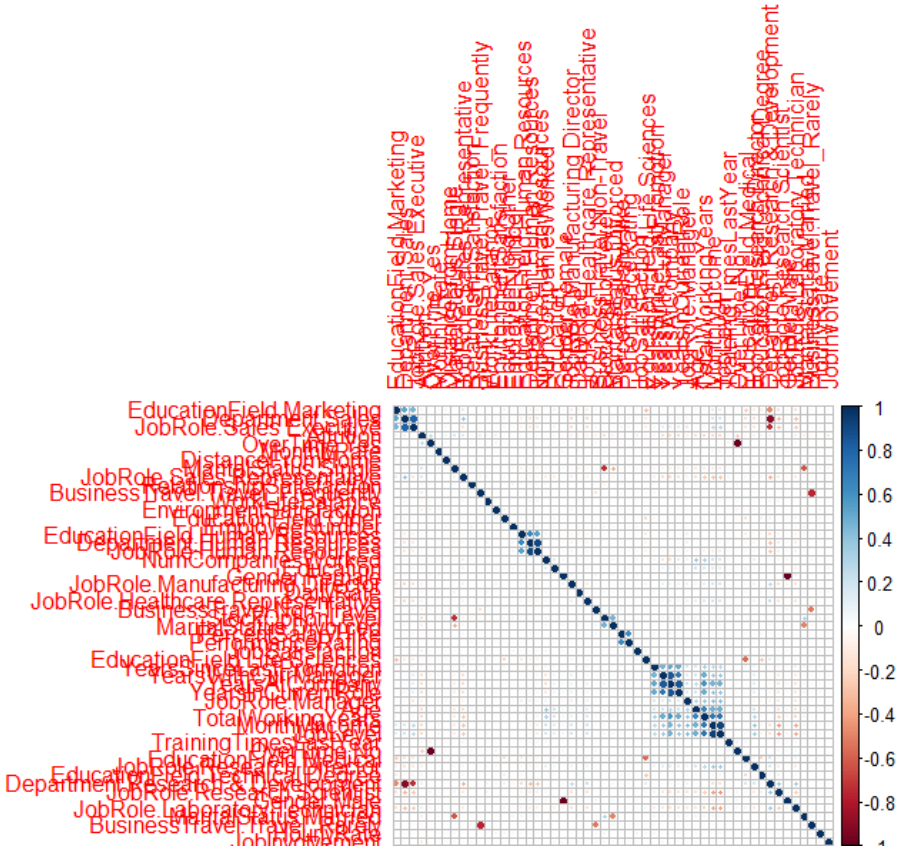


Figure 1 Correlation Matrix HRdataDummyFullRank, method spearman

4.1 Pre-processing procedures

In addition to the already created data sets, for some ML algorithms additional pre-processing procedures got conducted. Some models produce better results when the data is preprocessed. For example, centering and scaling will positively affect K-Nearest Neighbors (KNN), but will only hinder the interpretability of tree based models. The preprocessing procedures used in this study are centering, scaling, and box-cox transformation, and will be discussed next.

4.1.1 Centering and Scaling

The most straightforward and common data transformation is to center and scale the predictor variables. These are preprocessing procedures done for some of the models. To center a predictor variable, the average predictor value is subtracted from all the values. This results in the predictor having a zero mean. To scale a predictor variable, each value of the predictor variable is divided by its standard deviation. Scaling the data coerces the values to have a common standard deviation of one. These manipulations are generally used to improve the numerical stability of some calculations. Some models, benefit from the predictors being on a common scale (Kuhn & Johnson, 2016). The only real downside to these transformations is a loss of interpretability of the individual values, since the data is no longer in the original units.

4.1.2 Box-cox transformation

Another preprocessing step used in this study is a transformation proposed by (Box & Cox, 1964). The purpose of this transformation, in this study, is to remove the distributional skewness in the predictors. An un-skewed distribution is one that is

roughly symmetric. This means that the probability of falling on either side of the distribution's mean is roughly equal. A right-skewed distribution has a large number of points on the left side of the distribution (smaller values) than on the right side (larger values).

The formula for the sample skewness statistic is:

$$skewness = \frac{\sum(x_i - \bar{x}\ )^3}{(n-1)v^{3/2}}$$

$$where\ v = \frac{\sum(x_i - \bar{x}\ )^2}{(n-1)}$$

where x is the predictor variable, n is the number of values, and $\bar{x}$ is the sample mean of the predictor. If the predictor distribution is roughly symmetric, the skewness values will be close to zero. As the distribution becomes more right skewed, the skewness statistic becomes larger. Similarly, as the distribution becomes more left skewed, the value becomes negative.

Replacing the data with the log, square root, or inverse may help to remove the skew. A statistical method used to empirically identify an appropriate transformation is the (Box & Cox, 1964) method. They propose a family of transformations that are indexed by a parameter, denoted as λ:

$$x^* = \begin{cases} \frac{x^\lambda - 1}{\lambda} & if\ \lambda \neq 0 \\ \log(x) & if\ \lambda = 0 \end{cases}$$

In addition to the log transformation, this family can identify square transformation (λ = 2), square root (λ =0 .5), inverse (λ = −1), and others in between. λ is estimated by using the maximum likelihood estimation of the predictor. This procedure would be applied independently to each predictor data that contain values greater than zero.

For the HRdataDummyFullRankLowCorr data, 26 predictors were not transformed due to zero or negative values. From the remaining 13, 4 predictors had λ estimates within 1 ± 0.2, and for these 4 no transformation was applied. The remaining 9 predictors had λ estimates between -1.3 and 1.6. For λ estimates within 0 ± 0.2 a log transformation is found to be reasonable.

5. Machine learning algorithms

In this chapter the machine learning algorithms used to make the predictive models are described. The descriptions will be concise. For a comprehensive explanation of the mathematics behind the models used in this study the author refers the reader to (Hastie, Tibshirani, & Friedman, 2009), and to the references used in text. This chapter is split into the three subchapters: Linear classification, nonlinear classification, and classification trees and rule based models. In each of these subchapters the corresponding machine learning algorithms get described.

5.1 Linear classification

In this section, the linear classification models used are being described, as well as the tuning parameters used to get to the optimal model. All the models in this section are trained based on the HRdataDummyFullRankLowCorr dataset.

5.1.1 Logistic regression

The logistic regression model is one of the basic linear models for classification. The model is very popular due to its simplicity and ability to make inferential statements about model terms. Logistic regression is a specific category of regression best used to predict for binary or categorical dependent variables. Logistic regression finds parameter values that maximizes the binomial likelihood function. Even though the equation used in logistic regression is nonlinear, it produces linear classification boundaries.

For logistic regression, formal statistical hypothesis tests can be conducted to assess whether the slope coefficients for each predictor are statistically significant. This application of logistic regression will also be used to check the hypotheses stated in chapter 3 (p. **Error! Reference source not found.**). A Z statistic is commonly used for t hese models, and is essentially a measure of the signal-to-noise ratio: the estimated slope is divided by its corresponding standard error. Using this statistic, the predictors can be ranked to understand which terms had the largest effect on the model (Kuhn & Johnson, 2016).

5.1.2 Linear Discriminant Analysis

Linear Discriminant Analysis is explained as deriving a z-score, which is a linear combination of two or more independent variables that will discriminate best between two (or more) different categories or groups. The z-scores calculated using the discriminant functions is then used to estimate the probabilities that an observation belongs to a class. Linear discriminant analysis could be formulated as a model that minimizes the total probability of misclassification. In this analysis, it is assumed that the predictors in each class shared a common covariance structure. The consequence of this assumption is that the class boundaries are linear functions of the predictors. (Hastie et al., 2009)

5.1.3 Penalized logistic regression

Some classification models utilize penalties (or regularization) to improve the fit to the data (Kuhn & Johnson, 2016). In this paper, a penalty term for the logistic regression model is included. Recall that logistic regression finds parameter values that maximizes the binomial likelihood function.

The method used for regularizing linear regression models in this paper is the Lasso and Elastic-Net Regularized Generalized Linear Models(glmnet). This glmnet model uses ridge and lasso penalties simultaneously and structures the penalty in the following way (Friedman, Hastie, & Tibshirani, 2010):

$$\log L(p) - \lambda \left[(1-a)\frac{1}{2}\sum_{j=1}^{P}\beta_j^2 + \alpha \sum_{j=1}^{P}|\beta_j| \right]$$

Here, the α value is the “mixing proportion” that toggles between the pure lasso penalty (when α = 1) and a pure ridge-regression-like penalty (α = 0). The other tuning parameter λ controls the total amount of penalization.

The model was tuned over a α from 0-1 divided into 10 equal steps, and a λ between 0.01 and 0.2 divided into 40 equal steps. The final model had a α of 0, and a λ 0.01, which suggests a small amount of penalization on a pure ridge-regression-like penalty.

5.2 Nonlinear classification models

The previous section described models that were intrinsically linear—the structure of the model would produce linear class boundaries. This section deals with some intrinsically nonlinear models. There are other nonlinear models that use trees or rules for modeling the data, these are discussed in the next section. All models in this section are trained based on the HRdataDummyFullRankLowCorr dataset.

5.2.1 Neural Networks

Neural networks (NN) are powerful nonlinear regression techniques inspired by theories about how the brain works (Goodfellow et al., 2016). The two classes (attrition yes/no) can be encoded into two binary columns of dummy variables and then used as the outcomes for the model. The outcome is modeled by an intermediary set of unobserved variables called hidden units. These hidden units are linear combinations of the original predictors. Each hidden unit is a linear combination of some or all of the predictor variables.

The structure of the model described here is the simplest neural network architecture: a single-layer feed-forward network. There are many other kinds (Goodfellow et al., 2016), such as models where there are more than one layer of hidden units (i.e., there is a layer of hidden units that models the other hidden units), or the model architectures have loops going both directions between layers.

Neural networks for classification have a significant potential for over-fitting. When optimizing the entropy, weight decay attenuates the size of the parameter estimates. This can lead to much smoother classification boundaries. Also, model averaging helps reduce over-fitting (Kuhn & Johnson, 2016). In this case, the class probability estimates would be averaged across networks and these average values would be used to classify samples. In this study both single NN’s and averaged NN’s are being tested.

(Kuhn & Johnson, 2016) found that spatial sign transformation can have a significant positive impact on the performance of neural networks. For these data that was also the case, so the results of the NN and Averaged NN are based on data pre-processed including the spatial sign transformation (Serneels, De Nolf, & Van Espen, 2006).

The models were tuned over the number of units in the hidden layer ranging from 1 to 10, as well as a of weight decay between 0 and 2 (λ = 0, 0.1, 1, 2). The best NN used only one hidden unit with a λ = 0.1. The averaged NN was the average of 10 single NN and similar as the single NN used only one hidden units and a λ = 0.1. In both these models only a single hidden unit is found to be optimal, which suggests the outcome can be linearly separated.

5.2.2 Flexible Discriminant Analysis

In the previous section, the motivation for classical linear discriminant analysis was based on minimizing the total probability of misclassification. It turns out that the same model can be derived in a completely different manner. (Hastie, Tibshirani, & Buja, 1994) describe a process where, for C classes, a set of C linear regression models can be fit to binary class indicators and show that the regression coefficients from these models can be post-processed to derive the discriminant coefficients. This allows the idea of linear discriminant analysis to be extended in a number of ways (Kuhn & Johnson, 2016). First, models such as the lasso, ridge regression, or MARS, can be extended to create discriminant variables. For example, MARS can be used to create a set of hinge functions that result in discriminant functions that are nonlinear combinations of the original predictors. As another example, the lasso can create discriminant functions with feature selection. This conceptual framework is referred to as flexible discriminant analysis (FDA).

An FDA model was tuned and trained with a first-degree MARS hinge functions where the number of retained terms ranged from 1 to 30 and with a degree 1 and 2. The optimal model had 29 retained terms with a degree of 1.

5.2.3 Support Vector Machines

Support vector machines are a class of statistical models first developed in the mid-1960s by Vladimir Vapnik. In later years, the model has evolved considerably into one of the most flexible and effective machine learning tools available (Kuhn & Johnson, 2016). A support vector machine constructs a hyperplane or set of hyperplanes in higher dimensional space for achieving class separation. The intuition here is that a good separation is achieved by the hyperplane that has the largest distance to the nearest training data points of any class- the larger the margin the lower the generalization error of the classifier. For this reason, it is also referred to as maximum margin classifier (Kuhn & Johnson, 2016).

Alternate versions of the support vector machine model also exist (Kuhn & Johnson, 2016), such as least squares support vector machines (Suykens & Vandewalle, 1999), relevance vector machines (Tipping, 2001), and import vector machines (J. Zhu & Hastie, 2005). There are several approaches to using SVMs. In this paper, we evaluated the radial basis function kernel. It should be noted that support vector machines can be negatively affected by including non-informative predictors in the model. The equation for this kernel is:

$$K(x, x') = \exp(-\gamma \|x - x\|^2)$$

The radial basis function kernel was tuned over a cost of 2 with a power ranging from -4 till 4. The optimal model results in σ = 0.009475476 and a cost of 2.

5.2.4 K-Nearest Neighbors

K-Nearest Neighbors (KNN) uses a sample's geographic neighborhood to predict the sample's classification. KNN for classification predicts a new sample using the K closest samples from the training set. "Closeness" is determined by a distance metric, and the choice of metric depends on predictor characteristics. For any distance metric, it is important to recall that the original measurement scales of the predictors affect the

resulting distance calculations. This implies that if predictors are on widely different scales, the distance value between samples will be biased towards predictors with larger scales. (Kuhn & Johnson, 2016)

The neighborhood range evaluated for tuning was between 1 and 451. The optimal model was found to have a neighborhood of 151, which is quite large. A large number of neighbors stimulates underfitting and a drop in corresponding predictive performance will be the result.

5.2.5 Naïve Bayes

Bayes' Rule answers the question "based on the predictors that we have observed, what is the probability that the outcome is class C?" Thus, Bayes' Rule is essentially a probability statement. The underlying logic to using the Bayes' rule for machine learning is as follows: To train a target function fn: X → Y, which is the same as, P (Y|X), we use the training data to learn estimates of P (X|Y) and P(Y). Using these estimated probability distributions and Bayes' rule new X samples could then be classified. (Kuhn & Johnson, 2016)

The Naïve Bayes model simplifies the probabilities of the predictor values by assuming that all of the predictors are independent of the others. This is an extremely strong assumption. For most applications, it would be difficult to claim that this assumption is realistic. However, the assumption of independence yields a significant reduction in the complexity of the calculations.

These predictors were modeled using either a normal distribution or a nonparametric density (the density type was treated as a tuning parameter), and a Laplace correction ranging between 0 and 2. The Laplace correction is a smoothing technique which counters the effect of the predictor not having a training sample, and thus no posterior probability. For this dataset, the best tuned model had a Laplace of 0 and used the nonparametric density distribution.

5.3 Classification Trees and Rule-Based Models

Classification trees fall within the family of tree-based models and consist of nested if-then statements. Some benefits of trees are: they can be highly interpretable, can handle many types of predictors as well as missing data. Some weaknesses are: they suffer from model instability, and may not produce optimal predictive performance.

For tree models, the splitting procedure may be able to make more dynamic splits of the data, such as groups of two or more categories on either side of the split (Kuhn & Johnson, 2016). However, to do this, the algorithm must treat the categorical predictors as an ordered set of bits. Therefore, when fitting trees and rule-based models, a choice must be made regarding the treatment of categorical predictor data:

1. Each categorical predictor can be entered into the model as a single entity so that the model decides how to group or split the values. In this study this refers to using the HRdata set.
2. Categorical predictors are first decomposed into binary dummy variables. In this way, the resulting dummy variables are considered independently, forcing

binary splits for the categories. In this study this refers to using the HRdataDummy set.

Which approach is more appropriate depends on the data and the model. For example, if a subset of the categories are highly predictive of the outcome, the first approach is probably best. However, this choice can have a significant effect on the complexity of the model, and the performance. Although the trees are identifying similarly important information, the independent category tree is much easier to interpret than the grouped category tree. In this study, models will be created using both approaches described above.

5.3.1 Classification trees (CART, and C4.5)

The aim of classification trees is to partition the data into smaller, more homogeneous groups. Homogeneity in this context means that the nodes of the split are more pure (i.e. contain a larger proportion of one class in each node). A simple way to define purity in classification is by maximizing accuracy or equivalently by minimizing misclassification error. However, accuracy as a measure of purity can be misleading, since the measure's focus is on partitioning the data in a way that minimizes misclassification rather than a focus on partitioning the data in a way that place samples primarily in one class (Kuhn & Johnson, 2016). Two alternative measures, the Gini index and cross-entropy shift the focus from accuracy to purity. Here Gini is equal to:

$$G = \sum_{K=1}^{K} \hat{P}mk \left(1 - \hat{P}mk\right),$$

And the cross-entropy equation:

$$D = -\sum_{K=1}^{K} \hat{P}mk \log(\hat{P}mk)$$

Where $\hat{P}mk$ is the proportion of training observations in the m^{th} region that belongs to the k^{th} class. If the $\hat{P}mk$ is close to zero or one, then G and D will be small.

Gini is the criterion used by Classification and Regression Trees (CART) model. In this model pruning is applied via the cost of complexity. And the cross-entropy criterion is used by the C4.5 model. Both models are likely to overfit the data. CART uses a cost function as pruning method; and C4.5 uses either a simple elimination of a sub-tree, or raising a sub-tree so that it replaces a node further up the tree. Both these models are used in this study. While CART and C4.5 classification trees are the most widely used, there has been extensive research on and many other proposals for tree based models (Kuhn & Johnson, 2016).

The CART model was tuned over 30 different trees with different complexity parameter (cp) values. Any split that does not decrease the overall lack of fit by a factor of cp is not attempted. The C4.5 like tree was trained using the standard tuning parameters ranging the confidence factor between 0.01 and 0.5 and a minimum number of instances ranging between 1 and 3. The best tuned CART model had a cp of 0.0009040424 and 0.007232339, for the HRdata and HRdataDummy respectively.

And the best tuned C4.5 model had a confidence factor of 0.255 and 0.5, for the HRdata and HRdataDummy respectively, and both had a minimum number of instances of three.

5.3.2 Rule-Based Models: PART

(Frank & Witten, 1998) describe a rule model called PART. Here, a pruned C4.5 tree is created from the data and the path through the tree that covers the most samples is retained as a rule. The samples covered by the rule are discarded from the data set and the process is repeated until all samples are covered by at least one rule. Although the model uses trees to create the rules, each rule is created separately and has more potential freedom to adapt to the data.

Since this model uses C4.5 trees, the same tuning parameters are used. The final model had a cp of 0.01 and 0.5 for the HRdata and HRdataDummy respectively, and in both cases the model was pruned.

5.3.3 Bagged Trees

Bagging trees for classification uses an unpruned classification tree for modeling, in this case, the two classes of attrition. Each model in the ensemble is used to predict the class of the new sample. Each model has equal weight in the ensemble, and can be thought of as casting a vote for the class it thinks the new sample belongs to. The total number of votes within each class are then divided by the total number of models in the ensemble (M) to produce a predicted probability vector for the sample. The new sample is then classified into the group that has the most votes, and therefore the highest probability. (Kuhn & Johnson, 2016)

According to (Kuhn & Johnson, 2016), bagging performance often plateaus with about 50 trees, so 50 was selected as the number of trees for each of these models. Both of these ROC curves are smoother than curves produced with CART or C4.5, which is an indication of bagging's ability to reduce variance via the ensemble. Additionally, both bagging models have better AUCs than either of the previous tree models.

5.3.4 Random Forests

Random forests are quite similar to bagging, each tree in the forest casts a vote for the classification of a new sample, and the proportion of votes in each class across the ensemble is the predicted probability vector. However, the type of tree changes in the algorithm, and the tuning parameter of number of randomly selected predictors to choose from at each split (denoted as mtry) is now added (Breiman, 2002). The idea behind randomly sampling predictors during training is to de-correlate the trees in the forest.

To tune mtry, (Kuhn & Johnson, 2016) recommend starting with five values that are somewhat evenly spaced across the range from 2 to P, where P is the number of predictors. They also recommend starting with an ensemble of 1,000 trees and increasing that number if performance is not yet close to a plateau. This is applied to the HRdata and HRdataDummy sets, and resulted in mtry of 2 for both datasets.

5.3.5 Boosting

The idea behind boosting is to combine many weak classifiers (e.g., a classifier that predicts marginally better than random) into a strong classifier. Boosting can be applied

to any classification technique, but classification trees are a popular method for boosting, since these can be made into weak learners by restricting the tree depth to create trees with few splits. Since classification trees are a low bias/high variance technique, the ensemble of trees helps to drive down variance, producing a result that has low bias and low variance. There are many species of boosting algorithms, and in this paper two major ones Stochastic Gradient Boosting and C5.0, are used. (Kuhn & Johnson, 2016)

5.3.5.1 Stochastic Gradient Boosting

(Friedman, J., Hastie, T., & Tibshirani, 2000) worked to provide statistical insight of the AdaBoost algorithm. For the classification problem, they showed that it could be interpreted as a forward stagewise additive model that minimizes an exponential loss function (Kuhn & Johnson, 2016). This framework led to algorithmic generalizations such as Real AdaBoost, Gentle AdaBoost, and LogitBoost. These generalizations were put into a unifying framework called gradient boosting machines. The basic principles of gradient boosting are as follows: given a loss function (e.g., shrinkage) and a weak learner (e.g., classification trees), the algorithm seeks to find an additive model that minimizes the loss function. The gradient (e.g., residual) is calculated, and a model is added to the previous model, and the procedure continues for a user-specified number of iterations (Kuhn & Johnson, 2016).

When trees are used as the base learner, as in this study, basic gradient boosting has two tuning parameters: tree depth (or interaction depth) and number of iterations. Also, the model can be tuned over a loss function, in this case shrinkage is implemented. In this study a tuning parameter grid was constructed where interaction depth ranged from 1 to 9, number of trees ranged from 100 to 2,000, and shrinkage ranged from 0.01 to 0.1. This grid was applied to constructing a boosting model for the HRdataDummy and resulted in a model with: interaction depth of 1, 600 trees, and a shrinkage of 0.1.

5.3.5.2 C5.0

C5.0 is a more advanced version of the C4.5 classification model, and has additional features. It is claimed to be faster, use memory more efficient, gets similar results with smaller trees, offers support for boosting, allows the user to give the classes different weights, and offers winnowing (i.e. removal of unhelpful predictors). (Kuhn & Johnson, 2016)

In this study, several variations of the C5.0 model were evaluated:

- Single tree- and rule-based models
- Tree and rules with boosting (up to 100 iterations)
- Using all predictors and using the winnowed set

The final best tuned model based on the HRdataDummy set is a rule based model with 90 iterations where winnowing was not used.

6. Model Validation technique

It is inappropriate to validate the models on the same data the model is trained with, since the data is not new to the predictive model the results will be biased and show an over optimistic performance of the model. So, to properly validate the trained predictive models there is need for a test set (i.e. a set which the predictive model has not seen before). The author chose to split the data into a training and test set of 75% and 25% respectively. Since the rate of interest is under represented, stratified random sampling is used to split the data. This will ensure that the proportions of attrition are equal in each set. The number of recorders for the training set and test set are 1103 and 367 respectively.

6.1 Resampling method

Additionally, to increase the reliability of the training process outcomes the practitioner can choose to use a resampling technique. Generally during the resampling process a subset of samples are used to fit a model, and the remaining samples are used to estimate the efficacy of the model. The resampling technique try to improve the bias and variance properties of the trained models. Here the bias is the difference between the estimated and true value of performance, and variance is the certainty of the bias. For example, an unbiased and high variance method may produce very different results when repeated. Some of the more common resampling techniques are: K-fold cross-validation, Leave-one-out cross-validation, repeated k-fold cross-validation, generalized cross-validation, repeated training/test splits, and bootstrap.

No resampling method is uniformly better than another. Several factors should be considered before making the choice. In this case the sample size is small. For a small sample size (Kuhn & Johnson, 2016) suggests using repeated 10-fold cross-validation for several reasons: the bias and variance properties are good and, given the sample size, the computational costs are not large. The author chose for 5 repeats of a 10-fold cross-validation as the validation method used during this study.

7. Model evaluation metrics

Although many machine learning techniques can be used both for regression and classification, the model evaluation metric is very different. Metrics like RMSE and R2 are not appropriate in the context of classification. Some metrics that are useful for classification are accuracy, sensitivity, specificity, Positive predictive value, balanced accuracy, F1 score, and Receiver Operating Characteristic (ROC). To improve the interpretability of the evaluation metrics, Table 2 shows a confusion matrix and its basic measures (e.g. TP, FP, FN, and TN) as well as the relationship with the evaluation metrics used in this study. These will be explained next.

7.1 Accuracy

Accuracy is one of the simplest metrics. It reflects the agreement between the observed and predicted classes, and has the most straightforward interpretation. However, in situations where the costs are different, accuracy may not measure the important model characteristics. Also, the natural frequencies of each class must be taken into consideration. For example, in the current case a simple rule stating that all

employees will stay at the company will already result in an accuracy of 84%. Although 84% can be considered high in other predictive modeling applications, it is the base rate in our study. If the class imbalance would be even more severe, it could be the case that a predictive model can achieve almost perfect accuracy with only one rule. The formula for Accuracy is:

$$Accuracy = \frac{\#\ samples\ corectly\ predicted}{Total\ population} = \frac{TP + TN}{TP + FP + FN + TN}$$

Table 2 Confusion Matrix including evaluation metrics used in current study

<table>
<tr><td colspan="2" rowspan="2"></td><td colspan="2">True Condition</td><td colspan="2" rowspan="2"></td></tr>
<tr><td>Condition Positive</td><td>Condition Negative</td></tr>
<tr><td rowspan="2">Predicted condition</td><td>Predicted Condition Positive</td><td>True positive (TP)</td><td>False positive (FP)</td><td>PPV (or Precision) $\frac{TP}{TP + FP}$</td><td></td></tr>
<tr><td>Predicted Condition Negative</td><td>False negative (FN)</td><td>True negative (TN)</td><td colspan="2"></td></tr>
<tr><td colspan="2" rowspan="2"></td><td rowspan="2">Sensitivity (or Recall) $\frac{TP}{TP + FN}$</td><td>Specificity $\frac{TN}{TN + FP}$</td><td></td><td>Balanced Accuracy $\frac{1}{2} * \left(\frac{Sensitivity}{+Specificty}\right)$</td></tr>
<tr><td></td><td>F1 score $2 * \frac{PPV * Sensitivity}{PPV + Sensitivity}$</td><td></td></tr>
</table>

7.2 Sensitivity and specificity

For two class classification problems sensitivity and specificity are two additional statistics that can be relevant. The sensitivity (or recall) of the model is the rate that the event of interest is predicted correctly for all samples having the event, or

$$Sensitivity = \frac{\#\ samples\ with\ the\ event\ and\ predicted\ to\ have\ the\ event}{\#\ samples\ having\ the\ event} = \frac{TP}{TP + FN}$$

The sensitivity is sometimes considered the true positive rate since it measures the accuracy in the event population. Conversely, the specificity is defined as the rate that nonevent samples are predicted as non events, or

$$Specificity = \frac{\#\ samples\ without\ the\ event\ and\ predicted\ as\ nonevents}{\#\ samples\ without\ the\ event} = \frac{TN}{TN + FP}$$

7.3 Positive predictive value

Like sensitivity and specificity, the positive predictive value (PPV), also known as precision, can be calculated using the confusion matrix. One often overlooked aspect

of sensitivity and specificity is that they are *conditional* measures. Sensitivity is the accuracy rate for only the event population (and specificity for the nonevents). Intuitively, if the event is rare, this should be reflected in the answer. The PPV takes this into account and is an unconditional measure for the positive condition. The formula is:

$$PPV = \frac{TP}{TP + FP}$$

(Kuhn & Johnson, 2016)

7.4 Balanced Accuracy

Since the accuracy can be a misleading performance measure (as described in the accuracy section), it may falsely suggest above-chance generalizability. To safeguards against reporting an optimistic accuracy estimate the balanced accuracy is introduced. The balanced accuracy can be defined as the average accuracy obtained on either class. Based on a confusion matrix the balanced accuracy is given by:

$$Balanced\ Accuracy = \frac{1}{2} * (Sensitivity + Specificty) = \frac{1}{2} * (\frac{TP}{TP + FN} + \frac{TN}{TN + FP})$$

If the classifier performs equally well on either class, this term reduces to the conventional accuracy. In contrast, if the conventional accuracy is high only because the classifier takes advantage of an imbalanced test set, then the balanced accuracy will drop to chance.

(Brodersen, Ong, Stephan, & Buhmann, 2010)

7.5 F score

The F1 score is a measure of a test's accuracy for binary classification problems. It considers both the sensitivity (or recall) and PPV (or precision). The F1 score can be interpreted as a weighted average of these two measures. The F1 score will result in a grade between 1 and 0 where 1 is considered best. The formula is:

$$F1 = 2 * \frac{PPV * Sensitivity}{PPV + Sensitivity}$$

Note, however, that the F1 score does not directly consider the true negatives in its calculation.

(Yang & Liu, 1999)

7.6 Receiver Operating Characteristic (ROC) Curve

The ROC curve is created by evaluating the class probabilities for the model across a continuum of thresholds. For each candidate threshold, the resulting true-positive rate (i.e., the sensitivity) and the false-positive rate (one minus the specificity) are plotted against each other. The default threshold is 50%, which is also the threshold used to indicate the results. This threshold can be changed to choose a new sensitivity and specificity trade-off, as discussed in the previous section.

The ROC curve can also be used for a quantitative assessment of the model. A perfect model that completely separates the two classes would have 100% sensitivity and specificity. Graphically, the ROC curve would be a single step between (0, 0) and (0, 1) and remain constant from (0, 1) to (1, 1). The area under the ROC curve for such a model would be one. A completely ineffective model would result in an ROC curve with an area under the ROC curve of 0.5.

One advantage of using ROC curves to characterize models is that, since it is a function of sensitivity and specificity, the curve is insensitive to disparities in the class proportions (Fawcett, 2006). Since altered cutoff values can completely change the other metrics described above, performance metrics that are independent of probability cutoffs (such as the area under the ROC curve) are likely to produce more meaningful contrasts between models. That is why this metric is chosen to select the best model on during the training process.

8. Remedies for severe class imbalance

When modeling discrete classes, the relative frequencies of the classes can have a significant impact on the effectiveness of the model. An imbalance occurs when one or more classes have very low proportions in the training data as compared to the other classes. This is the case in the current dataset. Several remedies for severe class imbalances exist. In this study the two remedies altered cutoff value and sampling are being implemented. Some other remedies include, e.g. model tuning on minority case accuracy, adjusting prior probabilities, cost-sensitive training, and unequal case weights (Kuhn & Johnson, 2016).

8.1 Alternated Cutoffs

When there are two possible outcome categories, a method for increasing the prediction accuracy of the minority class samples is to determine alternative cutoffs for the predicted probabilities which effectively changes the definition of a predicted event. The most straightforward approach is to use the ROC curve, since it calculates the sensitivity and specificity across a continuum of cutoffs. Using this curve, an appropriate balance between sensitivity and specificity can be determined.

Several techniques exist for determining a new cutoff. First, if there is a specific target that must be met for the sensitivity, or specificity, this point can be found on the ROC curve and the corresponding cutoff can be determined. Another approach is to find the point on the ROC curve that is closest (i.e. the shortest distance) to the perfect model (a model with 100% sensitivity and 100% specificity), which is associated with the upper left corner of the plot. Another approach for determining the cutoff uses Youden's J index, which measures the proportion of correctly predicted samples for both the event and nonevent groups. This index can be computed for each cutoff that is used to create the ROC curve. Youden and the upper left corner approach often result in very similar results. During this study the upper left corner approach is implemented.

It is worth noting that changing the cutoff does not change the core of the model, i.e. the same model parameters are being used. Changing the cutoff to increase the sensitivity does not increase the overall predictive effectiveness of the model. The main

impact that an alternative cutoff has is to make trade-offs between particular types of errors. For example, in a confusion matrix (Table 2), alternate cutoffs can only move samples up and down rows of the matrix. Thus, using an alternative cutoff does not induce further separation between the classes.

In our analysis, the alternate cutoff values were determined only for the most promising models. The alternate cutoff value was derived from the training set, since, if the test set was used it will no longer be an unbiased source to judge the model performance. However, using the training set does has it downsides, e.g. the results on the training are likely to have an optimistic bias and can thus lead to an inaccurate assessment of the sensitivity and specificity. However, these effects are being mitigated by using validation methods (described in chapter 6 p.23). A better estimation of the altered cutoff value would be by creating an additional evaluation set (i.e. an extra dataset not used during the training process), estimate the optimal cutoff value based on this dataset (validation set), assess the performance of the model using the test set.

8.2 Sampling Methods

When using alternated cutoff values, there is almost always a decrease in either sensitivity or specificity as one is increased. Unlike alternate cutoff values, sampling approaches have the benefit of enabling better trade-offs between sensitivity and specificity. There are several methods of sampling, and in this study the methods oversampling and synthetic minority over-sampling technique (SMOTE) are being evaluated.

Two general post hoc approaches are down-sampling and up-sampling the data. Up-sampling is any technique that simulates or imputes additional data points to improve balance across classes, while down-sampling refers to any technique that reduces the number of samples to improve the balance across classes. (Ling & Li, 1998) provide one approach to up-sampling in which cases from the minority classes are sampled with replacement until each class has approximately the same number. The training set in this study contain 178 cases of attrition and 925 cases of non-attrition. Applying up-sampling resulted in adding 747 randomly chosen replacement samples.

The synthetic minority over-sampling technique (SMOTE) (Chawla, Bowyer, Hall, & Kegelmeyer, 2002) is a data sampling procedure that uses both up-sampling and down-sampling and can be considered a hybrid sampling approach. This approach has three operational parameters: the amount of up-sampling, the amount of down-sampling, and the number of neighbors that are used to impute new cases. To up-sample for the minority class, SMOTE synthesizes new cases. To do this, a data point is randomly selected from the minority class and its K-nearest neighbors (KNNs) are determined, in this study we used a k equal to five. The new synthetic data point is a random combination of the predictors of the randomly selected data point and its neighbors. While the SMOTE algorithm adds new samples to the minority class via up-sampling, it also can down-sample cases from the majority class via random sampling in order to help balance the training set. Implementing this approach using both up and down sampling resulted in a training set with 534 cases of attrition and 712 cases of non-attrition.

It should be noted that when using modified versions of the training set, resampled estimates of model performance can become biased. For example, if the data are up-sampled, resampling procedures are likely to have the same sample in the cases that are used to build the model as well as the holdout set, leading to optimistic results. Despite this, resampling methods can still be effective at tuning the models.

9. Results of predictive modeling

The results of this study are split into two parts. In part 1 (this chapter) the results of the machine learning algorithms will be presented. Part 2 (Chapter 10, p.35) represents the results of the hypotheses and additional HR voluntary attrition findings. In this chapter, the author will provide the reader with a better context for interpreting the results by explaining the concept of measurement error. Next, we will discuss the results of the predictive models. This chapter will end with an evaluation of the severe class imbalance remedies.

9.1 Measurement error

The better we understand the measurement system and its limits, as well as the relationship between predictors and the response, the better we can foresee the limits of the model performance. To set expectations for the predictive modeling performance the author wants to inform the reader about two types of errors namely, measurement error in the outcome and measurement error in the predictors.

9.1.1 Measurement error in the outcome

During the modeling process the goal is to eliminate the model error. However, there is a component that cannot be eliminated through the modeling process. This is the case when the outcome contains significant measurement noise. As noise increases, the models used become virtually indistinguishable in terms of their predictive performance. This means that the advantages that some of the more complex models bring are only advantageous when the measurement system error is relatively low. (Kuhn & Johnson, 2016)

Some aspects that can increase the measurement error in the outcome is right censoring and left truncation (Carrión, Solano, Gamiz, & Debón, 2010). Simply put, each dataset can be thought of as an observation window in time. Here left truncation is the periods on the left of this observation window. This period will also include cases of voluntary leave, but these will remain unobserved, and only the cases for which attrition did not happen are included in the dataset. The opposite of this phenomenon happens during right censoring, this is the period on the right side of the observation window. In this study, it can be interpreted as the non-attrition cases in the observation window (i.e. dataset) that eventually did leave the organization after the observation window ended.

9.1.2 Measurement error in the predictors

Measurement errors in the predictors can cause considerable issues when building models, especially in terms of reproducibility of the results on future data sets. Although it is often assumed that predictors are measured without errors, this is not always the case. The effect of error in measurement in the predictors can be drastic. The effect

depends on several factors namely: The amount of randomness, the importance of the predictors, the type of models being used, as well as others. Future results may be poor because the underlying predictor data are different than the values used in the training set. (Kuhn & Johnson, 2016)

The dataset used in this study contains 34 different predictors. Although some predictors are expected to have low measurement error in the predictors, like for example age, gender, total work years, etc. Others are expected to include some noise, for example job satisfaction, work life balance, performance rating, etc.. The latter are expected to have noise, since there may be a difference in how people perceive the object (rater-to-rater noise). Also, it can be imagined that not all employees are totally honest when filling in these types of surveys for personal reasons. As noted before, the effect of these predictors on the model depend on several factors. And it is important for the reader to keep these in mind while interpreting the model. Also, although these predictors have noise, this does not mean they are unable to provide additional information about the outcome.

9.2 Comparison of the models

In this section the results of the models used in this study will be compared. As discussed in the chapter 7 (p.23), model metrics can be severely influenced by the cutoff value used; so, in this section the ROC, which is insensitive to the cutoff value, is chosen to compare the models. The results of the modeling process validated based on the test set are shown in Table 3.

For the linear classification models the results are all very similar with an approximate area under the ROC curve (AUC) of 0,83. This result is good compared to the other type of models, indicating attrition can be linearly separated.

The non-linear classification models include the best model based on AUC namely the support vector machine. However, the other non-linear classification models performed worse compared to the linear classification models, further confirming the linear relationship.

Classification trees and rule based models were expected to have difficulty with the class imbalance. Compared to the other two groups, the classification trees and rule based models had the worst performance. The more complex models did improve the AUC over the simpler tree and rule based models, with gradient boost machine performing the best with a AUC of 0.82.

To better compare the models Figure 2 was created which shows the 95% confidence interval of the AUC for each of the models represented in Table 3. This figure can be used to compare the effectiveness and performance of the models. The bars represent 95% confidence intervals of the AUC that were derived using 2000 stratified bootstrap replicates.

Based on this figure it can be inferred that the simpler tree and rule based models together with KNN, and Naïve Bayes perform significantly worse than the other models. Also for the tree and rule based models the dataset used (dummy or non-dummy) did not result in significantly different models, indicating the dummy dataset might be preferred due to being more interpretable.

Table 3 The test set results of the modeling process based on the normal training set.

Model	Accuracy	Sensitivity	Specificity	PPV	F1 Score	Balanced Accuracy	AUC
Linear Classification Models							
Logistic Regression	0.8719	0.38983	0.96429	0,77876	0,49462	0.67706	0.8303
Linear Discriminant Analysis	0.8692	0.42373	0.95455	0,64103	0.51020	0.68914	0.8397
Penalized logistic regression	0.8692	0.32203	0.97403	0.70370	0.44186	0.64803	0.8319
Non-linear Classification Models							
Neural Network	0.8692	0.37288	0.96429	0.66667	0.47826	0.66858	0.828
Average Neural Network	0.8556	0.35593	0.95130	0.58333	0.44211	0.65362	0.8259
Flexible Discriminant Analysis	0.8801	0.44068	0.96429	0.70270	0.54167	0.70248	0.7972
Support Vector Machine	0.8856	0.38983	0.98052	0.79310	0.52273	0.68517	**0.8489**
KNN	0.8392	0	1	NA	NA	0.5000	0.7644
Naïve Bayes	0.8392	0	1	NA	NA	0.5000	0.7595
Classification Trees and Rules based models							
RPART (Gini based)	0.8147	0.32203	0.90909	0.40426	0.35849	0.61556	0.6901
RPART Dummy	0.8556	0.33898	0.95455	0.58824	0.43011	0.64676	0.7145
J48 (cross-entropy based)	0.842	0.30508	0.94481	0.40426	0.35849	0.62494	0.6867
J48 Dummy	0.812	0.42373	0.88636	0.58824	0.43011	0.6581	0.6581
PART	0.7902	0.28814	0.88636	0.51429	0.38298	0.58725	0.6411
PART Dummy	0.7929	0.30508	0.88636	0.41667	0.42017	0.59572	0.607
Bagged Trees	0.8692	0.35593	0.96753	0.67742	0.46667	0.66173	0.7649
Bagged Trees Dummy	0.8583	0.30508	0.96429	0.62069	0.40909	0.63469	0.7632
Random Forest	0.8665	0.2551	1	1	0.28986	0.58475	0.7839
Random Forest Dummy	0.8583	0.11864	1	1	0.21212	0.55932	0.7859
GBM Dummy	0.8638	0.44068	0.94481	0.60465	0.50980	0.69274	0.8154
C5.0 Dummy	0.8774	0.35593	0.97727	0.75000	0.48276	0.66660	0.7912

The more complex trees AUCs all significantly overlap, indicating that the simpler random forest might be preferred over the more complex C5.0 model. GBM seems to slightly outperform the other tree based models.

The confidence intervals of AUC for the linear and more complex non-linear classification models all significantly overlap, indicating that the outcome can be linearly separated and that the less complex logistic regression model might be preferred over the more complex neural network. From these models the SVM seems to slightly outperform the rest.

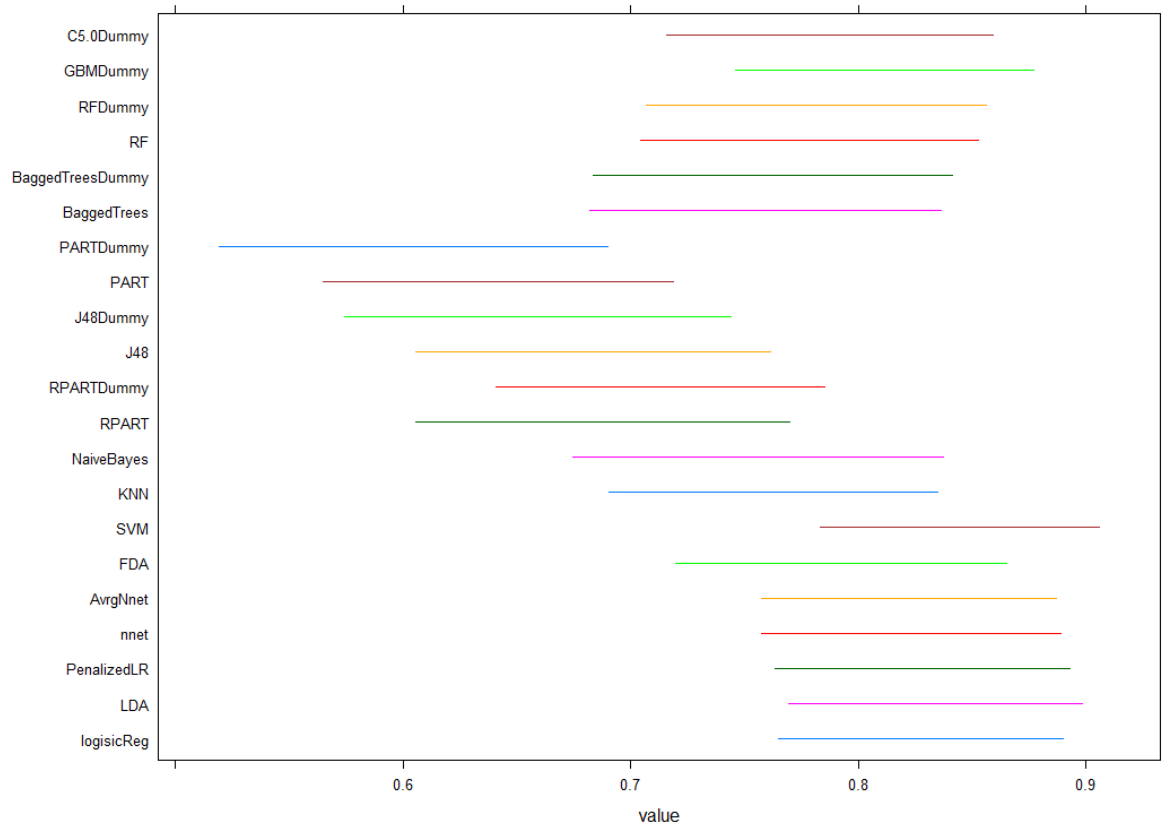


Figure 2 A plot of the test set ROC AUCs and their associated 95% confidence intervals. Here, the upper part of the graph represents the tree and rule based models, from 'c5.0 dummy' – 'RPART'; the middle with the non linear classification models, from 'NaiveBayes' – 'nnet'; and the lower part the linear classification models, from 'PenalizedLR' – 'logisticReg'.

9.3 Comparison of models after implementation of class imbalance remedies

Based on the previous results some models were selected to compare the results based on sampling and alternate cutoff value methods. The results of sampling are not expected to outperform the base case models based on the AUC, since the training set is sampled and is thus less comparative to the test set results. The implementation of these methods are, however, expected to result in more optimal sensitivity and specificity tradeoffs; and as a result in higher F1 scores and balanced accuracies. The alternate cutoff values are selected based on the point most close to the top left corner of the ROC based on the training set, as described in section 8.1 (p.26); and the results, represented in Table 4Table 3, are validated on the test set, and are representative for the models actual performance.

Table 4 Results most promising models base case vs. altered cutoff value vs. sampling results

Model	Accuracy	Sensitivity	Specificity	PPV	F1 Score	Balanced Accuracy	AUC
Logistic Regression	**0.8719**	**0.38983**	**0.96429**	**0,77876**	**0,49462**	**0.67706**	**0.8303**
Altered Cutoff	0.7901907	0.7288136	0.8019481	0.93916	0.82072	0.7654	
Logistic Regression UpSampled	0.7629	0.7288	0.7695	0.3772	0.49711	0.7491	0.8168
Logistic Regression SMOTE	0.7902	0.6949	0.8084	0.4100	0.51572	0.7517	0.815
Neural Network	**0.8692**	**0.37288**	**0.96429**	**0.66667**	**0.47826**	**0.66858**	**0.828**
Altered Cutoff	0.8174387	0.6949153	0.6949153	0.93501	0.79728	0.69492	
Neural Network UpSampled	0.8338	0.50847	0.89610	0.48387	0.49587	0.70229	0.7925
Neural Network SMOTE	0.8065	0.61017	0.84416	0.42857	0.50350	0.72716	0.7676
Flexible Discriminant Analysis	**0.8801**	**0.44068**	**0.96429**	**0.70270**	**0.54167**	**0.70248**	**0.7972**
Altered Cutoff	0.7329700	0.6610169	0.7467532	0.92000	0.76930	0,70389	
FDA UpSampled	0.782	0.6949	0.7987	0.3981	0.50617	0.7468	0.7729
FDA SMOTE	0.8338	0.47458	0.90260	0.48276	0.47863	0.68859	0.7569
Support Vector Machines	**0.8856**	**0.38983**	**0.98052**	**0.79310**	**0.52273**	**0.68517**	**0.8489**
Altered Cutoff	0.7574932	0.7796610	0.7532468	0.94694	0.85520	0.76645	
Support Vector Machines UpSampled	0.8638	0.44068	0.94481	0.60465	0.50980	0.69274	0.7753
Support Vector Machines SMOTE	0.8283	0.44068	0.90260	0.46429	0.45217	0.67164	0.7581
Random Forests Dummy	**0.8583**	**0.11864**	**1**	**1**	**0.21212**	**0.55932**	**0.7859**
Altered Cutoff	0.8801090	0.3050847	0.9902597	0.88150	0.45329	0.64767	
Random Forests Dummy UpSampled	0.8719	0.37288	0.96753	0.68750	0.48352	0.67021	0.7778
Random Forests Dummy SMOTE	0.8719	0.38983	0.96429	0.67647	0.49462	0.67706	0.7911
GBM dummy	**0.8638**	**0.44068**	**0.94481**	**0.60465**	**0.50980**	**0.69274**	**0.8154**
Altered Cutoff	0.8092643	0.6949153	0.8311688	0.93430	0.79702	0.76304	
GBM dummy UpSampled	0.8692	0.4068	0.9578	0.6486	0.5	0.6823	0.7751
GBM dummy SMOTE	0.8338	0.44068	0.90909	0.48148	0.46018	0.67488	0.7673

Table 5 Results of altered cutoff values based on the test set, base case vs. sampling results

Model	Threshold	Accuracy	Specificity	Sensitivity	PPV	F1 score	Balanced Accuracy
Logistic Regression	**0.2295564**	**0.8365123**	**0.7118644**	**0.8603896**	**0.9397163**	**0.81007**	**0.786127**
Logistic Regression UpSampled	0.5343487	0.7901907	0.7288136	0.8019481	0.9391635	0.82072	0.7653809
Logistic Regression SMOTE	0.5257691	0.8010899	0.6949153	0.8214286	0.9335793	0.79676	0.7581720
Neural Network	**0.1674948**	**0.8147139**	**0.7118644**	**0.8344156**	**0.9379562**	**0.80942**	**0.77314**
Neural Network UpSampled	0.1066153	0.7574932	0.7288136	0.7629870	0.9362550	0.81961	0.7459003
Neural Network SMOTE	0.3349407	0.7738420	0.6779661	0.7922078	0.9277567	0.78343	0.7350870
Support Vector Machines	**0.1770735**	**0.7874659**	**0.7627119**	**0.7922078**	**0.9457364**	**0.84442**	**0.7774599**
Support Vector Machines UpSampled	0.08109166	0.72752044	0.67796610	0.73701299	0.92276423	0.781646	0.7074895
Support Vector Machines SMOTE	0.1183862	0.6594005	0.7118644	0.6493506	0.9216590	0.80329	0.6806075
Random Forests Dummy	**0.1475000**	**0.7411444**	**0.7457627**	**0.7402597**	**0.9382716**	**0.83101**	**0.7430112**

Random Forests Dummy UpSampled	0.2065000	0.7247956	0.7457627	0.7207792	0.9367089	0.83040	0.7332710
Random Forests Dummy SMOTE	0.3885000	0.8174387	0.6610169	0.8474026	0.9288256	0.77237	0.75420975
GBM dummy	**0.1652439**	**0.7792916**	**0.7796610**	**0.7792208**	**0.9486166**	**0.85588**	**0.7794409**
GBM dummy UpSampled	0.002546103	0.754768392	0.677966102	0.769480519	0.9257813	0.78273	0.7237233
GBM dummy SMOTE	0.006469846	0.752043597	0.711864407	0.759740260	0.9322709	0.80729	0.735802334

For random forest the sampling did result in a better F1 score and balanced accuracy, as compared to altered cutoff. For all other models the F1 score was found to be optimal when altered cutoff values were used. The best performing model is Support Vector Machine (SVM) model with a corresponding F1 score and balanced accuracy of, respectively, 0.85520 and 0.76645. The second to best model is a Logistic Regression model with altered cutoff value, which results are similar as the SVM. It has a corresponding F1 score and balanced accuracy of, respectively, 0.82072 and 0.7654. Based on these metrics these two models significantly outperformed the other tested models.

9.3.1 Comparison of models with altered cutoffs based on test set

To further investigate the results of sampling, Table 5 represents the results of the models build on: the original dataset, up sampled data, and SMOTE data, with altered cutoff values selected based on the test set. So, the results may not be interpreted as actual results, since they will likely be over optimistic. However, the alternative cutoff values can be used as an evaluation method between the sampled and the base case to check whether actual predictive performance gains are likely to have been made. These results indicate that based on F1 score up-sampling could potentially outperform the normal dataset (logistic regression and NN), although the scores are very similar. And based on the balanced accuracy, applying sampling did not result in better predictive performance.

10. Results Predictor Importance and Significance

As indicated on the beginning of the previous chapter, this section can be considered part two of the results. In this chapter the hypotheses stated in section 3.1 (p.11) are tested. Also, some of the other significant predictors will be discussed. The logistic regression method was used to evaluate the significance, as this model provided, compared to all parametric models, the best results according to previous chapter. Notice that during the dataset creation some of the hypothesized predictors got removed from the dataset, due to high correlations with the other predictors. To be able to test the hypotheses, the author reentered these variables into the dataset, which is only used for this chapter, and deleted their corresponding correlation counterparts. All hypothesis tests are based on an alpha of 0.05. A list of predictors that are significant at the 0.05 level is presented below, ordered by their relative importance.

Figure 3 provides the reader with a visual representation of the relative importance of all predictors used in the model. In addition to the already found relevance of logistic regression in the previous chapter; this chapter will start with two goodness of fit evaluations for logistic regression. Next, based on the results of the model, the hypotheses will be tested. And this chapter will end with an evaluation and discussion of the other significant predictors related to voluntary turnover.

```
Coefficients:
                              Estimate Std. Error z value Pr(>|z|)
(Intercept)                 -2.462e+01  1.284e+01  -1.917 0.055257 .
OverTime.Yes                -1.993e+00  1.936e-01 -10.295  < 2e-16 ***
```

```
EnvironmentSatisfaction           4.390e-01  8.280e-02   5.302 1.15e-07 ***
NumCompaniesWorked               -1.945e-01  3.769e-02  -5.161 2.46e-07 ***
JobSatisfaction                   4.009e-01  8.153e-02   4.918 8.75e-07 ***
BusinessTravel.Travel_Frequently -1.020e+00  2.109e-01  -4.837 1.32e-06 ***
JobInvolvement                    3.452e-01  7.965e-02   4.334 1.46e-05 ***
YearsSinceLastPromotion          -1.667e-01  4.178e-02  -3.991 6.58e-05 ***
DistanceFromHome                 -3.464e-01  8.854e-02  -3.912 9.14e-05 ***
Age                               7.463e-01  2.122e-01   3.517 0.000436 ***
MaritalStatus.Single             -1.173e+00  3.448e-01  -3.403 0.000666 ***
RelationshipSatisfaction          2.588e-01  8.247e-02   3.138 0.001699 **
MonthlyIncome                     9.037e-01  2.924e-01   3.090 0.001999 **
YearsWithCurrManager              1.391e-01  4.607e-02   3.020 0.002527 **
YearsInCurrentRole                1.199e-01  4.466e-02   2.685 0.007245 **
`JobRole.Sales Representative`   -1.481e+00  5.524e-01  -2.681 0.007338 **
TrainingTimesLastYear             1.909e-01  7.293e-02   2.618 0.008850 **
WorkLifeBalance                   1.885e-01  7.240e-02   2.604 0.009208 **
`JobRole.Sales Executive`        -1.050e+00  4.469e-01  -2.349 0.018812 *
`JobRole.Laboratory Technician`  -1.068e+00  4.799e-01  -2.225 0.026107 *
Gender.Male                      -3.949e-01  1.845e-01  -2.141 0.032271 *

Signif. codes:  '***'= 0.001, '**'= 0.01, '*'= 0.05
```

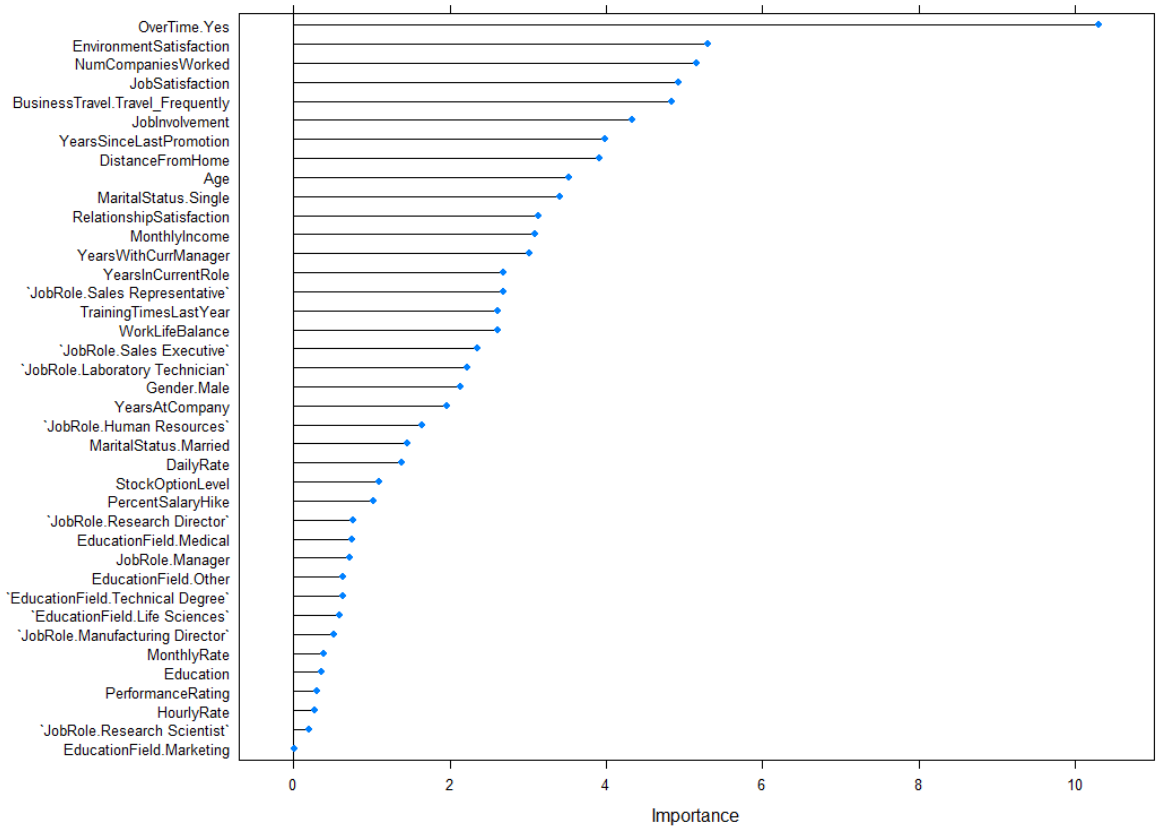


Figure 3 A graphical indication of the relative importance of all predictors.

10.1 Evaluation Goodness of Fit

Discrimination in linear regression models is generally measured using R2. Since this has no direct analog in logistic regression, various methods can be used instead to evaluated the goodness of fit of the model. In this study the methods likelihood ratio test and Hosmer-Lemeshow test are conducted, and will be discussed next.

10.1.1 Likelihood ratio tests

In linear regression analysis, one is concerned with partitioning variance via the sum of squares calculations. Variance in the criterion is essentially divided into variance accounted for by the predictors and residual variance. In logistic regression analysis, deviance is used instead of sum of squares calculations. Deviance is analogous to the sum of squares calculations in linear regression and is a measure of the lack of fit to the data in a logistic regression model like (Hosmer D.W. and Lemeshow, 1980).

If the distribution of the likelihood ratio Λ corresponding to a particular null and alternative hypothesis can be explicitly determined, then it can directly be used to form decision regions (to accept/reject the null hypothesis). In most cases, however, the exact distribution of the likelihood ratio corresponding to specific hypotheses is very difficult to determine. (Wilks, 1938) found that as the sample size approaches ∞, the test statistic $-2\log(\Lambda)$ for a nested model will be asymptotically chi-squared distributed (χ^2) with degrees of freedom equal to the difference in dimensionality. This means that for a great variety of hypotheses, a practitioner can compute the likelihood ratio Λ for the data and compare $-2\log(\Lambda)$ to the χ^2 value corresponding to a desired statistical significance as an approximate statistical test.

For the logistic model the null deviance and residual deviance are provided. With the null deviance defined as:

$$D\ null = -2\ln\frac{likelihood\ of\ null\ model}{likelihood\ of\ saturated\ model}$$

And with the residual deviance defined as:

$$D\ residual = -2\ln\frac{likelihood\ of\ saturated\ model}{likelihood\ of\ proposed\ model}$$

Here the null model estimates only one parameter to explain the data, the saturated model is a model that assumes each data point has its own parameter, and the proposed model assumes that the data points can be explained with p parameters + an intercept term.

Based on these two statistics, their corresponding degrees of freedom, and the findings of (Wilks, 1938) the following hypothesis can be tested:

$$H_1 = the\ proposed\ logitic\ regression\ model\ provides\ an\ adequate\ fit\ for\ the\ data.$$

With a p-value = 0 the null hypothesis can be rejected. So, no evidence was found that the model fits the data inadequately.

10.1.2 Hosmer-Lemeshow test

The Hosmer–Lemeshow test is another statistical test for goodness of fit for logistic regression models. The test assesses whether or not the observed event rates match expected event rates in subgroups of the model population. Logistic regression models provide an estimate of the probability of an outcome. It is desirable that the estimated probability of success is close to the true probability.

In a 1980 paper Hosmer and Lemeshow showed by simulation that, given that the predictors +1 is smaller than the number of groups (g), their test statistic approximately followed a chi-squared distribution χ^2 on g−2 degrees of freedom, when the model is correctly specified (Hosmer D.W. and Lemeshow, 1980). This means that given our fitted model, the p-value can be calculated as the right hand tail probability of the corresponding χ^2 distribution using the calculated test statistic. If the p-value is small, this is indicative of poor fit. Applying the Hosmer–Lemeshow test to our model resulted in: χ^2 equal to -4.32, degrees of freedom equal to 8, and a p-value equal to 1, indicating a good fit of the current model.

10.2 Hypothesis test results

Eight hypotheses are formed at the beginning of this study (p.11). Each of these will be discussed in turn. But first, the consequences of the preprocessing step box-cox will be explained.

A box-cox transformation was used as a preprocessing step to better handle the skewness in the data. Due to this transformation, the results give a better estimate of actual predictor importance and significance. Consequently however, the relationship between predictor and outcome becomes less interpretable. Out of 39 predictors 13 were considered to be transformed. From the significant predictors, the following predictors got transformed using this method: Age, Monthly income, Distance from home, Job involvement, and Work life balance. To present the relationship of these predictors with the outcome, plots were created with the y-ass representing the logistic model outcome, and the x-ass representing the original units (pre-box-cox transformation). In these plots, all other variables are held at a constant equal to their corresponding means.

Hypothesis 1: Age is negatively related to voluntary turnover.

Evidence was found to reject this null hypothesis. Although age is found to be a significant predictor for employee attrition ($p < 0.001$), in this dataset as age goes up so does the likelihood of attrition. However, it should be kept in mind that this evidence can be found due to the left truncation effect described at the beginning of the previous chapter. Age is one of the significant predictors that got transformed using the box-cox method, this hinders the interpretation of the estimate. Figure 4 shows the plot created as an indication of the effect of age on the outcome. As age goes up, so does the likelihood of attrition.

Hypothesis 2: Education is a significant predictor of voluntary turnover.

For this dataset employee education is not a significant indicator for employee voluntary turnover, and the null hypothesis cannot be rejected. Based on Figure 3, employee education level is not only insignificant it is also at the bottom of the list of relative importance to the outcome. This indicates that employee's voluntary leave behavior is similar regardless of their education level.

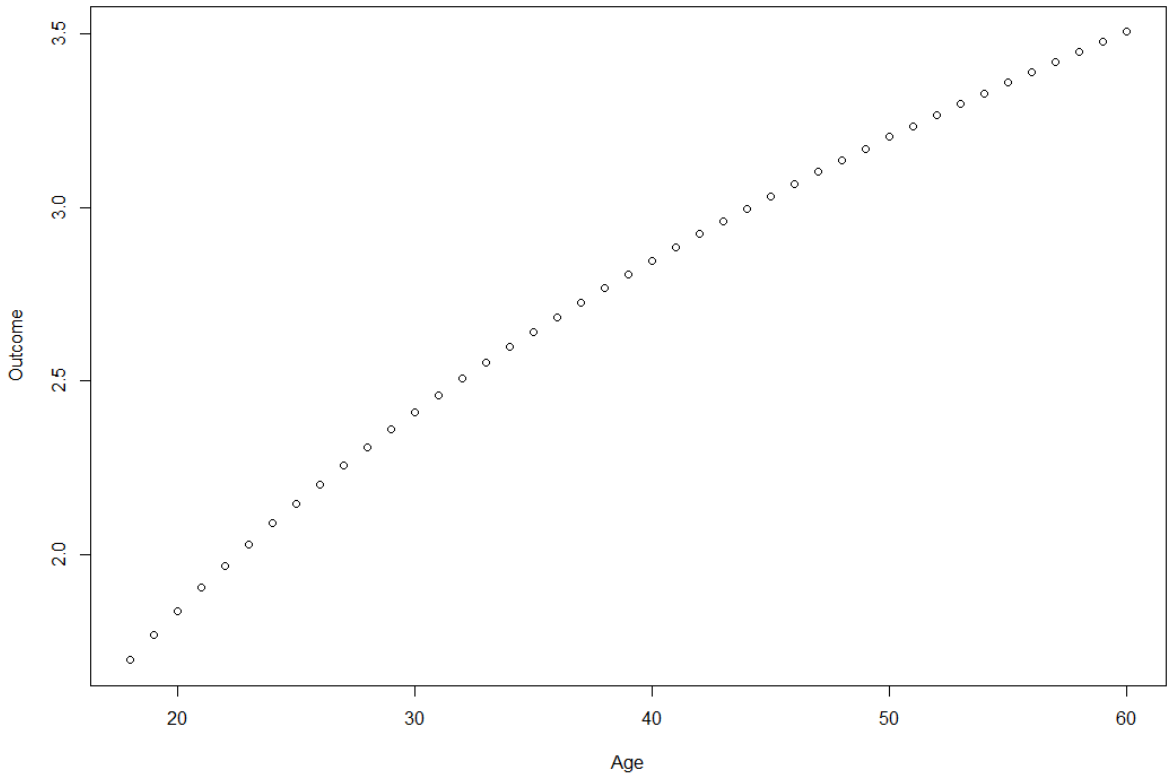


Figure 4 Effect Age on model Outcome with all other predictors held equal to their respective means

Hypothesis 3: Sex is a significant predictor of voluntary turnover.

Gender is found to be a significant predictor for employee voluntary turnover ($p < 0.05$), and thus, the null hypothesis can be rejected. In this dataset if the employee is a male the odds ratio of voluntary leave is 0.67(exp(-3.949e-01)) indicating that the voluntary turnover odds decrease with 33% if the employee is male compared to female. This hypothesis is thus found to be true. However, out of all significant predictors, gender is at the bottom of the list of importance to the outcome.

Hypothesis 4: Tenure is negatively related to voluntary turnover.

Tenure in this dataset is indicated as years at company. Although years at company was found to be negatively related to voluntary turnover; with a $p = 0.051$ it is not significant and therefore the null hypothesis cannot be rejected.

Hypothesis 5: Pay is negatively related to voluntary turnover.

In this dataset pay is denoted as monthly income. There is no evidence to reject the null hypothesis. Although monthly income is found to be a significant predictor for voluntary turnover with a $p < 0.01$, the relation that was positive rather than negative, implying that as monthly income increases, so does the likelihood of employee voluntary turnover. Since monthly income is transformed by the box-cox procedure the results are, similar as age, hard to interpret. Figure 5 shows the relationship between pay and the outcome.

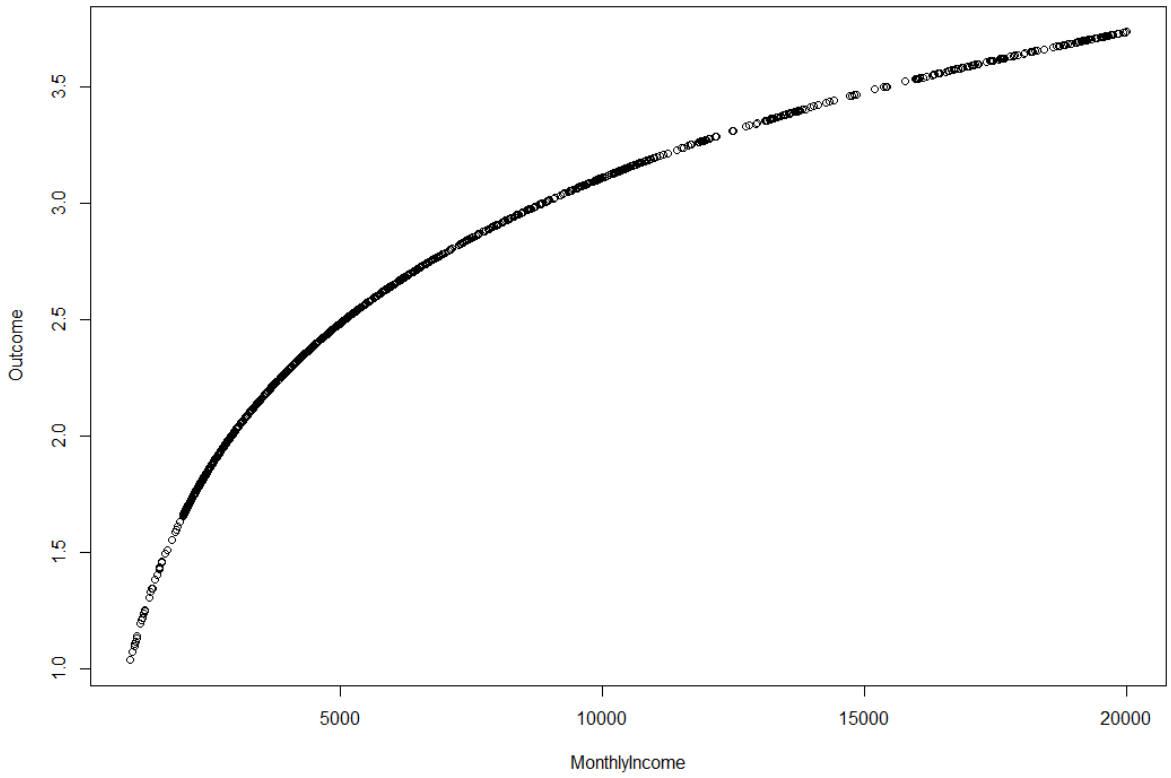


Figure 5 Effect Monthly Income on model Outcome with all other predictors held equal to their respective means

Hypothesis 6: Job involvement is negatively related to voluntary turnover.

Job involvement was found to be a significant indicator for voluntary turnover with a $p < 0.001$. However, contrary to the hypothesis as job involvement increases so does voluntary turnover. Like age and monthly income, job involvement is transformed by the box-cox procedure. A similar graph was plotted and is shown in Figure 6.

Hypothesis 7: Job satisfaction is negatively related to voluntary turnover.

Job satisfaction was found to be a significant predictor to voluntary turnover with a $p < 0.001$. However, contrary to the hypothesis it was found that as job satisfaction increases so does the likelihood of voluntary turnover. With an estimate equal to 0.4009 (= 4.009e-01) as job satisfaction increases with one the odds of voluntary turnover increase with 49% (1- exp(0.4009)).

Hypothesis 8: Employee performance is negatively related to voluntary turnover.

For this dataset employee performance is not a significant predictor for voluntary turnover, thus we cannot reject the null hypothesis. However, it should be stated that although employee performance is grade between 1 and 4, the former indicating bad performance the latter good; all employees in the current dataset had a performance of either 3 or 4. Employee performance could still be a significant predictor for voluntary turnover within a dataset containing the full spectrum of employee performance grades. However, when it comes to voluntary turnover, practitioners are probably mostly interested in the good performing employees.

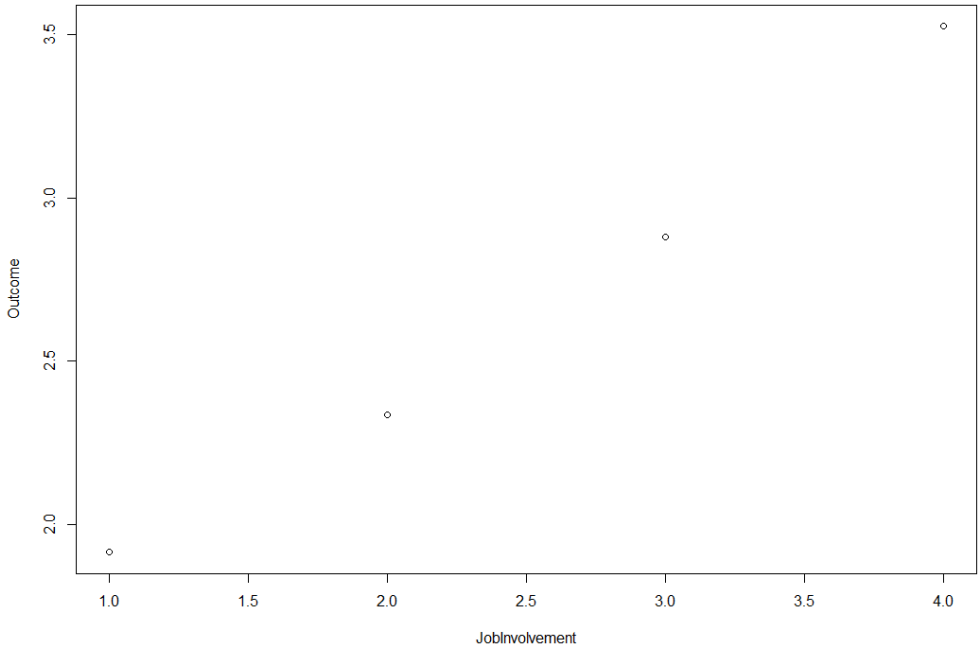


Figure 6 Effect of Job Involvement on model Outcome with all other predictors held equal to their respective means

10.3 Other significant predictors

In the current study overtime is by far the most important predictor for voluntary turnover and is also significant with a p< 0.001. As indicated in Table 1, overtime is equal to either yes or no, and indicates whether an employee is allowed to write and get payed for the extra hours he/she makes. The odds ratio for overtime is equal to 0.14, indicating if overtime is equal to 'yes' the turnover odds decreases with 86%. Usually in organizations lower career levels equal to employee overtime: yes, and higher career levels equal to overtime: no. Assuming this is true, in the case study organization higher career level employees are leaving the organization. This conclusion is in alignment with the relationships with voluntary turnover found in the hypotheses about age, and pay. In fact, during the initial creation of the datasets (Chapter 4 p.13) some predictors got deleted due to being highly correlated with others; and job level was found to have a 0.95 correlation with monthly income. The author suspects that the voluntary leave behavior for certain job levels will significantly differ from others.

Another important and significant predictor for voluntary turnover is environment satisfaction, with a p<0.001. Contrary to common sense, the odds ratio for environment satisfaction is equal to 1.55, indicating if environment satisfaction is increased by 1 the voluntary turnover odds increase by 55%.

Next in the list of important and significant predictors is the number of companies the employee has worked for in the past, with a p<0.001. Employees who worked for a larger number of companies are less likely to voluntarily leave the organization. The odds ratio is equal to 0.82, indicating that if number of companies is increased by one the odds for voluntary turnover decrease by 18%.

Employees that worked for many companies in the past can be considered more experienced. They will have better understanding of the differences between organizations, and will eventually settle in an organization and role which is most aligned with their own values. This settlement in the organization and in the role the employee wants to fulfil is likely to be associated with the 'years since last promotion' predictor, which is another important and significant predictor with a $p<0.001$. Here more years since last promotion reduces the likelihood of attrition. An increase of one year in years since last promotion decreases the odds ratio with 15%. It should be noted that this effect could be influenced by the left truncation effect, i.e. a high number of years since last promotion is likely to be over represented in the dataset compared to lower years since last promotion.

Another important and significant predictor for voluntary turnover is business travel, with a $p<0.001$. More specific, it was found that employees who travel frequently are less likely to voluntary leave, with a decrease in odds of 64% compared with employees that do not travel or travel rarely.

Next, distance from home was found to be a significant predictor to voluntary turnover, with a $p<0.001$. As the distance from home increases the likelihood of voluntary turnover decreases. Distance from home yet another predictor that is transformed by the box-cox procedure. A plot was created in a similar fashion as the other cases, and is shown in Figure 7.

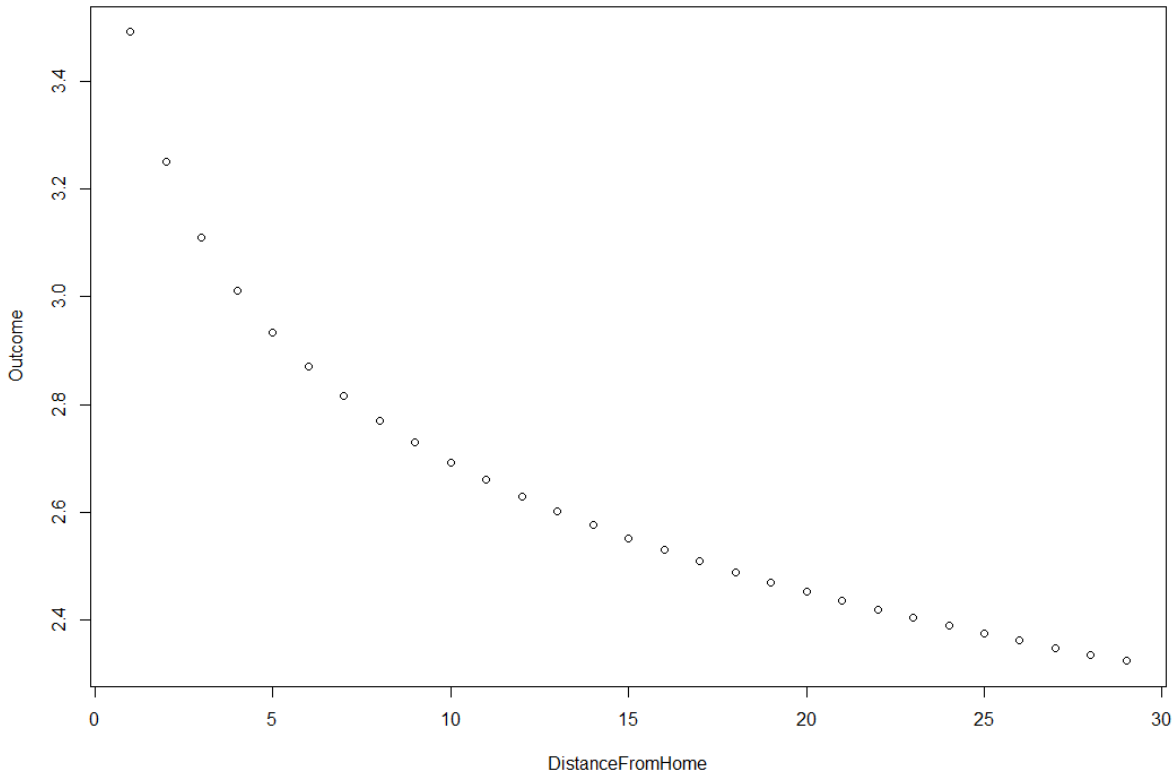


Figure 7 Effect of Distance from Home on model Outcome with all other predictors held equal to their respective means

Although marital status was found to be incomparable with previous studies, in this study single employees were found to be significantly different ($p<0.001$) from married or divorced employees. More specifically, the odds ratio is equal to 0.31; indicating an odd decrease of 69% when an employee is single.

The most important and significant predictors were discussed. Some other less important but still significant predictors are: *relationship satisfaction, years with current manager, years in current role, job role, training times last year,* and *work life balance*. Here *relationship satisfaction, years with current manager, years in current role,* and *training times* are positively related to voluntary turnover. And the job roles: *sales representative, sales executive,* and *laboratory technician*, were found to be significantly different from the other job roles and are all negatively related to voluntary turnover.

11. Related work

In their study, (X. Zhu et al., 2017) made a summary of previous research on employee turnover forecasting. It contains references dating back from 1982 until 2015. From these, the studies classified as response variable equal to probability are considered related to this study. Next, the author searched for related research after 2015 and found two additional paper. The author was unable to get access to the paper (Nagadevara, Srinivasan, & Valk, 2008), and is thus excluded from the table. Table 6 provides an overview of comparable studies and some characteristics of these studies.

Some characteristics that differentiates the current study from previous studies are: the number of methods used, number of samples, number of predictors, class-imbalance resolution, and the combination of model accuracy and predictor significance. In this study, multiple machine learning algorithms get compared. The sample size is greater than any of the other studies, indicating this study's outcome to be more robust. The dataset includes many predictors, including predictors such as monthly salary and monthly rate which are often excluded. This enabled the author to analyze the significance of multiple predictors related to voluntary turnover, improve the performance of the predictive models, and for practitioners the results give a better indication of actual model performance. Also, previous studies did often not address the class imbalance which exists in all datasets. In addition, the current study, contrary to previous studies, uses a publicly available dataset which makes the results 100% reproducible.

Table 6 Previous studies predicting voluntary turnover

Authors	Data Acquisition	Methods	# Samples (N)	# Predictors (P)	Test Set	Validation Method	Class-Imbalance Resolution	Model Accuracy Calculation	Predictor Significance Calculation
(Ng, Cram, & Jenkins, 1991)	Survey	Hazard proportional model	1002	8	No	No	No	No	Yes
(Balfour & Neff, 1993)	Employee records	Logistic regression	171	7	No	No	No	No	Yes
(Feeley & Barnett, 1997)	Survey	Social network, logistic regression	166	3	No	No	No	No	Yes
(Sexton, McMurtrey, Michalopoulos, & Smith, 2005)	Employee records	Multiple versions of NN	447	18	No	10-fold cross-validation	No	Yes	No
(Hong, Wei, & Chen, 2007)	Survey	Logistic regression, and probit regression	132	6	32%	No	No	No	Yes
(Saradhi & Palshikar, 2011)	Employee records	SVM, Random Forest, Naïve Bayes	1363	32	20%	No	Weighting	Yes	No
(Tews, Stafford, & Michel, 2014)	Survey	Logistic regression	290	10	No	No	No	No	Yes
(X. Zhu, 2016) Chapter 4	Employee records	Logistic regression, classification tree	731	8	40%	No	No	No	Yes
(Ribes, Touahri, & Perthame, 2017)	Employee records	SVM, RF, LDA, bagged trees	1000	11	20%	10-fold cross-validation	Down-sampling, Up-sampling, Weighting,	Yes	No

							SMOTE, and ROSE		
Current Study	Employee records	Logistic regression, LDA, penalized logistic regression, NN, FDA, SVM, KNN, Naïve Bayes, Simple decision trees and rule based models, bagged trees, random forest, Stochastic Gradient Boosting, and C5.0	1470	31	25%	5 repeats of 10-fold cross-validation	Up-sampling, SMOTE, and altered cutoffs	Yes	Yes

12. Discussion and conclusion

This chapter will provide the reader with a discussion of the results, the conclusion of this papers, and areas for future research.

12.1 Discussion

At the start of this paper four key questions got formed, in this section each of these will be discussed in turn. The first key question formed is: 'What machine learning algorithm is most appropriate for predicting employee voluntary leave?'. To answer this question 16 different machine learning algorithms got tested, and the results were analyzed. From this analysis, it was found that voluntary turnover can be linearly separated; indicating that the simpler more interpretable linear classification models might be preferable over the less interpretable nonlinear classification models. However, the model that was found to have the best performance based on the area under the ROC curve is the non-linear classification model Support Vector Machine.

The second key question is: 'Does sampling of the dataset increase the predictive performance of the models?'. A method for countering the effects of class imbalance is to apply alternated cutoff values (see section 8.1 for more detail, p.26). The normal cutoff value for predicting a case is 0.5. Based on the ROC curve an altered cutoff value was found, which provides the best trade-off between sensitivity and specificity. Notice that the model, which is based on the original data, remains unchanged. In contrast, sampling is a technique for which the models are trained on more balanced datasets with the aim to find even better trade-offs between sensitivity and specificity. Training on sampled data will result in structurally different models. Two additional datasets based on over-sampling, and synthetic minority over-sampling technique (SMOTE) (section 8.2, p.27), where created two answer the second key question. Our findings suggest that sampling does not significantly improve the performance of the models for voluntary turnover. However, only two sampling approaches got tested here, and there might still be others that will result in better performing models.

The third key question stated is: 'What results can be expected from predictive modeling of employee voluntary turnover?'. In this study based on the F1 score and balanced accuracy the best performing model is the support vector machine when applied with altered cutoff, with a F1 score and balanced accuracy of respectively, 0.86 and 0.77. This model has a corresponding sensitivity of 0.78, specificity of 0.75, and a positive predictive value (PPV) of 0.95. These results can be interpreted as: for sensitivity, assuming the employee is voluntary leaving the organization, this test has an accuracy of 78%; for specificity, assuming the employee is not leaving the organization, this test has an accuracy of 75%; and for PPV, for all employees identified by the model as voluntary leaving the organization, this model has an accuracy of 95%. A close second is the more interpretable logistic regression applied with altered cutoff, with a F1 score and balanced accuracy of respectively, 0.82 and 0.77; and with corresponding sensitivity, specificity, and PPV of, respectively, 0.73, 0.80, and 0.94. It should be noted that the altered cutoffs were found on the ROC curves of the training set. This may not produce optimal results on the test set, since the models are likely

to overfit the training data. Even better results can be expected from a model for which the altered cutoffs are based on an additional evaluation set.

And the final question: 'What are the significant predictors for determining employee voluntary leave?'. The significance of predictors is evaluated by fitting a logistic regression model to the data. The model was found to fit the data well (section 10.1 p.36). Out of 39 predictors 20 were found to be significant for predicting employee voluntary leave. The top 10 most important and significant predictors are: *Overtime, EnvironmentSatisfaction, NumCompaniesWorked, JobSatisfaction, BusinessTravel, JobInvolvement, YearsSinceLastPromotion, DistanceFromHome, Age*, and relationship status *Single*. For the full list, and interpretation of the relationship between predictor and outcome, the author refers the reader to chapter 10 (p. 35). In section 2 of this chapter (p.38) the hypothesis formed in chapter 3 (p.11) are tested. Out of the eight hypotheses formed, *education, tenure,* and *employee performance* where found to be insignificantly related to voluntary turnover. However, *tenure* is only slightly insignificant with a p=0.051; and this dataset did only contain employees with a performance rating of 3-4 on a 4-point scale. *Sex* was found to be a significant predictor, where males are less likely to leave compared to females. The remaining hypotheses, *age, pay, job involvement,* and *job satisfaction*, were all found to be significantly related to employee voluntary turnover. However, not always in the relationship described based on literature. For these data, *age*, and *pay* were found to be positively related to voluntary leave, indicating older successful employees are leaving the organization. And, *job involvement,* and *job satisfaction*, were found to be negatively related to voluntary leave, indicating employee's that are more involved and more satisfied with their job are leaving this organization.

12.2 Conclusion

The importance of predicting employee voluntary turnover and the application of machine learning in building predictive models are presented in this paper. The dataset used in this study is a publicly available dataset provided by IBM (McKinley, 2015). The dataset was found to be valid in being representative to an organization's actual data. Multiple datasets were created to effectively compare the results of 16 different machine learning methods. The support vector machine was found to produce the best model, closely followed by logistic regression. Based on these data, it was found that voluntary turnover can be linearly separated. Also, sampling was investigated as a remedy for the severe class imbalance in the data. Compared to altered cutoff values, it did not result in better performing models.

In addition, based on previous literature some hypotheses got formed. These hypotheses, in addition with the other predictors used in this study, were tested on their respective importance and significance. Out of 39, 20 were found to be significant for predicting voluntary turnover. However, some of the relationships found go against common sense and suggests areas for further research, which will be discussed next.

12.3 Future work

The results of analysis of the importance and significance of the predictors can be counterintuitive. For example, in this dataset the older more experienced employees are leaving the organization, which is in contrast to other studies. This does not mean

that this, or previous studies, are invalid. This merely means that for voluntary turnover there is variability in the outcome. This results in some predictors being significant for voluntary turnover in one organization, but can be found insignificant in others; or have a contradicting relationship with the outcome.

This finding is in accordance with (Rubenstein et al., 2017) which, besides investigating the correlation found between predictors and voluntary leave, investigated the effect of moderating factors on the relationship between the predictors and voluntary leave. In their study, they highlight the context-sensitive nature of individual-level voluntary leave. This implies that it is important for practitioners to analyze their own data to find, and be able to react to, the predictors that are important and significant in their organization. And for researchers to come up with a framework, so organizations can be categorized, and the results can be compared to their respective groups; as well as investigate the meta-analytic moderators (Rubenstein et al., 2017).

In addition, (Rubenstein et al., 2017) found that by far the strongest correlation with voluntary turnover is *intent to leave*. This would be an interesting variable for organizations to measure, and will most likely increase the performance of the models. Previous individual-level voluntary turnover predictive modeling research did not include this predictor yet, so this is another important contribution that can be made to literature.

Some other related avenues, future research should focus on including external available data. For example, job market opportunities, macro-economic factors, unemployment rate, activity of employee on work related social media site (e.g. LinkedIn in the Netherlands), etc., are all potentially interesting measures that can possibly influence the predictive performance of the models. This was impossible to do on the current dataset, since the dataset is anonymized and not enough is known about its background.

It would also be interesting to see the effect of the changes of variables (e.g., job satisfaction, or employee performance) in time on employee voluntary turnover, to see if this can bring additional predictive information.

And last, the predictive model results presented in this study can still be improved by, for example, feature engineering. It would be interesting to see based on these data the optimal performance of the logistic regression model vs. the support vector machine.

Acknowledgements

The author greatly acknowledges the constructive feedback from prof.dr.ir. H.A.M. Daniels, and dr. E.A.M. Caron; as well as the contribution of Accenture colleagues. More specifically I would like to thank C. Kerkdijk, R. Hendrix, E. Pupazan, and A. Loginovskaja for providing guidance and direction, without them the author would not have been able to successfully conduct this research.

13. References:

Akkermans, H. A. (Henricus A. (2014). *Supply chain dynamics: mastering disruptive change in innovation driven industries*. Educatieve Uitgeversgroep.

Allen, D. G., Bryant, P. C., & Vardaman, J. M. (2010). Retaining Talent: Replacing Misconceptions With Evidence-Based Strategies. *Academy of Management Perspectives*, *24*(2), 48–64. https://doi.org/10.5465/AMP.2010.51827775

Angrave, D., Charlwood, A., Kirkpatrick, I., & Stuart, M. (2016). HR and analytics: why HR is set to fail the big data challenge; HR and analytics: why HR is set to fail the big data challenge. *Human Resource Management Journal*, *26*(1), 1–11. https://doi.org/10.1111/1748-8583.12090

Balfour, D. L., & Neff, D. M. (1993). Predicting and Managing Turnover in Human Service Agencies: A Case Study of an Organization in Crisis. *Public Personnel Management*, *22*(3), 473–486. https://doi.org/10.1177/009102609302200310

Boles, J. S., Dudley, G. W., Onyemah, V., Rouziès, D., & Weeks, W. A. (2012). Sales Force Turnover and Retention: A Research Agenda. *Journal of Personal Selling and Sales Management*, *32*(1), 131–140. https://doi.org/10.2753/PSS0885-3134320111

Box, G. E. P., & Cox, D. R. (1964). An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological*, *26*(2), 211–252. https://doi.org/10.2307/2287791

Breiman, L. (2002). Manual on setting up, using, and understanding random forests v3. 1. *Technical Report, Http://oz.berkeley.edu/users/breiman, Statistics Department University of California Berkeley,* …, 29. https://doi.org/10.2776/85168

Brodersen, K. H., Ong, C. S., Stephan, K. E., & Buhmann, J. M. (2010). The balanced accuracy and its posterior distribution. In *Proceedings - International Conference on Pattern Recognition* (pp. 3121–3124). https://doi.org/10.1109/ICPR.2010.764

Carrión, A., Solano, H., Gamiz, M. L., & Debón, A. (2010). Evaluation of the Reliability of a Water Supply Network from Right-Censored and Left-Truncated Break Data. *Water Resources Management*, *24*(12), 2917–2935. https://doi.org/10.1007/s11269-010-9587-y

Chang, H. Y. (2009). Employee turnover: A novel prediction solution with effective feature selection. *WSEAS Transactions on Information Science and Applications*, *6*(3), 417–426. Retrieved from https://pdfs.semanticscholar.org/fa0f/a51a434091c5feb6e26ae9418872c67872a4.pdf

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. https://doi.org/10.1613/jair.953

Fawcett, T. (2006). An introduction to ROC analysis. Pattern Recognition Letters. *Pattern Recognition Letters*, *27*, 861–874. Retrieved from http://www.sciencedirect.com/science/article/pii/S016786550500303X

Feeley, T. H., & Barnett, G. a. (1997). Predicting Employee Turnover From

Communication Networks. *Human Communication Research*, *23*(3), 370–387. https://doi.org/10.1111/j.1468-2958.1997.tb00401.x

Frank, E., & Witten, I. H. (1998). Generating accurate rule sets without global optimization. *Proceedings of the Fifteenth International Conference on Machine Learning*, 144–151. https://doi.org/1-55860-556-8

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive Logistic Regression: A Statistical View of Boosting: Discussion. *The Annals of Statistics*, *28*(2), 374–377. Retrieved from http://projecteuclid.org/euclid.aos/1016218223

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, *33*(1), 1–22. https://doi.org/10.1016/j.drudis.2011.09.009

Glebbeek, A. C., Bax, E. H., The, S., Journal, M., Apr, N., Journal, M., … Bax, E. H. (2014). Is High Employee Turnover Really Harmful ? An Empirical Test Using Company Records IS HIGH EMPLOYEE TURNOVER REALLY HARMFUL ? AN EMPIRICAL TEST USING COMPANY RECORDS. *Academy of Management Journal*, *47*(2), 277–286. Retrieved from http://amj.aom.org/content/47/2/277.short

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning* (1st ed.). MIT Press. Retrieved from http://www.deeplearningbook.org/

Griffeth, R. W., Hom, P. W., & Gaertner, S. (2000). A Meta-Analysis of Antecedents and Correlates of Employee Turnover: Update, Moderator Tests, and Research Implications for the Next Millennium. *Journal of Management*, *26*(3), 463–488. https://doi.org/10.1177/014920630002600305

Hancock, J., Allen, D., & Bosco, F. (2013). Meta-analytic review of employee turnover as a predictor of firm performance. *Journal of Management*, *39*(3), 573–603. Retrieved from http://journals.sagepub.com/doi/abs/10.1177/0149206311424943

Hastie, T., Tibshirani, R., & Buja, A. (1994). Flexible Discriminant Analysis by Ooptimal Scoring. *Journal of the Amercian Statistical Association*, *89*(428), 1255–1270. https://doi.org/10.1080/01621459.1994.10476866

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning (2nd edition)*. https://doi.org/10.1007/978-0-387-84858-7

Hong, W., Wei, S. Y., & Chen, Y. F. (2007). A Comparative Test of Two Employee Turnover Prediction Models. *International Journal of Management*. Retrieved from http://search.proquest.com/openview/f56f89c41e7a8826ada9d02ac2f27d83/1?pq-origsite=gscholar&cbl=5703

Hosmer D.W. and Lemeshow, S. (1980). Goodness of fit tests for the multiple logistic regression model: Communications in Statistics. *Communications in Statistics-Theory and Methods.*, *9*(10), 1043–1069. Retrieved from http://www.tandfonline.com/doi/abs/10.1080/03610928008827941

Kacmar, K. M., Andrews, M. C., & System, R. (2006). SURE EVERYONE CAN BE REPLACED . . . BUT AT WHAT COST ? TURNOVER AS A PREDICTOR OF UNIT-LEVEL PERFORMANCE r ww ia w . n l V ua er nce si o . co n r ww ia w . n l V ua er nce si o . co n. *Academy of Management Journal*, *49*(1), 133–144.

https://doi.org/10.5465/amj.2006.20785670

Kaufman, B. E. (2014). The historical development of American HRM broadly viewed. *Human Resource Management Review*, *24*(3), 196–218. https://doi.org/10.1016/j.hrmr.2014.03.003

King, K. (2016). Data Analytics in Human Resources: A Case Study and Critical Review. *Human Resource Development Review*, *15*(4), 487– 495. Retrieved from http://journals.sagepub.com/doi/abs/10.1177/1534484316675818

Kuhn, M., & Johnson, K. (2016). *Applied Predictive Modeling* (5th ed.). New York: Springer. https://doi.org/10.1007/978-1-4614-6849-3

Ling, C. X., & Li, C. (1998). Data mining for direct marketing: Problems and solutions. In *KDD* (Vol. 98, pp. 73–79). Retrieved from http://www.csd.uwo.ca/~cling/papers/kdd98.pdf

Marler, J. H., & Boudreau, J. W. (2017). An evidence-based review of HR Analytics. *The International Journal of Human Resource Management*, 3–26. https://doi.org/10.1080/09585192.2016.1244699

Mcclure, B. (2003). The hidden value of intangibles. *Recuperado El*. Retrieved from https://scholar.google.nl/scholar?q=The+hidden+value+of+intangibles+McClure&btnG=&hl=nl&as_sdt=0%2C5

McKinley, S. I. (2015). SAMPLE DATA: HR Employee Attrition and Performance – IBM Analytics Communities. Retrieved July 24, 2017, from https://www.ibm.com/communities/analytics/watson-analytics-blog/hr-employee-attrition/

Nagadevara, V., Srinivasan, V., & Valk, R. (2008). Establishing a Link Between Employee Turnover and Withdrawal Behaviours : *Research and Practice in Human Resource Management*, *16*(2), 81–100. Retrieved from http://go.galegroup.com/ps/i.do?id=GALE%7CA200117637&sid=googleScholar&v=2.1&it=r&linkaccess=fulltext&issn=02185180&p=AONE&sw=w

Ng, S. H., Cram, F., & Jenkins, L. (1991). A Proportional Hazards Regression Analysis of Employee Turnover Among Nurses in New Zealand. *Human Relations*, *44*(12), 1313–1330. https://doi.org/10.1177/001872679104401205

Ongori, H. (2010). A review of the literature on employee turnover. *African Journal of Business Management*, *1*(4), 49–54. https://doi.org/10.1108/14634449710195471

Panetta, K. (2016). Gartner's Top 10 Strategic Technology Trends for 2017 - Smarter With Gartner. Retrieved April 10, 2017, from http://www.gartner.com/smarterwithgartner/gartners-top-10-technology-trends-2017/

Ribes, E., Touahri, K., & Perthame, B. (2017). Employee turnover prediction and retention policies design: a case study. Retrieved from http://arxiv.org/abs/1707.01377

Rubenstein, A. L., Eberly, M. B., Lee, T. W., & Mitchell, T. R. (2017). Surveying the forest: A meta-analysis, moderator investigation, and future-oriented discussion of the antecedents of voluntary employee turnover. *Personnel Psychology*.

https://doi.org/10.1111/peps.12226

Sagie, A., Birati, A., & Tziner, A. (2002). Assessing the Costs of Behavioral and Psychological Withdrawal: A New Model and an Empirical Illustration. *Applied Psychology*, *51*(1), 67–89. https://doi.org/10.1111/1464-0597.00079

Saradhi, V. V., & Palshikar, G. K. (2011). Employee churn prediction. *Expert Systems with Applications*, *38*(3), 1999–2006. Retrieved from http://www.sciencedirect.com/science/article/pii/S0957417410007621

Serneels, S., De Nolf, E., & Van Espen, P. J. (2006). Spatial sign preprocessing: A simple way to impart moderate robustness to multivariate estimators. *Journal of Chemical Information and Modeling*, *46*(3), 1402–1409. https://doi.org/10.1021/ci050498u

Sexton, R. S., McMurtrey, S., Michalopoulos, J. O., & Smith, A. M. (2005). Employee turnover: A neural network solution. *Computers and Operations Research*, *32*(10), 2635–2651. https://doi.org/10.1016/j.cor.2004.06.022

Shaw, J. D. (2011). Turnover rates and organizational performance. *Organizational Psychology Review*, *1*(3), 187–213. https://doi.org/10.1177/2041386610382152

Shmueli, G., Patel, N., & Bruce, P. (2010). Data mining for business intelligence. Hoboken. Retrieved from https://scholar.google.com/scholar?hl=nl&q=hmueli%2C+G.%2C+Patel%2C+N.+R.%2C+%26+Bruce%2C+P.+C.+%282010%29.+Data+mining+for+business+intelligence.+Hoboken.&btnG=&lr=

Sikaroudi, A. M. E., Ghousi, R., & EsmaieeliSikaroudi, A. (2015). A data mining approach to employee turnover prediction (case study : Arak automotive parts manufacturing) 1- Introduction. *Journal of Industrial and*, *8*(4), 106–121. Retrieved from http://www.jise.ir/article_10857_1428.html

Suykens, J. A. K., & Vandewalle, J. (1999). Least Squares Support Vector Machine Classifiers. *Neural Processing Letters 9*, 293–300. https://doi.org/10.1023/A:1018628609742

Tews, M. J., Stafford, K., & Michel, J. W. (2014). Life happens and people matter: Critical events, constituent attachment, and turnover among part-time hospitality employees. *International Journal of Hospitality Management*, *38*, 99–105. https://doi.org/10.1016/j.ijhm.2014.01.005

Tipping, M. (2001). {S}parse {B}ayesian {L}earning and the {R}elevance {V}ector {M}achine. *J. Mach. Learn. Res.*, *1*, 211–244. https://doi.org/10.1162/15324430152748236

Watson Analytics Support. (2015). HR Employee Attrition Sample Data Set Origin - Discussions. Retrieved June 12, 2017, from https://community.watsonanalytics.com/discussions/questions/18014/hr-employee-attrition-sample-data-set-origin.html

Wilks, S. S. (1938). The Large-Sample Distribution of the Likelihood Ratio for Testing Composite Hypotheses. *The Annals of Mathematical Statistics*, *9*(1), 60–62. https://doi.org/10.1214/aoms/1177732360

Witten, I., Frank, E., Hall, M., & Pal, C. (2016). Data Mining: Practical machine learning

tools and techniques. Retrieved from https://books.google.com/books?hl=nl&lr=&id=1SylCgAAQBAJ&oi=fnd&pg=PP1&dq=Witten,+I.+H.,+Frank,+E.,+Hall,+M.+A.,+%26+Pal,+C.+J.+(2016).+Data+Mining:+Practical+machine+learning+tools+and+techniques.+Morgan+Kaufmann.&ots=8HIJycgEx7&sig=2tUWfBEV2a-F3ycLsC4WP

Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '99* (pp. 42–49). https://doi.org/10.1145/312624.312647

Zhu, J., & Hastie, T. (2005). Kernel Logistic Regression and the Import Vector Machine. *Journal of Computational and Graphical Statistics*, *14*(Ivm), 185–205. https://doi.org/10.1198/106186005X25619

Zhu, X. (2016). Forecasting Employee Turnover in Large Organizations. Retrieved from http://trace.tennessee.edu/utk_graddiss/3985/

Zhu, X., Seaver, W., Sawhney, R., Ji, S., Holt, B., Sanil, G. B., & Upreti, G. (2017). Employee turnover forecasting for human resource management based on time series analysis. *Journal of Applied Statistics*, *44*(8), 1421–1440. https://doi.org/10.1080/02664763.2016.1214242

Appendix A: Employee Attrition R code

Table of Contents Appendix A

Session

during this study the author made use of two machines. The session information for both are presented below.
#machine 1:
sessionInfo()
Outcome:

```
R version 3.1.3 (2015-03-09)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows Server 2012 x64 (build 9200)
```

#machine 2:
sessionInfo()
Outcome:

```
R version 3.3.3 (2017-03-06)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows >= 8 x64 (build 9200)
```

Library Packages used during this study

```
# the packages used to generate the results in this study are:
library("caret")
library("corrplot")
library("doSNOW")
library("e1071")
library("earth")
library("glmnet")
library("kernlab")
library("klaR")
library("mda")
library("pls")
library("pROC")
library("rpart.plot")
library("rrcov")
library("sparseLDA")
library("lattice")
library("nnet")
library("C50")
library("gbm")
library("ipred")
library("partykit")
library("randomForest")
library("RWeka")
library("stringr")
library("plyr")
library("DMwR")
library("reshape")
library("ResourceSelection")
library("rms")
library("pscl")
library("arm")
```

Dataset validity

```
#dataset validation
HRdata <- Dataset_HR_Employee_Attrition

#Check 1. TotalWorkingYears>=YearsAtCompany
extractyears <- function(x) {
  if(HRdata[x, "TotalWorkingYears"] < HRdata[x, "YearsAtCompany"]){
    return(1)
  } else{
    return(0)
  }
}
workingyearasvalidity <- NULL
for (i in 1:nrow(HRdata)){
  workingyearasvalidity <- c(workingyearasvalidity, extractyears(i))
}
unique(workingyearasvalidity)
```

Outcome:

```
[1] 0
```

```
#Conclusion, the validity of TotalWorkingYears>=YearsAtCompany holds.

#Check 2. YearsAtCompany>=YearsInCurrentRole
extractyears2 <- function(x) {
  if(HRdata[x, "YearsAtCompany"] < HRdata[x, "YearsInCurrentRole"]){
    return(1)
  } else{
    return(0)
  }
}
YearsRoleValidity <- NULL
for (i in 1:nrow(HRdata)){
  YearsRoleValidity <- c(YearsRoleValidity, extractyears2(i))
}
unique(YearsRoleValidity)
```

Outcome:

```
 [1] 0
```

```
#Conclusion, the validity of YearsAtCompany>=YearsInCurrentRole holds.

#Check 3. YearsAtCompany>=YearsSinceLastPromotion
extractyears3 <- function(x) {
  if(HRdata[x, "YearsAtCompany"] < HRdata[x, "YearsSinceLastPromotion"]){
    return(1)
  } else{
    return(0)
  }
}
YearsPromotionValidity <- NULL
for (i in 1:nrow(HRdata)){
  YearsPromotionValidity <- c(YearsPromotionValidity, extractyears3(i))
}
unique(YearsPromotionValidity)
```

Outcome:

```
 [1] 0
```

```
#Conclusion, the validity of YearsAtCompany>=YearsSinceLastPromotion holds.

#Check 4. YearsAtCompany>=YearsWithCurrManager
extractyears4 <- function(x) {
  if(HRdata[x, "YearsAtCompany"] < HRdata[x, "YearsWithCurrManager"]){
    return(1)
  } else{
    return(0)
  }
}
YearsManagerValidity <- NULL
for (i in 1:nrow(HRdata)){
  YearsManagerValidity <- c(YearsManagerValidity, extractyears4(i))
}
unique(YearsManagerValidity)
```

Outcome:

```
 [1] 0
```

```
#Conclusion, the validity of YearsAtCompany>=YearsWithCurrManager holds.

#Check 5. MonthlyRate>=DailyRate
```

```
extractrate <- function(x) {
  if(HRdata[x, "MonthlyRate"] < HRdata[x, "DailyRate"]){
    return(1)
  } else{
    return(0)
  }
}
rateValidity <- NULL
for (i in 1:nrow(HRdata)){
  rateValidity <- c(rateValidity, extractrate(i))
}
unique(rateValidity)
```

Outcome:

```
[1] 0
```

```
#Conclusion, the validity of MonthlyRate>=DailyRate holds.

#Check 6. DailyRate>=HourlyRate
extractrate6 <- function(x) {
  if(HRdata[x, "DailyRate"] < HRdata[x, "HourlyRate"]){
    return(1)
  } else{
    return(0)
  }
}
rateValidity6 <- NULL
for (i in 1:nrow(HRdata)){
  rateValidity6 <- c(rateValidity6, extractrate6(i))
}
unique(rateValidity6)
```

Outcome:

```
[1] 0
```

```
#Conclusion, the validity of DailyRate>=HourlyRate holds.
```

Dataset Analytics

Frist dataset check

```
#check structure of the dataset
str(HRdata)
```

output:

```
Classes 'tbl_df', 'tbl' and 'data.frame':       1470 obs. of  35 variables:
 $ Attrition              : chr  "Yes" "No" "Yes" "No" ...
 $ EmployeeNumber         : num  1 2 4 5 7 8 10 11 12 13 ...
 $ Age                    : num  41 49 37 33 27 32 59 30 38 36 ...
 $ Gender                 : chr  "Female" "Male" "Male" "Female" ...
 $ MaritalStatus          : chr  "Single" "Married" "Single" "Married" ...
 $ NumCompaniesWorked     : num  8 1 6 1 9 0 4 1 0 6 ...
 $ EmployeeCount          : num  1 1 1 1 1 1 1 1 1 1 ...
 $ Over18                 : chr  "Y" "Y" "Y" "Y" ...
 $ TotalWorkingYears      : num  8 10 7 8 6 8 12 1 10 17 ...
 $ YearsAtCompany         : num  6 10 0 8 2 7 1 1 9 7 ...
 $ YearsInCurrentRole     : num  4 7 0 7 2 7 0 0 7 7 ...
 $ YearsSinceLastPromotion: num  0 1 0 3 2 3 0 0 1 7 ...
 $ YearsWithCurrManager   : num  5 7 0 0 2 6 0 0 8 7 ...
 $ TrainingTimesLastYear  : num  0 3 3 3 3 2 3 2 2 3 ...
 $ HourlyRate             : num  94 61 92 56 40 79 81 67 44 94 ...
 $ DailyRate              : num  1102 279 1373 1392 591 ...
 $ MonthlyRate            : num  19479 24907 2396 23159 16632 ...
 $ MonthlyIncome          : num  5993 5130 2090 2909 3468 ...
 $ PercentSalaryHike      : num  11 23 15 11 12 13 20 22 21 13 ...
 $ StandardHours          : num  80 80 80 80 80 80 80 80 80 80 ...
 $ StockOptionLevel       : num  0 1 0 0 1 0 3 1 0 2 ...
 $ OverTime               : chr  "Yes" "No" "Yes" "Yes" ...
 $ Education              : num  2 1 2 4 1 2 3 1 3 3 ...
 $ EducationField         : chr  "Life Sciences" "Life Sciences" "Other" "Life Sciences" ...
 $ Department             : chr  "Sales" "Research & Development" "Research & Development" "Research & Development" .
.
 $ BusinessTravel         : chr  "Travel_Rarely" "Travel_Frequently" "Travel_Rarely" "Travel_Frequently" ...
 $ JobLevel               : num  2 2 1 1 1 1 1 1 3 2 ...
 $ JobRole                : chr  "Sales Executive" "Research Scientist" "Laboratory Technician" "Research Scientist"
..
 $ DistanceFromHome       : num  1 8 2 3 2 2 3 24 23 27 ...
 $ JobInvolvement         : num  3 2 2 3 3 3 4 3 2 3 ...
 $ PerformanceRating      : num  3 4 3 3 3 3 4 4 4 3 ...
 $ EnvironmentSatisfaction : num  2 3 4 4 1 4 3 4 4 3 ...
 $ JobSatisfaction        : num  4 2 3 3 2 4 1 3 3 3 ...
 $ RelationshipSatisfaction: num  1 4 2 3 4 3 1 2 2 2 ...
 $ WorkLifeBalance        : num  1 3 3 3 3 2 2 3 3 2 ...
```

#the results indicate that some chr string variables should become factors
#check if this is true
unique(HRdata $Attrition)
unique(HRdata $Gender)
unique(HRdata $MaritalStatus)
unique(HRdata $Over18)
unique(HRdata $OverTime)
unique(HRdata $EducationField)
unique(HRdata $Department)
unique(HRdata $BusinessTravel)
unique(HRdata $JobRole)
outcome:

```
> unique(HRdata$Attrition)
[1] "Yes" "No"
> unique(HRdata$Gender)
[1] "Female" "Male"
> unique(HRdata$MaritalStatus)
[1] "Single"   "Married"  "Divorced"
> unique(HRdata$Over18)
[1] "Y"
> unique(HRdata$OverTime)
[1] "Yes" "No"
> unique(HRdata$EducationField)
[1] "Life Sciences"    "Other"            "Medical"          "Marketing"
"Technical Degree"
[6] "Human Resources"
> unique(HRdata$Department)
[1] "Sales"                "Research & Development" "Human Resources"
> unique(HRdata$BusinessTravel)
[1] "Travel_Rarely"     "Travel_Frequently" "Non-Travel"
> unique(HRdata$JobRole)
[1] "Sales Executive"           "Research Scientist"        "Laboratory T
echnician"     "Manufacturing Director"
```

```
[5] "Healthcare Representative" "Manager"                    "Sales Repres
entative"      "Research Director"
[9] "Human Resources"
```

#the results indicate that all of them should be factors instead of string characters.
#also Over18 seems to have only one unique value, which is uninformative related to the outcome.

#change the chr string to factor.
HRdata$Attrition <- as.factor(HRdata$Attrition)
HRdata$Gender <- as.factor(HRdata$Gender)
HRdata$MaritalStatus <- as.factor(HRdata$MaritalStatus)
HRdata$OverTime <- as.factor(HRdata$OverTime)
HRdata$EducationField <- as.factor(HRdata$EducationField)
HRdata$Department <- as.factor(HRdata$Department)
HRdata$BusinessTravel <- as.factor(HRdata$BusinessTravel)
HRdata$JobRole <- as.factor(HRdata$JobRole)
HRdata$Over18<- as.factor(HRdata$Over18)

#Check summary of dataset for interesting results and missing data.
summary.data.frame(HRdata)
Outcome:

```
 Attrition  EmployeeNumber        Age           Gender     MaritalStatus NumCompaniesWorked
 No :1233   Min.   :   1.0   Min.   :18.00   Female:588   Divorced:327   Min.   :0.000
 Yes: 237   1st Qu.: 491.2   1st Qu.:30.00   Male  :882   Married :673   1st Qu.:1.000
            Median :1020.5   Median :36.00                Single  :470   Median :2.000
            Mean   :1024.9   Mean   :36.92                               Mean   :2.693
            3rd Qu.:1555.8   3rd Qu.:43.00                               3rd Qu.:4.000
            Max.   :2068.0   Max.   :60.00                               Max.   :9.000

 EmployeeCount Over18   TotalWorkingYears YearsAtCompany   YearsInCurrentRole
 Min.   :1     Y:1470   Min.   : 0.00     Min.   : 0.000   Min.   : 0.000
 1st Qu.:1              1st Qu.: 6.00     1st Qu.: 3.000   1st Qu.: 2.000
 Median :1              Median :10.00     Median : 5.000   Median : 3.000
 Mean   :1              Mean   :11.28     Mean   : 7.008   Mean   : 4.229
 3rd Qu.:1              3rd Qu.:15.00     3rd Qu.: 9.000   3rd Qu.: 7.000
 Max.   :1              Max.   :40.00     Max.   :40.000   Max.   :18.000

 YearsSinceLastPromotion YearsWithCurrManager TrainingTimesLastYear   HourlyRate
 Min.   : 0.000          Min.   : 0.000       Min.   :0.000         Min.   : 30.00
 1st Qu.: 0.000          1st Qu.: 2.000       1st Qu.:2.000         1st Qu.: 48.00
 Median : 1.000          Median : 3.000       Median :3.000         Median : 66.00
 Mean   : 2.188          Mean   : 4.123       Mean   :2.799         Mean   : 65.89
 3rd Qu.: 3.000          3rd Qu.: 7.000       3rd Qu.:3.000         3rd Qu.: 83.75
 Max.   :15.000          Max.   :17.000       Max.   :6.000         Max.   :100.00

   DailyRate       MonthlyRate    MonthlyIncome   PercentSalaryHike StandardHours StockOptionLevel
 Min.   : 102.0   Min.   : 2094   Min.   : 1009   Min.   :11.00     Min.   :80    Min.   :0.0000
 1st Qu.: 465.0   1st Qu.: 8047   1st Qu.: 2911   1st Qu.:12.00     1st Qu.:80    1st Qu.:0.0000
 Median : 802.0   Median :14236   Median : 4919   Median :14.00     Median :80    Median :1.0000
 Mean   : 802.5   Mean   :14313   Mean   : 6503   Mean   :15.21     Mean   :80    Mean   :0.7939
 3rd Qu.:1157.0   3rd Qu.:20462   3rd Qu.: 8379   3rd Qu.:18.00     3rd Qu.:80    3rd Qu.:1.0000
 Max.   :1499.0   Max.   :26999   Max.   :19999   Max.   :25.00     Max.   :80    Max.   :3.0000

 OverTime     Education            EducationField                Department
 No :1054   Min.   :1.000   Human Resources : 27   Human Resources       : 63
 Yes: 416   1st Qu.:2.000   Life Sciences   :606   Research & Development:961
            Median :3.000   Marketing       :159   Sales                 :446
            Mean   :2.913   Medical         :464
            3rd Qu.:4.000   Other           : 82
            Max.   :5.000   Technical Degree:132

          BusinessTravel    JobLevel                         JobRole    DistanceFromHome
 Non-Travel       : 150   Min.   :1.000   Sales Executive          :326   Min.   : 1.000
 Travel_Frequently: 277   1st Qu.:1.000   Research Scientist       :292   1st Qu.: 2.000
 Travel_Rarely    :1043   Median :2.000   Laboratory Technician    :259   Median : 7.000
                          Mean   :2.064   Manufacturing Director   :145   Mean   : 9.193
                          3rd Qu.:3.000   Healthcare Representative:131   3rd Qu.:14.000
                          Max.   :5.000   Manager                  :102   Max.   :29.000
                                          (Other)                  :215
 JobInvolvement PerformanceRating EnvironmentSatisfaction JobSatisfaction RelationshipSatisfaction
 Min.   :1.00   Min.   :3.000     Min.   :1.000           Min.   :1.000   Min.   :1.000
 1st Qu.:2.00   1st Qu.:3.000     1st Qu.:2.000           1st Qu.:2.000   1st Qu.:2.000
 Median :3.00   Median :3.000     Median :3.000           Median :3.000   Median :3.000
 Mean   :2.73   Mean   :3.154     Mean   :2.722           Mean   :2.729   Mean   :2.712
 3rd Qu.:3.00   3rd Qu.:3.000     3rd Qu.:4.000           3rd Qu.:4.000   3rd Qu.:4.000
 Max.   :4.00   Max.   :4.000     Max.   :4.000           Max.   :4.000   Max.   :4.000

 WorkLifeBalance
 Min.   :1.000
 1st Qu.:2.000
 Median :3.000
 Mean   :2.761
 3rd Qu.:3.000
 Max.   :4.000
```

#no missing values found.
#Employee count seems also uninformative.
#PerformanceRating seems to have only two values 3 and 4. All employees in this dataset are rated to perform good.

#handle uninformative predictors.
summary.data.frame(HRdata[, nearZeroVar(HRdata)])

Outcome:

```
 EmployeeCount    Over18          StandardHours
  Min.   :1     Length:1470        Min.   :80
  1st Qu.:1     Class :character   1st Qu.:80
  Median :1     Mode  :character   Median :80
  Mean   :1                        Mean   :80
```

```
 3rd Qu.:1                          3rd Qu.:80
 Max.   :1                          Max.   :80
```

```
HRdata<-HRdata[ , -nearZeroVar(HRdata)]
#nearZeroVar indicates which columns should be removed.
#The columns EmployeeCount, Over18, and StandardHours are removed.

#Next, let's transform Attrition into binary variable.
# 1=Yes, 0=No
extractAttrition <- function(x) {
  if(HRdata[x, "Attrition"]== "Yes"){
    return(1)
  } else{
    return(0)
  }
}
AttritionBinary <- NULL
for (i in 1:nrow(HRdata)){
  AttritionBinary <- c(AttritionBinary, extractAttrition(i))
}
#check correctness
head(AttritionBinary)
head(HRdata[ , "Attrition"])
```

Outcome:

```
> head(AttritionBinary)
[1] 1 0 1 0 0 0
> head(HRdata[ , "Attrition"])
[1] Yes No  Yes No  No  No
Levels: No Yes
```

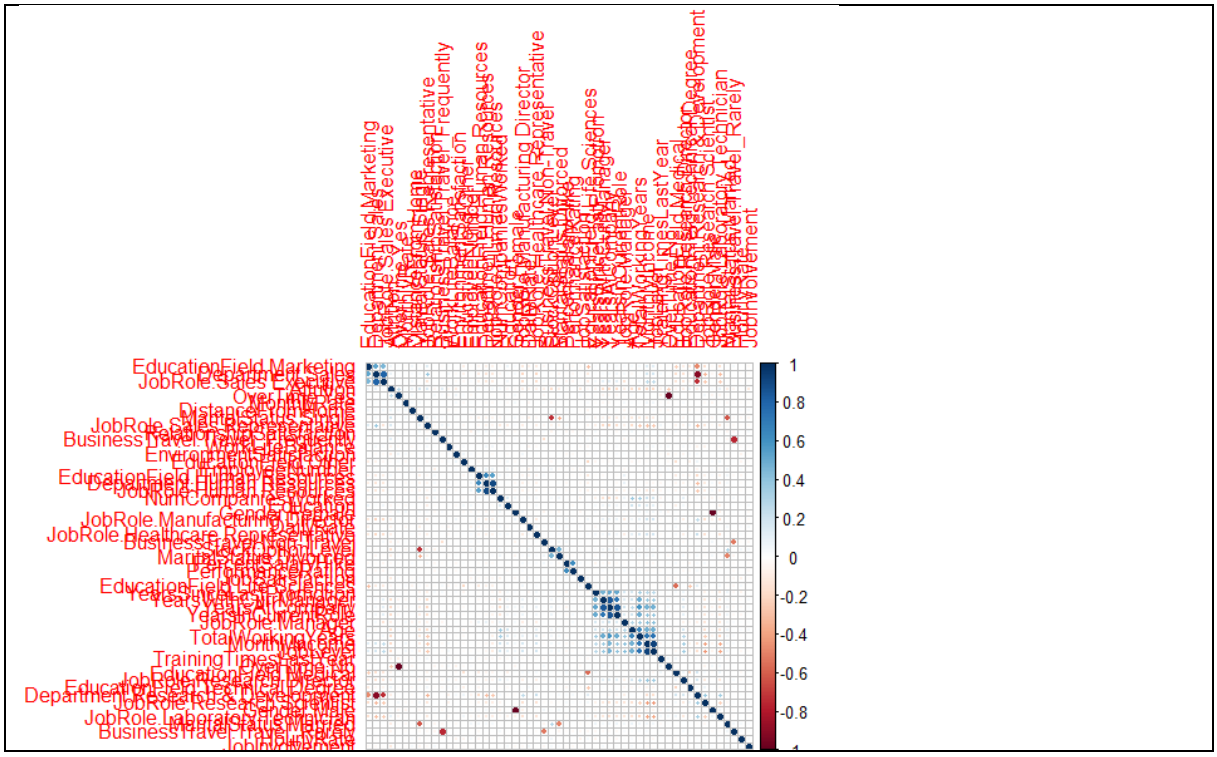
```
#transform into binary
HRdata[ , "Attrition"]<-AttritionBinary
```

Dummy variable creation

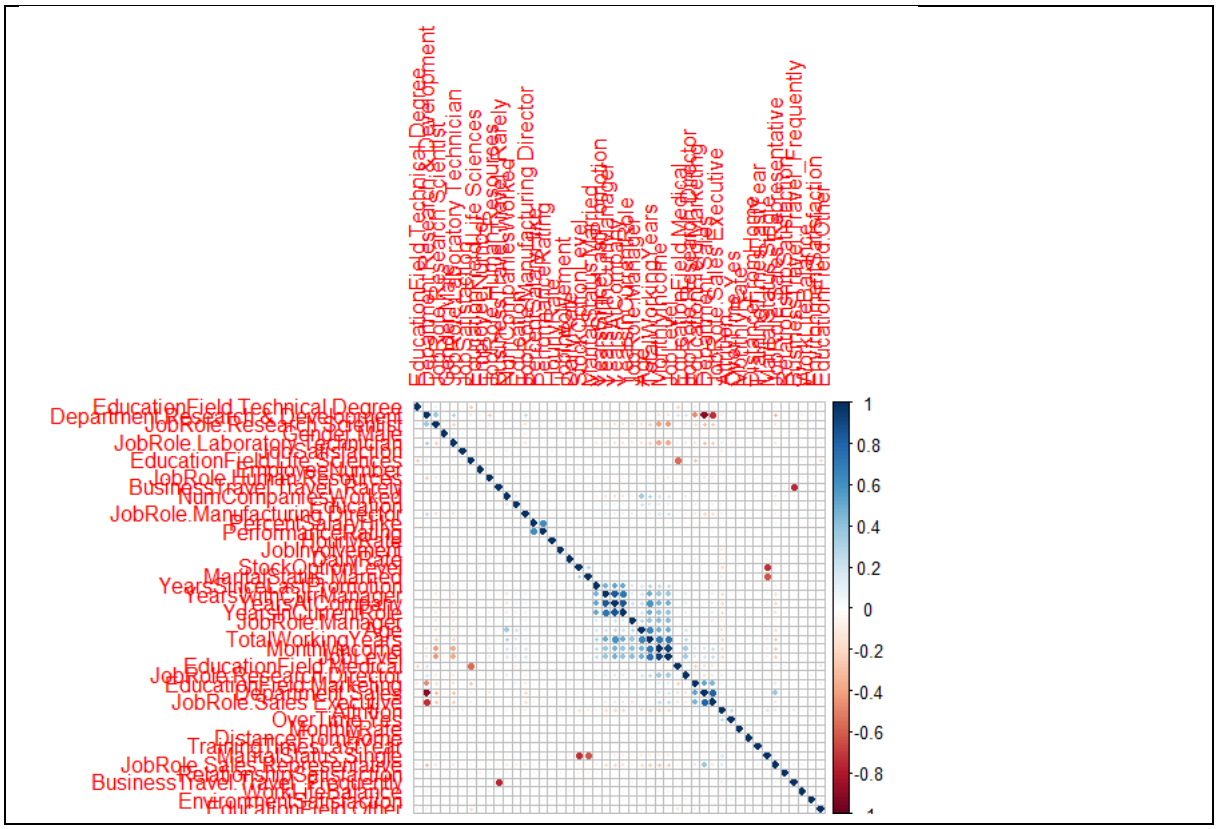
```
#Create dummy variables. Is necessary for some models and correlation check.
#One with all predictors for interpretability, and one with #class -1 predictors.
#FullRank=TRUE: factors are encoded so that there are no linear dependencies induced
between the columns
DummyVars <- dummyVars(~ Gender + MaritalStatus + EducationField + Department +
BusinessTravel + JobRole, data = HRdata, levelsOnly = FALSE, fullRank = FALSE)
HRdataDummyVars <- predict(DummyVars, HRdata)
DummyVars2 <- dummyVars(~ Gender + MaritalStatus + EducationField + Department +
BusinessTravel + JobRole, data = HRdata, levelsOnly = FALSE, fullRank = TRUE)
HRdataDummyVars2 <- predict(DummyVars2, HRdata)
#All predictors
HRdataDummy<-subset(HRdata, , -c(Gender, MaritalStatus, EducationField, Department,
BusinessTravel, JobRole))
HRdataDummy<-cbind(HRdataDummy, HRdataDummyVars)
#class -1 predictors, or no linear dependencies in dummyset.
HRdataDummyFullRank<-subset(HRdata, , -c(Gender, MaritalStatus, EducationField,
Department, BusinessTravel, JobRole))
HRdataDummyFullRank<-cbind(HRdataDummyFullRank, HRdataDummyVars2)
#HRdataDummy has 52 columns
#HRdataDummyFullRank has 46 columns
```

PCA and correlation check

#Create a correlation matrix, to get a feel for the data.
#first on dummy without full rank
correlations <- cor(HRdataDummy, method = c("spearman"))
corrplot(correlations, order = "hclust")
Outcome:



#next, on dummy with full rank
correlations2 <- cor(HRdataDummyFullRank, method = c("spearman"))
corrplot(correlations2, order = "hclust")
Outcome:



#notice a small decrease in correlations between HRdataDummy vs. HRdataDummyFullRank, and just couple of high correlations.

#Find highly correlated variables Full Rank.
highCorr <- findCorrelation(correlations, cutoff = .75)
length(highCorr)

Outcome:

```
5
```

#5 correlations were found with a between correlation of .75 or higher
#highCorr contains the column numbers of the highly correlated predictors
str(HRdataDummyFullRank[, highCorr])

Outcome:

```
'data.frame':  1470 obs. of  5 variables:
 $ JobLevel                          : num  2 2 1 1 1 1 1 1 3 2 ...
 $ YearsAtCompany                    : num  6 10 0 8 2 7 1 1 9 7 ...
 $ Department.Research & Development: num  0 1 1 1 1 1 1 1 1 1 ...
 $ Department.Sales                  : num  1 0 0 0 0 0 0 0 0 0 ...
 $ BusinessTravel.Travel_Rarely      : num  1 0 1 0 1 0 1 1 0 1 ...
```

#create dataset without the correlations Full Rank
HRdataDummyFullRankLowCorr<- HRdataDummyFullRank[, -highCorr]

#Check the difference with and without Full Rank.
#Find highly correlated variables.
highCorr <- findCorrelation(correlations, cutoff = .75)
length(highCorr)
#8 correlations were found with a between correlation of .75 or higher
#Notice that the number is higher compared to Full Rank.

#highCorr contains the column numbers of the highly correlated predictors
str(HRdataDummyFullRank[, highCorr])

Outcome:

```
'data.frame':  1470 obs. of  8 variables:
 $ JobLevel                     : num  2 2 1 1 1 1 1 1 3 2 ...
 $ YearsAtCompany               : num  6 10 0 8 2 7 1 1 9 7 ...
 $ JobRole.Laboratory Technician: num  0 0 1 0 1 1 1 1 0 0 ...
 $ JobRole.Manager              : num  0 0 0 0 0 0 0 0 0 0 ...
 $ JobRole.Sales Representative : num  0 0 0 0 0 0 0 0 0 0 ...
 $ MaritalStatus.Married        : num  0 1 0 1 1 0 1 0 0 1 ...
 $ EducationField.Medical       : num  0 0 0 0 1 0 1 0 0 1 ...
 $ JobRole.Research Director    : num  0 0 0 0 0 0 0 0 0 0 ...
```

#Also the except for the first two all the other variables are different

#check correlation of attrition vs all other predictors
corAttrition <- apply(HRdataDummy,2, function(col)cor(col, HRdataDummy$Attrition))
corAttrition

Outcome:

```
                   Attrition                          Age             NumCompaniesWorked
                1.0000000000                -0.1592050069                0.0434937391
             TotalWorkingYears                YearsAtCompany             YearsInCurrentRole
                -0.1710632461                -0.1343922140                -0.1605450043
      YearsSinceLastPromotion           YearsWithCurrManager           TrainingTimesLastYear
                -0.0330187751                -0.1561993159                -0.0594777986
                  HourlyRate                    DailyRate                  MonthlyRate
                -0.0068455496                -0.0566519919                0.0151702125
               MonthlyIncome               PercentSalaryHike             StockOptionLevel
                -0.1598395824                -0.0134782021                -0.1371449189
                   Education                     JobLevel             DistanceFromHome
                -0.0313728196                -0.1691047509                0.0779235830
              JobInvolvement           PerformanceRating        EnvironmentSatisfaction
                -0.1300159568                0.0028887517                -0.1033689783
             JobSatisfaction     RelationshipSatisfaction              WorkLifeBalance
                -0.1034811261                -0.0458722789                -0.0639390472
              Gender.Female                  Gender.Male          MaritalStatus.Divorced
                -0.0294532532                0.0294532532                -0.0877163459
      MaritalStatus.Married          MaritalStatus.Single                  OverTime.No
                -0.0909836512                0.1754185536                -0.2461179942
               OverTime.Yes  EducationField.Human Resources   EducationField.Life Sciences
                0.2461179942                0.0364661968                -0.0327031477
    EducationField.Marketing       EducationField.Medical         EducationField.Other
                0.0557806657                -0.0469987159                -0.0178975168
EducationField.Technical Degree    Department.Human Resources Department.Research & Development
                0.0693545948                0.0168320096                -0.0852929276
           Department.Sales      BusinessTravel.Non-Travel  BusinessTravel.Travel_Frequently
                0.0808552021                -0.0744572993                0.1151427655
BusinessTravel.Travel_Rarely JobRole.Healthcare Representative      JobRole.Human Resources
                -0.0495378384                -0.0786960496                0.0362150821
JobRole.Laboratory Technician              JobRole.Manager  JobRole.Manufacturing Director
                0.0982904855                -0.0833163842                -0.0829939241
   JobRole.Research Director     JobRole.Research Scientist      JobRole.Sales Executive
                -0.0888698417                -0.0003595713                0.0197743685
JobRole.Sales Representative
                0.1572342701
```

Next, principal component analysis to find out the main variance in the data.
#BoxCox corrects the skewness in the data.
#Center centers the variables.
#scale makes the scales of all variables the same.
pcaObject <- preProcess(HRdataDummyFullRankLowCorr, method = c("BoxCox", "center", "scale", "pca"))
pcaObject

Outcome:

```
Created from 1470 samples and 41 variables

Pre-processing:
```

```
  - Box-Cox transformation (14)
  - centered (41)
  - ignored (0)
  - principal component signal extraction (41)
  - scaled (41)

Lambda estimates for Box-Cox transformation:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-1.3000  0.3250  0.7500  0.6643  1.1000  1.6000

PCA needed 33 components to capture 95 percent of the variance
```

#out of 41 variables, 33 components were created to capture 95 percent of the varian ce.
#still many components are needed to explain the variance in the data.

Hypothesis check

#Hypotheses basic check, and first impression.

#first let's look at the outcome variable Attrition.
summary(HRdata$Attrition)
Outcome:

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.0000  0.0000  0.0000  0.1612  0.0000  1.0000
```

#make attrition a factor so basic plots can be made.
HRdata$Attrition<-as.factor(HRdata$Attrition)
summary(HRdata$Attrition)
Outcome:

```
  No  Yes
1233  237
```

#Only 16,12%, 237 out of 1470, is labeled as attrition.
#The rate of interest (Attrition yes) is under represented.
#This should be kept in mind for predictive model training.

#Hypothesis 1: Employee salary is a significant predictor of voluntary turnover.

#first create groups out of the salary.
#for this the function cut is used. This "breaks" MonthlyIncome in 5 pieces
HRdata$SalaryCut <- cut(HRdata$MonthlyIncome, breaks = 5, labels = FALSE)

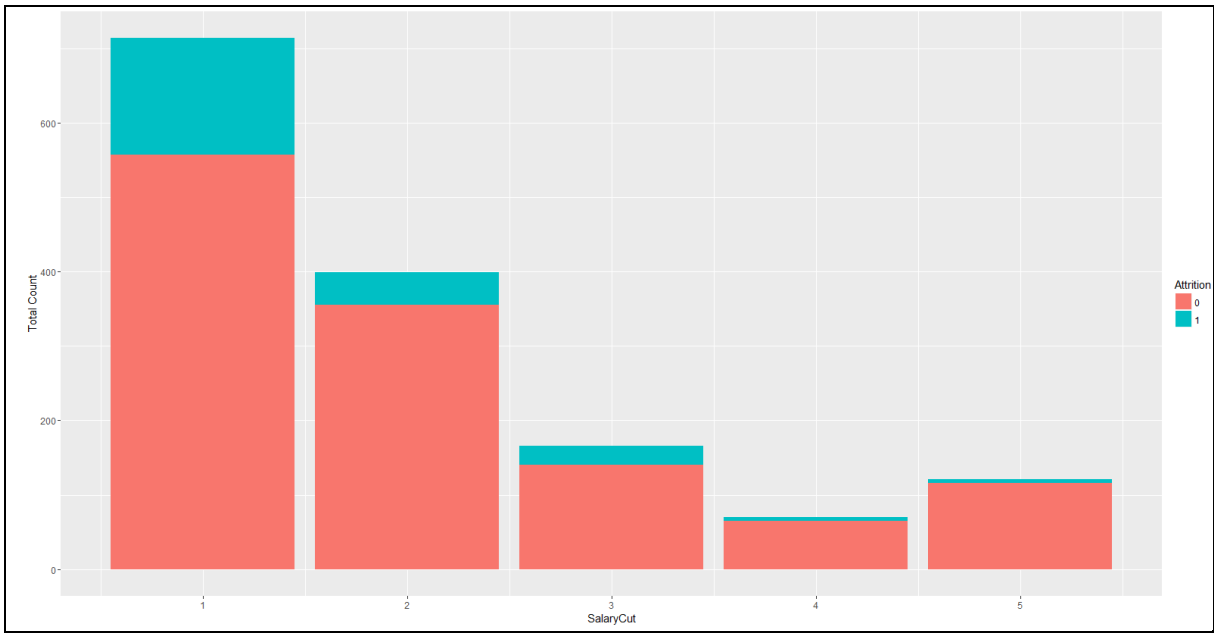
#create a plot to investigate income and attrition.
table(HRdata$SalaryCut)
Outcome:

```
  1   2   3   4   5
714 399 166  70 121
```

```
ggplot(HRdata, aes(x = SalaryCut, fill = factor(Attrition))) +
  geom_bar() +
  xlab("SalaryCut") +
  ylab("Total Count") +
```

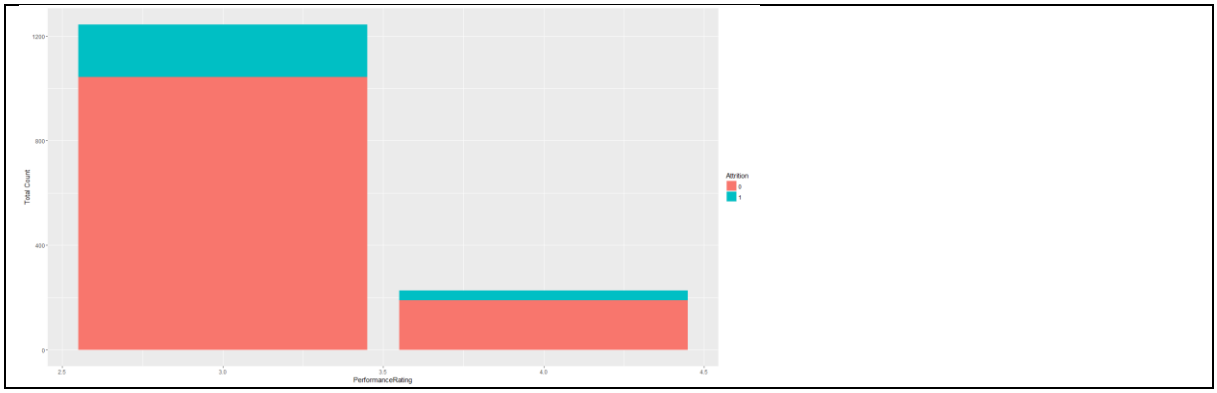
```
  labs(fill = "Attrition")
```

Outcome:




```
#The first impression confirms hypothesis 1.
#delete HRdata$SalaryCut
HRdata$SalaryCut<-NULL

#Hypothesis 2: Employee performance is a significant predictor of voluntary turnover.
#create a plot to investigate Employee performance and attrition.
ggplot(HRdata, aes(x = PerformanceRating, fill = factor(Attrition))) +
  geom_bar() +
  xlab("PerformanceRating") +
  ylab("Total Count") +
  labs(fill = "Attrition")
```

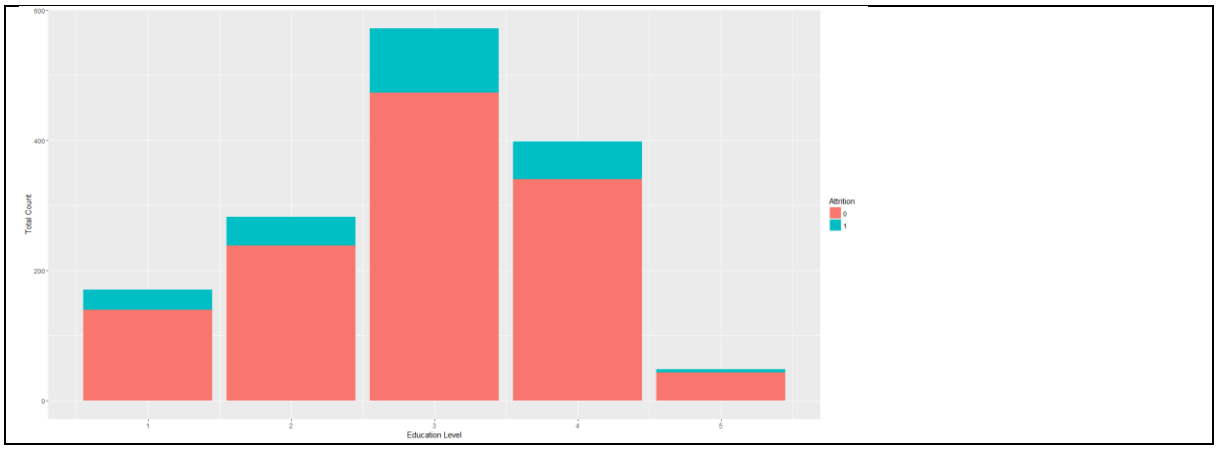
Outcome:



```
#again, the performance has only two unique values.
#only a fraction gets the performance rating 4.
#the proportion of leave for a performance rating of 3 seems only marginally more than 4.

#Hypothesis 3: Employee education level is a significant predictor of voluntary turnover.
#create a plot to investigate Employee education and attrition.
ggplot(HRdata, aes(x = Education, fill = factor(Attrition))) +
  geom_bar() +
  xlab("Education Level") +
```

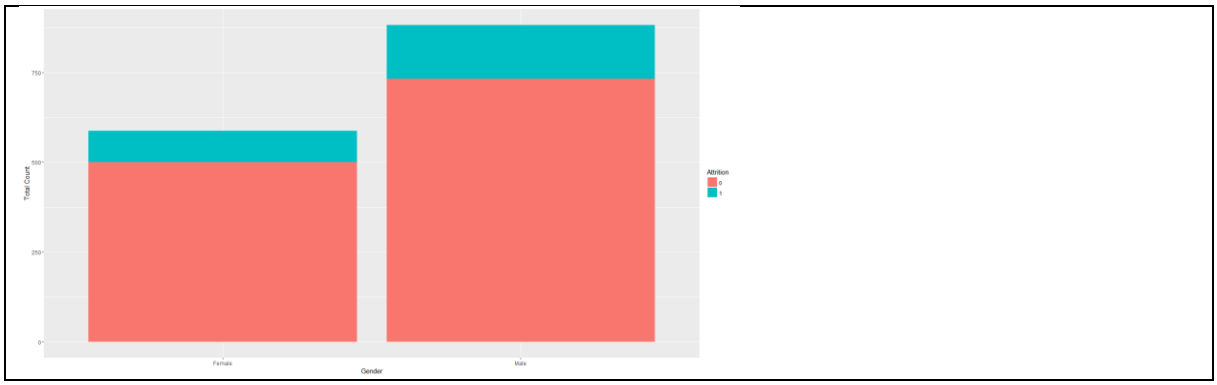
```
  ylab("Total Count") +
  labs(fill = "Attrition")
```

Outcome:




```
#an education level of 5 seems to be informative for employee attrition
#the proportions of attrition with a level of education below 5 seem equally distributed.

#Hypothesis 4: Female employees are more likely to voluntary leave than male employees.
#create a plot to investigate gender and attrition.
ggplot(HRdata, aes(x = Gender, fill = factor(Attrition))) +
  geom_bar() +
  xlab("Gender") +
  ylab("Total Count") +
  labs(fill = "Attrition")
```

Outcome:




```
#Gender seems uninformative for attrition.
#Looking at the plot and hypothesis, the opposite seems to be more likely.

#Hypothesis 5: Age is a significant factor of voluntary turnover.
#create a plot to investigate age and attrition.
ggplot(HRdata, aes(x = Age, fill = factor(Attrition))) +
  geom_bar() +
  xlab("Age") +
  ylab("Total Count") +
  labs(fill = "Attrition")
```

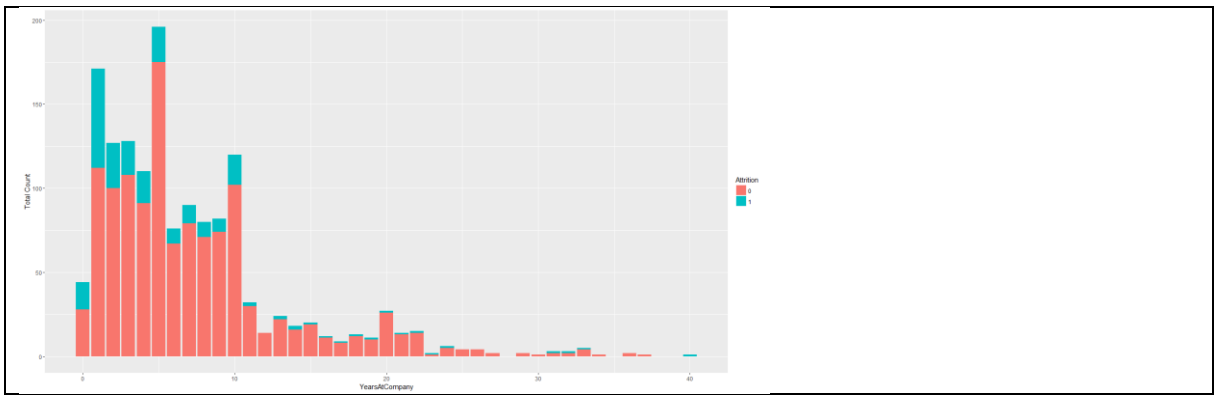
Outcome:

#The dataset does not seem to contain many old/retirement employees.
#Younger staff seems to be more likely to leave compared to older staff.

#Hypothesis 6: Years of service at company is a significant factor of voluntary turnover.
#create a plot to investigate years of service and attrition.

```
ggplot(HRdata, aes(x = YearsAtCompany, fill = factor(Attrition))) +
  geom_bar() +
  xlab("YearsAtCompany") +
  ylab("Total Count") +
  labs(fill = "Attrition")
```

Outcome:

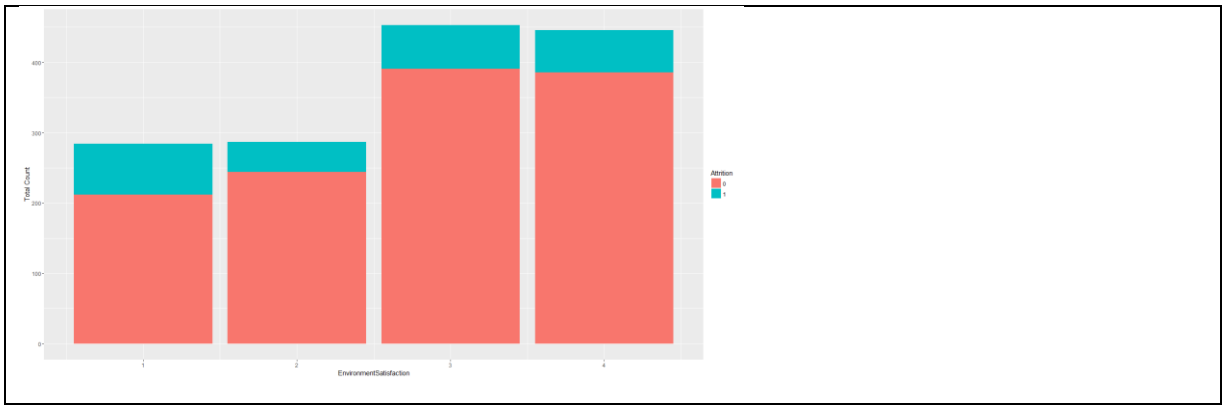


#years at company does seem informative for attrition.
#notice that the dataset is skewed to the right.

#Hypothesis 7: Work environment is a significant factor of voluntary turnover.
#create a plot to investigate work environment and attrition.

```
ggplot(HRdata, aes(x = EnvironmentSatisfaction, fill = factor(Attrition))) +
  geom_bar() +
  xlab("EnvironmentSatisfaction") +
  ylab("Total Count") +
  labs(fill = "Attrition")
```

Outcome:



#The environment does not seem to be informative for attrition, in this dataset.

#Next, the author decided that there is no causal link between EmployeeNumber and attrition, since uninformative predictors can decrease the performance of some models this variable is deleted from the datasets.

```
HRdata$EmployeeNumber<-NULL
HRdataDummy$EmployeeNumber<-NULL
HRdataDummyFullRank$EmployeeNumber<-NULL
HRdataDummyFullRankLowCorr$EmployeeNumber<-NULL
```

#Training and test set creation.

#The author chose to split the data into a training set of 75% and thus a test set of 25%.
#Since the rate of interest is under represented, stratified random sampling is used to split the data.

```
#first, create a dataset of the outcome.
classes <- HRdata[, "Attrition"]

# Set the random number seed so the results can be reproduced
set.seed(1247)
# By default, the numbers are returned as a list. Using list = FALSE, a matrix of row numbers is generated.
# These samples are allocated to the training set.
trainingRows <- createDataPartition(classes, p = .75, list = FALSE)

# Subset the data into objects for training using integer sub-setting.
trainClasses <- classes[trainingRows]
trainHRdata <- HRdata[trainingRows, ]
trainHRdataDummy <- HRdataDummy[trainingRows, ]
trainHRdataDummyFullRank <- HRdataDummyFullRank[trainingRows, ]
trainHRdataDummyFullRankLowCorr <- HRdataDummyFullRankLowCorr[trainingRows, ]
# Do the same for the test set using negative integers.
testClasses <- classes[-trainingRows]
testHRdata <- HRdata[-trainingRows, ]
testHRdataDummy <- HRdataDummy[-trainingRows, ]
testHRdataDummyFullRank <- HRdataDummyFullRank[-trainingRows, ]
testHRdataDummyFullRankLowCorr <- HRdataDummyFullRankLowCorr[-trainingRows, ]
```

Predictive Modeling

```
#to make computations go faster multi-core training is used.
cl <- makeCluster(6, type = "SOCK")
registerDoSNOW(cl)

#Shutdown cluster
stopCluster(cl)
```

#cross validation of the models

```
#with the caret function train, predictive models can be trained and tuned.
#within this function, trControl the method of cross validation.
#next, the author will set the specifics for the traincontrol settings used in this paper.
#It was chosen to use a repeated 10-fold cross-validation.
ctrl <- trainControl(method = "repeatedcv", repeats = 5, summaryFunction =
twoClassSummary, classProbs = TRUE, savePredictions = TRUE)

#-------------------------------------------------------
```

#create functions to check model results

```
#-------------------------------------------------------

# a = Machine learning algorithm
# b = dataset

#first, the function to get trainset results

trainresultsfunction <- function(a, b){
  #confusion matrix and roc curve for train set.
  #The basic predict call evaluates new samples, and type = "prob" returns the class
probabilities.
TrainPredict <- predict(a, b[, -1], type = "prob")
TrainPredict$class <- predict(a, b[, -1])
TrainPredict$outcome <- b$Attrition
TrainPredict$outcome <- as.factor(TrainPredict$outcome)

#Confusion matrix for trainset:
cm <- confusionMatrix(data = TrainPredict$class, reference = TrainPredict$outcome, positive
= "X1")
print(cm)

#plot ROC curve and Area under the curve statistic
RocTrain <- roc(response = TrainPredict$outcome, predictor = TrainPredict$X1, levels =
rev(levels(TrainPredict$outcome)))
plot(RocTrain, type = "s", print.thres = c(.5),
    print.thres.pch = 3,
    print.thres.pattern = "",
    print.thres.cex = 1.2,
    col = "red", legacy.axes = TRUE,
    print.thres.col = "red")
print(auc(RocTrain))

return(TrainPredict)
}
```

```
#Second, the function to get testset results

testresultsfunction <- function(a, b, c){
  #confusion matrix and roc curve for test set.
  #The basic predict call evaluates new samples, and type = "prob" returns the class
probabilities.
  TestPredict <- predict(a, b[, -1], type = "prob")
  TestPredict$class <- predict(a, b[, -1])
  TestPredict$outcome <- b$Attrition
  TestPredict$outcome <- as.factor(TestPredict$outcome)

  #Confusion matrix for testset:
  cm <- confusionMatrix(data = TestPredict$class, reference = TestPredict$outcome, positive
= "X1")
  print(cm)
  cmbyClass <- cm$byClass
  cmbyClass <- as.data.frame(cmbyClass)

  Fscore <-
(2*cmbyClass$cmbyClass[1]*cmbyClass$cmbyClass[3]/(cmbyClass$cmbyClass[1]+cmbyCla
ss$cmbyClass[3]))

  print(paste(c("F1 score =", Fscore), collapse = " "))

  #plot ROC curve and Area under the curve statistic
  RocTest <- roc(response = TestPredict$outcome, predictor = TestPredict$X1, levels =
rev(levels(TestPredict$outcome)))
  plot(RocTest,
       type = "s",
       add = TRUE,
       print.thres = c(.5),
       print.thres.pch = 16, legacy.axes = TRUE,
       print.thres.pattern = "",
       print.thres.cex = 1.2)

  legend(.75, .2,
         c(str_c("Testset ", c), str_c("Trainset ", c)),
         lwd = c(1, 1),
         col = c("black", "red"),
         pch = c(16, 3))

  print(auc(RocTest))

  return(TestPredict)
}
#create objects to save results in
trainresults <- NULL
testresults <- NULL
```

#Linear classification models

```
#make attrition useful for model classification

#first make outcome variable a factor to be useful for classification.
```

```
trainHRdataDummyFullRankLowCorr$Attrition <-
as.factor(trainHRdataDummyFullRankLowCorr$Attrition)
testHRdataDummyFullRankLowCorr$Attrition <-
as.factor(testHRdataDummyFullRankLowCorr$Attrition)
#change the name of factor, so it can be computed.
trainHRdataDummyFullRankLowCorr$Attrition <-
make.names(trainHRdataDummyFullRankLowCorr$Attrition, unique = FALSE, allow_ =
TRUE)
testHRdataDummyFullRankLowCorr$Attrition <-
make.names(testHRdataDummyFullRankLowCorr$Attrition, unique = FALSE, allow_ =
TRUE)

#first make outcome variable a factor.
trainHRdataDummy$Attrition <- as.factor(trainHRdataDummy$Attrition)
testHRdataDummy$Attrition <- as.factor(testHRdataDummy$Attrition)
#change the name of factor, so it can be computed.
trainHRdataDummy$Attrition <- make.names(trainHRdataDummy$Attrition, unique = FALSE,
allow_ = TRUE)
testHRdataDummy$Attrition <- make.names(testHRdataDummy$Attrition, unique = FALSE,
allow_ = TRUE)
```

#Logistic Regression

```
set.seed(1247)
logisticReg <- train(trainHRdataDummyFullRankLowCorr[ , -1], y =
trainHRdataDummyFullRankLowCorr$Attrition,  method = "glm", preProc = c("BoxCox",
"center", "scale"), metric = “ROC”, trControl = ctrl)
logisticReg
```

Outcome:

```
Generalized Linear Model

1103 samples
  39 predictor
   2 classes: 'X0', 'X1'

Pre-processing: Box-Cox transformation (13), centered (39), scaled (39)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 992, 992, 992, 993, 994, 993, ...
Resampling results:

  ROC        Sens       Spec
  0.8380446  0.9615101  0.4566013
```

```
#the predictions for this analysis is contained in the sub-object pred.
head(logisticReg$pred)
```

Outcome:

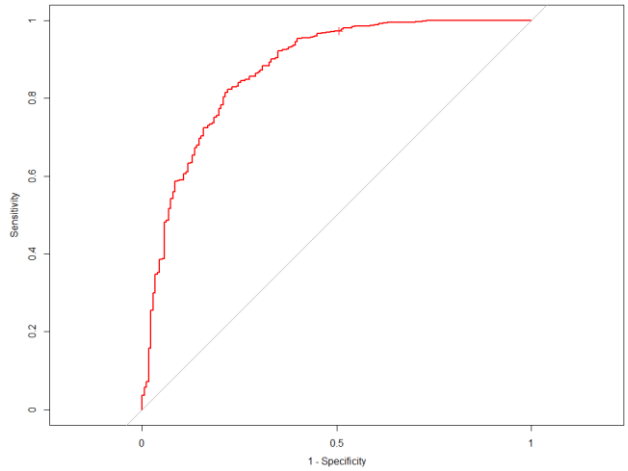
```
  pred obs        X0           X1 rowIndex parameter    Resample
1   X1  X1 0.3289703 0.6710297002        1      none Fold01.Rep1
2   X0  X0 0.9749180 0.0250820405        6      none Fold01.Rep1
3   X1  X1 0.2889912 0.7110088106       16      none Fold01.Rep1
4   X0  X0 0.9992801 0.0007199291       17      none Fold01.Rep1
5   X1  X0 0.2274435 0.7725564938       39      none Fold01.Rep1
6   X0  X0 0.9437310 0.0562690283       42      none Fold01.Rep1
```

```
#results of trained model
```

trainresults$logisticReg <- trainresultsfunction(logisticReg, trainHRdataDummyFullRankLowCorr)

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 900  90
        X1  25  88

               Accuracy : 0.8957
                 95% CI : (0.8762, 0.9132)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : 3.419e-08

                  Kappa : 0.5482
 Mcnemar's Test P-Value : 2.401e-09

            Sensitivity : 0.49438
            Specificity : 0.97297
         Pos Pred Value : 0.77876
         Neg Pred Value : 0.90909
             Prevalence : 0.16138
         Detection Rate : 0.07978
   Detection Prevalence : 0.10245
      Balanced Accuracy : 0.73368

       'Positive' Class : X1

[1] "F1 score = 0.494623655913979"
Area under the curve: 0.8739
```

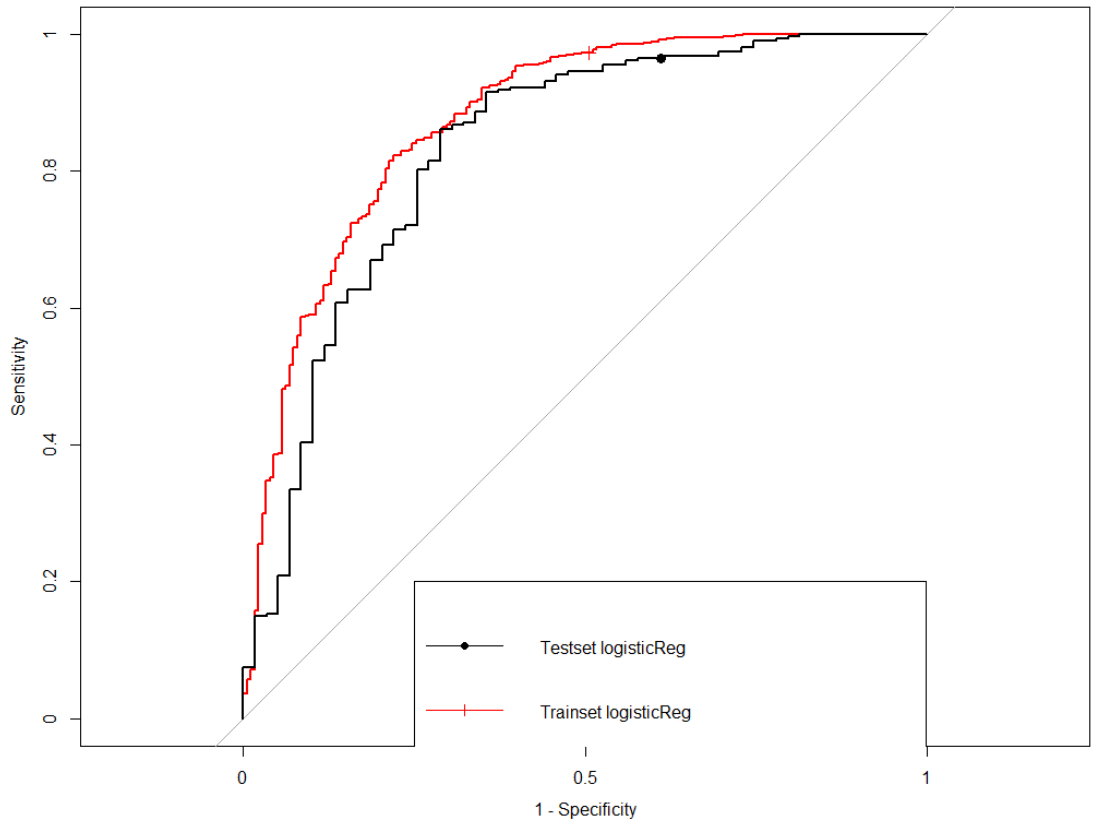


#results of trained model on test set
testresults$logisticReg <- testresultsfunction(logisticReg, trainHRdataDummyFullRankLowCorr, "logisticReg")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 297  36
        X1  11  23

               Accuracy : 0.8719
```

```
                  95% CI : (0.8334, 0.9044)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.0480056

                   Kappa : 0.4273
 Mcnemar's Test P-Value : 0.0004639

            Sensitivity : 0.38983
            Specificity : 0.96429
         Pos Pred Value : 0.67647
         Neg Pred Value : 0.89189
             Prevalence : 0.16076
         Detection Rate : 0.06267
   Detection Prevalence : 0.09264
      Balanced Accuracy : 0.67706

       'Positive' Class : X1

Area under the curve: 0.8303
```



#Linear Discriminant Analysis

```
set.seed(1247)
ldaFit <- train(trainHRdataDummyFullRankLowCorr[ , -1], y =
trainHRdataDummyFullRankLowCorr$Attrition, method = "lda", preProc = c("BoxCox",
"center", "scale"), metric = "ROC", trControl = ctrl)
ldaFit
```

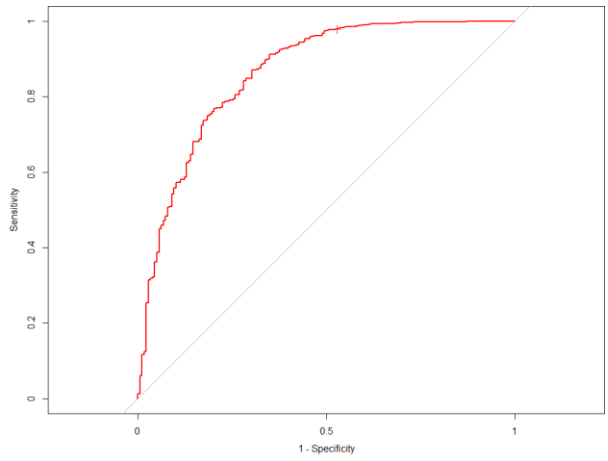
Outcome:

```
Linear Discriminant Analysis
```

```
1103 samples
  39 predictor
   2 classes: 'X0', 'X1'

Pre-processing: Box-Cox transformation (13), centered (39), scaled (39)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 992, 992, 992, 993, 994, 993, ...
Resampling results:

  ROC        Sens       Spec
  0.8302441  0.9660519  0.4498039
```

#results of trained model
trainresults$ldaFit <- trainresultsfunction(ldaFit, trainHRdataDummyFullRankLowCorr)
Outcome:

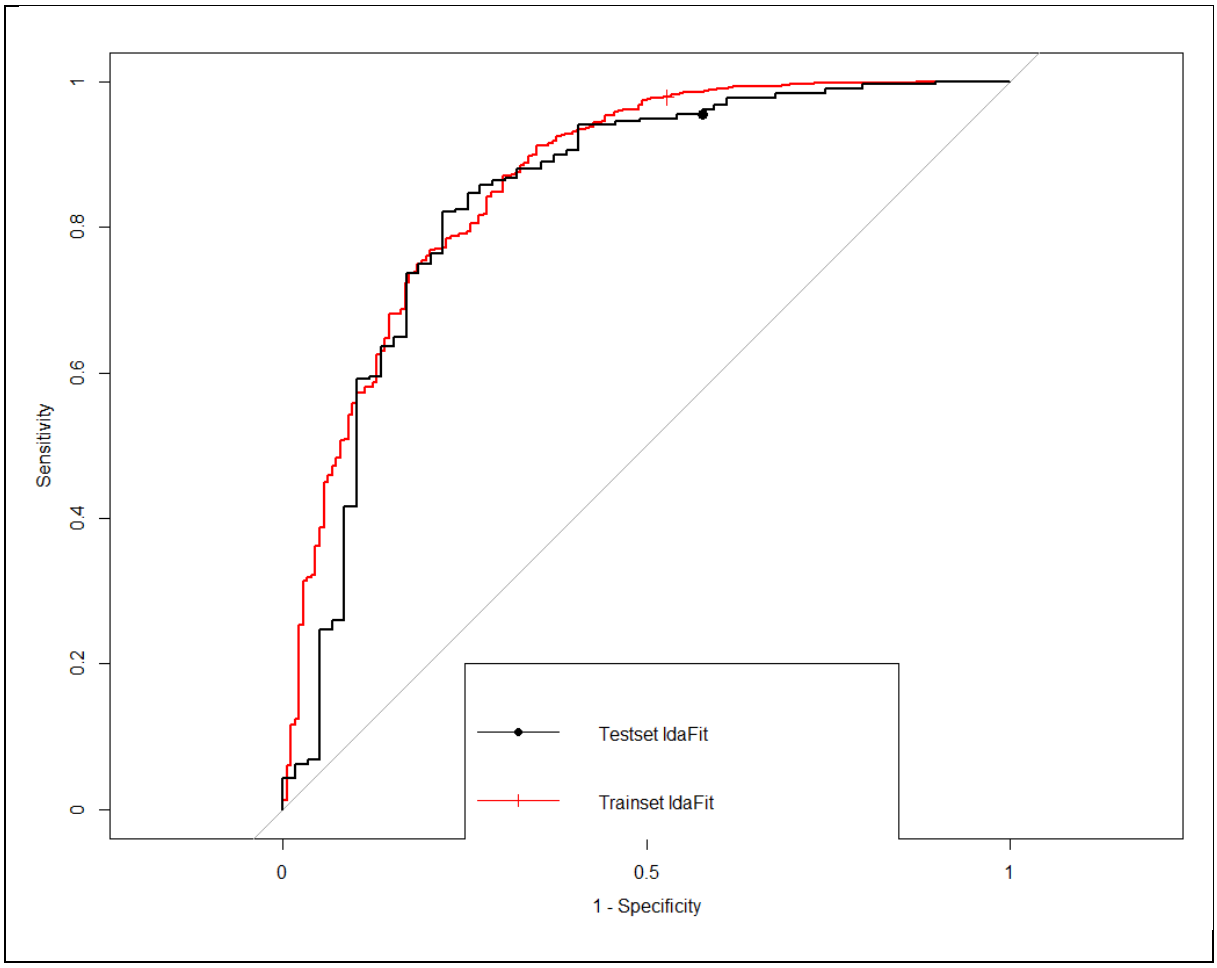
```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 905  94
        X1  20  84

               Accuracy : 0.8966
                 95% CI : (0.8772, 0.914)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : 2.037e-08

                  Kappa : 0.5411
 Mcnemar's Test P-Value : 8.083e-12

            Sensitivity : 0.47191
            Specificity : 0.97838
         Pos Pred Value : 0.80769
         Neg Pred Value : 0.90591
             Prevalence : 0.16138
         Detection Rate : 0.07616
   Detection Prevalence : 0.09429
      Balanced Accuracy : 0.72514

       'Positive' Class : X1

Area under the curve: 0.8633
```



#results of trained model on test set
testresults$ldaFit <- testresultsfunction(ldaFit, testHRdataDummyFullRankLowCorr, "ldaFit")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 294  34
        X1  14  25

               Accuracy : 0.8692
                 95% CI : (0.8304, 0.902)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.064977

                  Kappa : 0.4383
 Mcnemar's Test P-Value : 0.006099

            Sensitivity : 0.42373
            Specificity : 0.95455
         Pos Pred Value : 0.64103
         Neg Pred Value : 0.89634
             Prevalence : 0.16076
         Detection Rate : 0.06812
   Detection Prevalence : 0.10627
      Balanced Accuracy : 0.68914

       'Positive' Class : X1

[1] "F1 score = 0.510204081632653"
Area under the curve: 0.8397
```



#Penalized logistic regression

```
# Specify the tuning values for training
glmnGrid <- expand.grid(.alpha = seq(0, 1, length = 10), .lambda = seq(.01, .2, length = 40))

#train the model
set.seed(1247)
glmnTuned <- train(trainHRdataDummyFullRankLowCorr[ , -1], y =
trainHRdataDummyFullRankLowCorr$Attrition, method = "glmnet", tuneGrid = glmnGrid,
preProc = c("BoxCox", "center", "scale"), metric = "ROC", trControl = ctrl)
glmnTuned
```

Outcome:

```
glmnet

1103 samples
  39 predictor
   2 classes: 'X0', 'X1'

Pre-processing: Box-Cox transformation (13), centered (39), scaled (39)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 993, 992, 993, 993, 992, 994, ...
Resampling results across tuning parameters:


  alpha  lambda       ROC        Sens       Spec
  0.0    0.01000000  0.8388901  0.9790276  0.409411765
  0.0    0.01487179  0.8384322  0.9837821  0.394836601
  0.0    0.01974359  0.8376994  0.9865965  0.378039216
```

```
  0.0    0.02461538  0.8371455  0.9885414  0.352026144
  0.0    0.02948718  0.8364904  0.9894086  0.336405229
  0.0    0.03435897  0.8358813  0.9902735  0.319346405
  0.0    0.03923077  0.8351582  0.9915708  0.298954248
  0.0    0.04410256  0.8345746  0.9924381  0.282091503
  0.0    0.04897436  0.8338869  0.9930832  0.270653595
  0.0    0.05384615  0.8334509  0.9935157  0.256143791
  0.0    0.05871795  0.8332058  0.9945979  0.241568627
  0.0    0.06358974  0.8325966  0.9956802  0.222352941
  0.0    0.06846154  0.8321102  0.9963277  0.204379085
  0.0    0.07333333  0.8316511  0.9969752  0.194248366
  0.0    0.07820513  0.8310447  0.9974077  0.180718954
```

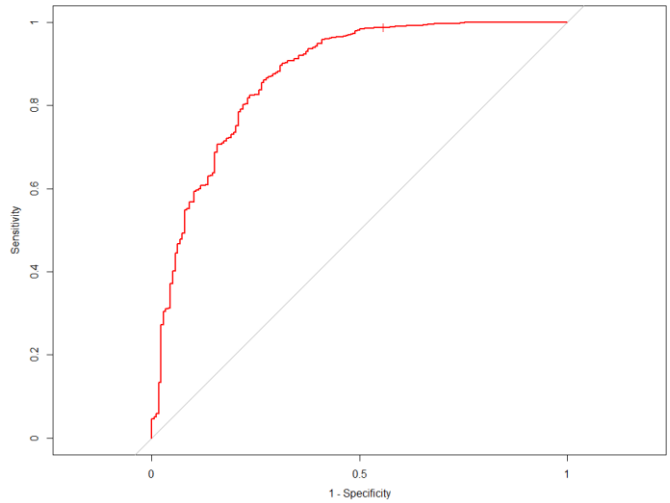
This list was reduced!

```
ROC was used to select the optimal model using  the largest value.
The final values used for the model were alpha = 0 and lambda = 0.01.
```

#results of trained model
trainresults$glmnTuned <- trainresultsfunction(glmnTuned,
trainHRdataDummyFullRankLowCorr)

Outcome:

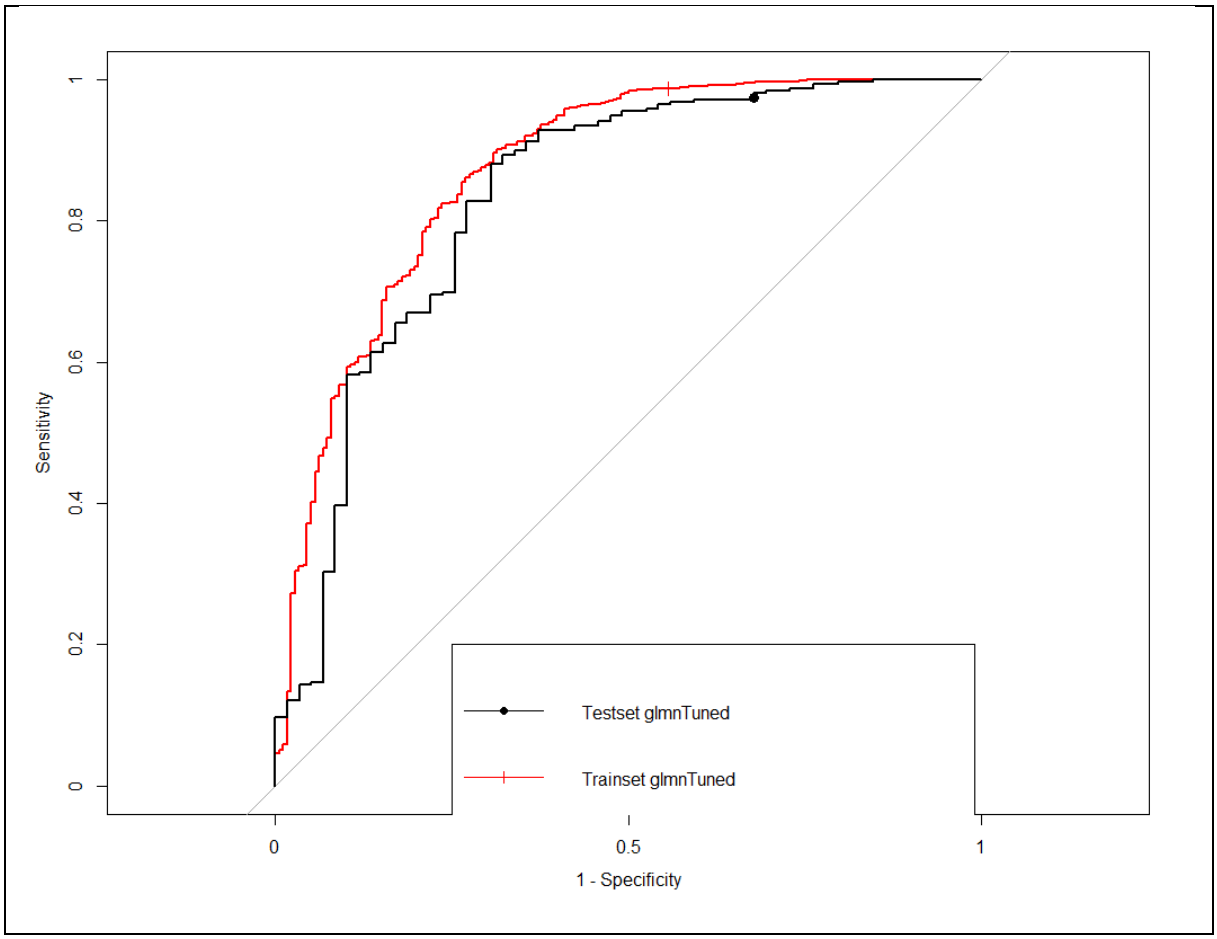
```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 913  99
        X1  12  79

               Accuracy : 0.8994
                 95% CI : (0.8801, 0.9165)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : 4.070e-09

                  Kappa : 0.5368
 Mcnemar's Test P-Value : 3.275e-16

            Sensitivity : 0.44382
            Specificity : 0.98703
         Pos Pred Value : 0.86813
         Neg Pred Value : 0.90217
             Prevalence : 0.16138
         Detection Rate : 0.07162
   Detection Prevalence : 0.08250
      Balanced Accuracy : 0.71542

       'Positive' Class : X1

Area under the curve: 0.87
```



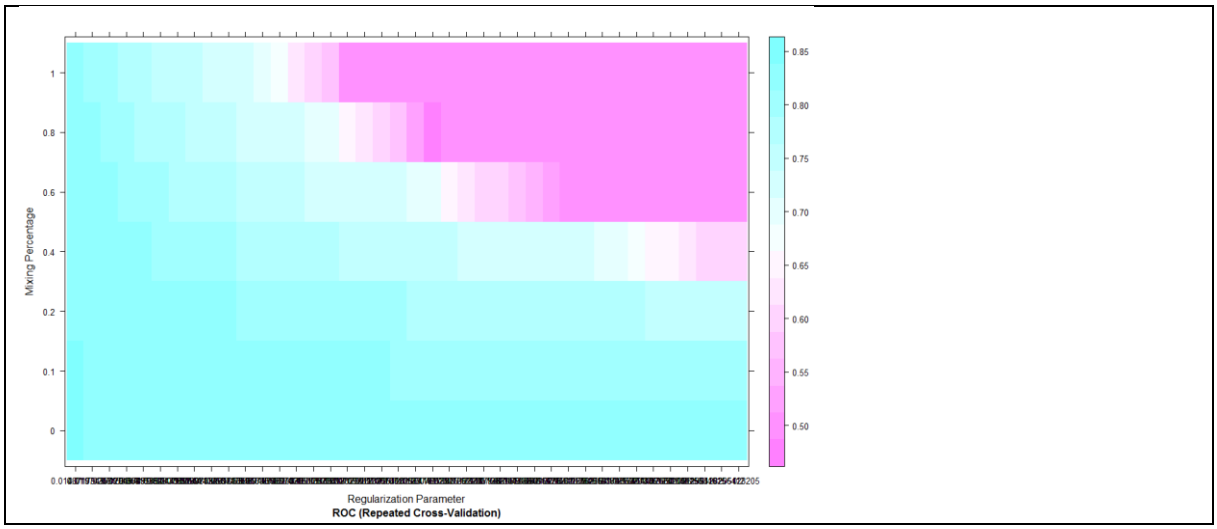
#results of trained model on test set
testresults$glmnTuned <- testresultsfunction(glmnTuned, testHRdataDummyFullRankLowCorr, "glmnTuned")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 300  40
        X1   8  19

               Accuracy : 0.8692
                 95% CI : (0.8304, 0.902)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.06498

                  Kappa : 0.3792
 Mcnemar's Test P-Value : 7.66e-06

            Sensitivity : 0.32203
            Specificity : 0.97403
         Pos Pred Value : 0.70370
         Neg Pred Value : 0.88235
             Prevalence : 0.16076
         Detection Rate : 0.05177
   Detection Prevalence : 0.07357
      Balanced Accuracy : 0.64803

       'Positive' Class : X1

[1] "F1 score = 0.441860465116279"
Area under the curve: 0.8319
```



#Create a heatmap.
plot(glmnTuned, plotType = "level")
Outcome:



#Nonlinear classification models

#Neural Networks

#create grid

```
nnetGrid <- expand.grid(.size = 1:10, .decay = c(0, .1, 1, 2))

maxSize <- max(nnetGrid$.size)
numWts <- 1*(maxSize * (length(trainHRdataDummyFullRankLowCorr) ) + maxSize + 1)

#train model
#spatialSign increases the predictive performance
set.seed(1247)
nnetFit <- train(trainHRdataDummyFullRankLowCorr[ , -1],
trainHRdataDummyFullRankLowCorr$Attrition, method = "nnet", metric = "ROC", preProc =
c("BoxCox", "center", "scale", "spatialSign"), tuneGrid = nnetGrid, trace = FALSE, maxit =
2000, MaxNWts = numWts, trControl = ctrl)
nnetFit
```

Outcome:

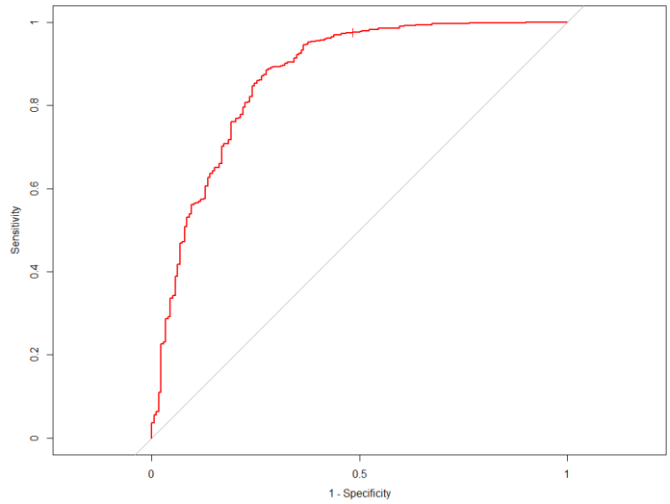
```
Neural Network

1103 samples
  39 predictor
   2 classes: 'X0', 'X1'

Pre-processing: Box-Cox transformation (13), centered (39), scaled (39), s
patial
 sign transformation (39)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 992, 992, 992, 993, 994, 993, ...
Resampling results across tuning parameters:

  size  decay  ROC        Sens       Spec
   1    0.0    0.6331973  0.9556942  0.339346405
   1    0.1    0.8403970  0.9641000  0.465294118
   1    1.0    0.8325750  0.9974053  0.137973856
   1    2.0    0.8244572  1.0000000  0.000000000
   2    0.0    0.6156003  0.9424848  0.384771242
   2    0.1    0.8236169  0.9576134  0.462026144
   2    1.0    0.8326925  0.9937284  0.202026144
   2    2.0    0.8236968  1.0000000  0.002222222
   3    0.0    0.6417868  0.9331884  0.379738562
   3    0.1    0.8095311  0.9411664  0.447516340
   3    1.0    0.8328273  0.9937307  0.215424837
   3    2.0    0.8242260  1.0000000  0.005555556
Note: list was reduced

ROC was used to select the optimal model using  the largest value.
The final values used for the model were size = 1 and decay = 0.1.
```

```
#results of trained model
trainresults$nnetFit <- trainresultsfunction(nnetFit, trainHRdataDummyFullRankLowCorr)
```

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 902  86
        X1  23  92

               Accuracy : 0.9012
                 95% CI : (0.882, 0.9182)
    No Information Rate : 0.8386
```

```
    P-Value [Acc > NIR] : 1.325e-09

                  Kappa : 0.574
 Mcnemar's Test P-Value : 2.876e-09

            Sensitivity : 0.51685
            Specificity : 0.97514
         Pos Pred Value : 0.80000
         Neg Pred Value : 0.91296
             Prevalence : 0.16138
         Detection Rate : 0.08341
   Detection Prevalence : 0.10426
      Balanced Accuracy : 0.74599

       'Positive' Class : X1

Area under the curve: 0.867
```

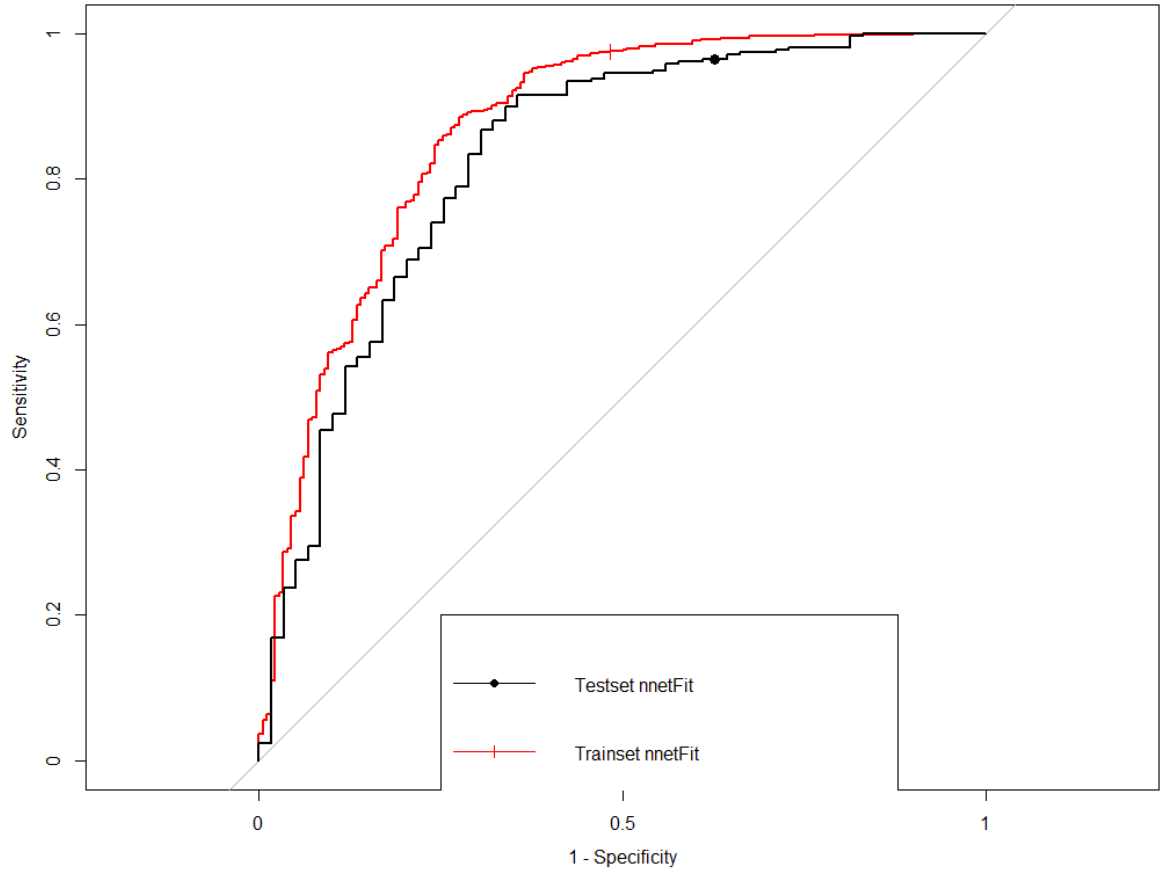


#results of trained model on test set
testresults$nnetFit <- testresultsfunction(nnetFit, testHRdataDummyFullRankLowCorr, "nnetFit")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 297  37
        X1  11  22

               Accuracy : 0.8692
                 95% CI : (0.8304, 0.902)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.064977

                  Kappa : 0.4102
 Mcnemar's Test P-Value : 0.000308

            Sensitivity : 0.37288
            Specificity : 0.96429
         Pos Pred Value : 0.66667
         Neg Pred Value : 0.88922
             Prevalence : 0.16076
```

```
          Detection Rate : 0.05995
    Detection Prevalence : 0.08992
       Balanced Accuracy : 0.66858

        'Positive' Class : X1

[1] "F1 score = 0.478260869565217"
Area under the curve: 0.828
```




```
#-----------------------
#train average nnet model
#-----------------------

#create grid
rdaGrid <- expand.grid(.gamma = seq(0, 1, length = 10), .lambda = seq(0, 1, length = 10))

set.seed(1247)
nnetFit2 <- train(x = trainHRdataDummyFullRankLowCorr[ , -1], y =
trainHRdataDummyFullRankLowCorr$Attrition, method = "avNNet", metric = "ROC", preProc
= c("BoxCox", "center", "scale", "spatialSign"), tuneGrid = avnnetGrid, trace = FALSE,
trControl = ctrl)
nnetFit2
```

Outcome:

```
Model Averaged Neural Network

1103 samples
  39 predictor
```

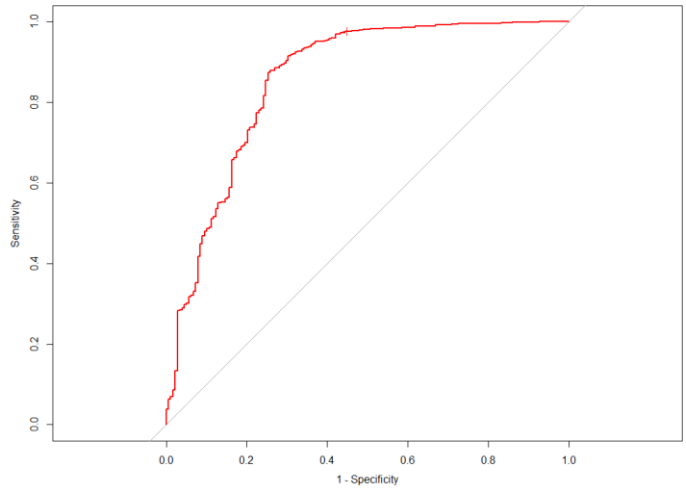
```
   2 classes: 'X0', 'X1'

Pre-processing: Box-Cox transformation (13), centered (39), scaled (39), s
patial sign transformation (39)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 992, 992, 992, 993, 994, 993, ...
Resampling results across tuning parameters:

  size  decay  ROC        Sens       Spec
   1    0.0    0.4937108  1.0000000  0.000000000
   1    0.1    0.8355051  0.9712436  0.455228758
   1    1.0    0.8242448  1.0000000  0.007777778
   1    2.0    0.8060022  1.0000000  0.000000000
   2    0.0    0.5278147  1.0000000  0.000000000
   2    0.1    0.8318003  0.9625993  0.449738562
   2    1.0    0.8231309  1.0000000  0.014509804
   2    2.0    0.8041041  1.0000000  0.000000000
   3    0.0    0.5093318  1.0000000  0.000000000
   3    0.1    0.8315763  0.9608509  0.443660131
   3    1.0    0.8263758  0.9987050  0.050588235
   3    2.0    0.8044978  1.0000000  0.000000000
   4    0.0    0.5195749  1.0000000  0.000000000
   4    0.1    0.8326432  0.9617251  0.453071895
   4    1.0    0.8247662  0.9995699  0.060522876
   4    2.0    0.8051703  1.0000000  0.000000000
   5    0.0    0.5321935  1.0000000  0.000000000
   5    0.1    0.8308858  0.9621482  0.456535948
   5    1.0    0.8248748  1.0000000  0.058300654
   5    2.0    0.8018000  1.0000000  0.000000000
   6    0.0    0.5539379  1.0000000  0.000000000
   6    0.1    0.8350617  0.9619402  0.448562092
   6    1.0    0.8232154  0.9993525  0.068300654
   6    2.0    0.8023413  1.0000000  0.000000000
   7    0.0    0.5543868  1.0000000  0.000000000
   7    0.1    0.8305635  0.9595605  0.454052288
   7    1.0    0.8247775  0.9991374  0.067516340
   7    2.0    0.8036769  1.0000000  0.000000000
   8    0.0    0.5831814  1.0000000  0.000000000
   8    0.1    0.8271659  0.9632235  0.441895425
   8    1.0    0.8230947  0.9991328  0.076143791
   8    2.0    0.8033158  1.0000000  0.000000000
   9    0.0    0.5599956  1.0000000  0.000000000
   9    0.1    0.8331060  0.9630178  0.448169935
   9    1.0    0.8249645  0.9993525  0.062549020
   9    2.0    0.8042971  1.0000000  0.000000000
  10    0.0    0.5501331  1.0000000  0.000000000
  10    0.1    0.8324096  0.9623749  0.440653595
  10    1.0    0.8245509  0.9995676  0.067320261
  10    2.0    0.8019499  1.0000000  0.000000000

Tuning parameter 'bag' was held constant at a value of 10
ROC was used to select the optimal model using  the largest value.
The final values used for the model were size = 1, decay = 0.1 and bag = 1
0.
```

#results of trained model
trainresults$nnetFit2 <- trainresultsfunction(nnetFit2, trainHRdataDummyFullRankLowCorr)

Outcome:

```
Confusion Matrix and Statistics
```

```
          Reference
Prediction  X0  X1
        X0 902  80
        X1  23  98

               Accuracy : 0.9066
                 95% CI : (0.8879, 0.9231)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : 3.587e-11

                  Kappa : 0.6038
 Mcnemar's Test P-Value : 3.432e-08

            Sensitivity : 0.55056
            Specificity : 0.97514
         Pos Pred Value : 0.80992
         Neg Pred Value : 0.91853
             Prevalence : 0.16138
         Detection Rate : 0.08885
   Detection Prevalence : 0.10970
      Balanced Accuracy : 0.76285

       'Positive' Class : X1

Area under the curve: 0.856
```

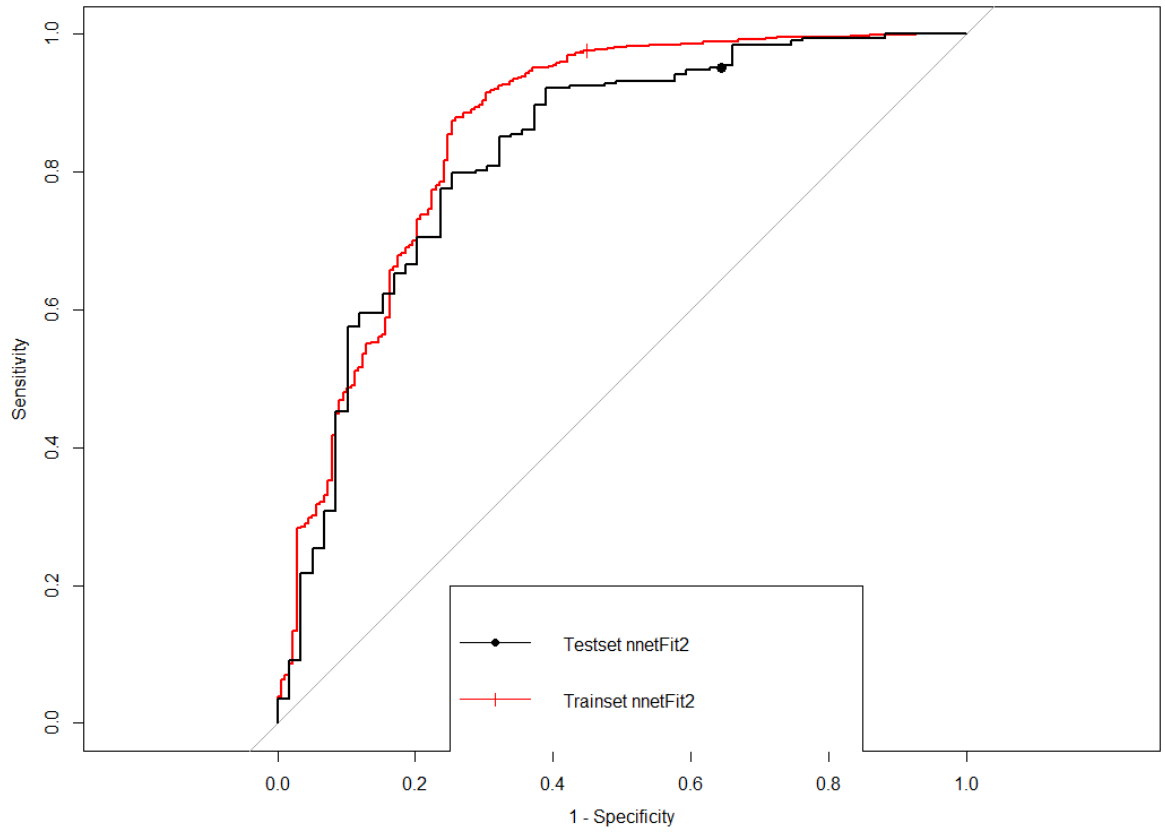


#results of trained model on test set
testresults$nnetFit2 <- testresultsfunction(nnetFit2, testHRdataDummyFullRankLowCorr, "nnetFit2")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 293  38
        X1  15  21

               Accuracy : 0.8556
                 95% CI : (0.8154, 0.8899)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.219061
```

```
                 Kappa : 0.3647
 Mcnemar's Test P-Value : 0.002512

            Sensitivity : 0.35593
            Specificity : 0.95130
         Pos Pred Value : 0.58333
         Neg Pred Value : 0.88520
             Prevalence : 0.16076
         Detection Rate : 0.05722
   Detection Prevalence : 0.09809
      Balanced Accuracy : 0.65362

       'Positive' Class : X1

[1] "F1 score = 0.442105263157895"
Area under the curve: 0.8259
```



#Flexible Discriminant Analysis

#Train FDA over number of components from 1 to 30 and a degree of 1 and 2.
set.seed(1247)
fdaFit <- train(x = trainHRdataDummyFullRankLowCorr[, -1], y = trainHRdataDummyFullRankLowCorr$Attrition, method = "earth", metric = "ROC", preProc = c("BoxCox", "center", "scale"), tuneGrid = expand.grid(.nprune = 1:30, .degree = 1:2), trControl = ctrl)
fdaFit

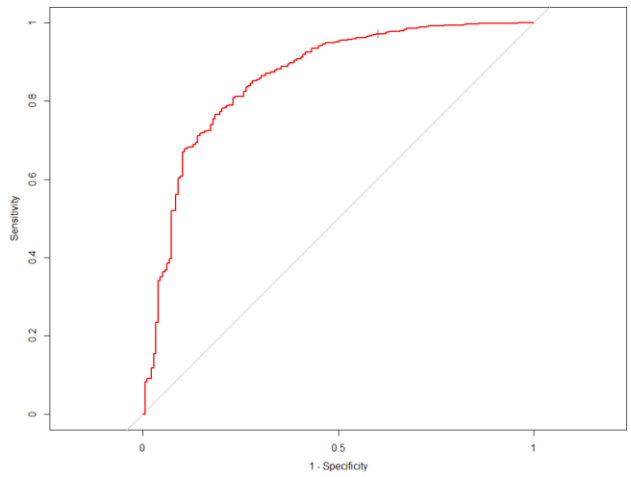
Outcome:

```
Multivariate Adaptive Regression Spline
```

```
1103 samples
  39 predictor
   2 classes: 'X0', 'X1'

Pre-processing: Box-Cox transformation (13), centered (39), scaled (39)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 992, 994, 992, 992, 993, 993, ...
Resampling results across tuning parameters:

  degree  nprune  ROC        Sens       Spec
  1       25      0.8083973  0.9571739  0.3620915
  1       26      0.8084876  0.9567438  0.3632026
  1       27      0.8080336  0.9565288  0.3620915
  1       28      0.8083617  0.9563137  0.3632026
  1       29      0.8095219  0.9569565  0.3643137
  1       30      0.8094213  0.9571716  0.3654248
  2       25      0.7510179  0.9437821  0.3062745
  2       26      0.7494129  0.9437868  0.3051634
  2       27      0.7493593  0.9437868  0.3041830
  2       28      0.7482044  0.9442193  0.3053595
  2       29      0.7479579  0.9442193  0.3065359
  2       30      0.7480327  0.9444343  0.3053595

ROC was used to select the optimal model using  the largest value.
The final values used for the model were nprune = 29 and degree = 1.
```

#results of trained model
trainresults$fdaFit <- trainresultsfunction(fdaFit, trainHRdataDummyFullRankLowCorr)
Outcome:

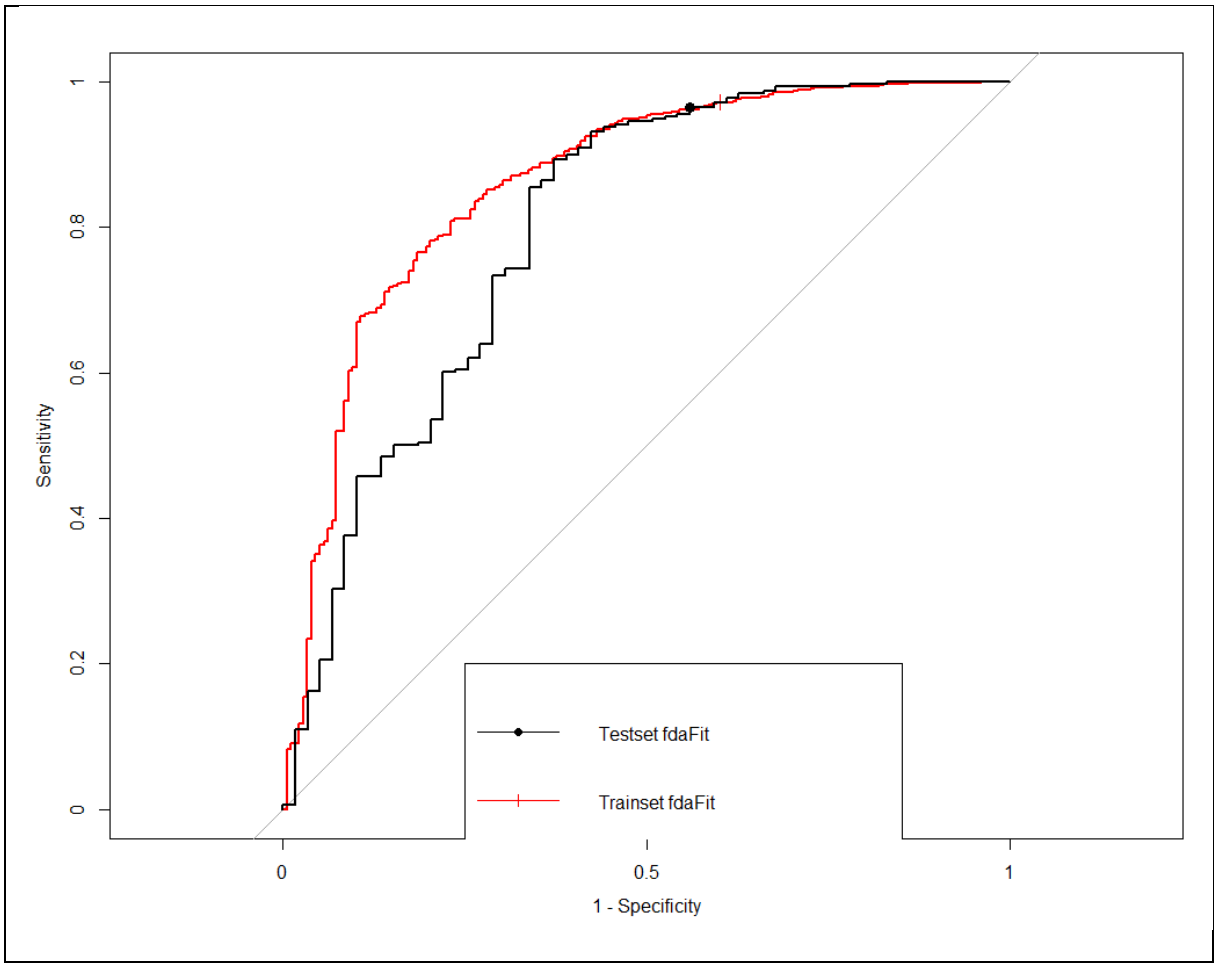
```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 899 107
        X1  26  71

               Accuracy : 0.8794
                 95% CI : (0.8587, 0.8981)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : 8.126e-05

                  Kappa : 0.4542
 Mcnemar's Test P-Value : 4.009e-12

            Sensitivity : 0.39888
            Specificity : 0.97189
         Pos Pred Value : 0.73196
         Neg Pred Value : 0.89364
             Prevalence : 0.16138
         Detection Rate : 0.06437
   Detection Prevalence : 0.08794
      Balanced Accuracy : 0.68538

       'Positive' Class : X1

Area under the curve: 0.8594
```



#results of trained model on test set
testresults$fdaFit <- testresultsfunction(fdaFit, testHRdataDummyFullRankLowCorr, "fdaFit")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 297  33
        X1  11  26

               Accuracy : 0.8801
                 95% CI : (0.8424, 0.9115)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.016982

                  Kappa : 0.4768
 Mcnemar's Test P-Value : 0.001546

            Sensitivity : 0.44068
            Specificity : 0.96429
         Pos Pred Value : 0.70270
         Neg Pred Value : 0.90000
             Prevalence : 0.16076
         Detection Rate : 0.07084
   Detection Prevalence : 0.10082
      Balanced Accuracy : 0.70248

       'Positive' Class : X1

[1] "F1 score = 0.541666666666667"
Area under the curve: 0.7972
```



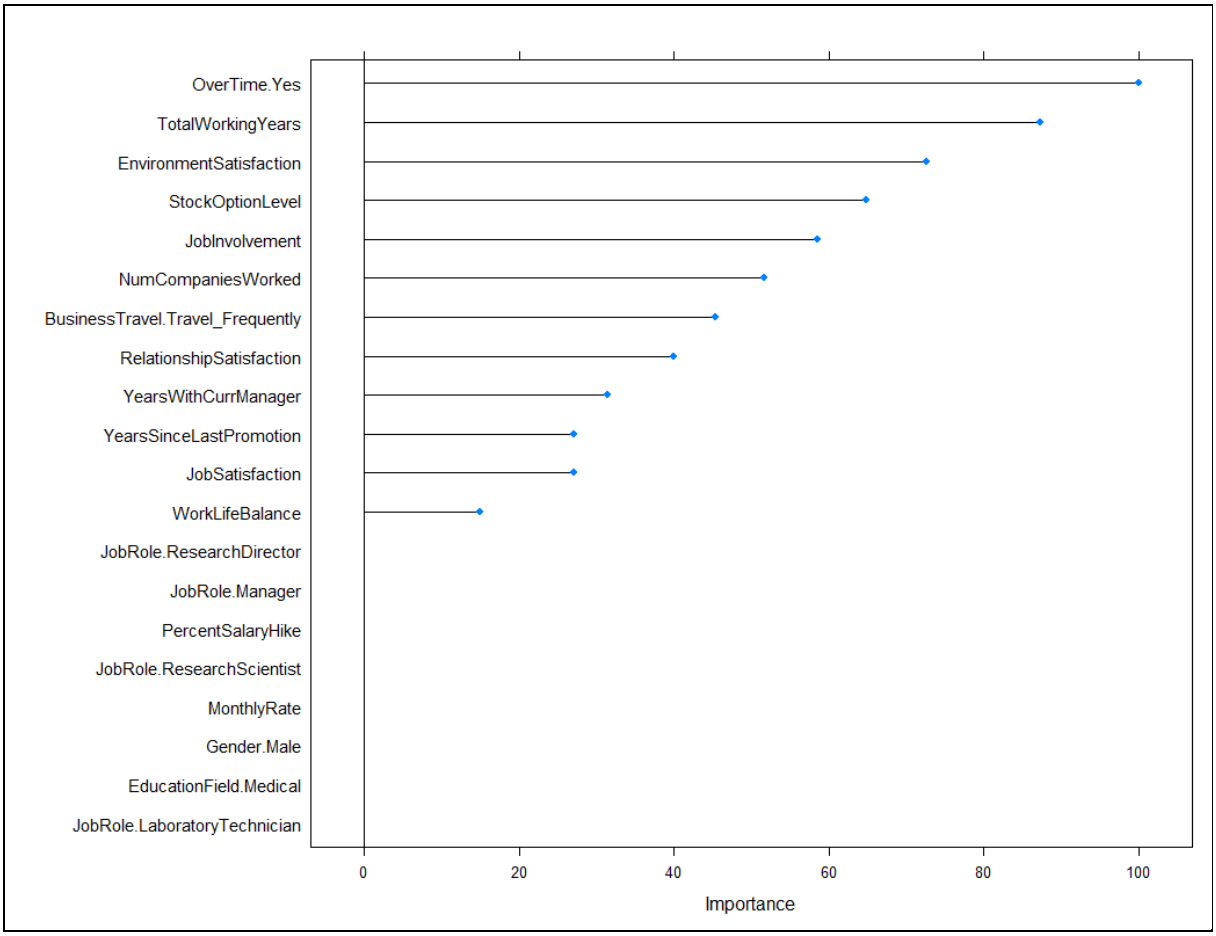
#check the variable importance
fdaimp <- varImp(fdaFit, scale = FALSE)
fdaimp

Outcome:

```
earth variable importance

  only 20 most important variables shown (out of 39)

                                Overall
OverTime.Yes                     100.00
TotalWorkingYears                 87.29
EnvironmentSatisfaction           72.56
StockOptionLevel                  64.85
JobInvolvement                    58.48
NumCompaniesWorked                51.62
BusinessTravel.Travel_Frequently  45.41
RelationshipSatisfaction          39.99
YearsWithCurrManager              31.36
YearsSinceLastPromotion           27.07
JobSatisfaction                   27.07
WorkLifeBalance                   14.86
MaritalStatus.Married              0.00
DailyRate                          0.00
PercentSalaryHike                  0.00
Education                          0.00
Gender.Male                        0.00
MonthlyRate                        0.00
EducationField.Marketing           0.00
```

```
EducationField.LifeSciences          0.00
```

#plot the variable importance
plot(fdaimp , top = 20, scales = list(y = list(cex = .95)))
Outcome:



Support Vector Machines

#create tuning parameters
set.seed(1247)
sigmaRangeReduced <- sigest(as.matrix(trainHRdataDummyFullRankLowCorr[, -1]))
svmRGridReduced <- expand.grid(.sigma = sigmaRangeReduced[1], .C = 2^(seq(-4, 4)))

#train the model
set.seed(1247)
svmFit <- train(x = trainHRdataDummyFullRankLowCorr[, -1], y = trainHRdataDummyFullRankLowCorr$Attrition, method = "svmRadial", metric = "ROC", preProc = c("BoxCox", "center", "scale"), tuneGrid = svmRGridReduced, fit = FALSE, trControl = ctrl)
svmFit
Outcome:

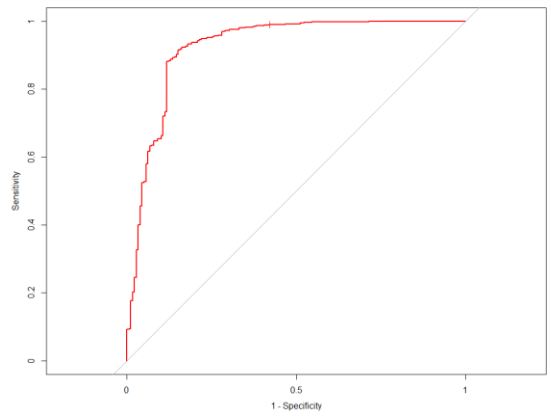
```
Support Vector Machines with Radial Basis Function Kernel

1103 samples
  39 predictor
   2 classes: 'X0', 'X1'

Pre-processing: Box-Cox transformation (13), centered (39), scaled (39)
```

```
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 992, 992, 992, 993, 993, 993, ...
Resampling results across tuning parameters:

  C        ROC        Sens       Spec
   0.0625  0.8367184  0.9628144  0.4480392
   0.1250  0.8366713  0.9641094  0.4358824
   0.2500  0.8366593  0.9636816  0.4401307
   0.5000  0.8367560  0.9625993  0.4405229
   1.0000  0.8369186  0.9641117  0.4427451
   2.0000  0.8370395  0.9703763  0.4058170
   4.0000  0.8342243  0.9764236  0.3707843
   8.0000  0.8234219  0.9749112  0.3392157
  16.0000  0.8088556  0.9777349  0.2840523

Tuning parameter 'sigma' was held constant at a value of 0.009475476
ROC was used to select the optimal model using  the largest value.
The final values used for the model were sigma = 0.009475476 and C = 2.
```

#results of trained model
trainresults$svmFit <- trainresultsfunction(svmFit, trainHRdataDummyFullRankLowCorr)

Outcome:

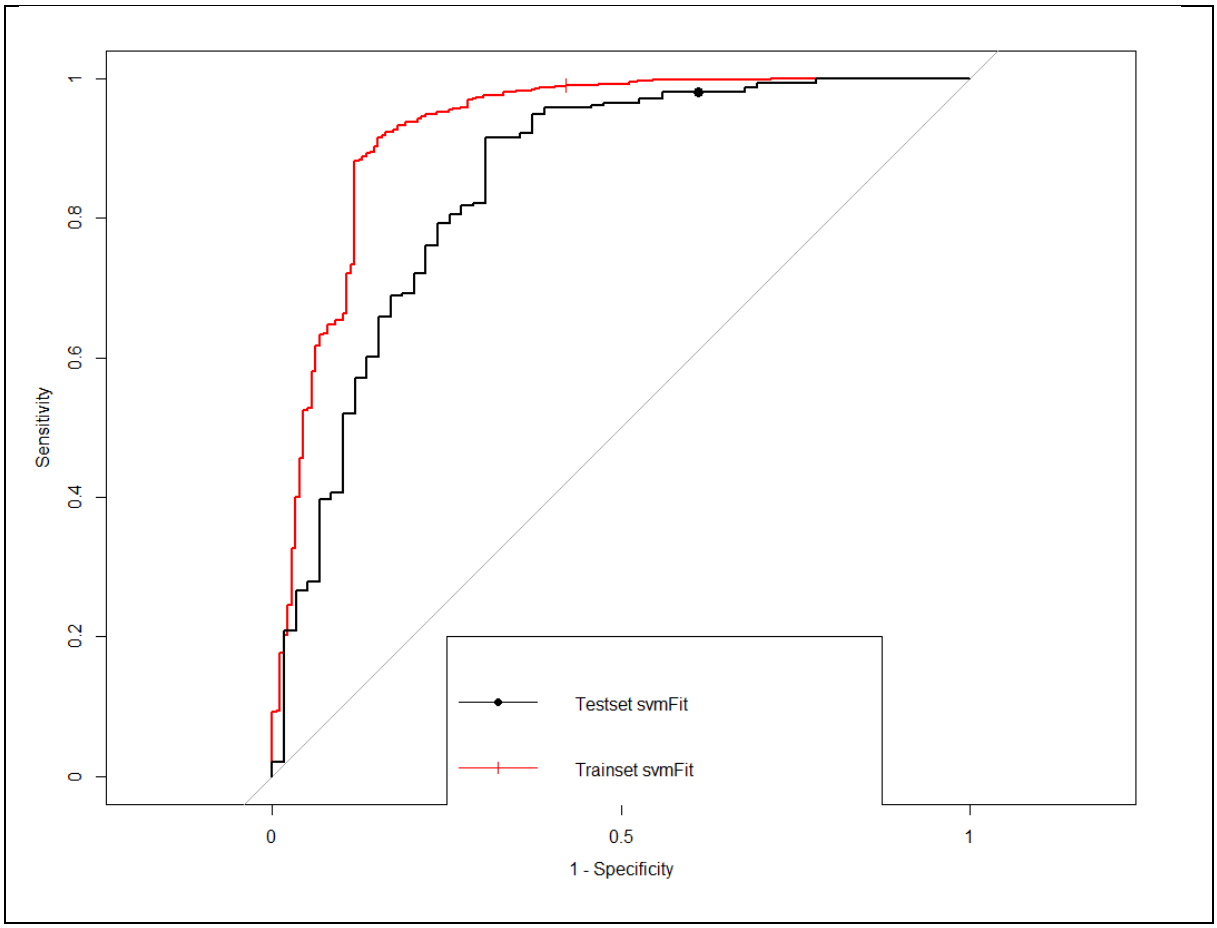
```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 916  75
        X1   9 103

               Accuracy : 0.9238
                 95% CI : (0.9066, 0.9388)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6691
 Mcnemar's Test P-Value : 1.321e-12

            Sensitivity : 0.57865
            Specificity : 0.99027
         Pos Pred Value : 0.91964
         Neg Pred Value : 0.92432
             Prevalence : 0.16138
         Detection Rate : 0.09338
   Detection Prevalence : 0.10154
      Balanced Accuracy : 0.78446

       'Positive' Class : X1

Area under the curve: 0.9243
```



#results of trained model on test set
testresults$svmFit <- testresultsfunction(svmFit, testHRdataDummyFullRankLowCorr, "svmFit")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 302  36
        X1   6  23

               Accuracy : 0.8856
                 95% CI : (0.8485, 0.9163)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.007567

                  Kappa : 0.4662
 Mcnemar's Test P-Value : 7.648e-06

            Sensitivity : 0.38983
            Specificity : 0.98052
         Pos Pred Value : 0.79310
         Neg Pred Value : 0.89349
             Prevalence : 0.16076
         Detection Rate : 0.06267
   Detection Prevalence : 0.07902
      Balanced Accuracy : 0.68517

       'Positive' Class : X1

[1] "F1 score = 0.522727272727273"
Area under the curve: 0.8489
```



#K-Nearest Neighbors

#Train model
set.seed(1247)
knnFit <- train(x = trainHRdataDummy[, -1], y = trainHRdataDummy$Attrition, method = "knn", metric = "ROC", preProc = c("BoxCox", "center", "scale"), tuneGrid = data.frame(.k = c(4*(0:5)+1, 20*(1:5)+1, 50*(2:9)+1)), trControl = ctrl)
knnFit

Outcome:

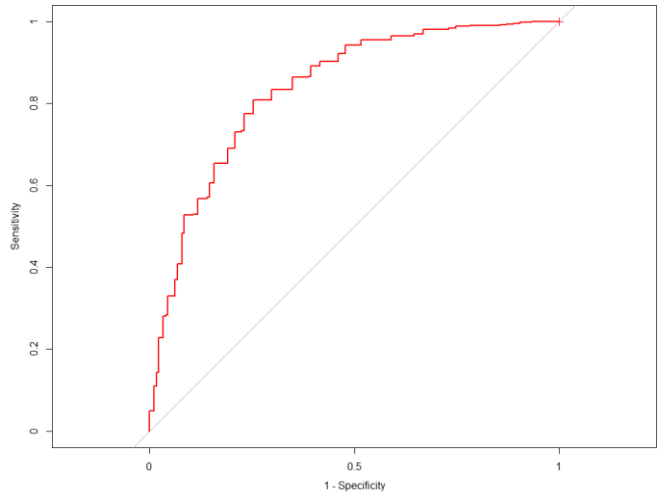
```
k-Nearest Neighbors

1103 samples
  51 predictor
   2 classes: 'X0', 'X1'

Pre-processing: Box-Cox transformation (14), centered (51), scaled (51)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 992, 992, 992, 993, 994, 993, ...
Resampling results across tuning parameters:

  k    ROC        Sens       Spec
    1  0.5547712  0.9115685  0.197973856
    5  0.6908449  0.9716830  0.156797386
    9  0.7351462  0.9872347  0.097647059
   13  0.7512537  0.9956826  0.075359477
   17  0.7557581  0.9971926  0.059346405
   21  0.7592417  0.9991351  0.045882353
   41  0.7734339  1.0000000  0.003333333
```

```
   61  0.7879072  1.0000000  0.000000000
   81  0.7968381  1.0000000  0.000000000
  101  0.8053464  1.0000000  0.000000000
  151  0.8137418  1.0000000  0.000000000
  201  0.8117384  1.0000000  0.000000000
  251  0.8099499  1.0000000  0.000000000
  301  0.8085165  1.0000000  0.000000000
  351  0.8008097  1.0000000  0.000000000
  401  0.8016349  1.0000000  0.000000000
  451  0.8035886  1.0000000  0.000000000

ROC was used to select the optimal model using  the largest value.
The final value used for the model was k = 151.
```

#results of trained model
trainresults$knnFit <- trainresultsfunction(knnFit, trainHRdataDummy)

Outcome:

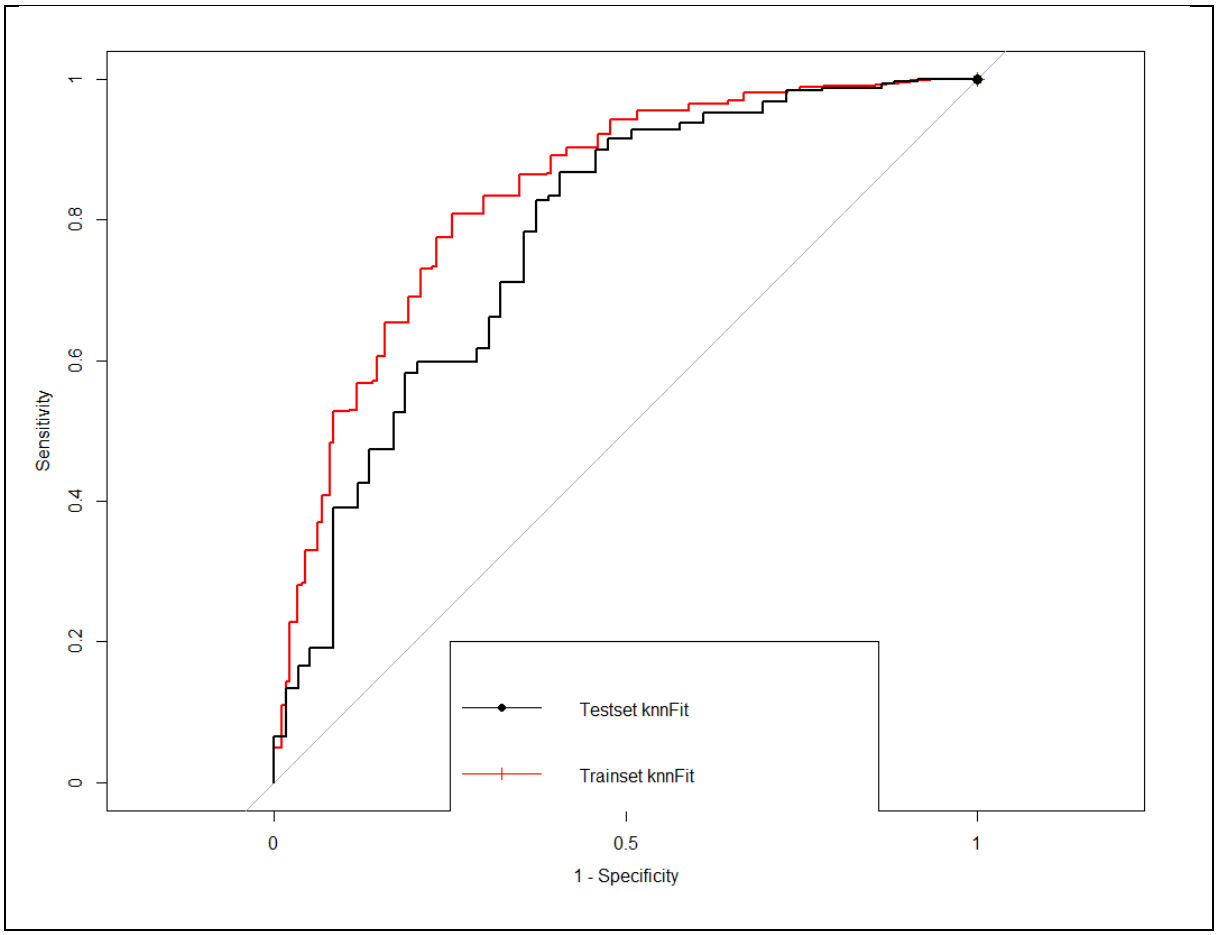
```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 925 178
        X1   0   0

               Accuracy : 0.8386
                 95% CI : (0.8156, 0.8599)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : 0.52

                  Kappa : 0
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.0000
            Specificity : 1.0000
         Pos Pred Value :    NaN
         Neg Pred Value : 0.8386
             Prevalence : 0.1614
         Detection Rate : 0.0000
   Detection Prevalence : 0.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : X1

Area under the curve: 0.8272
```



#results of trained model on test set
testresults$knnFit <- testresultsfunction(knnFit, testHRdataDummy, "knnFit")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 308  59
        X1   0   0

               Accuracy : 0.8392
                 95% CI : (0.7976, 0.8753)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.5347

                  Kappa : 0
 Mcnemar's Test P-Value : 4.321e-14

            Sensitivity : 0.0000
            Specificity : 1.0000
         Pos Pred Value :    NaN
         Neg Pred Value : 0.8392
             Prevalence : 0.1608
         Detection Rate : 0.0000
   Detection Prevalence : 0.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : X1

[1] "F1 score = NaN"
Area under the curve: 0.7644
```



#Naive Bayes

```
#create Naive Bayes grid
nbgrid <- expand.grid(.usekernel = c(TRUE, FALSE), .fL = seq(0, 2, length.out = 4), .adjust = 1)

#train model
set.seed(1247)
nbFit <- train(x = trainHRdataDummyFullRankLowCorr[ , -1], y = trainHRdataDummyFullRankLowCorr$Attrition, method = "nb", metric = "ROC", preProc = c("BoxCox", "center", "scale"), tuneGrid = nbgrid, trControl = ctrl)
nbFit
```

Outcome:

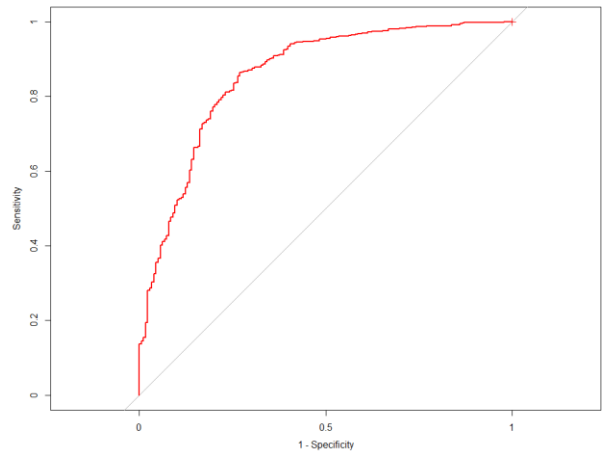
```
Naive Bayes

1103 samples
  39 predictor
   2 classes: 'X0', 'X1'

Pre-processing: Box-Cox transformation (13), centered (39), scaled (39)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 992, 992, 992, 993, 994, 993, ...
Resampling results across tuning parameters:

  usekernel  fL   ROC        Sens       Spec
  FALSE      0.0  0.7401731  0.6366604  0.766739288
  FALSE      0.5  0.7401731  0.6366604  0.766739288
  FALSE      1.0  0.7401731  0.6366604  0.766739288
  FALSE      1.5  0.7401731  0.6366604  0.766739288
```

```
  FALSE      2.0  0.7401731  0.6366604  0.766739288
   TRUE      0.0  0.8078821  0.9997849  0.001111111
   TRUE      0.5  0.8078821  0.9997849  0.001111111
   TRUE      1.0  0.8078821  0.9997849  0.001111111
   TRUE      1.5  0.8078821  0.9997849  0.001111111
   TRUE      2.0  0.8078821  0.9997849  0.001111111

Tuning parameter 'adjust' was held constant at a value of 1
ROC was used to select the optimal model using  the largest value.
The final values used for the model were fL = 0, usekernel = TRUE and adj
ust = 1.
```

#Note model were kernel density estimate is used outperformed normal density

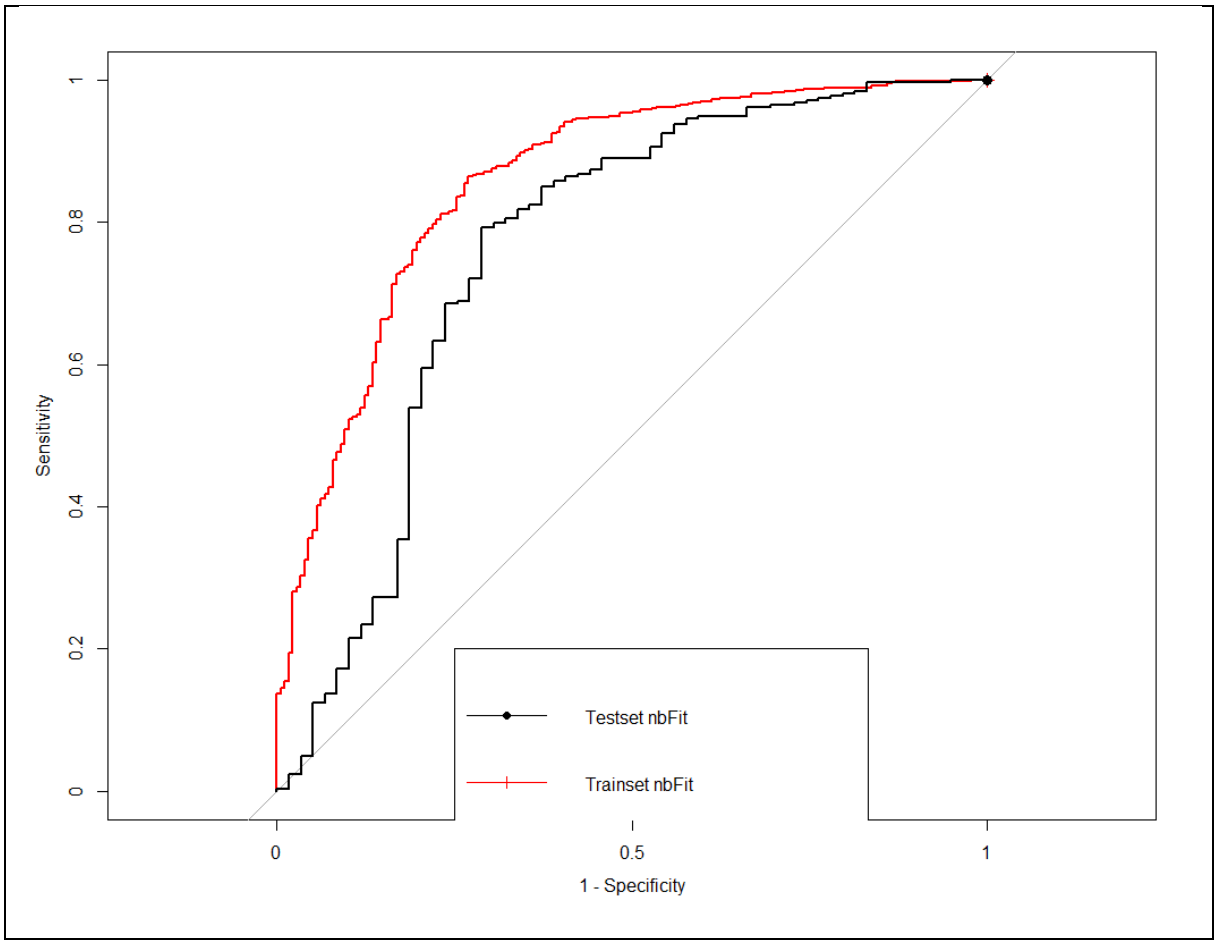
#results of trained model
trainresults$nbFit <- trainresultsfunction(nbFit, trainHRdataDummyFullRankLowCorr)
Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 925 178
        X1   0   0

               Accuracy : 0.8386
                 95% CI : (0.8156, 0.8599)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : 0.52

                  Kappa : 0
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.0000
            Specificity : 1.0000
         Pos Pred Value :    NaN
         Neg Pred Value : 0.8386
             Prevalence : 0.1614
         Detection Rate : 0.0000
   Detection Prevalence : 0.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : X1

Area under the curve: 0.857
```



#results of trained model on test set
testresults$nbFit <- testresultsfunction(nbFit, testHRdataDummyFullRankLowCorr, "nbFit")
Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 308  59
        X1   0   0

               Accuracy : 0.8392
                 95% CI : (0.7976, 0.8753)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.5347

                  Kappa : 0
 Mcnemar's Test P-Value : 4.321e-14

            Sensitivity : 0.0000
            Specificity : 1.0000
         Pos Pred Value :    NaN
         Neg Pred Value : 0.8392
             Prevalence : 0.1608
         Detection Rate : 0.0000
   Detection Prevalence : 0.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : X1

[1] "F1 score = NaN"
Area under the curve: 0.7595
```



Classification Trees and Rule-Based Models

#Classification Trees

```
#------------------------------
#rpart non dummy train set
#------------------------------

set.seed(1247)
rpartFit <- train(x = trainHRdata[, -1],
                  y = trainHRdata$Attrition,
                  method = "rpart",
                  tuneLength = 30,
                  metric = "ROC",
                  trControl = ctrl)
rpartFit
```

Outcome:

```
CART

1103 samples
  30 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
```

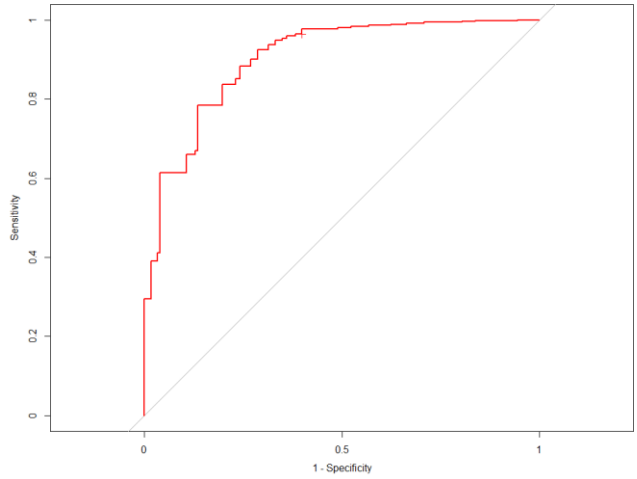
```
Summary of sample sizes: 992, 992, 992, 993, 994, 993, ...
Resampling results across tuning parameters:

  cp            ROC        Sens       Spec
  0.0000000000  0.7230258  0.9227887  0.3174510
  0.0009040424  0.7234989  0.9234409  0.3174510
  0.0018080847  0.7226876  0.9238733  0.3151634
  0.0027121271  0.7202946  0.9260285  0.3128758
  0.0036161694  0.7203436  0.9286232  0.3105882
  0.0045202118  0.7148741  0.9301286  0.3050327
  0.0054242542  0.7147050  0.9303460  0.3050327
  0.0063282965  0.7144686  0.9379383  0.2994771
  0.0072323389  0.7144686  0.9379383  0.2994771
  0.0081363812  0.7091616  0.9388032  0.2916993
  0.0090404236  0.7044311  0.9411875  0.2861438
  0.0099444660  0.6967399  0.9420524  0.2783660
  0.0108485083  0.6888061  0.9466059  0.2715686
  0.0117525507  0.6888061  0.9466059  0.2715686
  0.0126565931  0.6872033  0.9522137  0.2659477
  0.0135606354  0.6872033  0.9522137  0.2659477
  0.0144646778  0.6865412  0.9526461  0.2624183
  0.0153687201  0.6786785  0.9537260  0.2545752
  0.0162727625  0.6777687  0.9545979  0.2533333
  0.0171768049  0.6737032  0.9550281  0.2533333
  0.0180808472  0.6737032  0.9550281  0.2533333
  0.0189848896  0.6614408  0.9587027  0.2433333
  0.0198889319  0.6614408  0.9587027  0.2433333
  0.0207929743  0.6565543  0.9597779  0.2366667
  0.0216970167  0.6565543  0.9597779  0.2366667
  0.0226010590  0.6416361  0.9600070  0.2119608
  0.0235051014  0.6384054  0.9606592  0.2097386
  0.0244091437  0.6384054  0.9606592  0.2097386
  0.0253131861  0.6173281  0.9651893  0.1670588
  0.0262172285  0.6173281  0.9651893  0.1670588

ROC was used to select the optimal model using  the largest value.
The final value used for the model was cp = 0.0009040424.
```

#results of trained model
trainresults$rpartFit <- trainresultsfunction(rpartFit, trainHRdata)

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 892  71
        X1  33 107

               Accuracy : 0.9057
                 95% CI : (0.8869, 0.9223)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : 6.716e-11

                  Kappa : 0.6188
 Mcnemar's Test P-Value : 0.0002855

            Sensitivity : 0.60112
            Specificity : 0.96432
         Pos Pred Value : 0.76429
         Neg Pred Value : 0.92627
```

```
             Prevalence : 0.16138
         Detection Rate : 0.09701
   Detection Prevalence : 0.12693
      Balanced Accuracy : 0.78272

       'Positive' Class : X1

Area under the curve: 0.8845
```

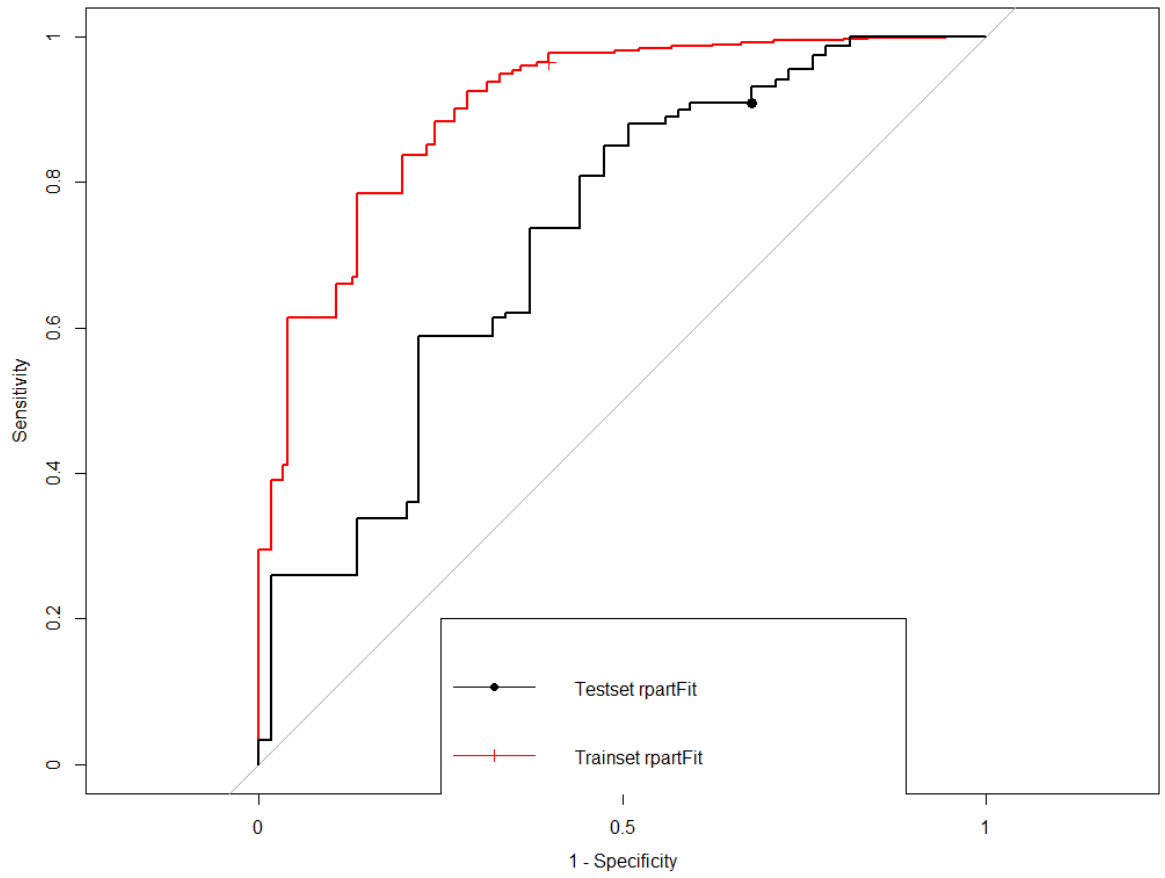


#results of trained model on test set
testresults$rpartFit <- testresultsfunction(rpartFit, testHRdata)

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 280  40
        X1  28  19

               Accuracy : 0.8147
                 95% CI : (0.7711, 0.8531)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.9096

                  Kappa : 0.2518
 Mcnemar's Test P-Value : 0.1822

            Sensitivity : 0.32203
            Specificity : 0.90909
         Pos Pred Value : 0.40426
         Neg Pred Value : 0.87500
             Prevalence : 0.16076
         Detection Rate : 0.05177
   Detection Prevalence : 0.12807
      Balanced Accuracy : 0.61556

       'Positive' Class : X1
```

[1] "F1 score = 0.358490566037736"
Area under the curve: 0.6901




```
#------------------------------
#rpart dummy train set
#------------------------------

#train model
set.seed(1247)
rpartFitDummy <- train(x = trainHRdataDummy[, -1],
                       y = trainHRdataDummy$Attrition,
                       method = "rpart",
                       tuneLength = 30,
                       metric = "ROC",
                       trControl = ctrl)
rpartFitDummy
```

Outcome:

```
CART

1103 samples
  51 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 992, 992, 992, 993, 994, 993, ...
```

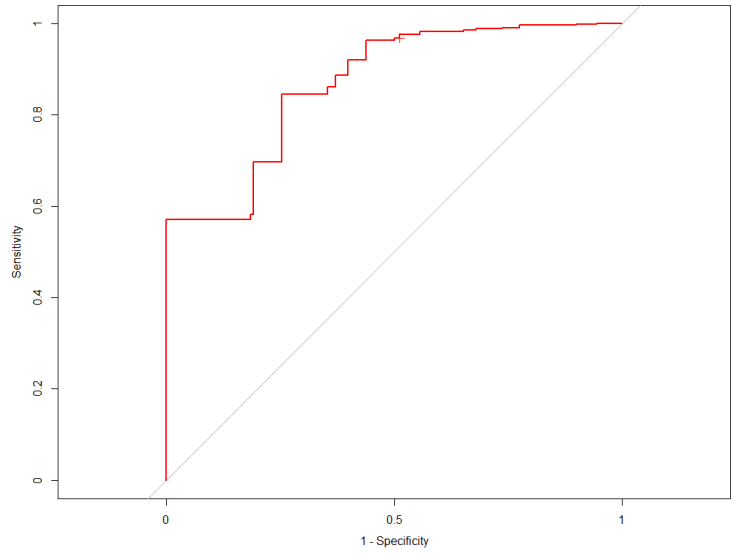
```
Resampling results across tuning parameters:

  cp            ROC        Sens       Spec
  0.0000000000  0.6957270  0.9212903  0.2964706
  0.0009040424  0.6957270  0.9212903  0.2964706
  0.0018080847  0.6999908  0.9221552  0.2942484
  0.0027121271  0.7023316  0.9238850  0.2930719
  0.0036161694  0.7041678  0.9290767  0.2908497
  0.0045202118  0.7021433  0.9318887  0.2828105
  0.0054242542  0.7027168  0.9325339  0.2828105
  0.0063282965  0.7047712  0.9388149  0.2748366
  0.0072323389  0.7047712  0.9388149  0.2748366
  0.0081363812  0.6989168  0.9401075  0.2681699
  0.0090404236  0.6980900  0.9420594  0.2636601
  0.0099444660  0.6968498  0.9431417  0.2614379
  0.0108485083  0.6945832  0.9474731  0.2557516
  0.0117525507  0.6945832  0.9474731  0.2557516
  0.0126565931  0.6935093  0.9504932  0.2559477
  0.0135606354  0.6935093  0.9504932  0.2559477
  0.0144646778  0.6926781  0.9507106  0.2524183
  0.0153687201  0.6821177  0.9520079  0.2467974
  0.0162727625  0.6755023  0.9533053  0.2343791
  0.0171768049  0.6726681  0.9537354  0.2343791
  0.0180808472  0.6674113  0.9541655  0.2277124
  0.0189848896  0.6556316  0.9578448  0.2186928
  0.0198889319  0.6556316  0.9578448  0.2186928
  0.0207929743  0.6531749  0.9584923  0.2120261
  0.0216970167  0.6528882  0.9591374  0.2098039
  0.0226010590  0.6508223  0.9610846  0.1950980
  0.0235051014  0.6475916  0.9617368  0.1928758
  0.0244091437  0.6475916  0.9617368  0.1928758
  0.0253131861  0.6264301  0.9692917  0.1286275
  0.0262172285  0.6264301  0.9692917  0.1286275

ROC was used to select the optimal model using  the largest value.
The final value used for the model was cp = 0.007232339.
```

#results of trained model
trainresults$rpartFitDummy <- trainresultsfunction(rpartFitDummy, trainHRdataDummy)

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 896  91
        X1  29  87

               Accuracy : 0.8912
                 95% CI : (0.8713, 0.909)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : 3.954e-07

                  Kappa : 0.5323
 Mcnemar's Test P-Value : 2.569e-08

            Sensitivity : 0.48876
            Specificity : 0.96865
         Pos Pred Value : 0.75000
         Neg Pred Value : 0.90780
             Prevalence : 0.16138
```

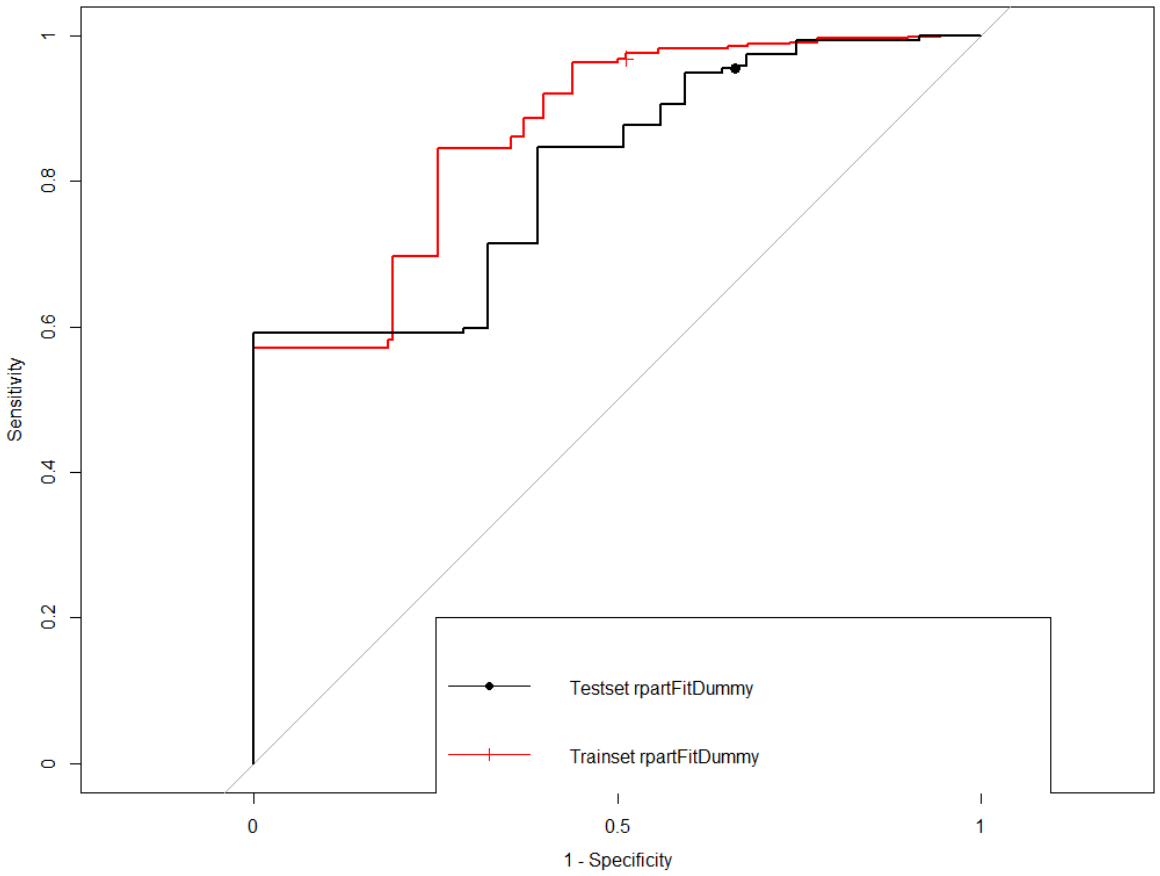
```
          Detection Rate : 0.07888
    Detection Prevalence : 0.10517
       Balanced Accuracy : 0.72871

        'Positive' Class : X1

Area under the curve: 0.7999
```



#results of trained model on test set
testresults$rpartFitDummy <- testresultsfunction(rpartFitDummy, testHRdataDummy)
Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 294  39
        X1  14  20

               Accuracy : 0.8556
                 95% CI : (0.8154, 0.8899)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.2190611

                  Kappa : 0.3542
 Mcnemar's Test P-Value : 0.0009784

            Sensitivity : 0.33898
            Specificity : 0.95455
         Pos Pred Value : 0.58824
         Neg Pred Value : 0.88288
             Prevalence : 0.16076
         Detection Rate : 0.05450
   Detection Prevalence : 0.09264
      Balanced Accuracy : 0.64676
```

```
      'Positive' Class : X1

[1] "F1 score = 0.43010752688172"
Area under the curve: 0.7145
```




```
#------------------------------
#C4.5/J48 non dummy trainset
#------------------------------

set.seed(1247)
j48Fit <- train(x = trainHRdata[, -1],
                y = trainHRdata$Attrition,
                method = "J48",
                metric = "ROC",
                trControl = ctrl)
j48Fit
```

Outcome:

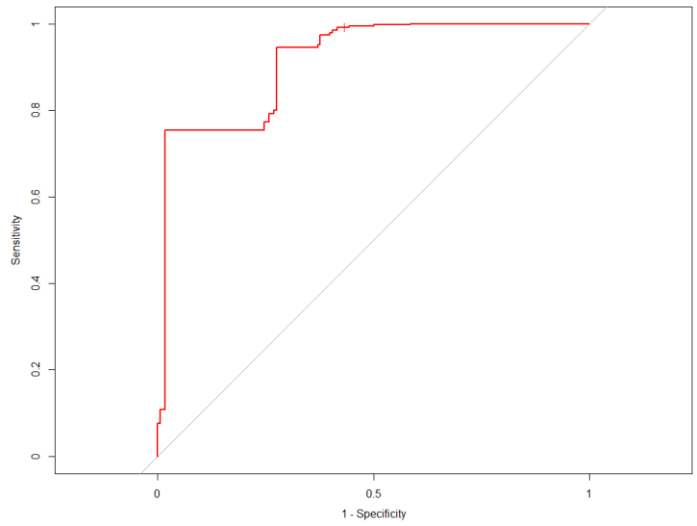
```
C4.5-like Trees

1103 samples
  30 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 992, 992, 992, 993, 994, 993, ...
Resampling results across tuning parameters:
```

```
  C      M  ROC        Sens       Spec
  0.010  1  0.5355405  0.9895979  0.03156863
  0.010  2  0.5199565  0.9924147  0.02124183
  0.010  3  0.5199419  0.9900257  0.02241830
  0.255  1  0.5658017  0.9319121  0.29562092
  0.255  2  0.5905983  0.9301753  0.28418301
  0.255  3  0.6147518  0.9314727  0.30104575
  0.500  1  0.5659947  0.8932071  0.33261438
  0.500  2  0.5953606  0.8999042  0.31222222
  0.500  3  0.6108905  0.9100818  0.32026144

ROC was used to select the optimal model using  the largest value.
The final values used for the model were C = 0.255 and M = 3.
```

#results of trained model
trainresults$j48Fit <- trainresultsfunction(j48Fit, trainHRdata)

Outcome:

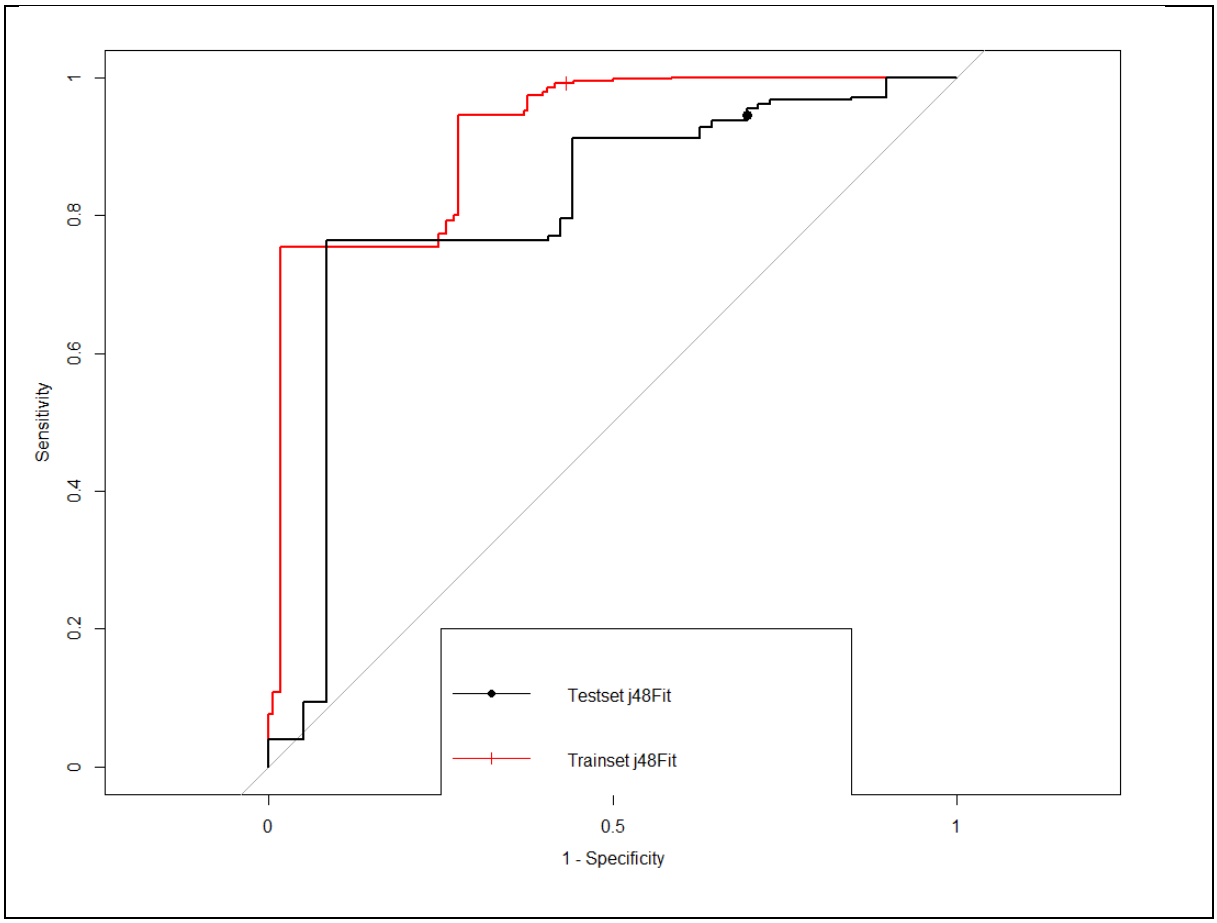
```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 917  77
        X1   8 101

               Accuracy : 0.9229
                 95% CI : (0.9056, 0.938)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6625
 Mcnemar's Test P-Value : 1.636e-13

            Sensitivity : 0.56742
            Specificity : 0.99135
         Pos Pred Value : 0.92661
         Neg Pred Value : 0.92254
             Prevalence : 0.16138
         Detection Rate : 0.09157
   Detection Prevalence : 0.09882
      Balanced Accuracy : 0.77938

       'Positive' Class : X1

Area under the curve: 0.8327
```



#results of trained model on test set
testresults$j48Fit <- testresultsfunction(j48Fit, testHRdata, "j48Fit")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 291  41
        X1  17  18

               Accuracy : 0.842
                 95% CI : (0.8005, 0.8778)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.478071

                  Kappa : 0.2991
 Mcnemar's Test P-Value : 0.002527

            Sensitivity : 0.30508
            Specificity : 0.94481
         Pos Pred Value : 0.51429
         Neg Pred Value : 0.87651
             Prevalence : 0.16076
         Detection Rate : 0.04905
   Detection Prevalence : 0.09537
      Balanced Accuracy : 0.62494

       'Positive' Class : X1

[1] "F1 score = 0.382978723404255"
Area under the curve: 0.6867
```




```
#------------------------------
#C4.5/J48 dummy trainset
#------------------------------

set.seed(1247)
j48FitDummy <- train(x = trainHRdataDummy[, -1],
                     y = trainHRdataDummy$Attrition,
                     method = "J48",
                     metric = "ROC",
                     trControl = ctrl)
j48FitDummy
```

Outcome:

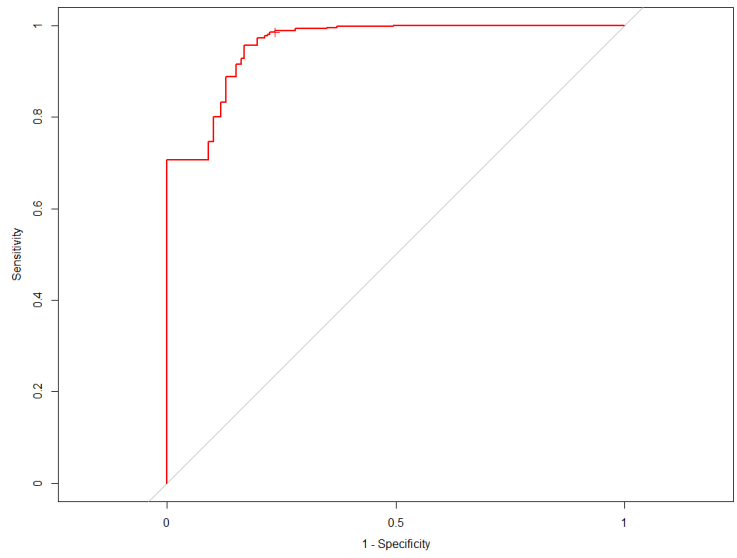
```
C4.5-like Trees

1103 samples
  51 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 992, 992, 992, 993, 994, 993, ...
Resampling results across tuning parameters:

  C      M  ROC        Sens       Spec
  0.010  1  0.5350743  0.9889551  0.03941176
  0.010  2  0.5361838  0.9900421  0.03692810
  0.010  3  0.5332913  0.9898200  0.03464052
  0.255  1  0.5569126  0.9204956  0.30542484
```

```
  0.255  2  0.5994401  0.9239271  0.32352941
  0.255  3  0.6115514  0.9276087  0.29418301
  0.500  1  0.5899274  0.8817765  0.33777778
  0.500  2  0.6163471  0.8875900  0.35032680
  0.500  3  0.6288828  0.9016386  0.33130719

ROC was used to select the optimal model using  the largest value.
The final values used for the model were C = 0.5 and M = 3.
```

#results of trained model
trainresults$j48Fit <- trainresultsfunction(j48FitDummy, trainHRdataDummy)

Outcome:

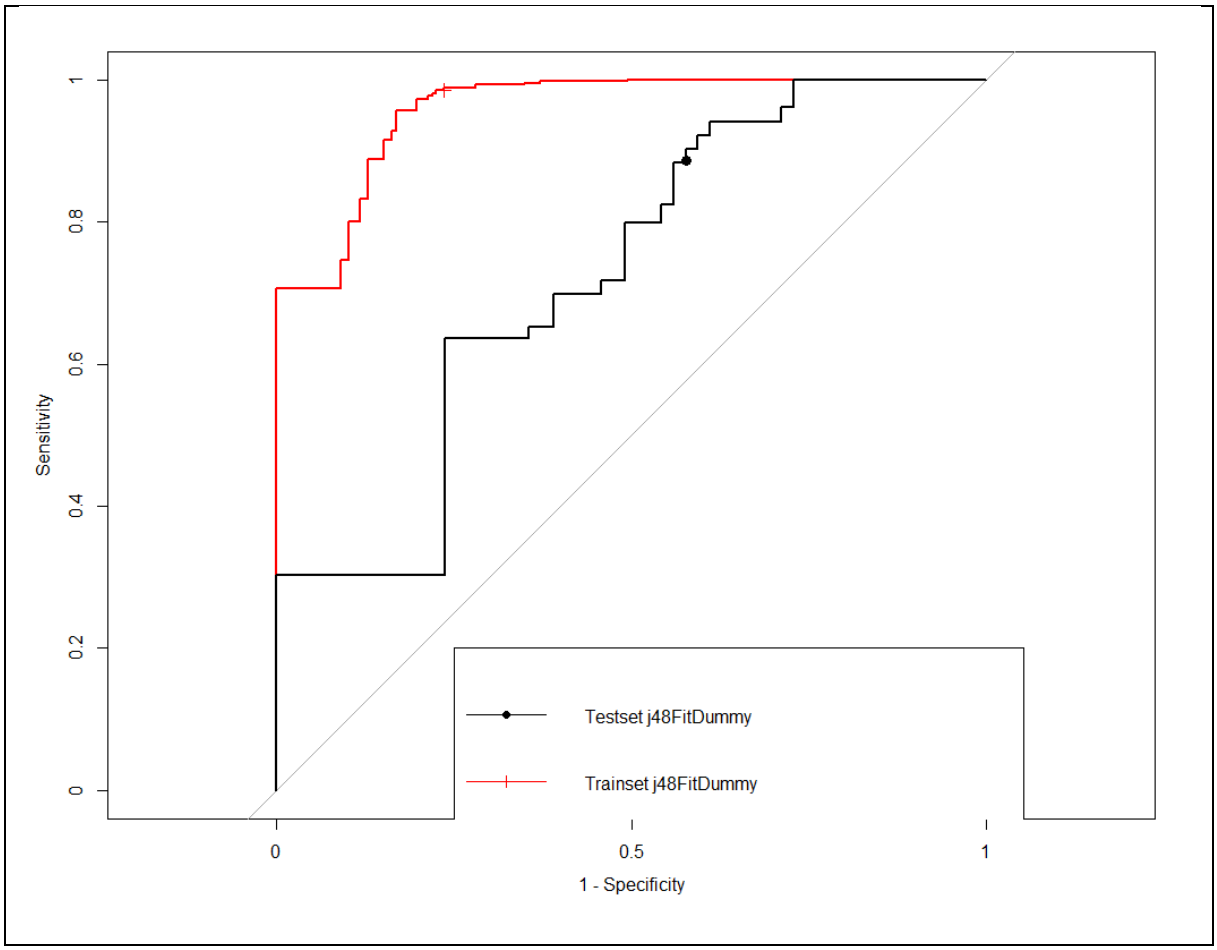
```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 911  42
        X1  14 136

               Accuracy : 0.9492
                 95% CI : (0.9346, 0.9614)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7997
 Mcnemar's Test P-Value : 0.0003085

            Sensitivity : 0.7640
            Specificity : 0.9849
         Pos Pred Value : 0.9067
         Neg Pred Value : 0.9559
             Prevalence : 0.1614
         Detection Rate : 0.1233
   Detection Prevalence : 0.1360
      Balanced Accuracy : 0.8745

       'Positive' Class : X1

Area under the curve: 0.9402
```



#results of trained model on test set
testresults$j48Fit <- testresultsfunction(j48FitDummy, testHRdataDummy, "j48FitDummy")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 273  34
        X1  35  25

               Accuracy : 0.812
                 95% CI : (0.7682, 0.8507)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.9298

                  Kappa : 0.308
 Mcnemar's Test P-Value : 1.0000

            Sensitivity : 0.42373
            Specificity : 0.88636
         Pos Pred Value : 0.41667
         Neg Pred Value : 0.88925
             Prevalence : 0.16076
         Detection Rate : 0.06812
   Detection Prevalence : 0.16349
      Balanced Accuracy : 0.65505

       'Positive' Class : X1

[1] "F1 score = 0.420168067226891"
Area under the curve: 0.6581
```




```
#------------------------------
#Rule based models: PART
#------------------------------

set.seed(1247)
partFit <- train(x = trainHRdata[, -1],
                 y = trainHRdata$Attrition,
                 method = "PART",
                 metric = "ROC",
                 trControl = ctrl)
partFit
```

Outcome:

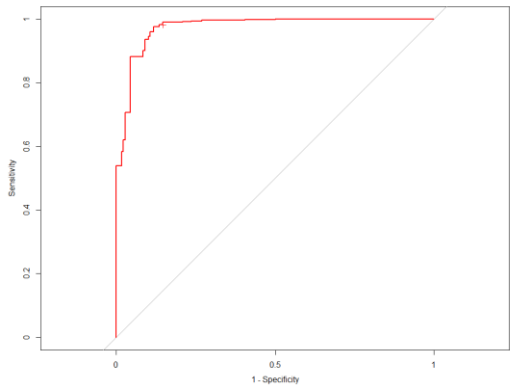
```
Rule-Based Classifier

1103 samples
  30 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 992, 992, 992, 993, 994, 993, ...
Resampling results across tuning parameters:

  threshold  pruned  ROC        Sens       Spec
  0.010      yes     0.6567082  0.8940019  0.3785621
  0.010      no      0.6303250  0.8823726  0.3749673
  0.255      yes     0.6479434  0.8933707  0.3617647
```

```
  0.255      no       0.6303250  0.8823726  0.3749673
  0.500      yes      0.6416703  0.8927092  0.3564052
  0.500      no       0.6303250  0.8823726  0.3749673

ROC was used to select the optimal model using  the largest value.
The final values used for the model were threshold = 0.01 and pruned = ye
s.
```

#results of trained model
trainresults$partFit <- trainresultsfunction(partFit, trainHRdata)

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 909  26
        X1  16 152

               Accuracy : 0.9619
                 95% CI : (0.9489, 0.9724)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8561
 Mcnemar's Test P-Value : 0.1649

            Sensitivity : 0.8539
            Specificity : 0.9827
         Pos Pred Value : 0.9048
         Neg Pred Value : 0.9722
             Prevalence : 0.1614
         Detection Rate : 0.1378
   Detection Prevalence : 0.1523
      Balanced Accuracy : 0.9183

       'Positive' Class : X1

Area under the curve: 0.967
```

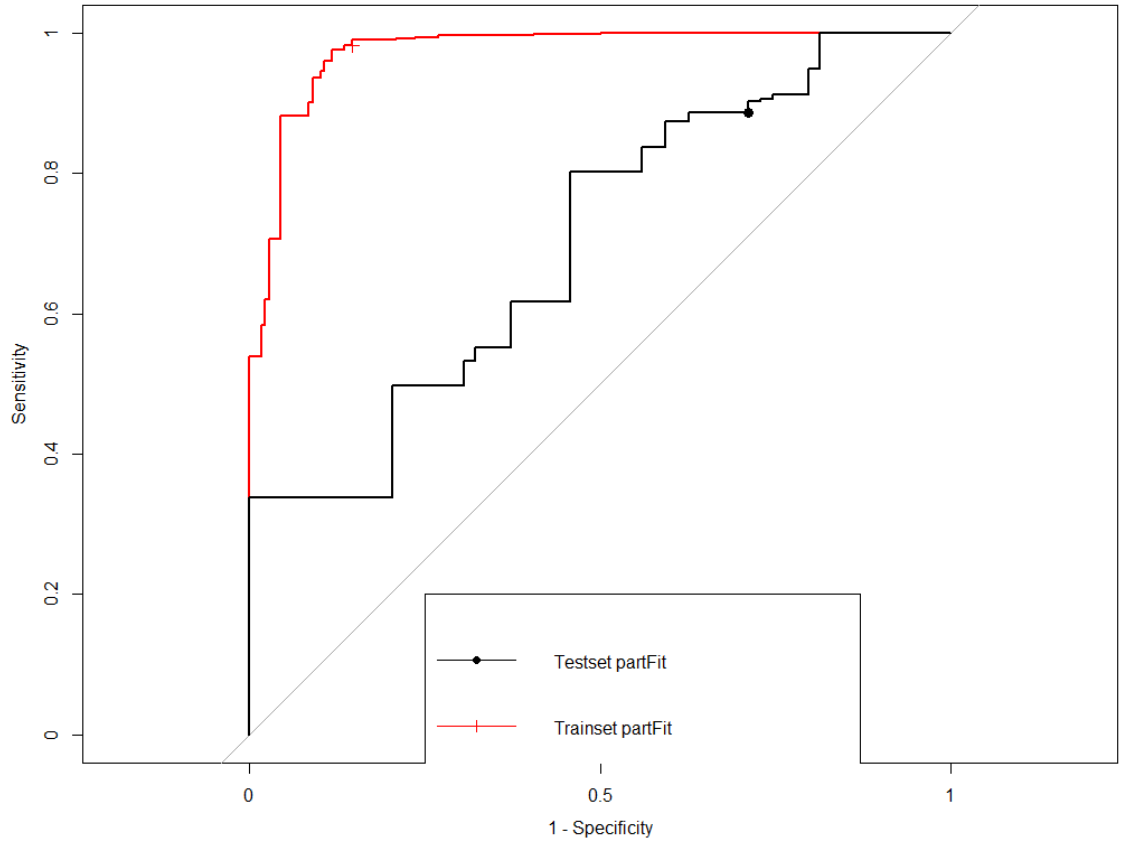


#results of trained model on test set
testresults$partFit <- testresultsfunction(partFit, testHRdata, "partFit")

Outcome:

```
Confusion Matrix and Statistics

          Reference
```

```
Prediction  X0  X1
        X0 273  42
        X1  35  17

               Accuracy : 0.7902
                 95% CI : (0.7449, 0.8307)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.9946

                  Kappa : 0.1833
 Mcnemar's Test P-Value : 0.4941

            Sensitivity : 0.28814
            Specificity : 0.88636
         Pos Pred Value : 0.32692
         Neg Pred Value : 0.86667
             Prevalence : 0.16076
         Detection Rate : 0.04632
   Detection Prevalence : 0.14169
      Balanced Accuracy : 0.58725

       'Positive' Class : X1

Area under the curve: 0.6411
```




```
#-----------------------------
#PART Dummy
#-----------------------------

set.seed(1247)
```

```
partFitDummy <- train(x = trainHRdataDummy[, -1],
           y = trainHRdataDummy$Attrition,
           method = "PART",
           metric = "ROC",
           trControl = ctrl)
partFitDummy
```

Outcome:

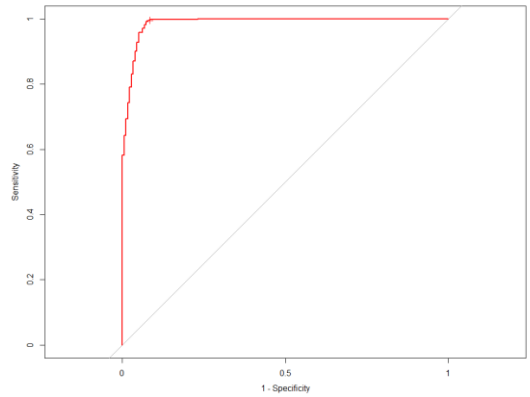
```
Rule-Based Classifier

1103 samples
  51 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 992, 992, 992, 993, 994, 993, ...
Resampling results across tuning parameters:

  threshold  pruned  ROC        Sens       Spec
  0.010      yes     0.6411306  0.8873329  0.3524183
  0.010      no      0.6351989  0.8812763  0.3862745
  0.255      yes     0.6468416  0.8903693  0.3805882
  0.255      no      0.6351989  0.8812763  0.3862745
  0.500      yes     0.6468416  0.8903693  0.3805882
  0.500      no      0.6351989  0.8812763  0.3862745

ROC was used to select the optimal model using  the largest value.
The final values used for the model were threshold = 0.5 and pruned = yes
.
```

```
#results of trained model
trainresults$partFitDummy <- trainresultsfunction(partFitDummy, trainHRdataDummy)
```

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 921  15
        X1   4 163

               Accuracy : 0.9828
                 95% CI : (0.9732, 0.9896)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.9347
 Mcnemar's Test P-Value : 0.02178

            Sensitivity : 0.9157
            Specificity : 0.9957
         Pos Pred Value : 0.9760
         Neg Pred Value : 0.9840
             Prevalence : 0.1614
         Detection Rate : 0.1478
   Detection Prevalence : 0.1514
      Balanced Accuracy : 0.9557

       'Positive' Class : X1
```

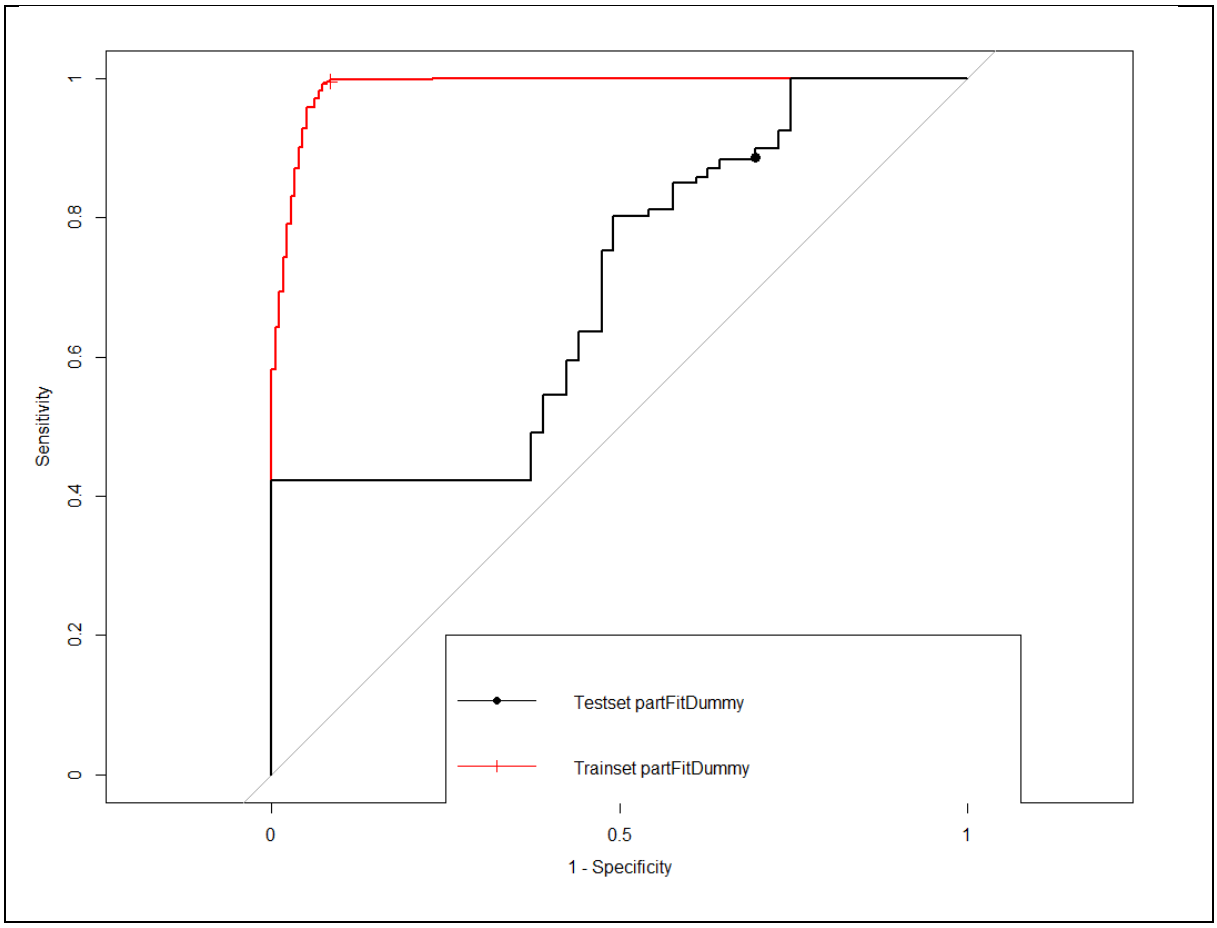
```
Area under the curve: 0.986
```



#results of trained model on test set
testresults$partFitDummy <- testresultsfunction(partFitDummy, testHRdataDummy, "partFitDummy")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 273  41
        X1  35  18

               Accuracy : 0.7929
                 95% CI : (0.7478, 0.8332)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.9921

                  Kappa : 0.1997
 Mcnemar's Test P-Value : 0.5663

            Sensitivity : 0.30508
            Specificity : 0.88636
         Pos Pred Value : 0.33962
         Neg Pred Value : 0.86943
             Prevalence : 0.16076
         Detection Rate : 0.04905
   Detection Prevalence : 0.14441
      Balanced Accuracy : 0.59572

       'Positive' Class : X1

Area under the curve: 0.607
```




```
#------------------------------
#Bagged Trees
#------------------------------

set.seed(1247)
treebagFit <- train(x = trainHRdata[, -1],
                    y = trainHRdata$Attrition,
                    method = "treebag",
                    nbagg = 50,
                    metric = "ROC",
                    trControl = ctrl)
treebagFit
```

Outcome:

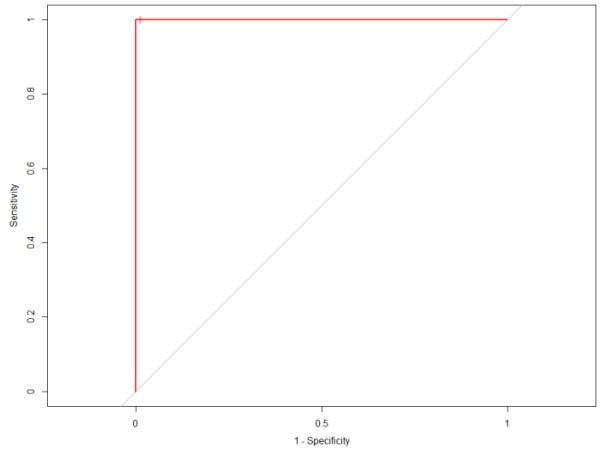
```
Bagged CART

1103 samples
  30 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 992, 992, 992, 993, 994, 993, ...
Resampling results:

  ROC        Sens       Spec
  0.7955515  0.9692871  0.2412418
```

#results of trained model

trainresults$partFit <- trainresultsfunction(partFit, trainHRdata)

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 925   2
        X1   0 176

               Accuracy : 0.9982
                 95% CI : (0.9935, 0.9998)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9933
 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 0.9888
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9978
             Prevalence : 0.1614
         Detection Rate : 0.1596
   Detection Prevalence : 0.1596
      Balanced Accuracy : 0.9944

       'Positive' Class : X1

Area under the curve: 1
```



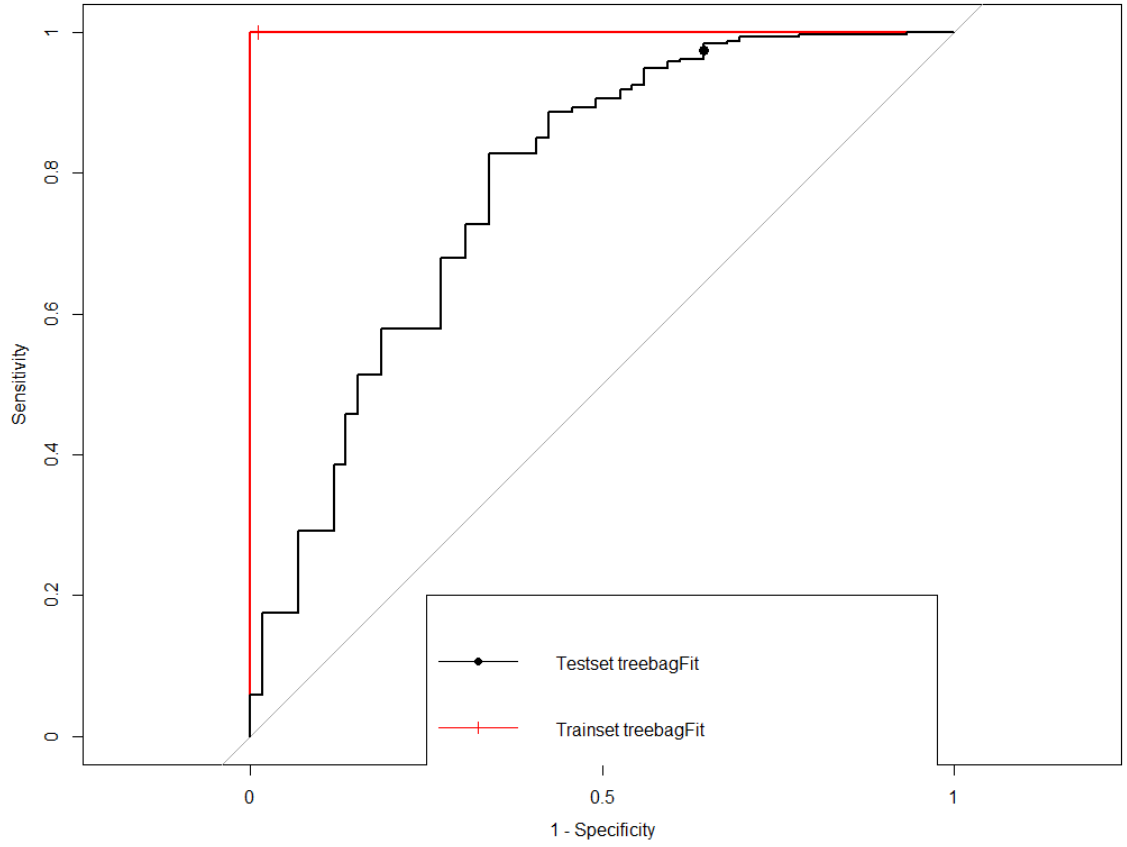
#results of trained model on test set

testresults$partFit <- testresultsfunction(partFit, testHRdata, "partFit")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 298  38
        X1  10  21

               Accuracy : 0.8692
                 95% CI : (0.8304, 0.902)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.06498
```

```
                 Kappa : 0.4002
 Mcnemar's Test P-Value : 9.735e-05

            Sensitivity : 0.35593
            Specificity : 0.96753
         Pos Pred Value : 0.67742
         Neg Pred Value : 0.88690
             Prevalence : 0.16076
         Detection Rate : 0.05722
   Detection Prevalence : 0.08447
      Balanced Accuracy : 0.66173

       'Positive' Class : X1

[1] "F1 score = 0.466666666666667"
Area under the curve: 0.7649
```




```
#------------------------------
```

#Bagged Trees Dummy set

```
#------------------------------

set.seed(1247)
treebagFitDummy <- train(x = trainHRdataDummy[, -1],
                         y = trainHRdataDummy$Attrition,
                         method = "treebag",
                         nbagg = 50,
                         metric = "ROC",
                         trControl = ctrl)
treebagFitDummy
```

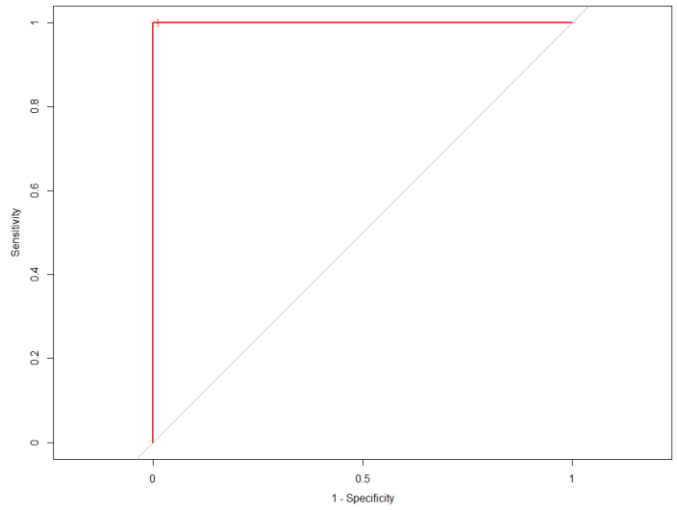
Outcome:

```
Bagged CART

1103 samples
  51 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 992, 992, 992, 993, 994, 993, ...
Resampling results:

  ROC        Sens       Spec
  0.7777671  0.9664633  0.2241176
```

#results of trained model
trainresults$treebagFitDummy <- trainresultsfunction(treebagFitDummy, trainHRdataDummy)

Outcome:

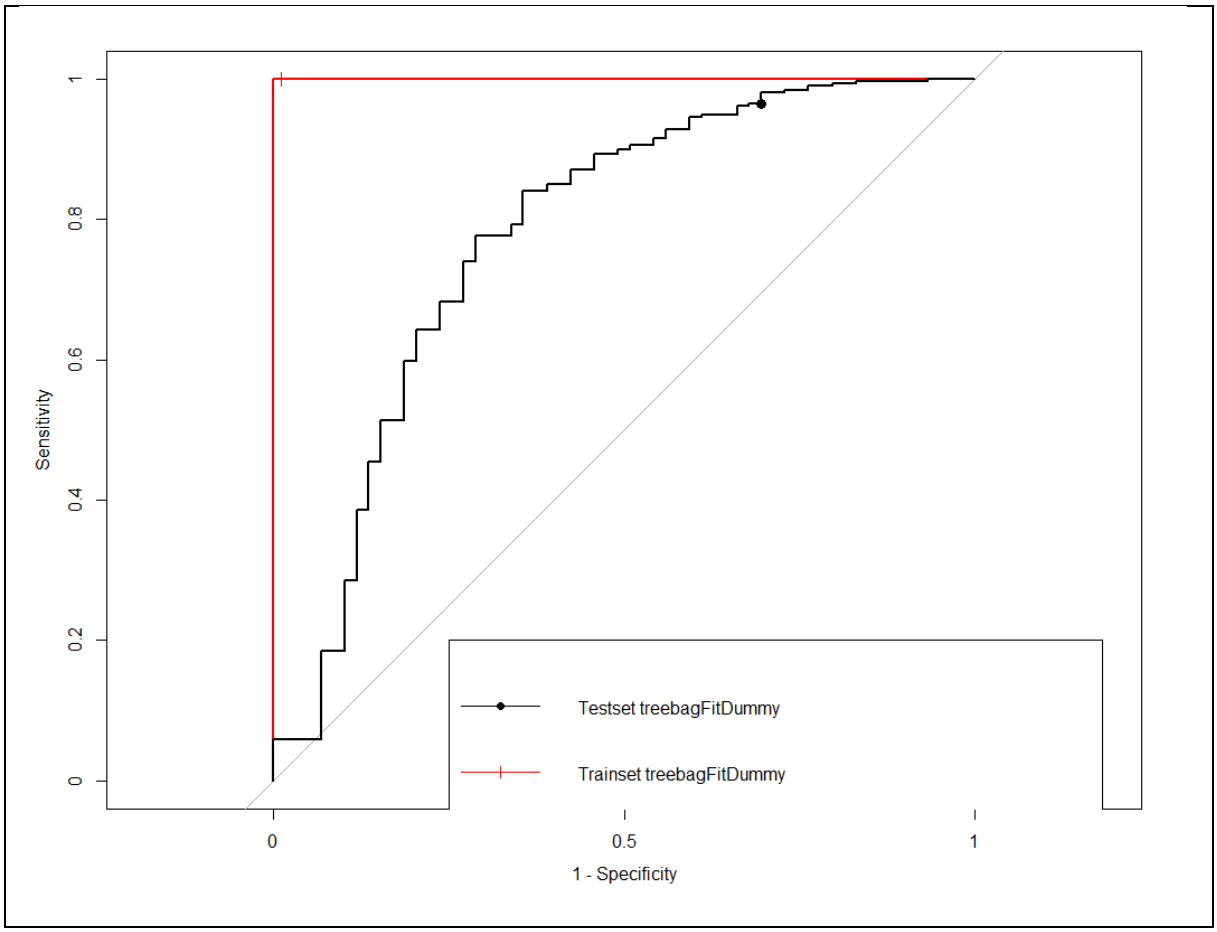
```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 925   2
        X1   0 176

               Accuracy : 0.9982
                 95% CI : (0.9935, 0.9998)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9933
 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 0.9888
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9978
             Prevalence : 0.1614
         Detection Rate : 0.1596
   Detection Prevalence : 0.1596
      Balanced Accuracy : 0.9944

       'Positive' Class : X1

Area under the curve: 1
```



#results of trained model on test set
testresults$treebagFitDummy <- testresultsfunction(treebagFitDummy, testHRdataDummy, "treebagFitDummy")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 297  41
        X1  11  18

               Accuracy : 0.8583
                 95% CI : (0.8184, 0.8923)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.1784

                  Kappa : 0.3391
 Mcnemar's Test P-Value : 5.781e-05

            Sensitivity : 0.30508
            Specificity : 0.96429
         Pos Pred Value : 0.62069
         Neg Pred Value : 0.87870
             Prevalence : 0.16076
         Detection Rate : 0.04905
   Detection Prevalence : 0.07902
      Balanced Accuracy : 0.63469

       'Positive' Class : X1

[1] "F1 score = 0.409090909090909"
Area under the curve: 0.7632
```




```
#------------------------------
#Random Forests
#------------------------------

mtryValues <- c(2, 10, 15, 20, 25, 31)
set.seed(1247)
rfFit <- train(x = trainHRdata[, -1],
               y = trainHRdata$Attrition,
               method = "rf",
               ntree = 1000,
               tuneGrid = data.frame(mtry = mtryValues),
               importance = TRUE,
               metric = "ROC",
               trControl = ctrl)
rfFit
```

Outcome:

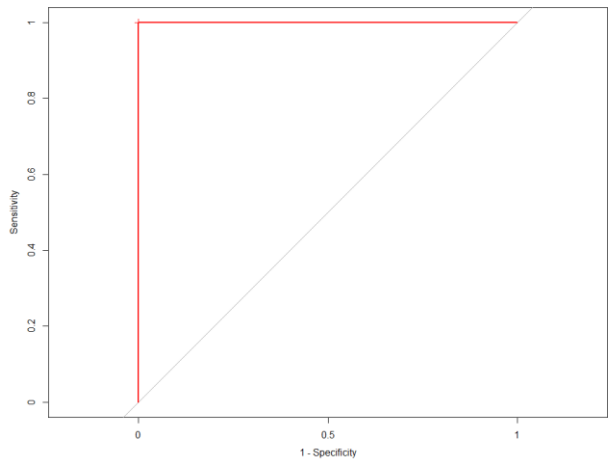
```
Random Forest

1103 samples
  30 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 992, 992, 992, 993, 994, 993, ...
Resampling results across tuning parameters:

  mtry  ROC        Sens       Spec
```

```
   2    0.8274313  0.9971926  0.07973856
  10    0.8182447  0.9894016  0.16816993
  15    0.8160469  0.9861547  0.18732026
  20    0.8135821  0.9820547  0.19967320
  25    0.8120986  0.9792380  0.22111111
  31    0.8102681  0.9768630  0.22777778

ROC was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
```

#results of trained model
trainresults$rfFit <- trainresultsfunction(rfFit, trainHRdata)

Outcome:

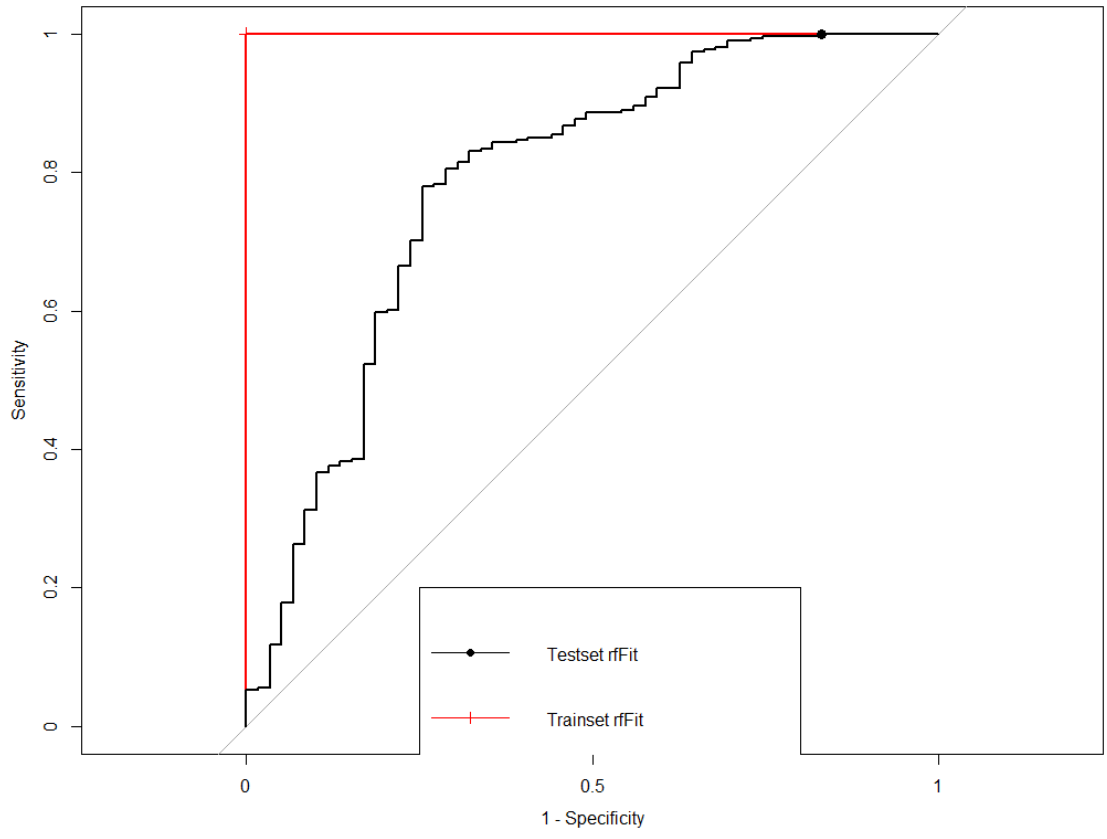
```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 925   0
        X1   0 178

               Accuracy : 1
                 95% CI : (0.9967, 1)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1
 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.1614
         Detection Rate : 0.1614
   Detection Prevalence : 0.1614
      Balanced Accuracy : 1.0000

       'Positive' Class : X1

Area under the curve: 1
```



#results of trained model on test set
testresults$rfFit <- testresultsfunction(rfFit, testHRdata, "rfFit")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 308  49
        X1   0  10

               Accuracy : 0.8665
                 95% CI : (0.8274, 0.8996)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.08614

                  Kappa : 0.2551
 Mcnemar's Test P-Value : 7.025e-12

            Sensitivity : 0.16949
            Specificity : 1.00000
         Pos Pred Value : 1.00000
         Neg Pred Value : 0.86275
             Prevalence : 0.16076
         Detection Rate : 0.02725
   Detection Prevalence : 0.02725
      Balanced Accuracy : 0.58475

       'Positive' Class : X1

[1] "F1 score = 0.289855072463768"
Area under the curve: 0.7839
```



#----------

```
#Random Forests Dummy set
#-----------------------------

mtryValuesDummy <- c(2, 10, 20, 30, 40, 52)
set.seed(1247)
rfFitDummy <- train(x = trainHRdataDummy[, -1],
          y = trainHRdataDummy$Attrition,
          method = "rf",
          ntree = 1000,
          tuneGrid = data.frame(mtry = mtryValuesDummy),
          importance = TRUE,
          metric = "ROC",
          trControl = ctrl)
rfFitDummy
```

Outcome:

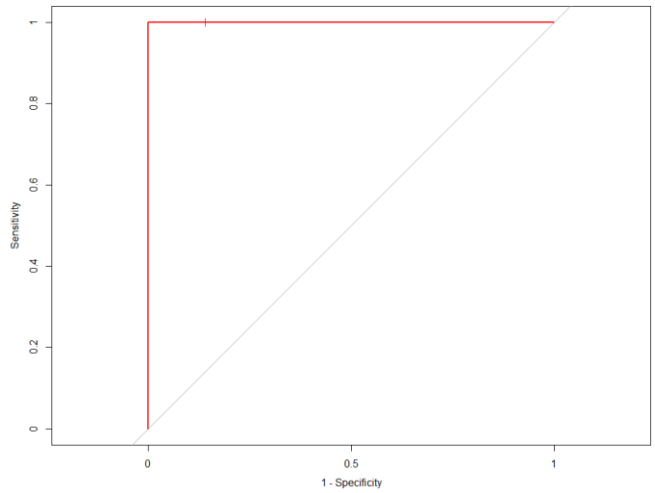
```
Random Forest

1103 samples
  51 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 992, 992, 992, 993, 994, 993, ...
Resampling results across tuning parameters:

  mtry  ROC        Sens       Spec
   2    0.8282182  0.9991351  0.05143791
  10    0.8120767  0.9900397  0.16150327
  20    0.8046020  0.9822604  0.19189542
  30    0.8006318  0.9794530  0.20535948
  40    0.7989807  0.9772838  0.21098039
  52    0.7975457  0.9740486  0.20862745

ROC was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
```

#results of trained model
trainresults$rfFitDummy <- trainresultsfunction(rfFitDummy, trainHRdataDummy)

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 925  25
        X1   0 153

               Accuracy : 0.9773
                 95% CI : (0.9667, 0.9853)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9112
 Mcnemar's Test P-Value : 1.587e-06

            Sensitivity : 0.8596
            Specificity : 1.0000
         Pos Pred Value : 1.0000
```

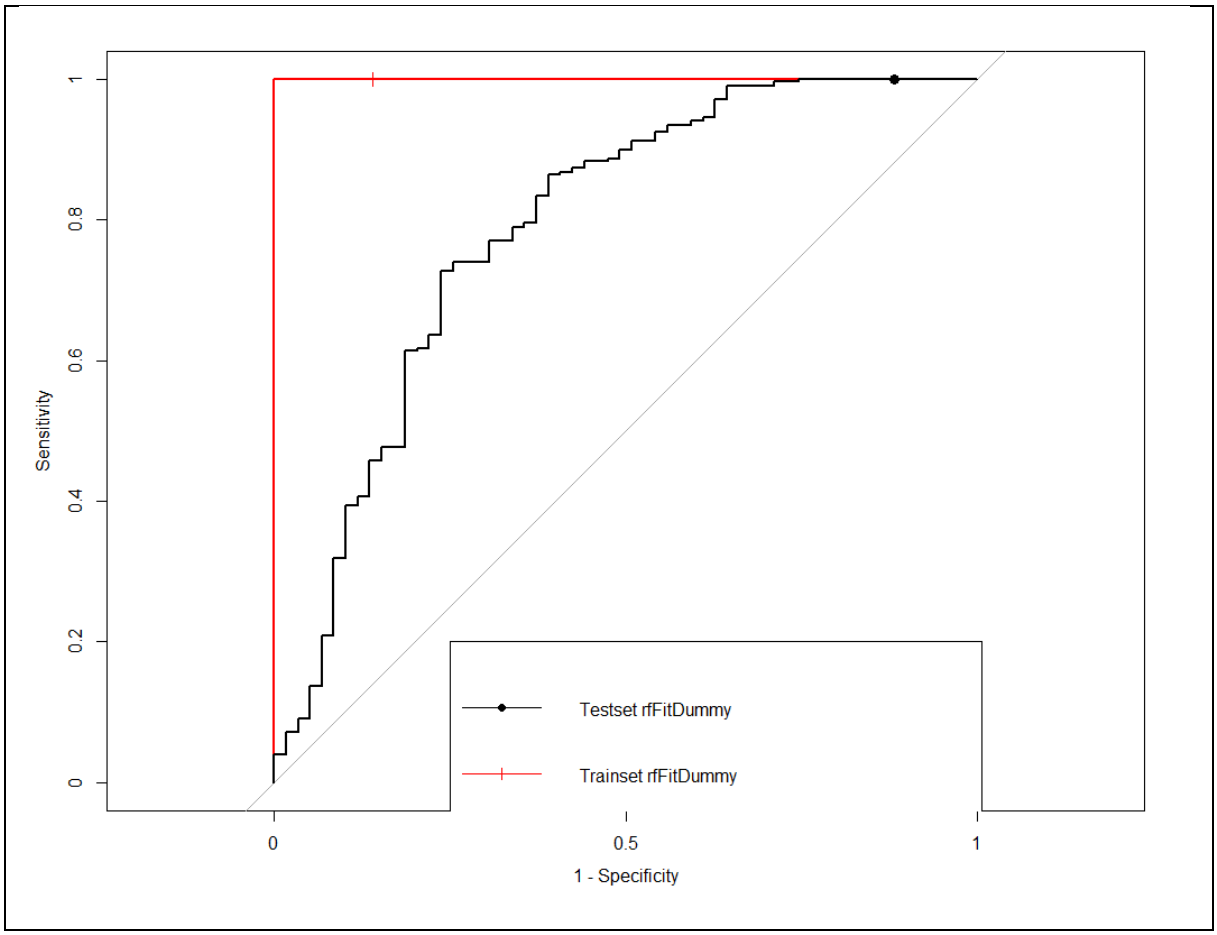
```
          Neg Pred Value : 0.9737
              Prevalence : 0.1614
          Detection Rate : 0.1387
    Detection Prevalence : 0.1387
       Balanced Accuracy : 0.9298

        'Positive' Class : X1

Area under the curve: 1
```



#results of trained model on test set
testresults$rfFitDummy <- testresultsfunction(rfFitDummy, testHRdataDummy, "rfFitDummy")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 308  52
        X1   0   7

               Accuracy : 0.8583
                 95% CI : (0.8184, 0.8923)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.1784

                  Kappa : 0.1843
 Mcnemar's Test P-Value : 1.522e-12

            Sensitivity : 0.11864
            Specificity : 1.00000
         Pos Pred Value : 1.00000
         Neg Pred Value : 0.85556
             Prevalence : 0.16076
         Detection Rate : 0.01907
   Detection Prevalence : 0.01907
      Balanced Accuracy : 0.55932

       'Positive' Class : X1

[1] "F1 score = 0.212121212121212"
Area under the curve: 0.7859
```




```
#------------------------------
#Boosting: Gradient Boosting Machines
#------------------------------

gbmGrid <- expand.grid(interaction.depth = c(1, 3, 5, 7, 9),
                       n.trees = (1:15)*100,
                       shrinkage = c(.01, .1),
                       n.minobsinnode = 10)
```

#the method gbm needs the predictors to be of the same type (numeric, ordered, or factor). That is why gbm is only executed on the dummy training set.

```
#------------------------------
#Gradient Boosting Machines dummy set
#------------------------------

set.seed(1247)
gbmFitDummy <- train(x = trainHRdataDummy[, -1],
                     y = trainHRdataDummy$Attrition,
                     method = "gbm",
                     tuneGrid = gbmGrid,
                     metric = "ROC",
                     verbose = FALSE,
                     trControl = ctrl)
gbmFitDummy
```

Outcome:

```
Stochastic Gradient Boosting

1103 samples
  51 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 992, 992, 992, 993, 994, 993, ...
Resampling results across tuning parameters:

  shrinkage  interaction.depth  n.trees  ROC        Sens       Spec
  0.01       1                   100     0.7520932  1.0000000  0.00000000
  0.01       1                   200     0.7744828  0.9993502  0.01555556
  0.01       1                   300     0.7881055  0.9976204  0.03790850
  0.01       1                   400     0.8009539  0.9956732  0.06496732
  0.01       1                   500     0.8114195  0.9943782  0.08307190
  0.01       1                   600     0.8179557  0.9937260  0.11104575
  0.01       1                   700     0.8234029  0.9937237  0.12901961
  0.01       1                   800     0.8278312  0.9935086  0.15026144
  0.01       1                   900     0.8312533  0.9917789  0.17169935
  0.01       1                  1000     0.8339333  0.9904839  0.19091503
  0.01       1                  1100     0.8364020  0.9896143  0.21895425
  0.01       1                  1200     0.8384327  0.9887447  0.25052288
  0.01       1                  1300     0.8409396  0.9883146  0.26189542
  0.01       1                  1400     0.8425855  0.9867952  0.28444444
  0.01       1                  1500     0.8436796  0.9867976  0.29222222
  0.01       3                   100     0.7805504  0.9995676  0.01555556
  0.01       3                   200     0.8010011  0.9939364  0.07601307
  0.01       3                   300     0.8123053  0.9922066  0.12895425
  0.01       3                   400     0.8187787  0.9902595  0.16267974
  0.01       3                   500     0.8244944  0.9885297  0.19307190
  0.01       3                   600     0.8279645  0.9876671  0.22568627
  0.01       3                   700     0.8311173  0.9874497  0.25150327
  0.01       3                   800     0.8333486  0.9861477  0.27176471
  0.01       3                   900     0.8364593  0.9850678  0.28862745
  0.01       3                  1000     0.8375149  0.9833333  0.30091503
  0.01       3                  1100     0.8385839  0.9831159  0.31437908
  0.01       3                  1200     0.8395808  0.9822511  0.32222222
  0.01       3                  1300     0.8402667  0.9807293  0.33568627
  0.01       3                  1400     0.8412079  0.9794320  0.34241830
  0.01       3                  1500     0.8419168  0.9783497  0.34346405
  0.01       5                   100     0.7954710  0.9991328  0.02568627
  0.01       5                   200     0.8119993  0.9943735  0.10300654
  0.01       5                   300     0.8199939  0.9913394  0.17071895
  0.01       5                   400     0.8265990  0.9889575  0.21235294
  0.01       5                   500     0.8308797  0.9878775  0.24712418
  0.01       5                   600     0.8331628  0.9863604  0.27189542
  0.01       5                   700     0.8344650  0.9852828  0.29098039
  0.01       5                   800     0.8359463  0.9839808  0.30660131
  0.01       5                   900     0.8366358  0.9828962  0.31117647
  0.01       5                  1000     0.8379351  0.9820243  0.32013072
  0.01       5                  1100     0.8385342  0.9800795  0.32901961
  0.01       5                  1200     0.8393080  0.9792146  0.33013072
  0.01       5                  1300     0.8393487  0.9774848  0.33803922
  0.01       5                  1400     0.8393068  0.9761851  0.33575163
  0.01       5                  1500     0.8393676  0.9761851  0.34254902
  0.01       7                   100     0.7995776  0.9982702  0.02673203
  0.01       7                   200     0.8151414  0.9926414  0.11973856
  0.01       7                   300     0.8233312  0.9900397  0.19666667
  0.01       7                   400     0.8275936  0.9880902  0.22549020
  0.01       7                   500     0.8312094  0.9876578  0.26156863
```

```
  0.01       7                   600      0.8333506  0.9852805  0.27954248
  0.01       7                   700      0.8351079  0.9835484  0.29653595
  0.01       7                   800      0.8359870  0.9822534  0.30895425
  0.01       7                   900      0.8365397  0.9813838  0.31679739
  0.01       7                  1000      0.8363373  0.9803039  0.32124183
  0.01       7                  1100      0.8366863  0.9790112  0.32228758
  0.01       7                  1200      0.8370614  0.9781463  0.32450980
  0.01       7                  1300      0.8374840  0.9764165  0.32679739
  0.01       7                  1400      0.8372265  0.9753343  0.32673203
  0.01       7                  1500      0.8374652  0.9757667  0.33013072
  0.01       9                   100      0.8016337  0.9984853  0.03679739
  0.01       9                   200      0.8144669  0.9926344  0.13901961
  0.01       9                   300      0.8222977  0.9891748  0.19862745
  0.01       9                   400      0.8269018  0.9876625  0.23470588
  0.01       9                   500      0.8288386  0.9852805  0.26372549
```

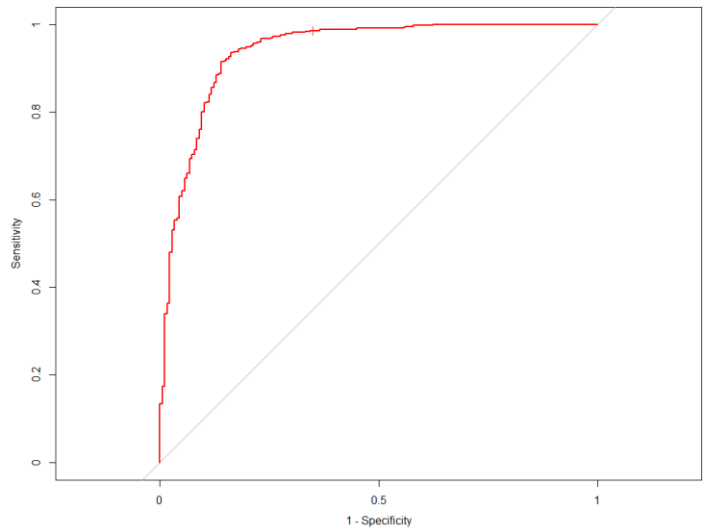
Note: the list was reduced!

```
Tuning parameter 'n.minobsinnode' was held constant at a value of 10
ROC was used to select the optimal model using  the largest value.
The final values used for the model were n.trees = 600, interaction.depth
= 1, shrinkage = 0.1
 and n.minobsinnode = 10.
```

#results of trained model
trainresults$gbmFitDummy <- trainresultsfunction(gbmFitDummy, trainHRdataDummy)

Outcome:

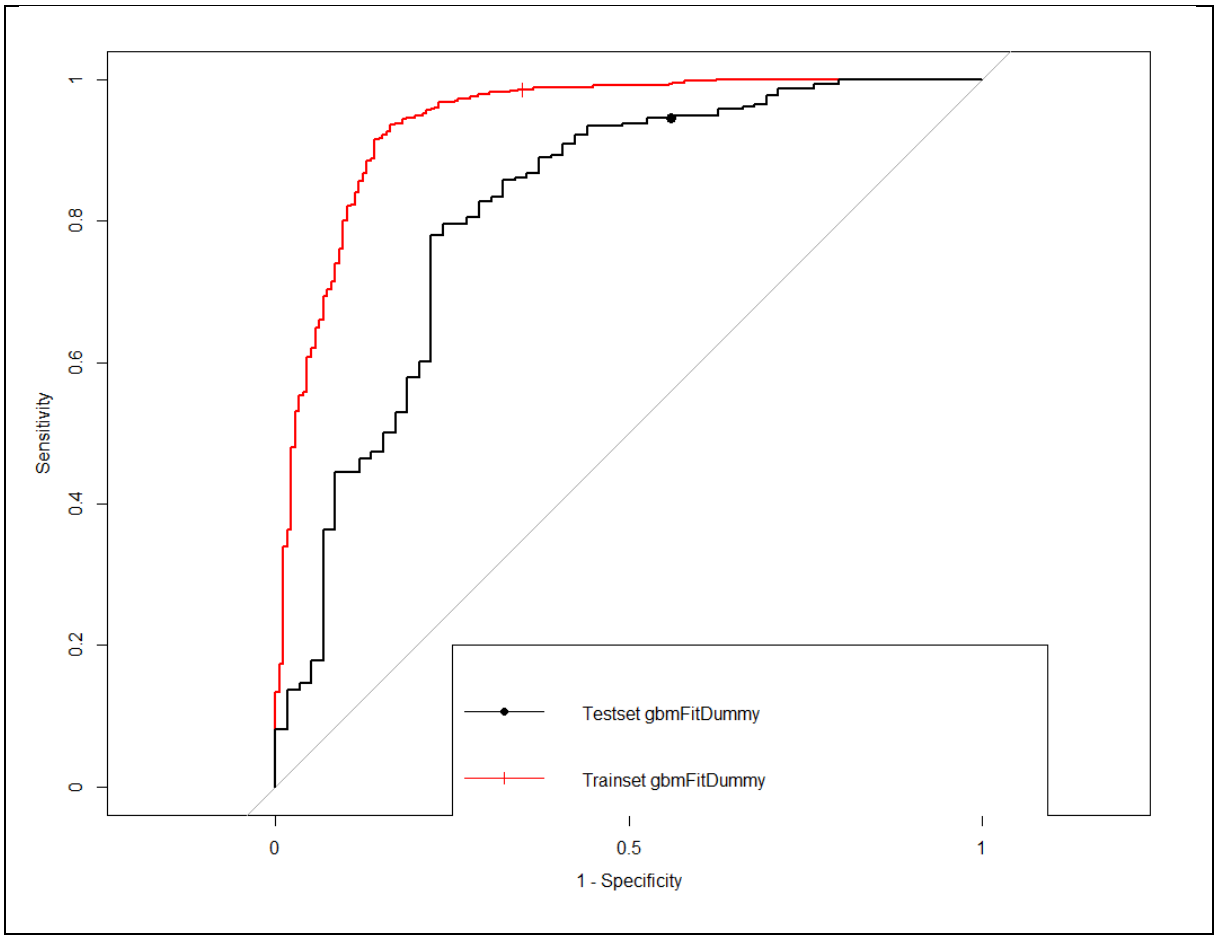
```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 911  62
        X1  14 116

               Accuracy : 0.9311
                 95% CI : (0.9145, 0.9453)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7143
 Mcnemar's Test P-Value : 6.996e-08

            Sensitivity : 0.6517
            Specificity : 0.9849
         Pos Pred Value : 0.8923
         Neg Pred Value : 0.9363
             Prevalence : 0.1614
         Detection Rate : 0.1052
   Detection Prevalence : 0.1179
      Balanced Accuracy : 0.8183

       'Positive' Class : X1

Area under the curve: 0.9394
```



#results of trained model on test set
testresults$gbmFitDummy <- testresultsfunction(gbmFitDummy, testHRdataDummy, "gbmFitDummy")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 291  33
        X1  17  26

               Accuracy : 0.8638
                 95% CI : (0.8244, 0.8972)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.11193

                  Kappa : 0.4329
 Mcnemar's Test P-Value : 0.03389

            Sensitivity : 0.44068
            Specificity : 0.94481
         Pos Pred Value : 0.60465
         Neg Pred Value : 0.89815
             Prevalence : 0.16076
         Detection Rate : 0.07084
   Detection Prevalence : 0.11717
      Balanced Accuracy : 0.69274

       'Positive' Class : X1

[1] "F1 score = 0.509803921568627"
Area under the curve: 0.8154
```




```
#------------------------------
#C5.0
#------------------------------

c50Grid <- expand.grid(trials = c(1:9, (1:10)*10),
                       model = c("tree", "rules"),
                       winnow = c(TRUE, FALSE))

c50Fit <- train(x = trainHRdata[, -1],
                y = trainHRdata$Attrition,
                method = "C5.0",
                tuneGrid = c50Grid,
                metric = "ROC",
                verbose = FALSE,
                trControl = ctrl)

#Running this model produces the error: "either a tree or rules must be provided"
#Since these are provided, it is concluded that c5.0 does not work on trainHRdata dataset.

#------------------------------
#C5.0 dummy set
#------------------------------

c50FitDummy <- train(x = trainHRdataDummy[, -1],
                y = trainHRdataDummy$Attrition,
```

```
        method = "C5.0",
        tuneGrid = c50Grid,
        metric = "ROC",
        verbose = FALSE,
        trControl = ctrl)
c50FitDummy
```

Outcome:

```
C5.0

1103 samples
  51 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 992, 992, 992, 993, 994, 993, ...
Resampling results across tuning parameters:

  model  winnow  trials  ROC        Sens       Spec
  rules  FALSE     1     0.4507860  0.9470407  0.2935294
  rules  FALSE     2     0.6116739  0.9461851  0.2923529
  rules  FALSE     3     0.7103982  0.9329804  0.3086928
  rules  FALSE     4     0.7396424  0.9182702  0.3708497
  rules  FALSE     5     0.7505268  0.9446470  0.3057516
  rules  FALSE     6     0.7586321  0.9338359  0.3496732
  rules  FALSE     7     0.7672699  0.9506989  0.3113072
  rules  FALSE     8     0.7723512  0.9409724  0.3596078
  rules  FALSE     9     0.7732234  0.9524217  0.3222876
  rules  FALSE    10     0.7786578  0.9459374  0.3458824
  rules  FALSE    20     0.7959418  0.9554675  0.3350327
  rules  FALSE    30     0.8048714  0.9595652  0.3403268
  rules  FALSE    40     0.8109593  0.9632328  0.3447059
  rules  FALSE    50     0.8119380  0.9636653  0.3368627
  rules  FALSE    60     0.8146937  0.9651847  0.3234641
  rules  FALSE    70     0.8157873  0.9660355  0.3268627
  rules  FALSE    80     0.8176024  0.9658205  0.3315686
  rules  FALSE    90     0.8183820  0.9662599  0.3337255
  rules  FALSE   100     0.8177644  0.9660332  0.3326797
  rules   TRUE     1     0.4326071  0.9571716  0.1675163
  rules   TRUE     2     0.5134970  0.9586933  0.1729412
  rules   TRUE     3     0.6601917  0.9476625  0.2389542
  rules   TRUE     4     0.6937326  0.9520126  0.2352288
  rules   TRUE     5     0.7031959  0.9491725  0.2475163
  rules   TRUE     6     0.7083315  0.9511477  0.2272549
  rules   TRUE     7     0.7143054  0.9524264  0.2431373
  rules   TRUE     8     0.7077651  0.9524357  0.2409150
  rules   TRUE     9     0.7170084  0.9530926  0.2443791
  rules   TRUE    10     0.7215055  0.9550257  0.2433987
  rules   TRUE    20     0.7363420  0.9589154  0.2486928
  rules   TRUE    30     0.7385781  0.9563090  0.2475163
  rules   TRUE    40     0.7432173  0.9578191  0.2438562
  rules   TRUE    50     0.7434752  0.9591141  0.2486928
  rules   TRUE    60     0.7428521  0.9597662  0.2385621
  rules   TRUE    70     0.7439902  0.9604137  0.2486928
  rules   TRUE    80     0.7458488  0.9606358  0.2452941
  rules   TRUE    90     0.7485768  0.9601964  0.2545098
  rules   TRUE   100     0.7497564  0.9619331  0.2521569
  tree   FALSE     1     0.5872516  0.9256685  0.3169281
  tree   FALSE     2     0.5678580  0.9651964  0.1943137
  tree   FALSE     3     0.6850062  0.9243198  0.2977124
  tree   FALSE     4     0.7079254  0.9608836  0.2235948
```

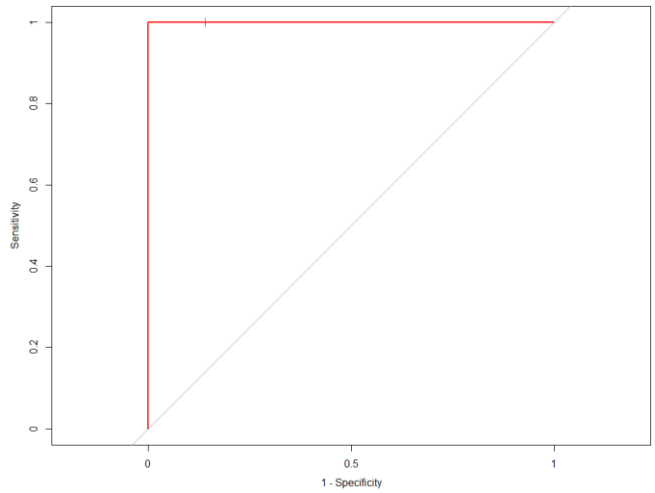
```
  tree    FALSE      5     0.7269227  0.9394250  0.2778431
  tree    FALSE      6     0.7371385  0.9621529  0.2249020
  tree    FALSE      7     0.7457089  0.9507013  0.2920915
  tree    FALSE      8     0.7523029  0.9664914  0.2157516
  tree    FALSE      9     0.7587161  0.9545956  0.2854248
  tree    FALSE     10     0.7620169  0.9675713  0.2429412
  tree    FALSE     20     0.7849037  0.9727489  0.2416340
  tree    FALSE     30     0.7975436  0.9744811  0.2507190
  tree    FALSE     40     0.8031911  0.9746891  0.2472549
  tree    FALSE     50     0.8057415  0.9753343  0.2302614
  tree    FALSE     60     0.8078063  0.9757644  0.2360131
  tree    FALSE     70     0.8085484  0.9762015  0.2392157
  tree    FALSE     80     0.8107055  0.9770617  0.2458824
  tree    FALSE     90     0.8114283  0.9772838  0.2470588
  tree    FALSE    100     0.8124837  0.9764165  0.2449020
  tree     TRUE      1     0.5475425  0.9396774  0.1969935
  tree     TRUE      2     0.4984825  0.9651706  0.1532026
  tree     TRUE      3     0.6443040  0.9334035  0.2338562
  tree     TRUE      4     0.6751334  0.9604231  0.1956209
  tree     TRUE      5     0.6878512  0.9439925  0.2361438
  tree     TRUE      6     0.6960811  0.9664656  0.1943791
  tree     TRUE      7     0.7047369  0.9556615  0.2358824
  tree     TRUE      8     0.7111375  0.9662576  0.1976471
  tree     TRUE      9     0.7135427  0.9560916  0.2245098
  tree     TRUE     10     0.7165999  0.9651683  0.1998039
  tree     TRUE     20     0.7275286  0.9640977  0.2078431
  tree     TRUE     30     0.7332628  0.9666924  0.2047059
  tree     TRUE     40     0.7387336  0.9673469  0.2014379
  tree     TRUE     50     0.7416204  0.9679874  0.2126144
  tree     TRUE     60     0.7392242  0.9679944  0.2103922
  tree     TRUE     70     0.7390053  0.9701543  0.2080392
  tree     TRUE     80     0.7395830  0.9692824  0.2103268
  tree     TRUE     90     0.7411626  0.9686396  0.2103268
  tree     TRUE    100     0.7437485  0.9699369  0.2114379

ROC was used to select the optimal model using  the largest value.
The final values used for the model were trials = 90, model = rules and w
innow = FALSE.
```

#results of trained model
trainresults$c50FitDummy <- trainresultsfunction(c50FitDummy, trainHRdataDummy)

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 925   0
        X1   0 178

               Accuracy : 1
                 95% CI : (0.9967, 1)
    No Information Rate : 0.8386
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1
 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
```

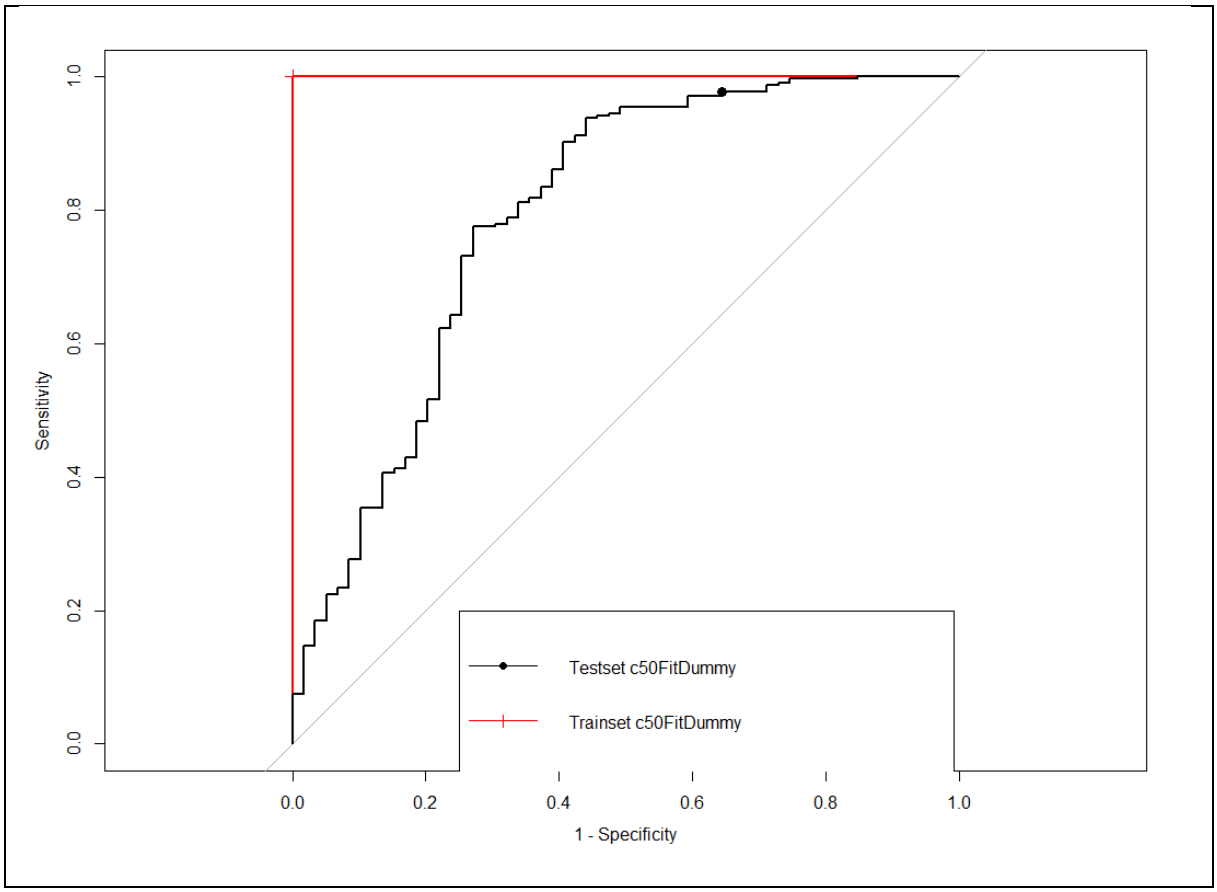
```
          Neg Pred Value : 1.0000
              Prevalence : 0.1614
          Detection Rate : 0.1614
    Detection Prevalence : 0.1614
       Balanced Accuracy : 1.0000

        'Positive' Class : X1

Area under the curve: 1
```



#results of trained model on test set
testresults$c50FitDummy <- testresultsfunction(c50FitDummy, testHRdataDummy, "c50FitDummy")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 301  38
        X1   7  21

               Accuracy : 0.8774
                 95% CI : (0.8394, 0.9091)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.02456

                  Kappa : 0.4231
 Mcnemar's Test P-Value : 7.744e-06

            Sensitivity : 0.35593
            Specificity : 0.97727
         Pos Pred Value : 0.75000
         Neg Pred Value : 0.88791
             Prevalence : 0.16076
         Detection Rate : 0.05722
   Detection Prevalence : 0.07629
      Balanced Accuracy : 0.66660

       'Positive' Class : X1

[1] "F1 score = 0.482758620689655"
Area under the curve: 0.7912
```

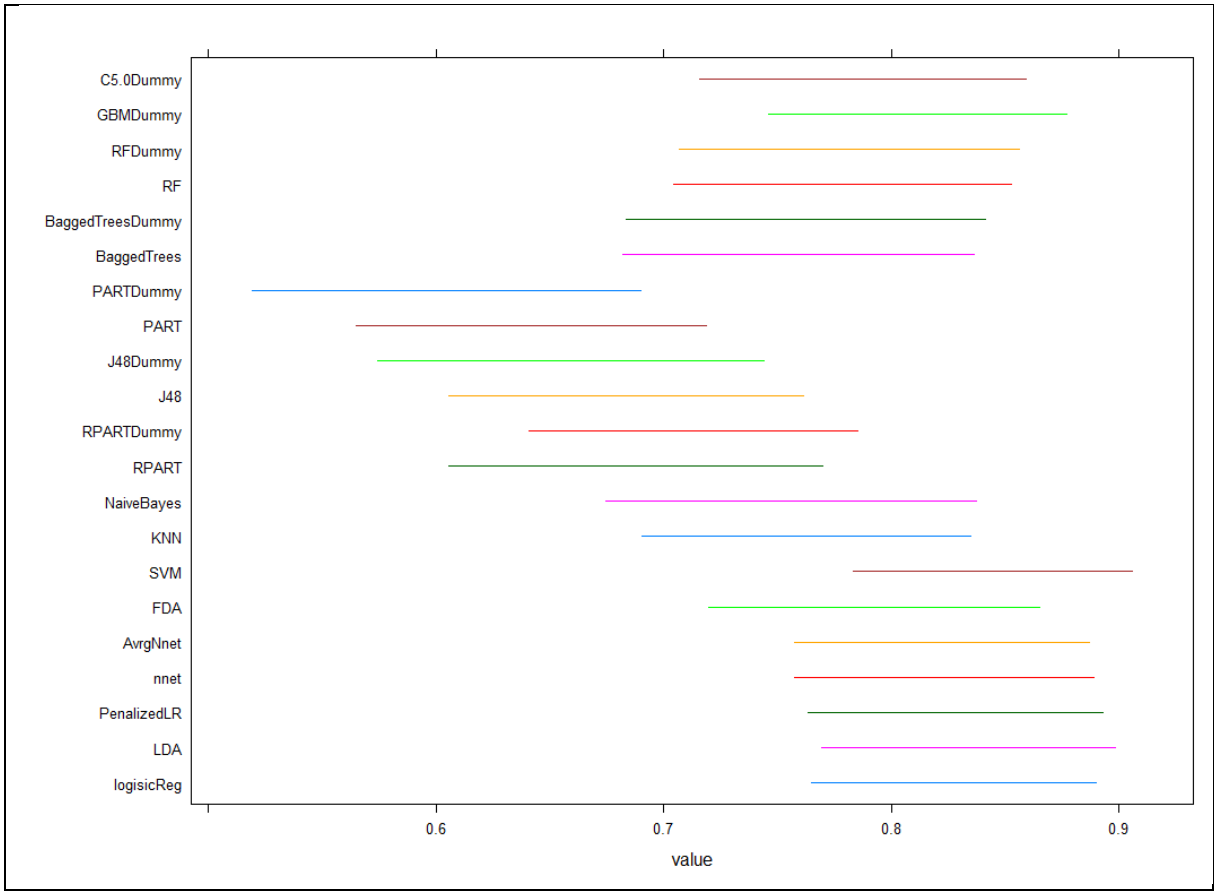


Compare the model resluts.

```
#make function to get the Confidence Interval values of the ROC

ROCCItestfunction <- function(xx){
  RocTest <- roc(response = xx$outcome, predictor = xx$X1, levels =
rev(levels(xx$outcome)))
  ROCCITest <- ci.auc(RocTest, method = "b", boot.n = 2000, boot.stratified = TRUE )
  print(ROCCITest)
  return(ROCCITest)
}

#collect value in dataframe
ROCCI <- NULL

ROCCI$logisicReg <- ROCCItestfunction(testresults$logisticReg)
ROCCI$LDA <- ROCCItestfunction(testresults$ldaFit)
ROCCI$PenalizedLR <- ROCCItestfunction(testresults$glmnTuned)
ROCCI$nnet <- ROCCItestfunction(testresults$nnetFit)
ROCCI$AvrgNnet <- ROCCItestfunction(testresults$nnetFit2)
ROCCI$FDA <- ROCCItestfunction(testresults$fdaFit)
ROCCI$SVM <- ROCCItestfunction(testresults$svmFit)
ROCCI$KNN <- ROCCItestfunction(testresults$knnFit)
ROCCI$NaiveBayes <- ROCCItestfunction(testresults$nbFit)
ROCCI$RPART <- ROCCItestfunction(testresults$rpartFit)
ROCCI$RPARTDummy <- ROCCItestfunction(testresults$rpartFitDummy)
ROCCI$J48 <- ROCCItestfunction(testresults$j48Fit)
```

```
ROCCI$J48Dummy <- ROCCItestfunction(testresults$j48FitDummy)
ROCCI$PART <- ROCCItestfunction(testresults$partFit)
ROCCI$PARTDummy <- ROCCItestfunction(testresults$partFitDummy)
ROCCI$BaggedTrees <- ROCCItestfunction(testresults$treebagFit)
ROCCI$BaggedTreesDummy <- ROCCItestfunction(testresults$treebagFitDummy)
ROCCI$RF <- ROCCItestfunction(testresults$rfFit)
ROCCI$RFDummy <- ROCCItestfunction(testresults$rfFitDummy)
ROCCI$GBMDummy <- ROCCItestfunction(testresults$gbmFitDummy)
ROCCI$C5.0Dummy <- ROCCItestfunction(testresults$c50FitDummy)

ROCCI <- data.frame(ROCCI)

ROCCI <- melt(ROCCI)

#plot the values
stripplot(variable ~ value,
        data = ROCCI,
        groups = variable,
        type = "l")
```

Outcome:



#Sampling Methods

#upsampling, downsampling, and the hybrid from: synthetic minority over-sampling technique (SMOTE)

```
trainHRdata$Attrition <- as.factor(trainHRdata$Attrition)
summary.default(trainHRdata$Attrition)
```

Outcome:

```
 X0  X1
925 178
```

```
#downsampling would result in a training set of 178x2= 356
#this is to low to expect good results
```

```
#apply upsampling
set.seed(1247)
trainHRdataDummyFullRankLowCorrUpSampled <-
upSample(trainHRdataDummyFullRankLowCorr,
trainHRdataDummyFullRankLowCorr$Attrition)
summary.default(trainHRdataDummyFullRankLowCorrUpSampled$Attrition)
```

Outcome:

```
 X0  X1
925 925
```

```
set.seed(1247)
trainHRdataUpSampled <- upSample(trainHRdata, trainHRdata$Attrition)

set.seed(1247)
trainHRdataDummyUpSampled <- upSample(trainHRdataDummy,
trainHRdataDummy$Attrition)

set.seed(1247)
trainHRdataDummyFullRankUpSampled <- upSample(trainHRdataDummyFullRank,
trainHRdataDummyFullRank$Attrition)

#apply SMOTE
set.seed(1247)
trainHRdataSMOTE <- SMOTE(Attrition ~., data = trainHRdata, perc.over = 100, k = 5,
perc.under = 100)
#Does not work on this dataset, since the predictors should be of the same type.

set.seed(1247)
trainHRdataDummySMOTE <- SMOTE(Attrition ~., data = trainHRdataDummy, perc.over =
200, k = 5, perc.under = 200)
summary.default(trainHRdataDummySMOTE$Attrition)
```

Outcome:

```
 X0  X1
712 534
```

```
set.seed(1247)
trainHRdataDummyFullRankSMOTE <- SMOTE(Attrition ~., data =
trainHRdataDummyFullRank, perc.over = 200, k = 5, perc.under = 200)

set.seed(1247)
```

```
trainHRdataDummyFullRankLowCorrSMOTE <- SMOTE(Attrition ~., data =
trainHRdataDummyFullRankLowCorr, perc.over = 200, k = 5, perc.under = 200)

#-----------------------------------------------

#Use newly created datasets to train models
#-----------------------------------------------

#-------------------------------------
#Logistic Regression
#-------------------------------------

#Logistic Regression UpSampled

trainHRdataDummyFullRankLowCorrUpSampled$Class<-NULL

set.seed(1247)
logisticRegUpSampled <- train(trainHRdataDummyFullRankLowCorrUpSampled[ , -1], y =
trainHRdataDummyFullRankLowCorrUpSampled$Attrition,  method = "glm", preProc =
c("BoxCox", "center", "scale"), metric = "ROC", trControl = ctrl)
logisticRegUpSampled
```

Outcome:

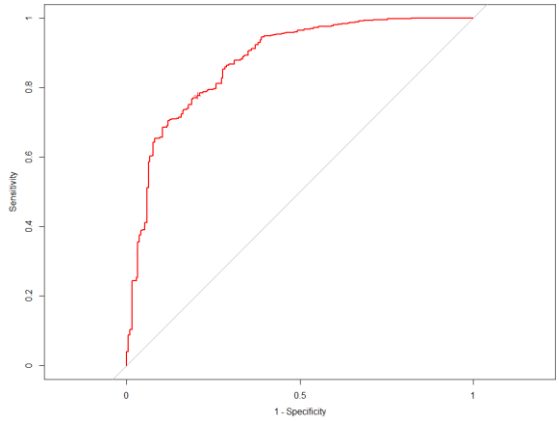
```
Generalized Linear Model

1850 samples
  39 predictor
   2 classes: 'X0', 'X1'

Pre-processing: Box-Cox transformation (13), centered (39), scaled (39)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 1664, 1664, 1664, 1665, 1666, 1665, ...
Resampling results:

  ROC        Sens       Spec
  0.8623527  0.7625877  0.7701753
```

```
trainresultsSampling <- NULL
testresultsSampling <- NULL

#results of trained model
trainresultsSampling$logisticRegUpSampled <- trainresultsfunction(logisticRegUpSampled,
trainHRdataDummyFullRankLowCorrUpSampled)
```

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 718 189
        X1 207 736

               Accuracy : 0.7859
                 95% CI : (0.7665, 0.8044)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5719
```

```
 Mcnemar's Test P-Value : 0.3929

            Sensitivity : 0.7957
            Specificity : 0.7762
         Pos Pred Value : 0.7805
         Neg Pred Value : 0.7916
             Prevalence : 0.5000
         Detection Rate : 0.3978
   Detection Prevalence : 0.5097
      Balanced Accuracy : 0.7859

       'Positive' Class : X1

Area under the curve: 0.8758
```

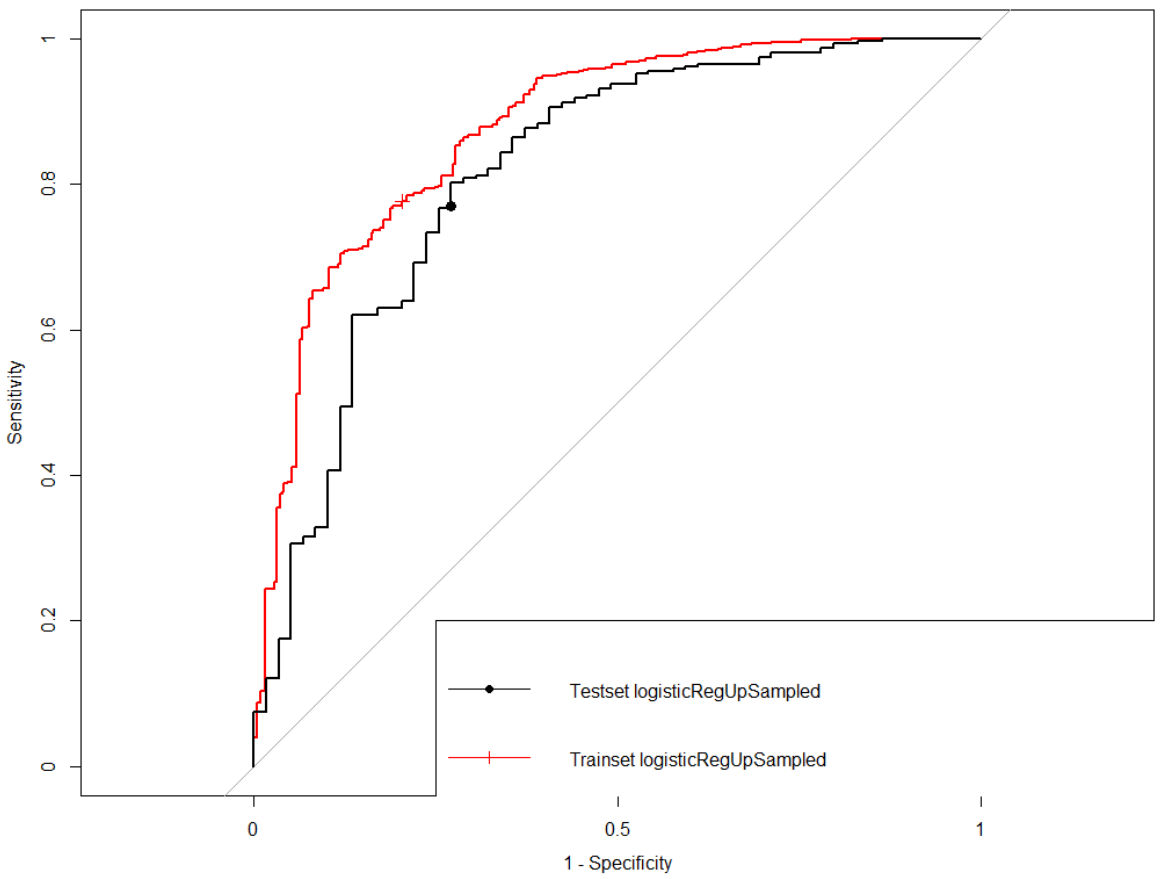


#results of trained model on test set
testresultsSampling$logisticRegUpSampled <- testresultsfunction(logisticRegUpSampled, testHRdataDummyFullRankLowCorr, "logisticRegUpSampled")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 237  16
        X1  71  43

               Accuracy : 0.7629
                 95% CI : (0.716, 0.8055)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.9999

                  Kappa : 0.3619
 Mcnemar's Test P-Value : 7.064e-09

            Sensitivity : 0.7288
            Specificity : 0.7695
         Pos Pred Value : 0.3772
         Neg Pred Value : 0.9368
             Prevalence : 0.1608
         Detection Rate : 0.1172
   Detection Prevalence : 0.3106
      Balanced Accuracy : 0.7491

       'Positive' Class : X1
```

```
[1] "F1 score = 0.497109826589595"
Area under the curve: 0.8168
```



#Logistic Regression SMOTE

set.seed(1247)
logisticRegSMOTE <- train(trainHRdataDummyFullRankLowCorrSMOTE[, -1], y = trainHRdataDummyFullRankLowCorrSMOTE$Attrition, method = "glm", preProc = c("BoxCox", "center", "scale"), metric = "ROC", trControl = ctrl)
logisticRegSMOTE

Outcome:

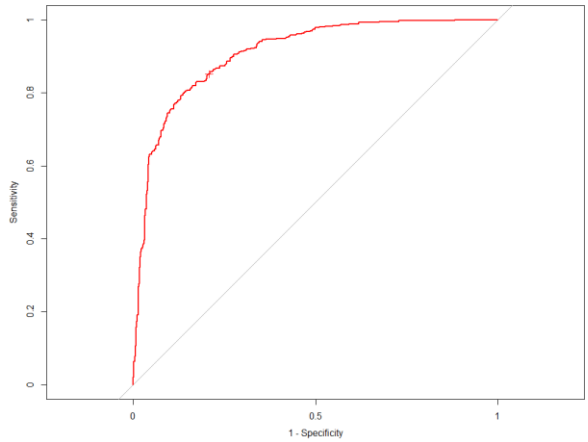
```
Generalized Linear Model

1246 samples
  39 predictor
   2 classes: 'X0', 'X1'

Pre-processing: Box-Cox transformation (14), centered (39), scaled (39)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 1121, 1121, 1121, 1122, 1121, 1122, ...
Resampling results:

  ROC       Sens       Spec
  0.891164  0.8376135  0.7744654
```

#results of trained model

trainresultsSampling$logisticRegSMOTE <- trainresultsfunction(logisticRegSMOTE, trainHRdataDummyFullRankLowCorrSMOTE)

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 606 111
        X1 106 423

               Accuracy : 0.8258
                 95% CI : (0.8036, 0.8465)
    No Information Rate : 0.5714
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.644
 Mcnemar's Test P-Value : 0.786

            Sensitivity : 0.7921
            Specificity : 0.8511
         Pos Pred Value : 0.7996
         Neg Pred Value : 0.8452
             Prevalence : 0.4286
         Detection Rate : 0.3395
   Detection Prevalence : 0.4246
      Balanced Accuracy : 0.8216

       'Positive' Class : X1

Area under the curve: 0.9078
```

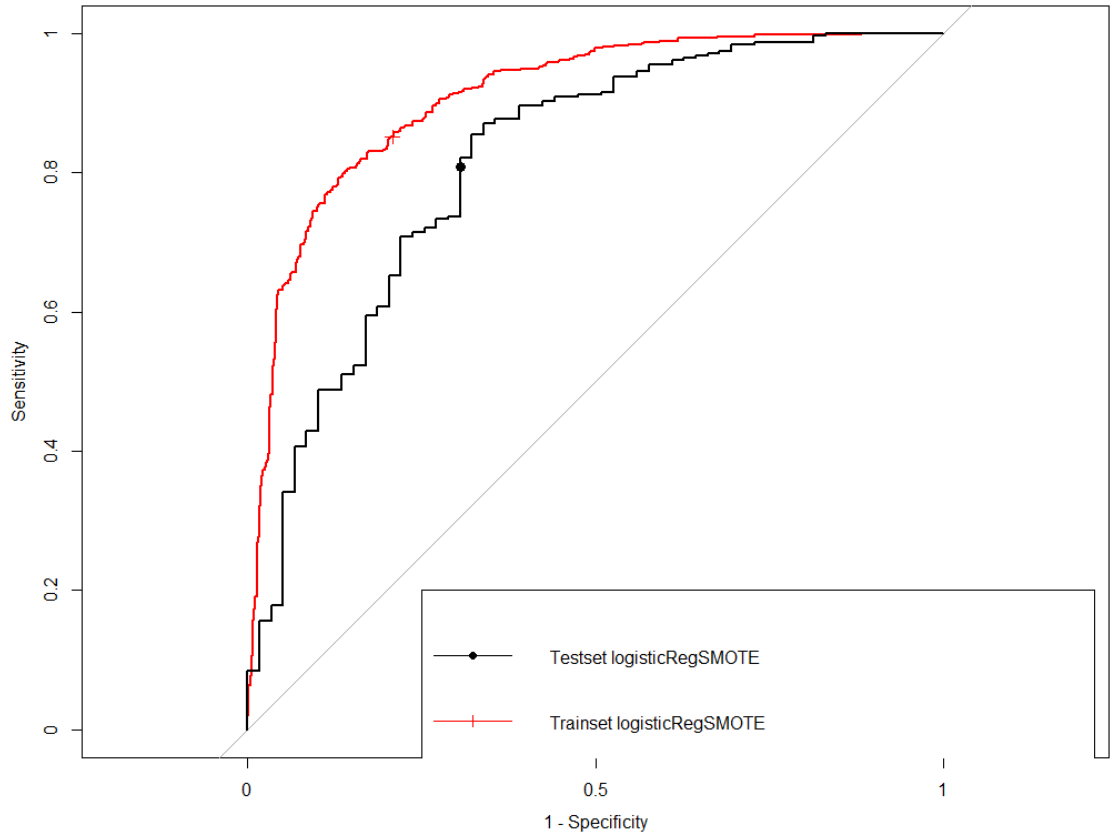


#results of trained model on test set
testresultsSampling$logisticRegSMOTE <- testresultsfunction(logisticRegSMOTE, testHRdataDummyFullRankLowCorr, "logisticRegSMOTE")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 249  18
        X1  59  41

               Accuracy : 0.7902
                 95% CI : (0.7449, 0.8307)
```

```
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.9946

                  Kappa : 0.393
 Mcnemar's Test P-Value : 5.154e-06

            Sensitivity : 0.6949
            Specificity : 0.8084
         Pos Pred Value : 0.4100
         Neg Pred Value : 0.9326
             Prevalence : 0.1608
         Detection Rate : 0.1117
   Detection Prevalence : 0.2725
      Balanced Accuracy : 0.7517

       'Positive' Class : X1

[1] "F1 score = 0.515723270440252"
Area under the curve: 0.815
```




```
#------------------------------------------
#Neural Networks
#------------------------------------------

#create grid
nnetGrid <- expand.grid(.size = 1:10, .decay = c(0, .1, 1, 2))

maxSize <- max(nnetGrid$.size)
numWts <- 1*(maxSize * (length(trainHRdataDummyFullRankLowCorr) ) + maxSize + 1)
```

#Neural Networks UpSampled

```
#spatialSign increases the predictive performance
set.seed(1247)
nnetFitUpSampled <- train(trainHRdataDummyFullRankLowCorrUpSampled[ , -1],
trainHRdataDummyFullRankLowCorrUpSampled$Attrition, method = "nnet", metric = "ROC",
preProc = c("BoxCox", "center", "scale", "spatialSign"), tuneGrid = nnetGrid, trace = FALSE,
maxit = 2000, MaxNWts = numWts, trControl = ctrl)
nnetFitUpSampled
```

Outcome:

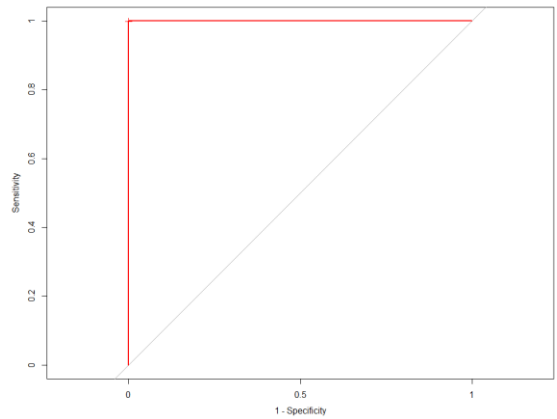
```
Neural Network

1850 samples
  39 predictor
   2 classes: 'X0', 'X1'

Pre-processing: Box-Cox transformation (13), centered (39), scaled (39), s
patial sign transformation (39)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 1664, 1664, 1664, 1665, 1666, 1665, ...
Resampling results across tuning parameters:

  size  decay  ROC        Sens       Spec
   1    0.0    0.8373854  0.8212062  0.7880902
   1    0.1    0.8690010  0.8196540  0.7584853
   1    1.0    0.8682373  0.7939154  0.7749299
   1    2.0    0.8650401  0.7863558  0.7799135
   2    0.0    0.8861367  0.7951403  0.8533754
   2    0.1    0.8969714  0.7889855  0.8244367
   2    1.0    0.8780763  0.8038710  0.7784011
   2    2.0    0.8656446  0.7803109  0.7799088
   3    0.0    0.9125570  0.8313020  0.8981814
   3    0.1    0.9191268  0.8220360  0.8940790
   3    1.0    0.8801631  0.8045348  0.7831557
   3    2.0    0.8661533  0.7818280  0.7805610
   4    0.0    0.9241239  0.8396470  0.9168186
   4    0.1    0.9336959  0.8298083  0.9291024
   4    1.0    0.8800960  0.8051870  0.7844530
   4    2.0    0.8654821  0.7781557  0.7816386
   5    0.0    0.9290908  0.8434409  0.9419986
   5    0.1    0.9396955  0.8456194  0.9643268
   5    1.0    0.8799024  0.8047546  0.7848831
   5    2.0    0.8655372  0.7770734  0.7822908
   6    0.0    0.9362701  0.8511641  0.9647288
   6    0.1    0.9477583  0.8516550  0.9783731
   6    1.0    0.8798791  0.8047546  0.7855283
   6    2.0    0.8653264  0.7753389  0.7837985
   7    0.0    0.9377642  0.8507924  0.9672814
   7    0.1    0.9588392  0.8661875  0.9881089
   7    1.0    0.8798790  0.8045418  0.7855283
   7    2.0    0.8652587  0.7744764  0.7827232
   8    0.0    0.9419095  0.8507340  0.9846237
   8    0.1    0.9621525  0.8661501  0.9933029
   8    1.0    0.8798977  0.8047592  0.7855283
   8    2.0    0.8652844  0.7753436  0.7844437
   9    0.0    0.9419214  0.8548995  0.9857340
   9    0.1    0.9671956  0.8668046  0.9950257
   9    1.0    0.8798653  0.8045442  0.7855283
```

```
   9    2.0    0.8652568  0.7742590  0.7827232
  10    0.0    0.9445508  0.8583614  0.9965498
  10    0.1    0.9713717  0.8782515  0.9971950
  10    1.0    0.8799071  0.8045418  0.7855283
  10    2.0    0.8651381  0.7736115  0.7831533

ROC was used to select the optimal model using  the largest value.
The final values used for the model were size = 10 and decay = 0.1.
```

#results of trained model
trainresultsSampling$nnetFitUpSampled <- trainresultsfunction(nnetFitUpSampled, trainHRdataDummyFullRankLowCorrUpSampled)

Outcome:

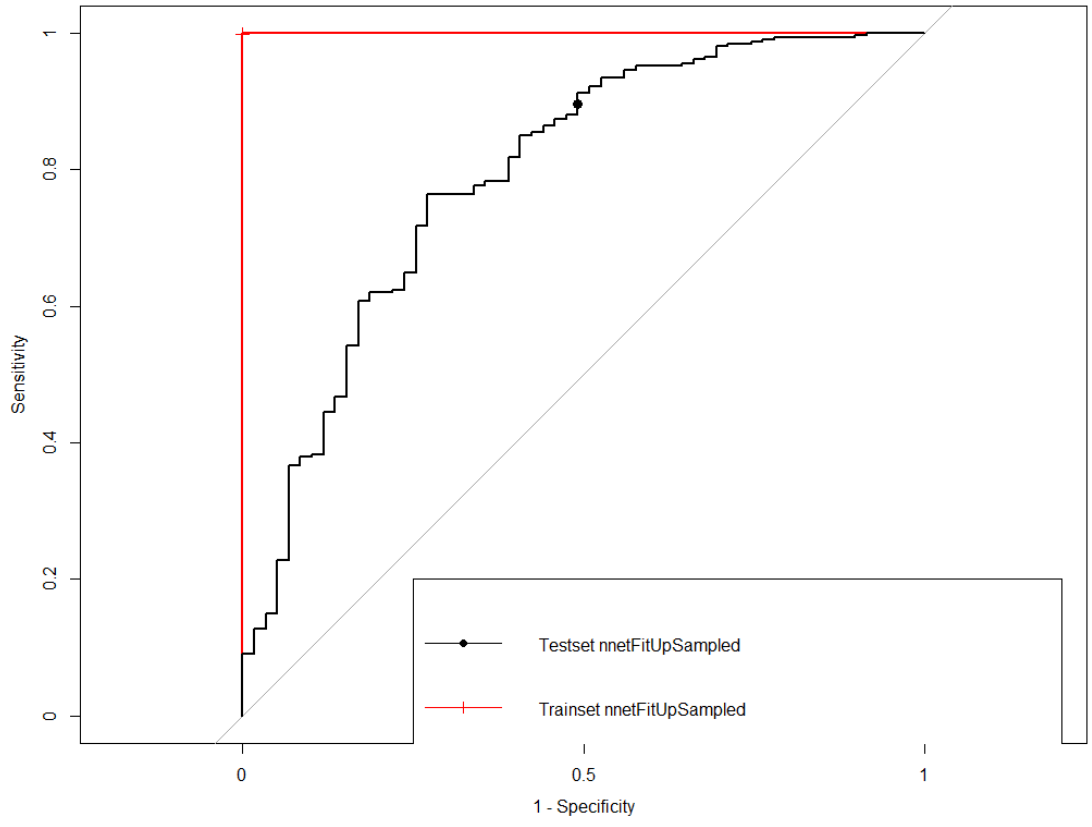
```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 923   0
        X1   2 925

               Accuracy : 0.9989
                 95% CI : (0.9961, 0.9999)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9978
 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 1.0000
            Specificity : 0.9978
         Pos Pred Value : 0.9978
         Neg Pred Value : 1.0000
             Prevalence : 0.5000
         Detection Rate : 0.5000
   Detection Prevalence : 0.5011
      Balanced Accuracy : 0.9989

       'Positive' Class : X1

Area under the curve: 1
```



#results of trained model on test set
testresultsSampling$nnetFitUpSampled <- testresultsfunction(nnetFitUpSampled, testHRdataDummyFullRankLowCorr, "nnetFitUpSampled")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 276  29
        X1  32  30

               Accuracy : 0.8338
                 95% CI : (0.7917, 0.8704)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.6440

                  Kappa : 0.3964
 Mcnemar's Test P-Value : 0.7979

            Sensitivity : 0.50847
            Specificity : 0.89610
         Pos Pred Value : 0.48387
         Neg Pred Value : 0.90492
             Prevalence : 0.16076
         Detection Rate : 0.08174
   Detection Prevalence : 0.16894
      Balanced Accuracy : 0.70229

       'Positive' Class : X1

[1] "F1 score = 0.495867768595041"
Area under the curve: 0.7925
```



#Neural Networks SMOTE

#spatialSign increases the predictive performance
set.seed(1247)
nnetFitUpSMOTE <- train(trainHRdataDummyFullRankLowCorrSMOTE[, -1], trainHRdataDummyFullRankLowCorrSMOTE$Attrition, method = "nnet", metric = "ROC", preProc = c("BoxCox", "center", "scale", "spatialSign"), tuneGrid = nnetGrid, trace = FALSE, maxit = 2000, MaxNWts = numWts, trControl = ctrl)
nnetFitUpSMOTE

Outcome:

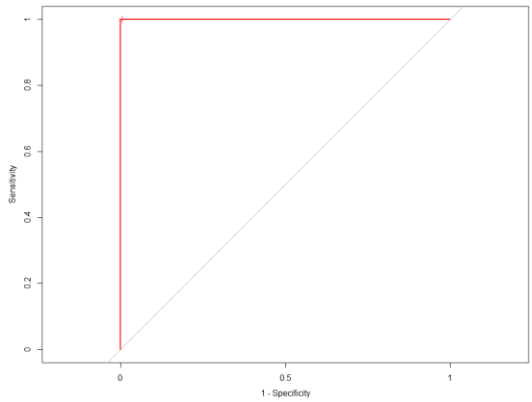
```
Neural Network

1246 samples
  39 predictor
   2 classes: 'X0', 'X1'

Pre-processing: Box-Cox transformation (14), centered (39), scaled (39), s
patial sign transformation (39)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 1121, 1121, 1121, 1122, 1121, 1122, ...
Resampling results across tuning parameters:

  size  decay  ROC        Sens       Spec
   1    0.0    0.8766043  0.8792097  0.7725926
   1    0.1    0.8976848  0.8578247  0.7734172
   1    1.0    0.8946517  0.8479890  0.7708036
   1    2.0    0.8887835  0.8608881  0.7562404
   2    0.0    0.8778907  0.8668271  0.7825856
   2    0.1    0.9142561  0.8536189  0.8052830
   2    1.0    0.8947731  0.8477074  0.7737876
   2    2.0    0.8896895  0.8586581  0.7554647
   3    0.0    0.8856142  0.8783646  0.8074423
   3    0.1    0.9257528  0.8637011  0.8431027
   3    1.0    0.8950995  0.8491119  0.7726555
   3    2.0    0.8896754  0.8575391  0.7550734
   4    0.0    0.8949660  0.8861581  0.8180922
   4    0.1    0.9378992  0.8828326  0.8700559
   4    1.0    0.8946383  0.8468701  0.7719008
   4    2.0    0.8895910  0.8569757  0.7557932
   5    0.0    0.8951367  0.8772066  0.8457372
   5    0.1    0.9484980  0.8929421  0.8884067
   5    1.0    0.8947368  0.8471518  0.7715234
   5    2.0    0.8895021  0.8569757  0.7557932
   6    0.0    0.8972280  0.8904734  0.8562683
   6    0.1    0.9528639  0.8959898  0.9086024
   6    1.0    0.8946169  0.8471518  0.7726415
   6    2.0    0.8894284  0.8561307  0.7554228
   7    0.0    0.9049533  0.8949296  0.8625926
   7    0.1    0.9565219  0.9128717  0.9131516
   7    1.0    0.8946210  0.8474335  0.7711391
   7    2.0    0.8893703  0.8564124  0.7550454
   8    0.0    0.9033950  0.8926643  0.8662683
   8    0.1    0.9611718  0.9120031  0.9258770
   8    1.0    0.8945690  0.8477152  0.7718798
   8    2.0    0.8893185  0.8564124  0.7550454
   9    0.0    0.8963317  0.8847613  0.8702096
   9    0.1    0.9640022  0.9151330  0.9389658
   9    1.0    0.8945167  0.8477152  0.7718798
   9    2.0    0.8892868  0.8561307  0.7557862
  10    0.0    0.9090557  0.8937911  0.8817470
```

```
  10    0.1    0.9651034  0.9187676  0.9319147
  10    1.0    0.8944221  0.8474335  0.7718798
  10    2.0    0.8892401  0.8555673  0.7554088

ROC was used to select the optimal model using  the largest value.
The final values used for the model were size = 10 and decay = 0.1.
```

#results of trained model
trainresultsSampling$nnetFitUpSMOTE <- trainresultsfunction(nnetFitUpSMOTE, trainHRdataDummyFullRankLowCorrSMOTE)

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 712   2
        X1   0 532

               Accuracy : 0.9984
                 95% CI : (0.9942, 0.9998)
    No Information Rate : 0.5714
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9967
 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 0.9963
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9972
             Prevalence : 0.4286
         Detection Rate : 0.4270
   Detection Prevalence : 0.4270
      Balanced Accuracy : 0.9981

       'Positive' Class : X1

Area under the curve: 1
```

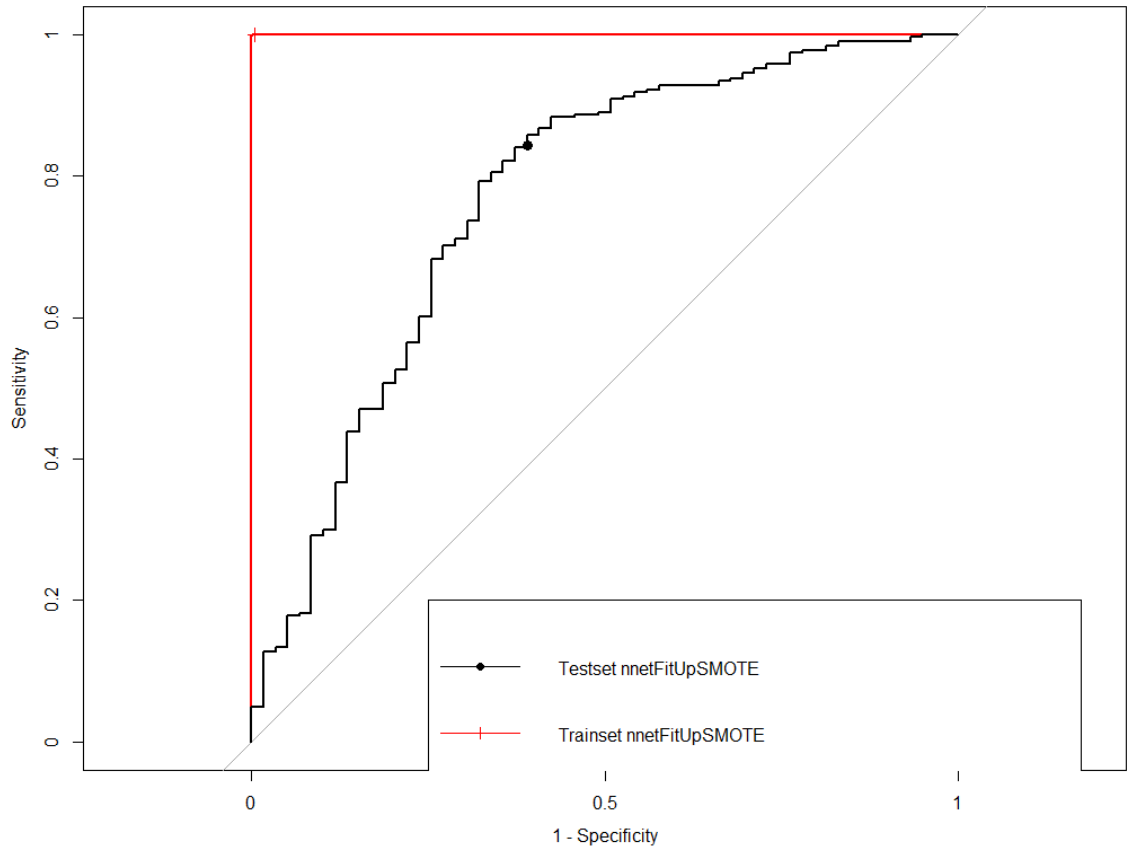


#results of trained model on test set
testresultsSampling$nnetFitUpSMOTE <- testresultsfunction(nnetFitUpSMOTE, testHRdataDummyFullRankLowCorr, "nnetFitUpSMOTE")

Outcome:

```
Confusion Matrix and Statistics
```

```
          Reference
Prediction  X0  X1
        X0 260  23
        X1  48  36

               Accuracy : 0.8065
                 95% CI : (0.7624, 0.8457)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.959541

                  Kappa : 0.3879
 Mcnemar's Test P-Value : 0.004396

            Sensitivity : 0.61017
            Specificity : 0.84416
         Pos Pred Value : 0.42857
         Neg Pred Value : 0.91873
             Prevalence : 0.16076
         Detection Rate : 0.09809
   Detection Prevalence : 0.22888
      Balanced Accuracy : 0.72716

       'Positive' Class : X1

[1] "F1 score = 0.503496503496503"
Area under the curve: 0.7676
```



#----------

#Flexible Discriminant Analysis

#----------

#Train FDA over number of components from 1 to 30 and a degree of 1 and 2.

#Train on upsampled trainingset.

```
set.seed(1247)
fdaFitupsampled <- train(x = trainHRdataDummyFullRankLowCorrUpSampled[ , -1], y =
trainHRdataDummyFullRankLowCorrUpSampled$Attrition, method = "earth", metric =
"ROC", preProc = c("BoxCox", "center", "scale"), tuneGrid = expand.grid(.nprune = 1:30,
.degree = 1:2), trControl = ctrl)
fdaFitupsampled
```

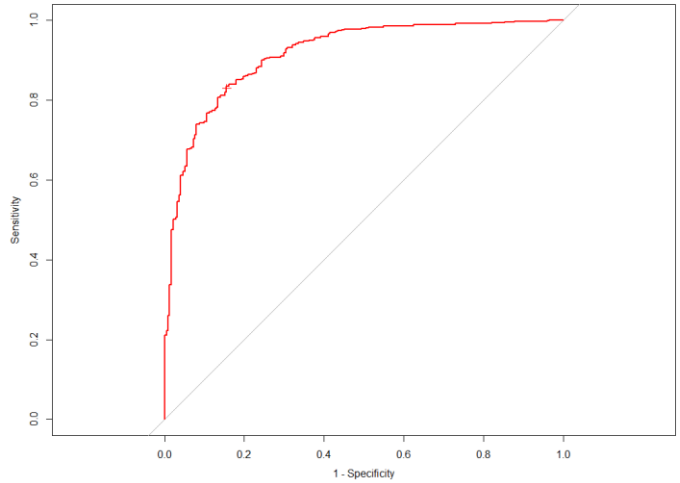
Outcome:

```
Multivariate Adaptive Regression Spline

1850 samples
  39 predictor
   2 classes: 'X0', 'X1'

Pre-processing: Box-Cox transformation (13), centered (39), scaled (39)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 1664, 1664, 1664, 1665, 1666, 1665, ...
Resampling results across tuning parameters:
  1       28      0.8799666  0.7897616  0.8207784
  1       29      0.8834650  0.7925830  0.8220804
  1       30      0.8830586  0.7923726  0.8227115
  2        1      0.5000000  0.7400000  0.2600000
  2        2      0.6100958  0.7104745  0.5620991
  2        3      0.7193622  0.6708392  0.6646424
  2        4      0.7422760  0.7035133  0.6490697
  2        5      0.7622773  0.7182539  0.6722814
  2        6      0.7778210  0.7262272  0.7033848
  2        7      0.7886456  0.7329056  0.7117578
  2        8      0.7967406  0.7363417  0.7212950
  2        9      0.8079603  0.7424240  0.7424731
  2       10      0.8182017  0.7543315  0.7457574
  2       11      0.8232655  0.7530295  0.7522347
  2       12      0.8319331  0.7588663  0.7609046
  2       13      0.8395600  0.7631837  0.7721155
  2       14      0.8426392  0.7716223  0.7768724
  2       15      0.8468435  0.7722698  0.7908789
  2       16      0.8516156  0.7707527  0.7958696
  2       17      0.8543649  0.7767952  0.7978144
  2       18      0.8575320  0.7815591  0.8036419
  2       19      0.8606604  0.7826414  0.8090720
  2       20      0.8648038  0.7876064  0.8142660
  2       21      0.8684763  0.7889130  0.8131791
  2       22      0.8716962  0.7884970  0.8216199
  2       23      0.8739333  0.7932702  0.8237681
  2       24      0.8756527  0.7926367  0.8250865
  2       25      0.8785085  0.7954254  0.8294156
  2       26      0.8796350  0.7997499  0.8298457
  2       27      0.8811837  0.8006054  0.8311127
  2       28      0.8826468  0.8017134  0.8354418
  2       29      0.8843310  0.8042987  0.8343572
  2       30      0.8856178  0.8023656  0.8363067

ROC was used to select the optimal model using  the largest value.
The final values used for the model were nprune = 30 and degree = 2.
```

#results of trained model
trainresultsSampling$fdaFitupsampled <- trainresultsfunction(fdaFitupsampled, trainHRdataDummyFullRankLowCorrUpSampled)

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 767 144
        X1 158 781

               Accuracy : 0.8368
                 95% CI : (0.8191, 0.8533)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.6735
 Mcnemar's Test P-Value : 0.4544

            Sensitivity : 0.8443
            Specificity : 0.8292
         Pos Pred Value : 0.8317
         Neg Pred Value : 0.8419
             Prevalence : 0.5000
         Detection Rate : 0.4222
   Detection Prevalence : 0.5076
      Balanced Accuracy : 0.8368

       'Positive' Class : X1

Area under the curve: 0.9149
```

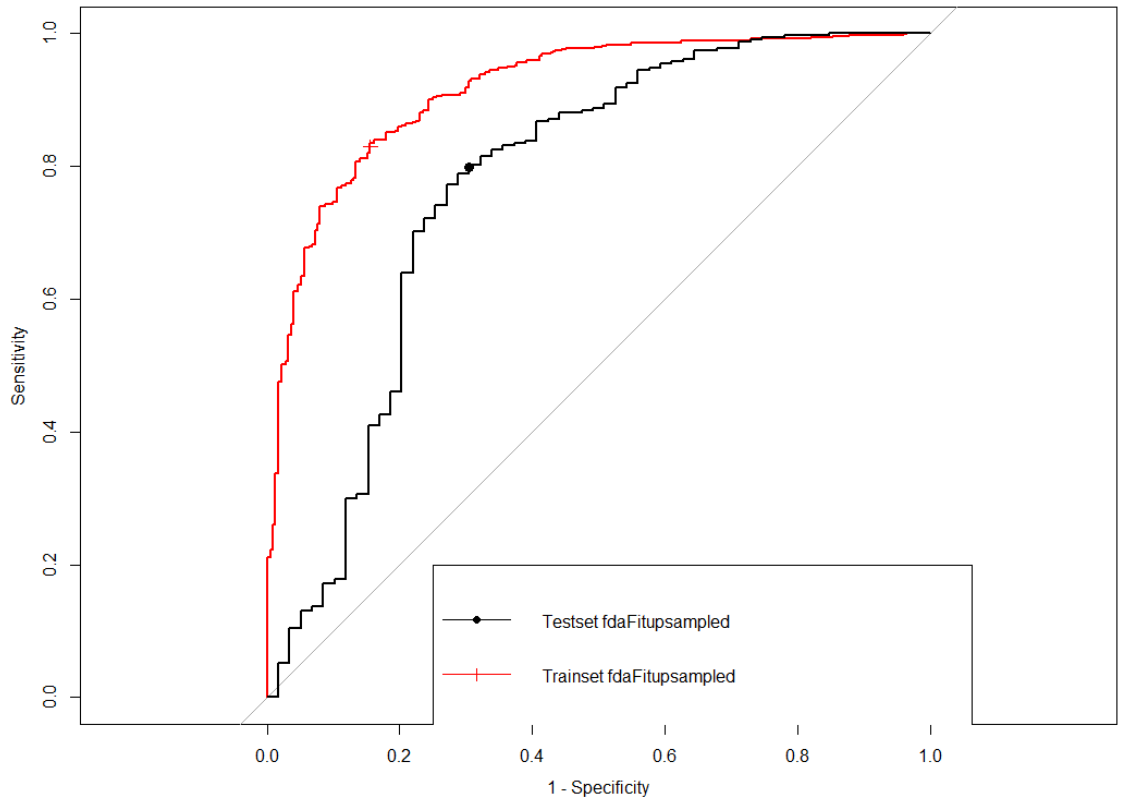


#results of trained model on test set
testresultsSampling$fdaFitupsampled <- testresultsfunction(fdaFitupsampled, testHRdataDummyFullRankLowCorr, "fdaFitupsampled")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
```

```
        X0 246  18
        X1  62  41

               Accuracy : 0.782
                 95% CI : (0.7362, 0.8232)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.9984

                  Kappa : 0.3793
 Mcnemar's Test P-Value : 1.528e-06

            Sensitivity : 0.6949
            Specificity : 0.7987
         Pos Pred Value : 0.3981
         Neg Pred Value : 0.9318
             Prevalence : 0.1608
         Detection Rate : 0.1117
   Detection Prevalence : 0.2807
      Balanced Accuracy : 0.7468

       'Positive' Class : X1

[1] "F1 score = 0.506172839506173"
Area under the curve: 0.7729
```



#train on SMOTE trainingset.

#Train FDA over number of components from 25 to 30 and a degree of 1 and 2.
set.seed(1247)
fdaFitSMOTE <- train(x = trainHRdataDummyFullRankLowCorrSMOTE[, -1], y = trainHRdataDummyFullRankLowCorrSMOTE$Attrition, method = "earth", metric = "ROC",

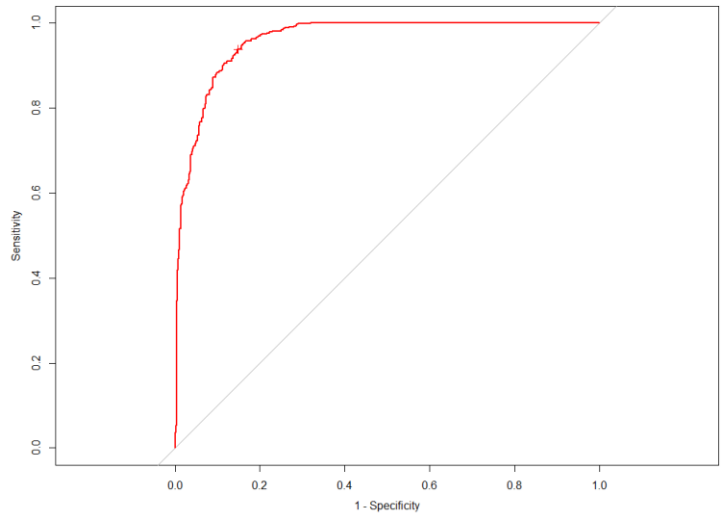
preProc = c("BoxCox", "center", "scale"), tuneGrid = expand.grid(.nprune = 25:30, .degree = 1:2), trControl = ctrl)
fdaFitSMOTE
Outcome:

```
Multivariate Adaptive Regression Spline

1246 samples
  39 predictor
   2 classes: 'X0', 'X1'

Pre-processing: Box-Cox transformation (14), centered (39), scaled (39)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 1121, 1121, 1121, 1122, 1121, 1122, ...
Resampling results across tuning parameters:

  degree  nprune  ROC        Sens       Spec
  1       25      0.9199323  0.9283177  0.7847170
  1       26      0.9199323  0.9283177  0.7847170
  1       27      0.9199323  0.9283177  0.7847170
  1       28      0.9199323  0.9283177  0.7847170
  1       29      0.9199323  0.9283177  0.7847170
  1       30      0.9199323  0.9283177  0.7847170
  2       25      0.9198561  0.8923670  0.7982460
  2       26      0.9200503  0.8957277  0.8016282
  2       27      0.9222879  0.8929108  0.8005101
  2       28      0.9238469  0.8963028  0.7967086
  2       29      0.9241004  0.9005282  0.7959539
  2       30      0.9245850  0.9008099  0.7978546

ROC was used to select the optimal model using  the largest value.
The final values used for the model were nprune = 30 and degree = 2.
```

#results of trained model
trainresultsSampling$fdaFitSMOTE <- trainresultsfunction(fdaFitSMOTE, trainHRdataDummyFullRankLowCorrSMOTE)
Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 668  79
        X1  44 455

               Accuracy : 0.9013
                 95% CI : (0.8834, 0.9173)
    No Information Rate : 0.5714
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7968
 Mcnemar's Test P-Value : 0.002172

            Sensitivity : 0.8521
            Specificity : 0.9382
         Pos Pred Value : 0.9118
         Neg Pred Value : 0.8942
             Prevalence : 0.4286
         Detection Rate : 0.3652
   Detection Prevalence : 0.4005
      Balanced Accuracy : 0.8951
```

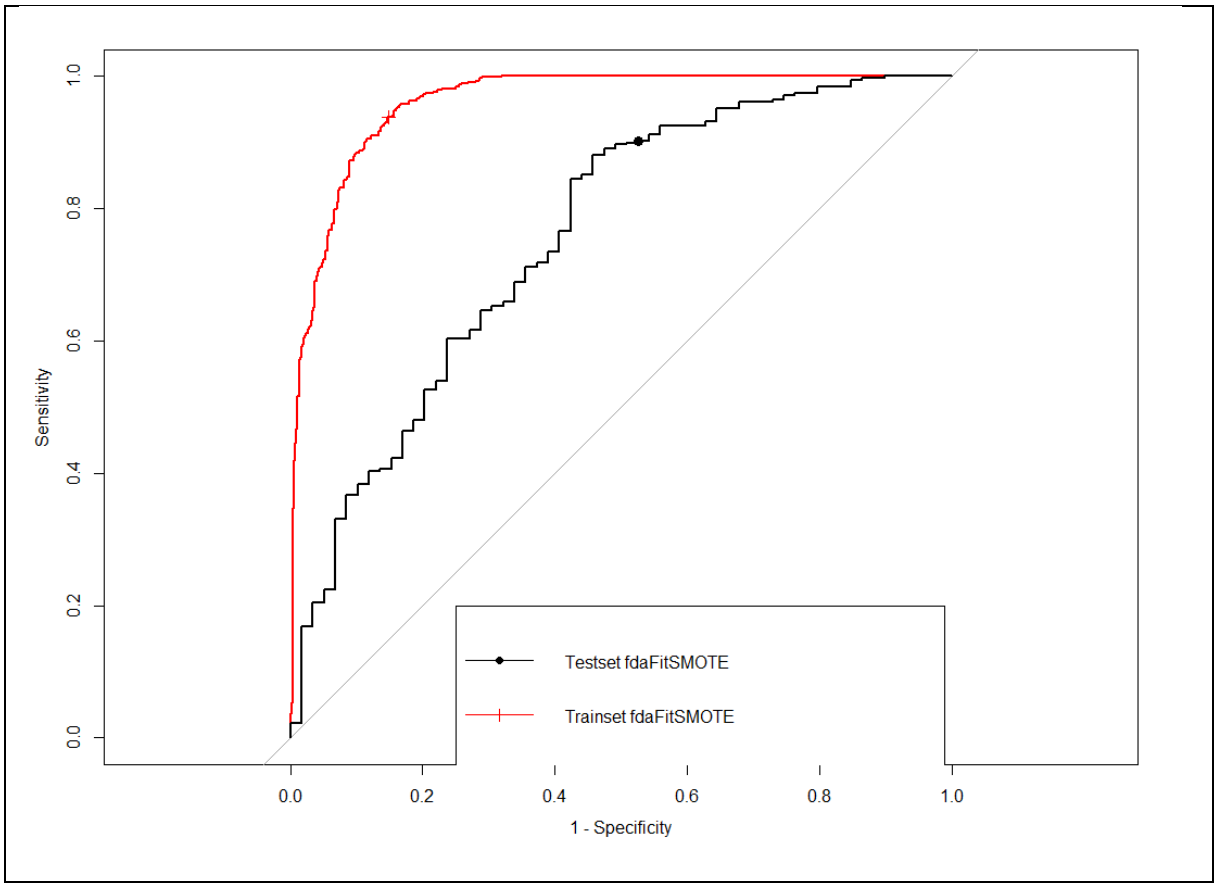
```
       'Positive' Class : X1

Area under the curve: 0.9604
```



#results of trained model on test set
testresultsSampling$fdaFitSMOTE <- testresultsfunction(fdaFitSMOTE, testHRdataDummyFullRankLowCorr, "fdaFitSMOTE")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 278  31
        X1  30  28

               Accuracy : 0.8338
                 95% CI : (0.7917, 0.8704)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.644

                  Kappa : 0.3798
 Mcnemar's Test P-Value : 1.000

            Sensitivity : 0.47458
            Specificity : 0.90260
         Pos Pred Value : 0.48276
         Neg Pred Value : 0.89968
             Prevalence : 0.16076
         Detection Rate : 0.07629
   Detection Prevalence : 0.15804
      Balanced Accuracy : 0.68859

       'Positive' Class : X1

[1] "F1 score = 0.478632478632479"
Area under the curve: 0.7569
```




```
#---------------------------------------
# Support Vector Machines
#---------------------------------------

#create tuning parameters
set.seed(1247)
sigmaRangeReduced <- sigest(as.matrix(trainHRdataDummyFullRankLowCorr[ , -1]))
svmRGridReduced <- expand.grid(.sigma = sigmaRangeReduced[1], .C = 2^(seq(-4, 4)))

# Support Vector Machines UpSampled

#train the model
set.seed(1247)
svmFitUpSampled <- train(x = trainHRdataDummyFullRankLowCorrUpSampled[ , -1], y =
trainHRdataDummyFullRankLowCorrUpSampled$Attrition, method = "svmRadial", metric =
"ROC", preProc = c("BoxCox", "center", "scale"), tuneGrid = svmRGridReduced, fit = FALSE,
trControl = ctrl)
svmFitUpSampled
```

Outcome:

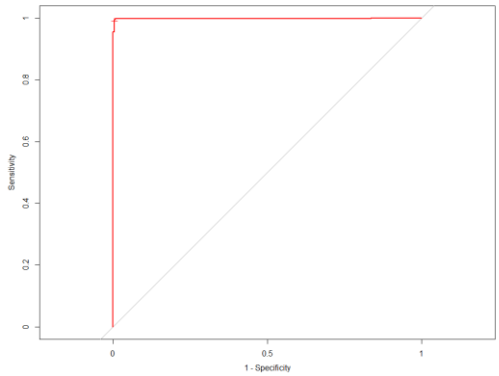
```
Support Vector Machines with Radial Basis Function Kernel

1850 samples
  39 predictor
   2 classes: 'X0', 'X1'

Pre-processing: Box-Cox transformation (13), centered (39), scaled (39)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 1664, 1664, 1664, 1665, 1666, 1665, ...
```

```
Resampling results across tuning parameters:

  C        ROC        Sens       Spec
   0.0625  0.8497826  0.7543432  0.7900468
   0.1250  0.8667428  0.8030154  0.7790323
   0.2500  0.8855392  0.8051800  0.7937331
   0.5000  0.9027779  0.8205213  0.8250888
   1.0000  0.9213240  0.8376017  0.8590439
   2.0000  0.9454673  0.8682679  0.8973165
   4.0000  0.9648639  0.8966246  0.9362225
   8.0000  0.9753479  0.9182492  0.9582702
  16.0000  0.9826495  0.9323095  0.9833450

Tuning parameter 'sigma' was held constant at a value of 0.009475476
ROC was used to select the optimal model using  the largest value.
The final values used for the model were sigma = 0.009475476 and C = 16.
```

#results of trained model
trainresultsSampling$svmFitUpSampled <- trainresultsfunction(svmFitUpSampled, trainHRdataDummyFullRankLowCorrUpSampled)

Outcome:

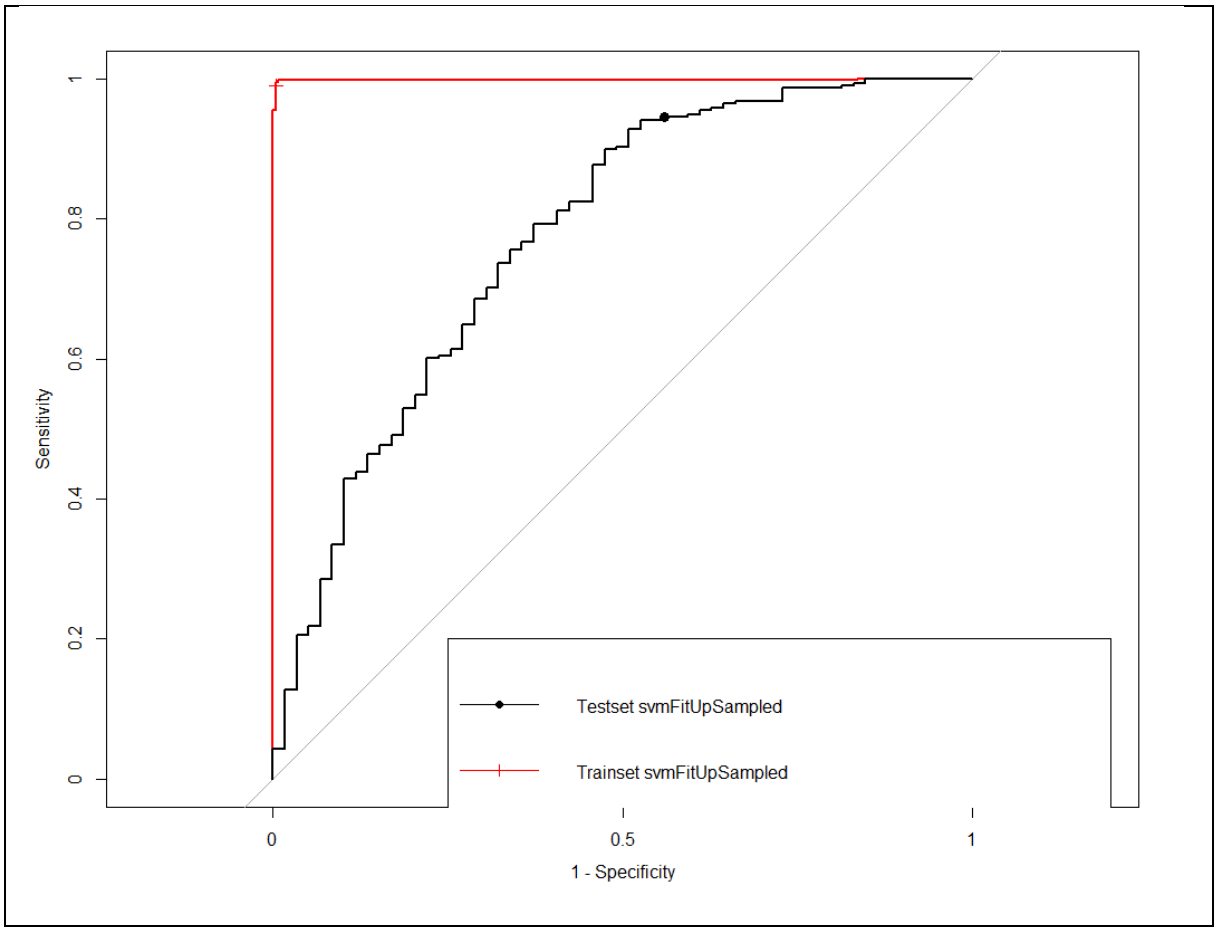
```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 916   4
        X1   9 921

               Accuracy : 0.993
                 95% CI : (0.988, 0.9963)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9859
 Mcnemar's Test P-Value : 0.2673

            Sensitivity : 0.9957
            Specificity : 0.9903
         Pos Pred Value : 0.9903
         Neg Pred Value : 0.9957
             Prevalence : 0.5000
         Detection Rate : 0.4978
   Detection Prevalence : 0.5027
      Balanced Accuracy : 0.9930

       'Positive' Class : X1

Area under the curve: 0.998
```



#results of trained model on test set
testresultsSampling$svmFitUpSampled <- testresultsfunction(svmFitUpSampled, testHRdataDummyFullRankLowCorr, "svmFitUpSampled")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 291  33
        X1  17  26

               Accuracy : 0.8638
                 95% CI : (0.8244, 0.8972)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.11193

                  Kappa : 0.4329
 Mcnemar's Test P-Value : 0.03389

            Sensitivity : 0.44068
            Specificity : 0.94481
         Pos Pred Value : 0.60465
         Neg Pred Value : 0.89815
             Prevalence : 0.16076
         Detection Rate : 0.07084
   Detection Prevalence : 0.11717
      Balanced Accuracy : 0.69274

       'Positive' Class : X1

[1] "F1 score = 0.509803921568627"
Area under the curve: 0.7753
```



 # Support Vector Machines SMOTE

#train the model
set.seed(1247)
svmFitSMOTE <- train(x = trainHRdataDummyFullRankLowCorrSMOTE[, -1], y = trainHRdataDummyFullRankLowCorrSMOTE$Attrition, method = "svmRadial", metric = "ROC", preProc = c("BoxCox", "center", "scale"), tuneGrid = svmRGridReduced, fit = FALSE, trControl = ctrl)
svmFitSMOTE

Outcome:

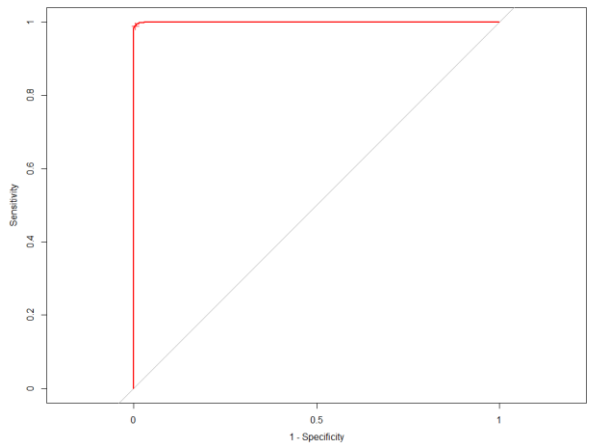
```
Support Vector Machines with Radial Basis Function Kernel

1246 samples
  39 predictor
   2 classes: 'X0', 'X1'

Pre-processing: Box-Cox transformation (14), centered (39), scaled (39)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 1121, 1121, 1121, 1122, 1121, 1122, ...
Resampling results across tuning parameters:

  C       ROC        Sens       Spec
   0.0625  0.8620929  0.7173435  0.8255835
   0.1250  0.8799273  0.8241002  0.7716212
   0.2500  0.8959591  0.8482590  0.7757652
   0.5000  0.9087934  0.8541510  0.7966806
   1.0000  0.9224847  0.8687793  0.8274004
   2.0000  0.9381621  0.8845110  0.8711670
```

```
   4.0000  0.9533407  0.9055829  0.8932914
   8.0000  0.9649410  0.9190336  0.9190846
  16.0000  0.9737213  0.9356103  0.9479804

Tuning parameter 'sigma' was held constant at a value of 0.009475476
ROC was used to select the optimal model using  the largest value.
The final values used for the model were sigma = 0.009475476 and C = 16.
```

#results of trained model
trainresultsSampling$svmFitSMOTE <- trainresultsfunction(svmFitSMOTE, trainHRdataDummyFullRankLowCorrSMOTE)

Outcome:

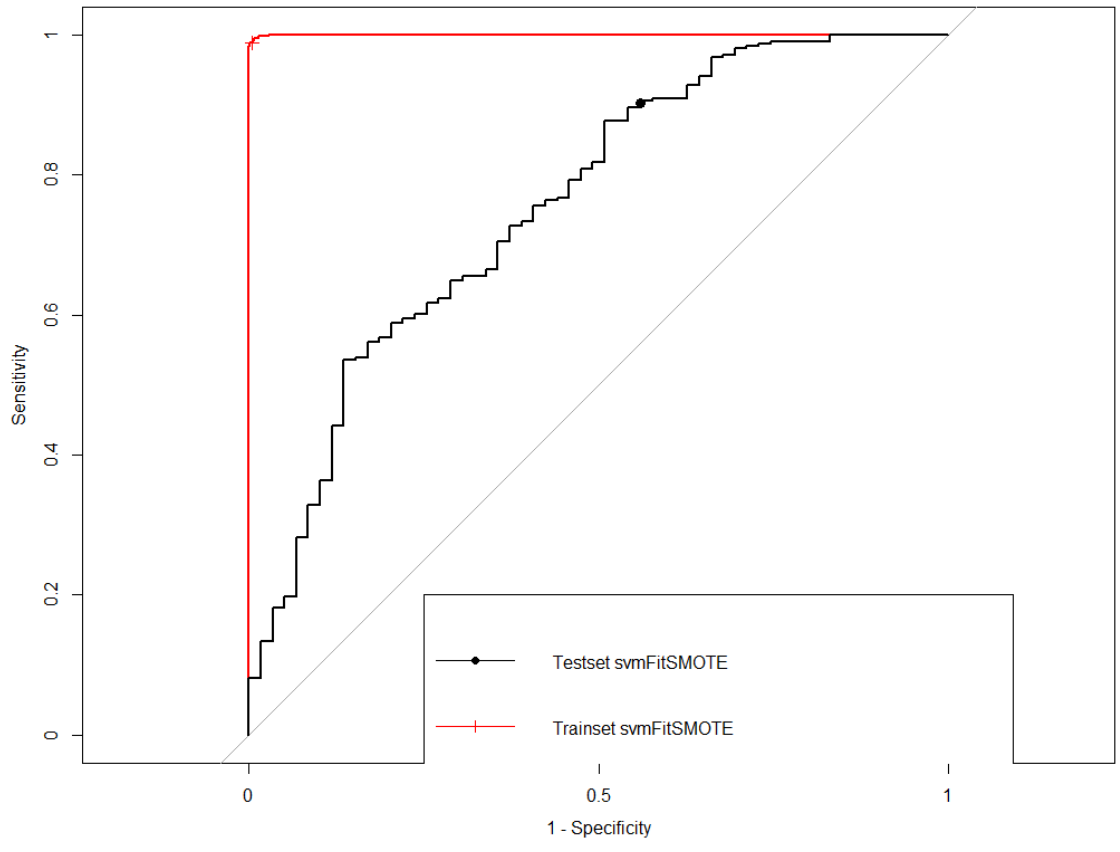
```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 704   2
        X1   8 532

               Accuracy : 0.992
                 95% CI : (0.9853, 0.9961)
    No Information Rate : 0.5714
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9836
 Mcnemar's Test P-Value : 0.1138

            Sensitivity : 0.9963
            Specificity : 0.9888
         Pos Pred Value : 0.9852
         Neg Pred Value : 0.9972
             Prevalence : 0.4286
         Detection Rate : 0.4270
   Detection Prevalence : 0.4334
      Balanced Accuracy : 0.9925

       'Positive' Class : X1

Area under the curve: 0.9999
```



#results of trained model on test set
testresultsSampling$svmFitSMOTE <- testresultsfunction(svmFitSMOTE, testHRdataDummyFullRankLowCorr, "svmFitSMOTE")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 278  33
        X1  30  26

               Accuracy : 0.8283
                 95% CI : (0.7858, 0.8655)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.7418

                  Kappa : 0.3505
 Mcnemar's Test P-Value : 0.8011

            Sensitivity : 0.44068
            Specificity : 0.90260
         Pos Pred Value : 0.46429
         Neg Pred Value : 0.89389
             Prevalence : 0.16076
         Detection Rate : 0.07084
   Detection Prevalence : 0.15259
      Balanced Accuracy : 0.67164

       'Positive' Class : X1

[1] "F1 score = 0.452173913043478"
Area under the curve: 0.7581
```



#----------

```
#Random Forests Dummy set
#----------------------------
mtryValuesDummy <- c(2, 10, 20, 30, 40, 52)

#Random Forests Dummy set UpSampled
set.seed(1247)
rfFitDummyUpSampled <- train(x = trainHRdataDummyUpSampled[, -1],
                             y = trainHRdataDummyUpSampled$Attrition,
                             method = "rf",
                             ntree = 1000,
                             tuneGrid = data.frame(mtry = mtryValuesDummy),
                             importance = TRUE,
                             metric = "ROC",
                             trControl = ctrl)
rfFitDummyUpSampled
```

Outcome:

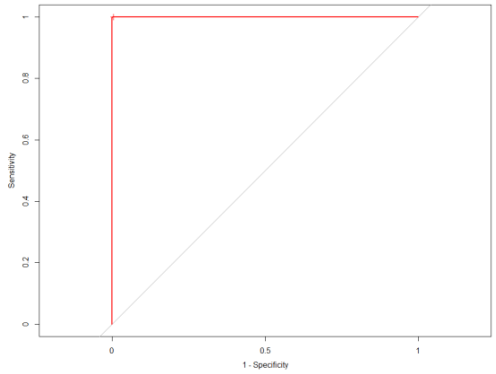
```
Random Forest

1850 samples
  51 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 1664, 1664, 1664, 1665, 1666, 1665, ...
Resampling results across tuning parameters:

  mtry  ROC        Sens       Spec
   2    0.9991652  0.9485133  0.9952408
  10    0.9994187  0.9517485  0.9976157
  20    0.9993058  0.9441842  0.9982655
  30    0.9991848  0.9413791  0.9987003
  40    0.9990302  0.9392146  0.9984853
  52    0.9989803  0.9394320  0.9984853

ROC was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 10.
```

```
#results of trained model
trainresultsSampling$rfFitDummyUpSampled <- trainresultsfunction(rfFitDummyUpSampled,
trainHRdataDummyUpSampled)
```

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 925   0
        X1   0 925

               Accuracy : 1
                 95% CI : (0.998, 1)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1
 Mcnemar's Test P-Value : NA
```

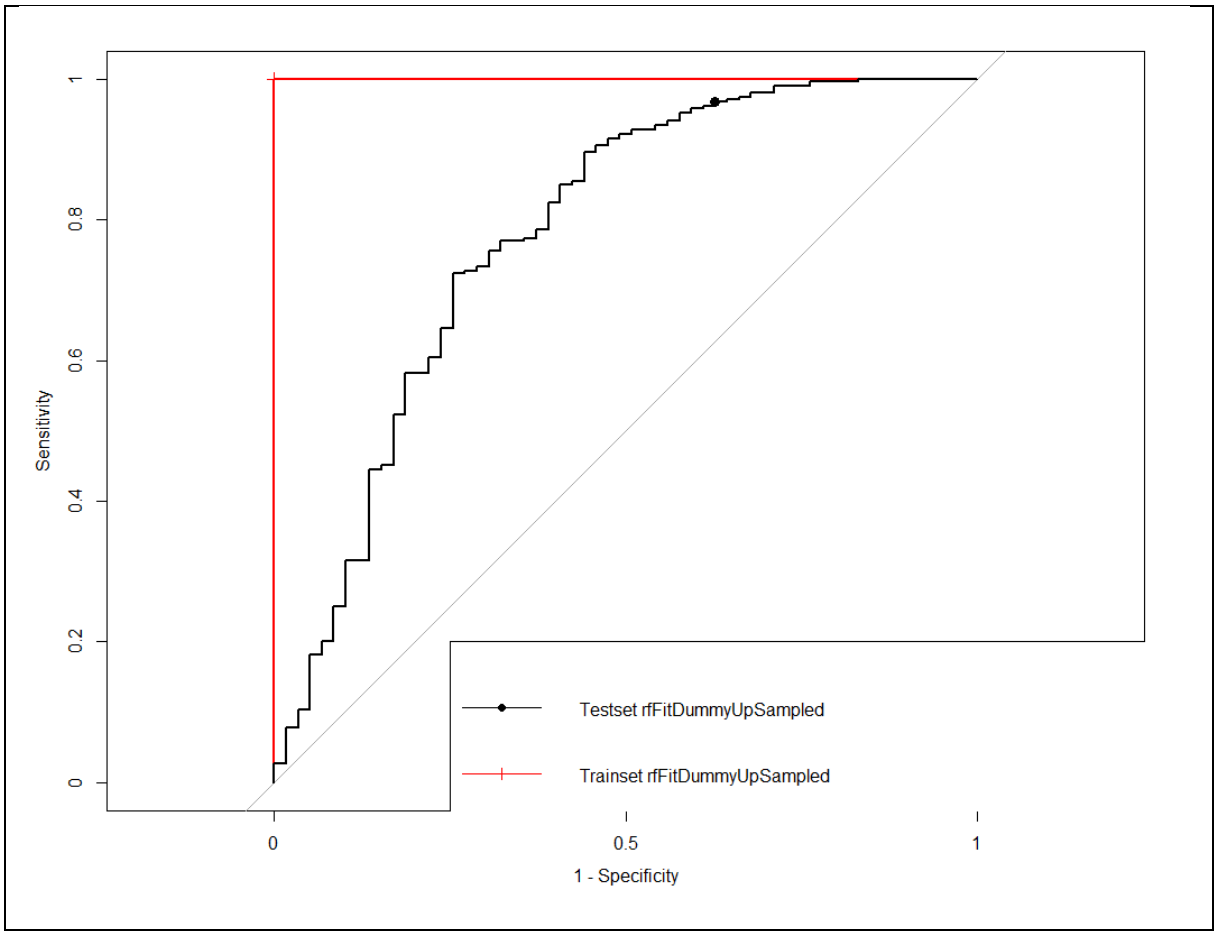
```
            Sensitivity : 1.0
            Specificity : 1.0
         Pos Pred Value : 1.0
         Neg Pred Value : 1.0
             Prevalence : 0.5
         Detection Rate : 0.5
   Detection Prevalence : 0.5
      Balanced Accuracy : 1.0

       'Positive' Class : X1

Area under the curve: 1
```



#results of trained model on test set
testresultsSampling$rfFitDummyUpSampled <- testresultsfunction(rfFitDummyUpSampled, testHRdataDummy, "rfFitDummyUpSampled")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 298  37
        X1  10  22

               Accuracy : 0.8719
                 95% CI : (0.8334, 0.9044)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.0480056

                  Kappa : 0.4177
 Mcnemar's Test P-Value : 0.0001491

            Sensitivity : 0.37288
            Specificity : 0.96753
         Pos Pred Value : 0.68750
         Neg Pred Value : 0.88955
             Prevalence : 0.16076
         Detection Rate : 0.05995
   Detection Prevalence : 0.08719
      Balanced Accuracy : 0.67021

       'Positive' Class : X1

[1] "F1 score = 0.483516483516484"
Area under the curve: 0.7778
```




```
#Random Forests Dummy set SMOTE
set.seed(1247)
rfFitDummySMOTE <- train(x = trainHRdataDummySMOTE[, -1],
                         y = trainHRdataDummySMOTE$Attrition,
                         method = "rf",
                         ntree = 1000,
                         tuneGrid = data.frame(mtry = mtryValuesDummy),
                         importance = TRUE,
                         metric = "ROC",
                         trControl = ctrl)
rfFitDummySMOTE
```

Outcome:

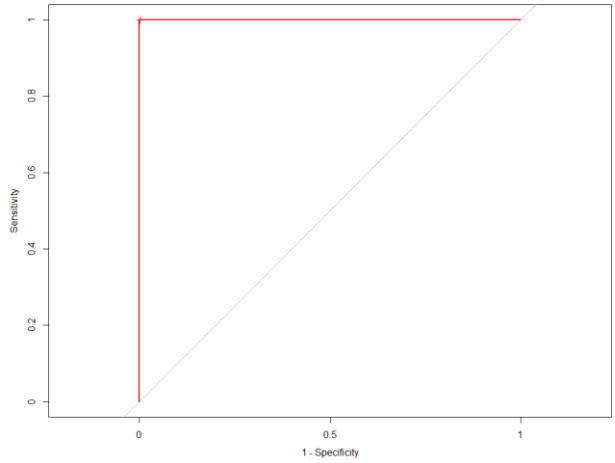
```
Random Forest

1246 samples
  51 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 1121, 1121, 1121, 1122, 1121, 1122, ...
Resampling results across tuning parameters:

  mtry  ROC        Sens       Spec
   2    0.9801181  0.9701956  0.8487212
  10    0.9794091  0.9558764  0.8707757
  20    0.9781573  0.9539085  0.8696506
  30    0.9768240  0.9508294  0.8677778
```

```
  40    0.9759455  0.9519444  0.8674144
  52    0.9749527  0.9527856  0.8614326

ROC was used to select the optimal model using  the largest value.
The final value used for the model was mtry = 2.
```

#results of trained model
trainresultsSampling$rfFitDummySMOTE <- trainresultsfunction(rfFitDummySMOTE, trainHRdataDummySMOTE)

Outcome:

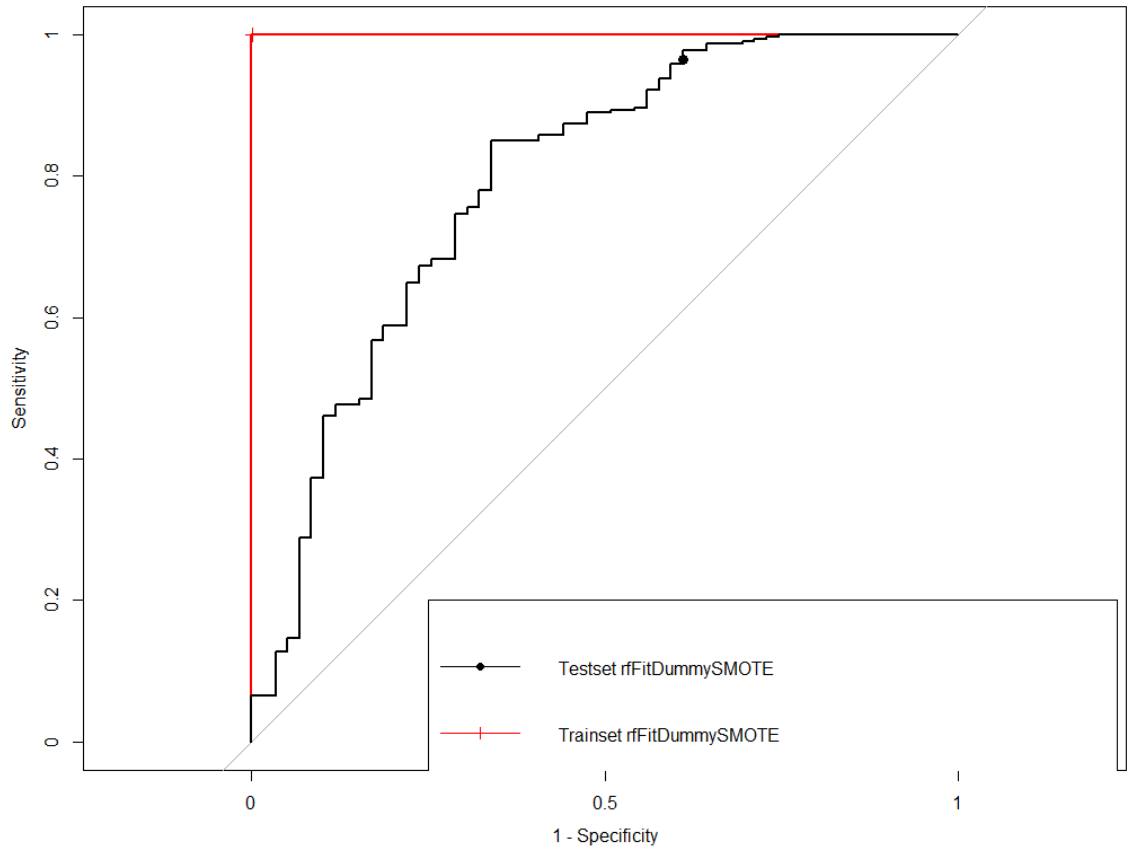
```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 712   1
        X1   0 533

               Accuracy : 0.9992
                 95% CI : (0.9955, 1)
    No Information Rate : 0.5714
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9984
 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9981
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9986
             Prevalence : 0.4286
         Detection Rate : 0.4278
   Detection Prevalence : 0.4278
      Balanced Accuracy : 0.9991

       'Positive' Class : X1

Area under the curve: 1
```



#results of trained model on test set
testresultsSampling$rfFitDummySMOTE <- testresultsfunction(rfFitDummySMOTE, testHRdataDummy, "rfFitDummySMOTE")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 297  36
        X1  11  23

               Accuracy : 0.8719
                 95% CI : (0.8334, 0.9044)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.0480056

                  Kappa : 0.4273
 Mcnemar's Test P-Value : 0.0004639

            Sensitivity : 0.38983
            Specificity : 0.96429
         Pos Pred Value : 0.67647
         Neg Pred Value : 0.89189
             Prevalence : 0.16076
         Detection Rate : 0.06267
   Detection Prevalence : 0.09264
      Balanced Accuracy : 0.67706

       'Positive' Class : X1

[1] "F1 score = 0.494623655913979"
Area under the curve: 0.7911
```



#----------

#Gradient Boosting Machines dummy set

```
#-----------------------------
gbmGrid <- expand.grid(interaction.depth = c(1, 3, 5, 7, 9),
                       n.trees = (1:15)*100,
                       shrinkage = c(.01, .1),
                       n.minobsinnode = 10)
```

#Gradient Boosting Machines dummy set UpSampled

```
set.seed(1247)
gbmFitDummyUpSampled <- train(x = trainHRdataDummyUpSampled[, -1],
                    y = trainHRdataDummyUpSampled$Attrition,
                    method = "gbm",
                    tuneGrid = gbmGrid,
                    metric = "ROC",
                    verbose = FALSE,
                    trControl = ctrl)
gbmFitDummyUpSampled
```

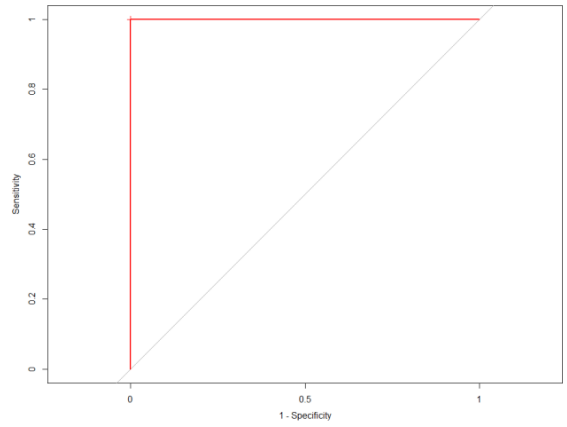
Outcome:

```
Stochastic Gradient Boosting

1850 samples
  51 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 1664, 1664, 1664, 1665, 1666, 1665, ...
Resampling results across tuning parameters:

  shrinkage  interaction.depth  n.trees  ROC        Sens       Spec
  0.10       7                  1200     0.9986824  0.9491748  0.9980552
  0.10       7                  1300     0.9987454  0.9491772  0.9982726
  0.10       7                  1400     0.9987881  0.9489621  0.9980552
  0.10       7                  1500     0.9988163  0.9511220  0.9980552
  0.10       9                   100     0.9877963  0.9093712  0.9755657
  0.10       9                   200     0.9963415  0.9314142  0.9967648
  0.10       9                   300     0.9977463  0.9394320  0.9980552
  0.10       9                   400     0.9985339  0.9411618  0.9980552
  0.10       9                   500     0.9986934  0.9446237  0.9976204
  0.10       9                   600     0.9990249  0.9487307  0.9978378
  0.10       9                   700     0.9991179  0.9500351  0.9976204
  0.10       9                   800     0.9992227  0.9504605  0.9976204
  0.10       9                   900     0.9992433  0.9513277  0.9976204
  0.10       9                  1000     0.9992672  0.9528354  0.9976204
  0.10       9                  1100     0.9993161  0.9519776  0.9976204
  0.10       9                  1200     0.9993284  0.9539224  0.9976204
  0.10       9                  1300     0.9993631  0.9543525  0.9976204
  0.10       9                  1400     0.9992977  0.9537097  0.9976204
  0.10       9                  1500     0.9993094  0.9534946  0.9976204
Note: list was shortened!

Tuning parameter 'n.minobsinnode' was held constant at a value of 10
ROC was used to select the optimal model using  the largest value.
The final values used for the model were n.trees = 1300, interaction.dept
h = 9, shrinkage = 0.1
 and n.minobsinnode = 10.
```

#results of trained model
trainresultsSampling$gbmFitDummyUpSampled <-
trainresultsfunction(gbmFitDummyUpSampled, trainHRdataDummyUpSampled)

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 925   0
        X1   0 925

               Accuracy : 1
                 95% CI : (0.998, 1)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1
 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0
            Specificity : 1.0
         Pos Pred Value : 1.0
         Neg Pred Value : 1.0
             Prevalence : 0.5
         Detection Rate : 0.5
   Detection Prevalence : 0.5
      Balanced Accuracy : 1.0

       'Positive' Class : X1

Area under the curve: 1
```

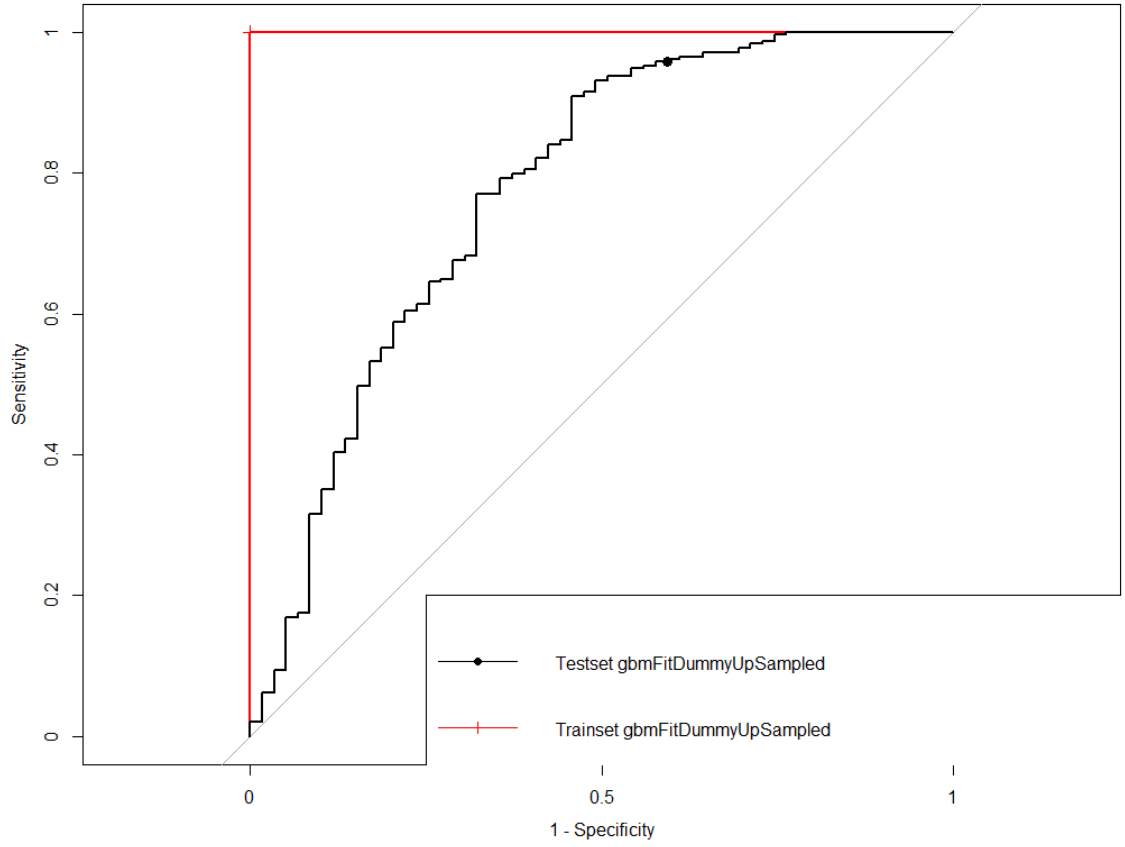


#results of trained model on test set
testresultsSampling$gbmFitDummyUpSampled <-
testresultsfunction(gbmFitDummyUpSampled, testHRdataDummy,
"gbmFitDummyUpSampled")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 295  35
        X1  13  24

               Accuracy : 0.8692
```

```
                 95% CI : (0.8304, 0.902)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.064977

                  Kappa : 0.4293
 Mcnemar's Test P-Value : 0.002437

            Sensitivity : 0.4068
            Specificity : 0.9578
         Pos Pred Value : 0.6486
         Neg Pred Value : 0.8939
             Prevalence : 0.1608
         Detection Rate : 0.0654
   Detection Prevalence : 0.1008
      Balanced Accuracy : 0.6823

       'Positive' Class : X1

[1] "F1 score = 0.5"
Area under the curve: 0.7751
```



#Gradient Boosting Machines dummy set SMOTE

```
set.seed(1247)
gbmFitDummyUpSMOTE <- train(x = trainHRdataDummySMOTE[, -1],
                            y = trainHRdataDummySMOTE$Attrition,
                            method = "gbm",
                            tuneGrid = gbmGrid,
                            metric = "ROC",
                            verbose = FALSE,
```

trControl = ctrl)

gbmFitDummyUpSMOTE

Outcome:

```
Stochastic Gradient Boosting

1246 samples
  51 predictor
   2 classes: 'X0', 'X1'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 1121, 1121, 1121, 1122, 1121, 1122, ...
Resampling results across tuning parameters:

  shrinkage  interaction.depth  n.trees  ROC        Sens       Spec
  0.10       7                  1100     0.9738368  0.9615219  0.8711950
  0.10       7                  1200     0.9743225  0.9626448  0.8726555
  0.10       7                  1300     0.9743821  0.9620775  0.8745563
  0.10       7                  1400     0.9742881  0.9623592  0.8726695
  0.10       7                  1500     0.9741201  0.9615141  0.8741929
  0.10       9                   100     0.9643828  0.9550196  0.8554787
  0.10       9                   200     0.9685343  0.9595305  0.8569811
  0.10       9                   300     0.9709049  0.9586894  0.8633403
  0.10       9                   400     0.9724274  0.9614984  0.8648498
  0.10       9                   500     0.9727322  0.9615141  0.8640811
  0.10       9                   600     0.9731866  0.9612363  0.8689727
  0.10       9                   700     0.9734812  0.9617997  0.8719567
  0.10       9                   800     0.9739389  0.9617919  0.8719357
  0.10       9                   900     0.9741881  0.9620657  0.8734382
  0.10       9                  1000     0.9743662  0.9609546  0.8738015
  0.10       9                  1100     0.9747216  0.9606690  0.8749476
  0.10       9                  1200     0.9747051  0.9615180  0.8749406
  0.10       9                  1300     0.9748111  0.9609468  0.8734312
  0.10       9                  1400     0.9749029  0.9615180  0.8741719
  0.10       9                  1500     0.9750860  0.9612285  0.8753040
Note: list was shortened

Tuning parameter 'n.minobsinnode' was held constant at a value of 10
ROC was used to select the optimal model using  the largest value.
The final values used for the model were n.trees = 1500, interaction.dept
h = 9, shrinkage = 0.1
 and n.minobsinnode = 10.
```

#results of trained model

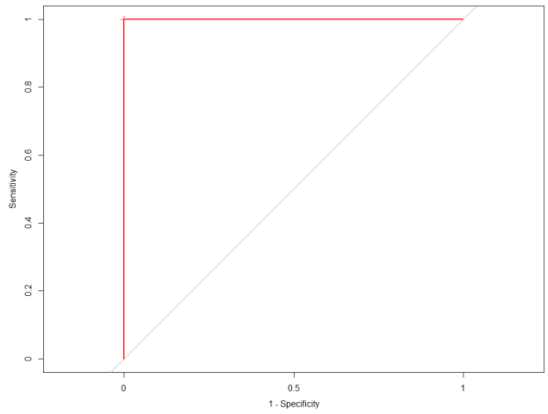
trainresultsSampling$gbmFitDummyUpSMOTE <-

trainresultsfunction(gbmFitDummyUpSMOTE, trainHRdataDummySMOTE)

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 712   0
        X1   0 534

               Accuracy : 1
                 95% CI : (0.997, 1)
    No Information Rate : 0.5714
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1
```

```
 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.4286
         Detection Rate : 0.4286
   Detection Prevalence : 0.4286
      Balanced Accuracy : 1.0000

       'Positive' Class : X1

Area under the curve: 1
```

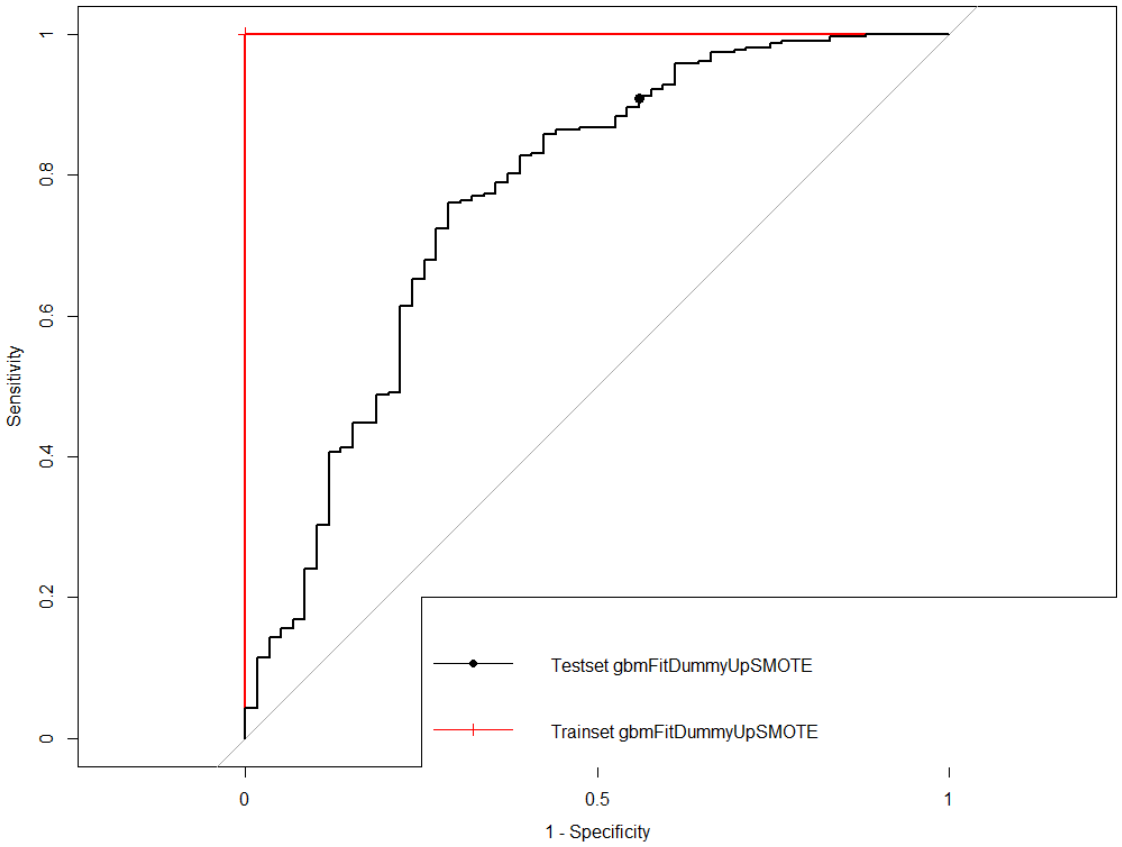


#results of trained model on test set
testresultsSampling$gbmFitDummyUpSMOTE <-
testresultsfunction(gbmFitDummyUpSMOTE, testHRdataDummy,
"gbmFitDummyUpSMOTE")

Outcome:

```
Confusion Matrix and Statistics

          Reference
Prediction  X0  X1
        X0 280  33
        X1  28  26

               Accuracy : 0.8338
                 95% CI : (0.7917, 0.8704)
    No Information Rate : 0.8392
    P-Value [Acc > NIR] : 0.6440

                  Kappa : 0.3622
 Mcnemar's Test P-Value : 0.6085

            Sensitivity : 0.44068
            Specificity : 0.90909
         Pos Pred Value : 0.48148
         Neg Pred Value : 0.89457
             Prevalence : 0.16076
         Detection Rate : 0.07084
   Detection Prevalence : 0.14714
      Balanced Accuracy : 0.67488

       'Positive' Class : X1
```

```
Area under the curve: 0.7673
```




```
#------------------------------------------------------
```

#Alternate Cutoff points

```
#------------------------------------------------------

#compare the results with Alternate Cutoff points

# Create alternate cutoff function
AlternateCutoffs <- function(xx, yy){
  # xx = testresults$(Machine learning algorithm)
  # yy = "Machine learning algorithm"
  RocTest <- roc(response = xx$outcome, predictor = xx$X1, levels =
rev(levels(xx$outcome)))


  Thresh <- coords( RocTest , x = "best", ret=c("threshold", "accuracy", "specificity",
"sensitivity", "ppv"),
                    best.method="closest.topleft")
print(Thresh)

Thresh

plot(RocTest , print.thres = c(.5, Thresh["threshold"]), type = "S",
     print.thres.pattern = "%.3f (Spec = %.2f, Sens = %.2f)",
```

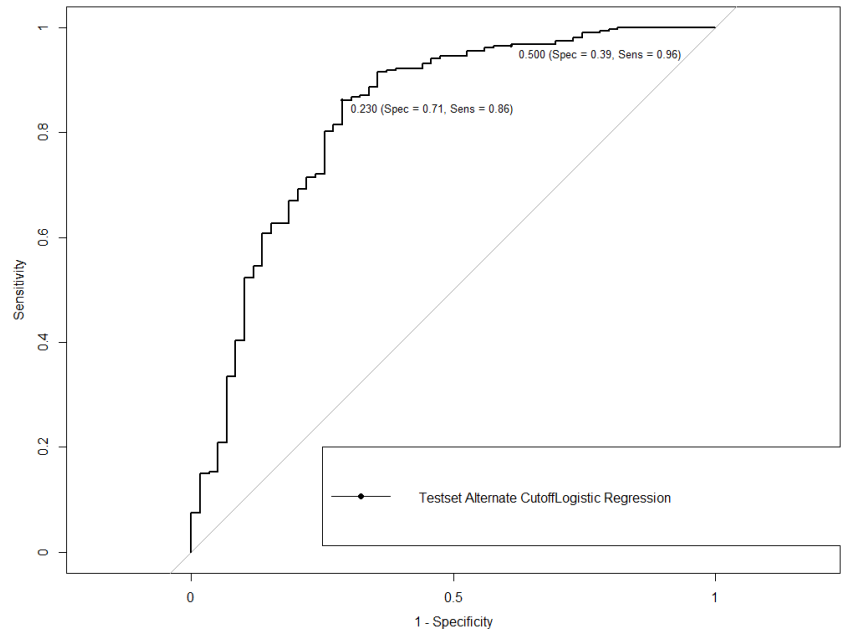
```
    print.thres.cex = .8, legacy.axes = TRUE)
legend(.75, .2,
       str_c("Testset Alternate Cutoff ", yy),
       lwd = 1,
       pch = 16)

ThreshY <- coords(RocTest , x = "best", ret="threshold",
                  best.method="youden")

cutText <- ifelse(Thresh["threshold"] == ThreshY,
                  "is the same as",
                  "is similar to")
print(cutText)

Fscore <- (2*Thresh["specificity"]*Thresh["ppv"]/(Thresh["specificity"]+Thresh["ppv"]))

print(paste(c("F1 score =", Fscore), collapse = " "))
}
```

#apply function on algorithms used

AlternateCutoffs(testresults$logisticReg, "Logistic Regression")

Outcome:

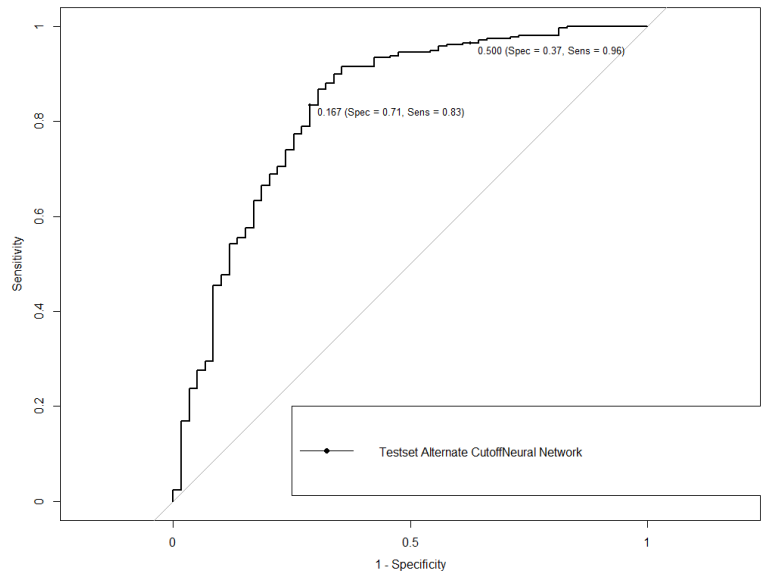
```
  threshold    accuracy specificity sensitivity         ppv
  0.1848141   0.7901907   0.7288136   0.8019481   0.9391635
[1] "is the same as"
[1] "F1 score = 0.820724828065837"
```



AlternateCutoffs(testresults$nnetFit, "Neural Network")

Outcome:

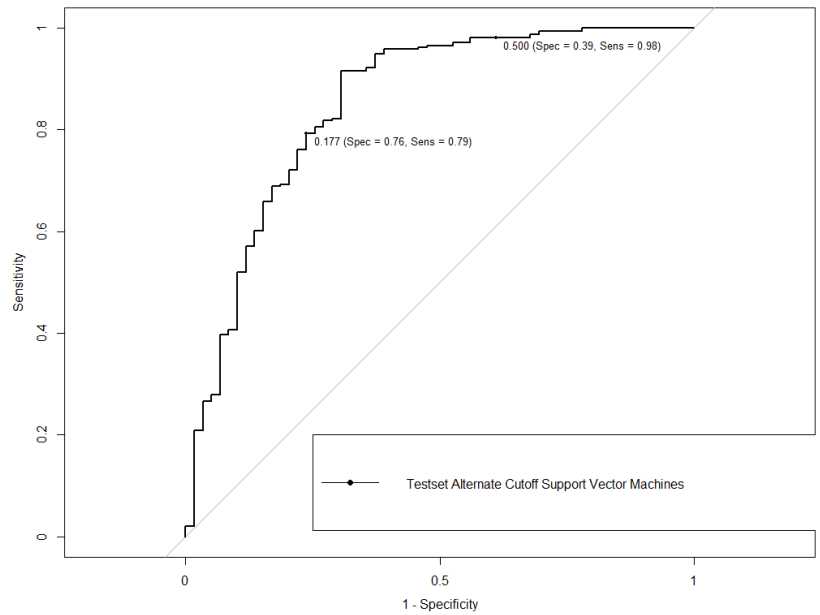
```
  threshold    accuracy specificity sensitivity         ppv
  0.1674948   0.8147139   0.7118644   0.8344156   0.9379562
      threshold
"is similar to"
[1] "F1 score = 0.809418469498706"
```



AlternateCutoffs(testresults$svmFit, "Support Vector Machines")

Outcome:

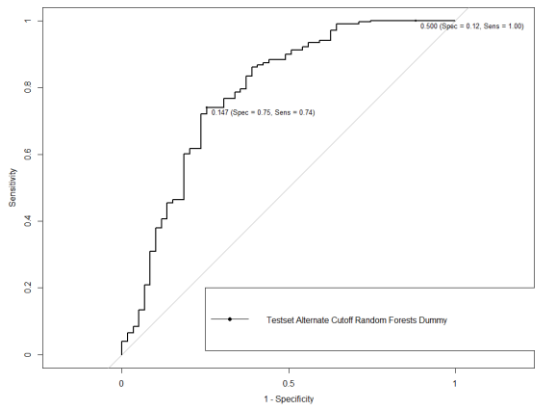
```
  threshold    accuracy specificity sensitivity         ppv
  0.1770735   0.7874659   0.7627119   0.7922078   0.9457364
      threshold
"is similar to"
[1] "F1 score = 0.844420518341921"
```



AlternateCutoffs(testresults$rfFitDummy, "Random Forests Dummy")

Outcome:

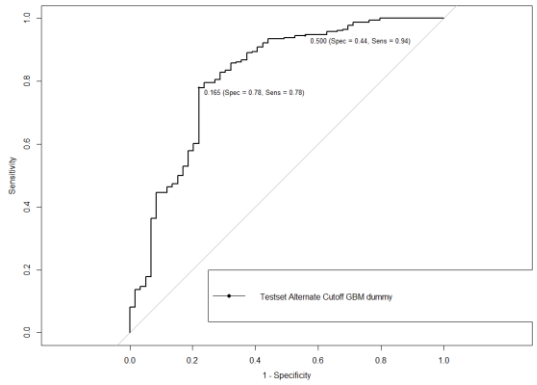
```
  threshold    accuracy specificity sensitivity         ppv
  0.1475000   0.7411444   0.7457627   0.7402597   0.9382716
      threshold
"is the same as"
[1] "F1 score = 0.831013916500994"
```



AlternateCutoffs(testresults$gbmFitDummy, "GBM dummy")

Outcome:

```
  threshold    accuracy specificity sensitivity         ppv
  0.1652439   0.7792916   0.7796610   0.7792208   0.9486166
      threshold
"is the same as"
[1] "F1 score = 0.855880300798512"
```

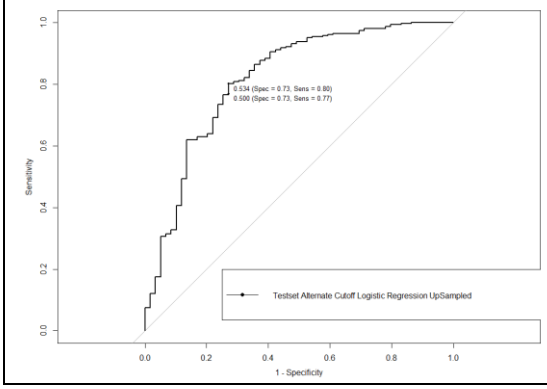


Results of alternate cutoffs on Sampling

AlternateCutoffs(testresultsSampling$logisticRegUpSampled, "Logistic Regression UpSampled")

Outcome:

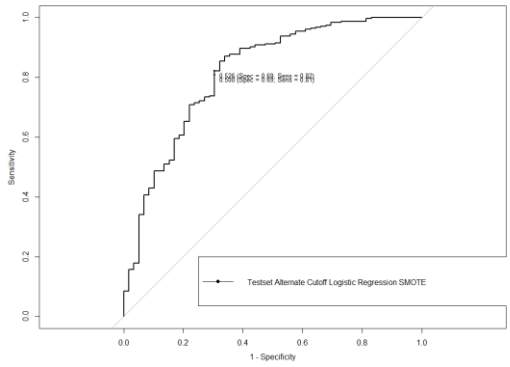
```
  threshold    accuracy specificity sensitivity         ppv
  0.5343487   0.7901907   0.7288136   0.8019481   0.9391635
      threshold
"is the same as"
[1] "F1 score = 0.820724828065837"
```



AlternateCutoffs(testresultsSampling$logisticRegSMOTE, "Logistic Regression SMOTE")

Outcome:

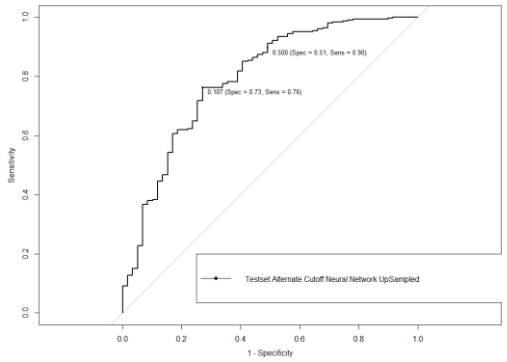
```
  threshold    accuracy specificity sensitivity         ppv
  0.5257691   0.8010899   0.6949153   0.8214286   0.9335793
      threshold
"is similar to"
[1] "F1 score = 0.796758583608572"
```



AlternateCutoffs(testresultsSampling$nnetFitUpSampled, "Neural Network UpSampled")

Outcome:

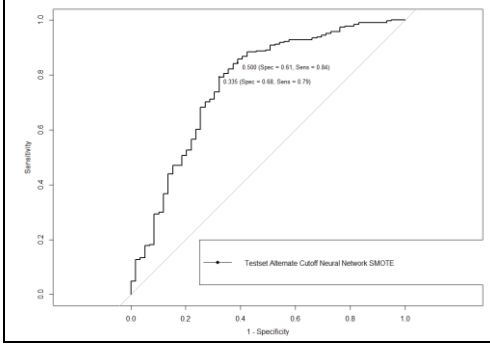
```
  threshold    accuracy specificity sensitivity         ppv
  0.1066153   0.7574932   0.7288136   0.7629870   0.9362550
       threshold
"is the same as"
[1] "F1 score = 0.819612296212183"
```



AlternateCutoffs(testresultsSampling$nnetFitUpSMOTE , "Neural Network SMOTE")

Outcome:

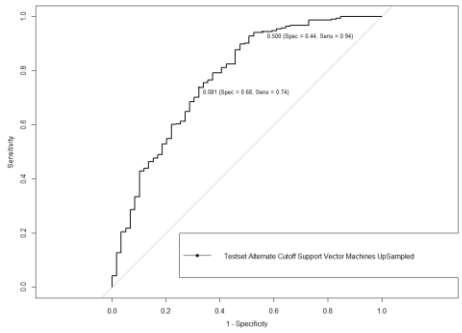
```
  threshold    accuracy specificity sensitivity         ppv
  0.3349407   0.7738420   0.6779661   0.7922078   0.9277567
       threshold
"is the same as"
[1] "F1 score = 0.783432332637663"
```



AlternateCutoffs(testresultsSampling$svmFitUpSampled, "Support Vector Machines UpSampled")

Outcome:

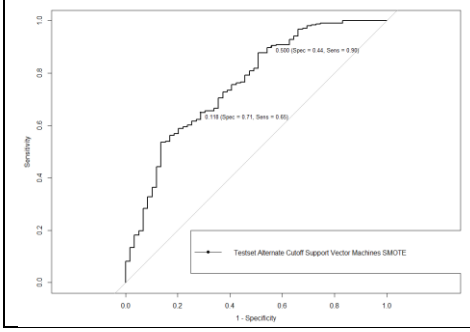
```
  threshold    accuracy specificity sensitivity         ppv
 0.08109166  0.72752044  0.67796610  0.73701299  0.92276423
      threshold
"is similar to"
[1] "F1 score = 0.781646795506392"
```



AlternateCutoffs(testresultsSampling$svmFitSMOTE, "Support Vector Machines SMOTE")

Outcome:

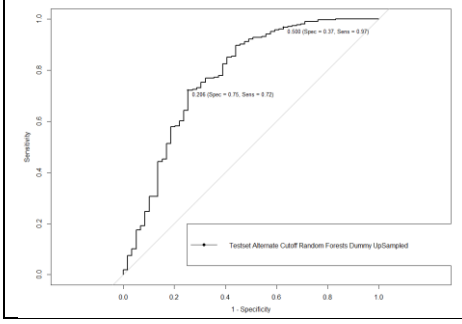
```
  threshold    accuracy specificity sensitivity         ppv
  0.1183862   0.6594005   0.7118644   0.6493506   0.9216590
      threshold
"is similar to"
[1] "F1 score = 0.803289662427082"
```



AlternateCutoffs(testresultsSampling$rfFitDummyUpSampled, "Random Forests Dummy UpSampled")

Outcome:

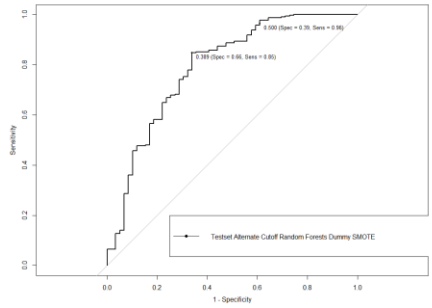
```
  threshold    accuracy specificity sensitivity         ppv
  0.2065000   0.7247956   0.7457627   0.7207792   0.9367089
      threshold
"is the same as"
[1] "F1 score = 0.830400408059169"
```



AlternateCutoffs(testresultsSampling$rfFitDummySMOTE, "Random Forests Dummy SMOTE")

Outcome:

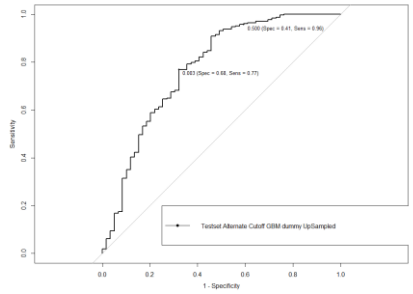
```
  threshold    accuracy specificity sensitivity         ppv
  0.3885000   0.8174387   0.6610169   0.8474026   0.9288256
       threshold
"is the same as"
[1] "F1 score = 0.772365126337355"
```



AlternateCutoffs(testresultsSampling$gbmFitDummyUpSampled, "GBM dummy UpSampled")

Outcome:

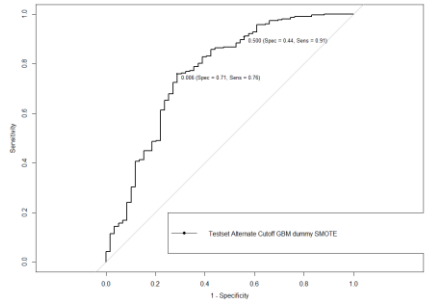
```
  threshold    accuracy specificity sensitivity         ppv
0.002546103 0.754768392 0.677966102 0.769480519 0.925781250
      threshold
"is similar to"
[1] "F1 score = 0.78272716013706"
```



AlternateCutoffs(testresultsSampling$gbmFitDummyUpSMOTE, "GBM dummy SMOTE")

Outcome:

```
  threshold    accuracy specificity sensitivity         ppv
0.006469846 0.752043597 0.711864407 0.759740260 0.932270916
       threshold
"is the same as"
[1] "F1 score = 0.807294233612617"
```

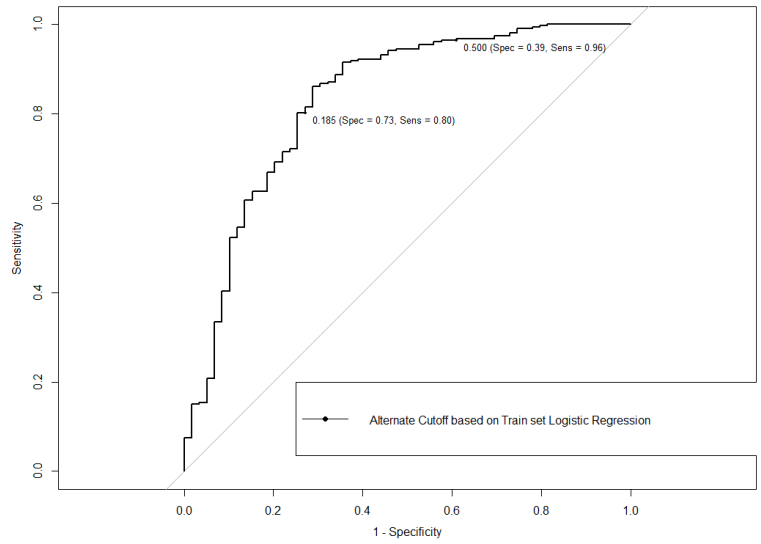


#----------

```
#apply function on algorithms used based on trainingset.
#------------------------------------------------------------

#First Create alternate cutoff function based on trainset.
AlternateCutoffstrain <- function(xx, yy, zz){
  # xx = trainresults$(Machine learning algorithm)
  # yy = "Machine learning algorithm"
  # zz = testresults$(Machine learning algorithm)
  RocTrain <- roc(response = xx$outcome, predictor = xx$X1, levels =
rev(levels(xx$outcome)))
  RocTest <- roc(response = zz$outcome, predictor = zz$X1, levels =
rev(levels(zz$outcome)))

  Threshtrain <- coords( RocTrain , x = "best", ret=c("threshold"),
                  best.method="closest.topleft")
  Threshtest <- coords( RocTest , x = Threshtrain, input = "threshold", ret=c("threshold",
"accuracy", "specificity", "sensitivity", "ppv"))

  print(Threshtest)

  plot(RocTest , print.thres = c(.5, Threshtrain), type = "S",
       print.thres.pattern = "%.3f (Spec = %.2f, Sens = %.2f)",
       print.thres.cex = .8, legacy.axes = TRUE)
  legend(.75, .2,
         str_c("Alternate Cutoff based on Train set ", yy),
         lwd = 1,
         pch = 16)

  ThreshY <- coords(RocTrain , x = "best", ret="threshold",
                  best.method="youden")

  cutText <- ifelse(Threshtrain == ThreshY,
                  "is the same as",
                  "is similar to")
  print(cutText)
  Fscore <-
(2*Threshtest["specificity"]*Threshtest["ppv"]/(Threshtest["specificity"]+Threshtest["ppv"]))

  print(paste(c("F1 score =", Fscore), collapse = " "))

}

AlternateCutoffstrain(trainresults$logisticReg, "Logistic Regression", testresults$logisticReg)
```

Outcome:

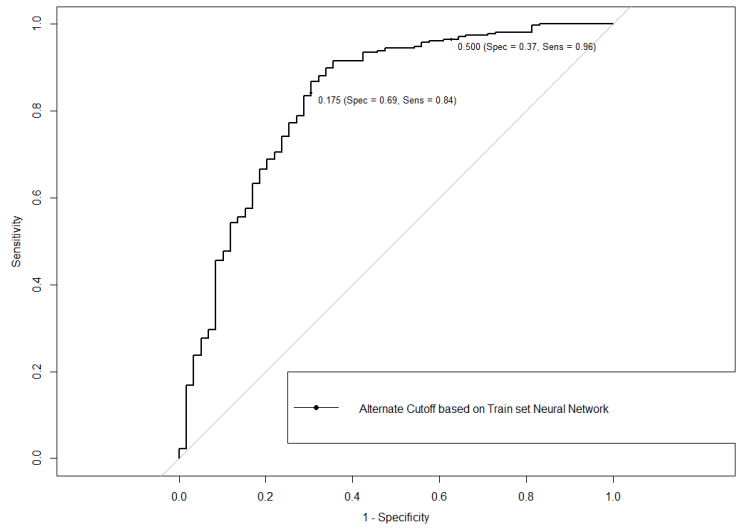
```
  threshold    accuracy specificity sensitivity         ppv
  0.1848141   0.7901907   0.7288136   0.8019481   0.9391635
[1] "is the same as"
[1] "F1 score = 0.820724828065837"
```



AlternateCutoffstrain(trainresults$nnetFit, "Neural Network", testresults$nnetFit)

Outcome:

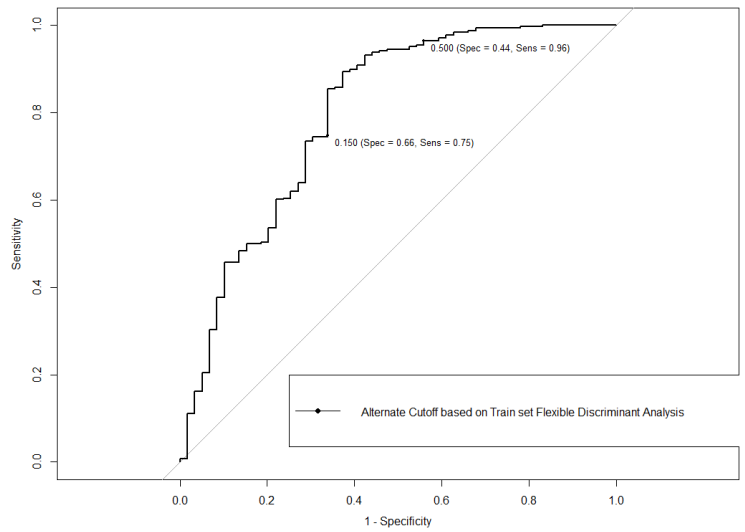
```
  threshold    accuracy specificity sensitivity         ppv
  0.1753550   0.8174387   0.6949153   0.8409091   0.9350181
[1] "is similar to"
[1] "F1 score = 0.797282078234102"
```



AlternateCutoffstrain(trainresults$fdaFit, "Flexible Discriminant Analysis", testresults$fdaFit)

Outcome:

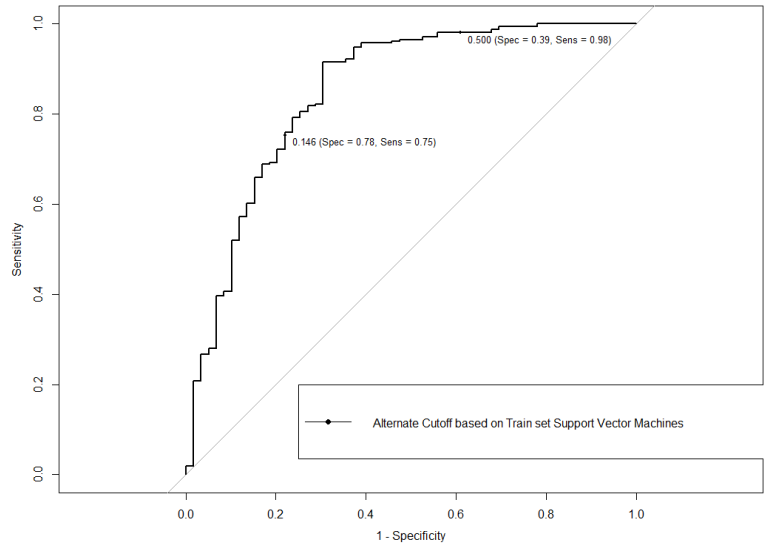
```
  threshold    accuracy specificity sensitivity         ppv
  0.1504595   0.7329700   0.6610169   0.7467532   0.9200000
[1] "is the same as"
[1] "F1 score = 0.769296740994854"
```



AlternateCutoffstrain(trainresults$svmFit, "Support Vector Machines", testresults$svmFit)

Outcome:

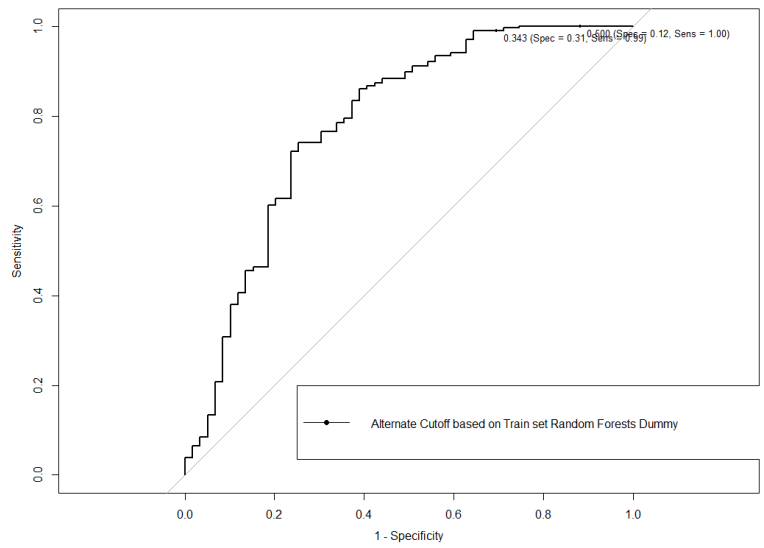
```
  threshold    accuracy specificity sensitivity         ppv
  0.1458358   0.7574932   0.7796610   0.7532468   0.9469388
[1] "is the same as"
[1] "F1 score = 0.855196730507252"
```



AlternateCutoffstrain(trainresults$rfFitDummy, "Random Forests Dummy", testresults$rfFitDummy)

Outcome:

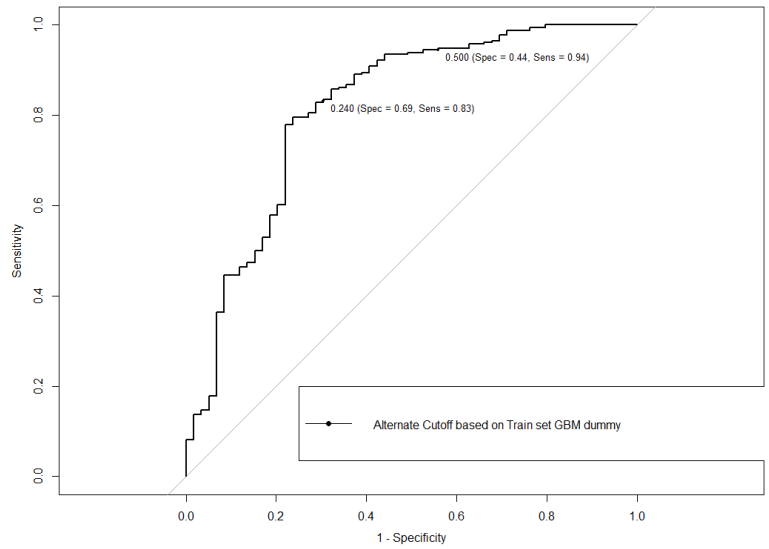
```
  threshold    accuracy specificity sensitivity         ppv
  0.3425000   0.8801090   0.3050847   0.9902597   0.8815029
[1] "is the same as"
[1] "F1 score = 0.453288197167981"
```



AlternateCutoffstrain(trainresults$gbmFitDummy, "GBM dummy", testresults$gbmFitDummy)

Outcome:

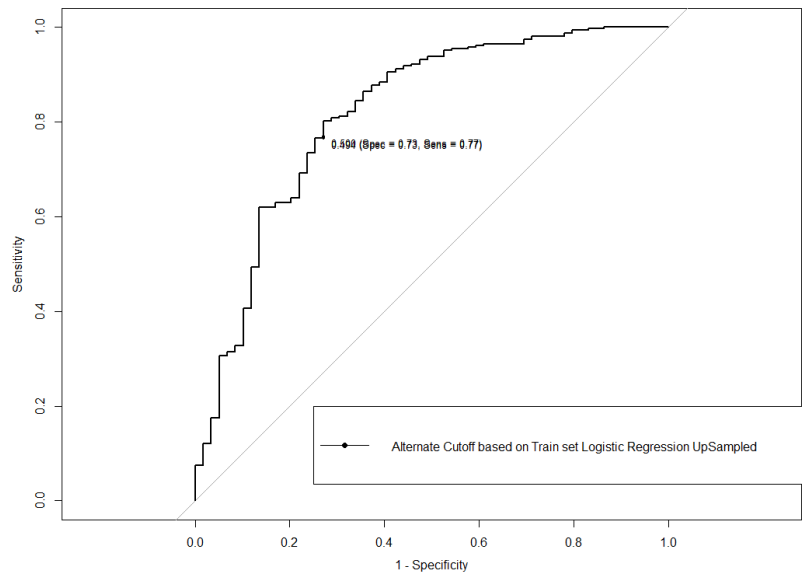
```
  threshold    accuracy specificity sensitivity         ppv
  0.2399141   0.8092643   0.6949153   0.8311688   0.9343066
[1] "is the same as"
[1] "F1 score = 0.797023312324398"
```



AlternateCutoffstrain(trainresultsSampling$logisticRegUpSampled, "Logistic Regression UpSampled",
testresultsSampling$logisticRegUpSampled)

Outcome:

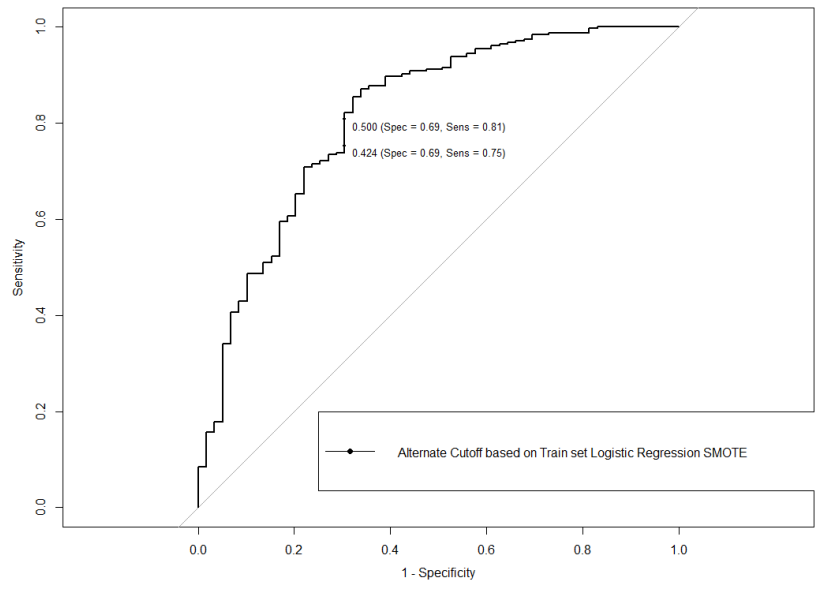
```
  threshold    accuracy specificity sensitivity         ppv
  0.4943143   0.7602180   0.7288136   0.7662338   0.9365079
[1] "is similar to"
[1] "F1 score = 0.819709208400646"
```



AlternateCutoffstrain(trainresultsSampling$logisticRegSMOTE, "Logistic Regression SMOTE",
testresultsSampling$logisticRegSMOTE)

Outcome:

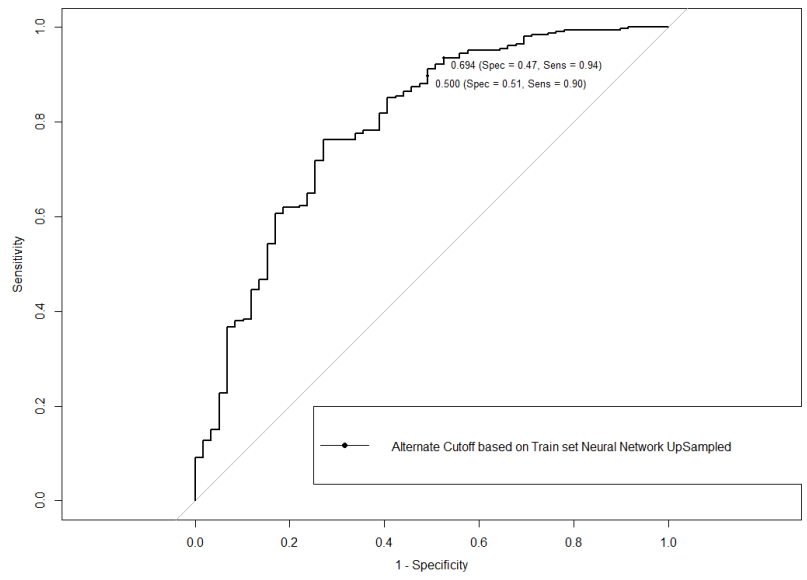
```
  threshold    accuracy specificity sensitivity         ppv
  0.4240535   0.7438692   0.6949153   0.7532468   0.9280000
         best             best
threshold "is the same as" "is similar to"
[1] "F1 score = 0.794719692539059"
```



AlternateCutoffstrain(trainresultsSampling$nnetFitUpSampled, "Neural Network UpSampled",
testresultsSampling$nnetFitUpSampled)

Outcome:

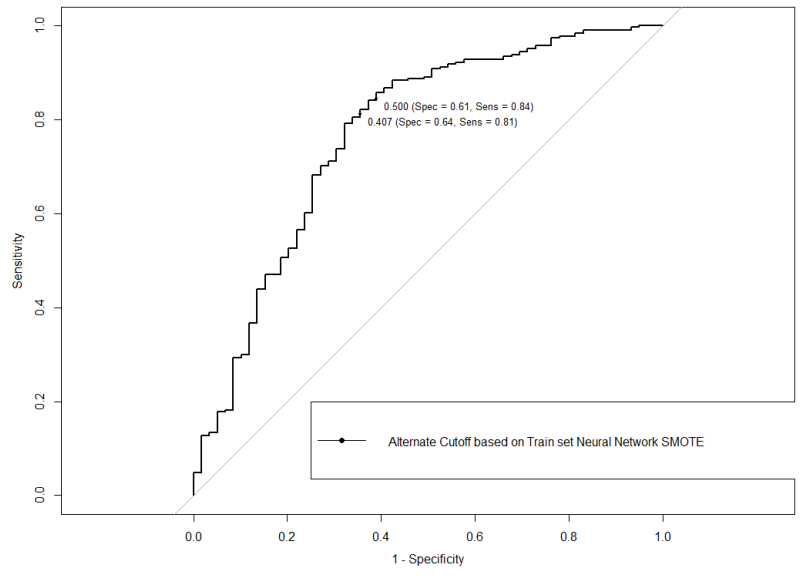
```
  threshold    accuracy specificity sensitivity         ppv
  0.6938504   0.8610354   0.4745763   0.9350649   0.9028213
[1] "is the same as"
[1] "F1 score = 0.622126215090264"
```



AlternateCutoffstrain(trainresultsSampling$nnetFitUpSMOTE , "Neural Network SMOTE",
testresultsSampling$nnetFitUpSMOTE)

Outcome:

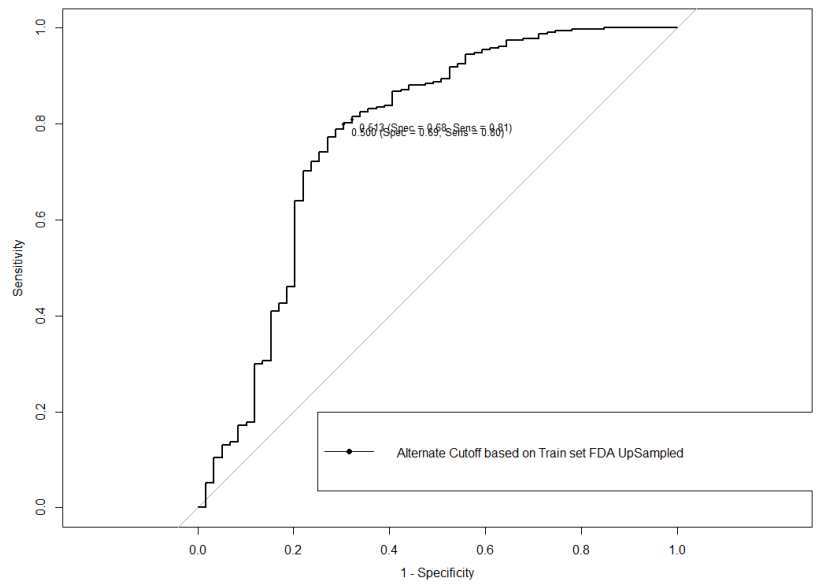
```
  threshold    accuracy specificity sensitivity         ppv
  0.4067063   0.7847411   0.6440678   0.8116883   0.9225092
[1] "is the same as"
[1] "F1 score = 0.758543596295113"
```



AlternateCutoffstrain(trainresultsSampling$fdaFitupsampled, "FDA UpSampled",
testresultsSampling$fdaFitupsampled)

Outcome:

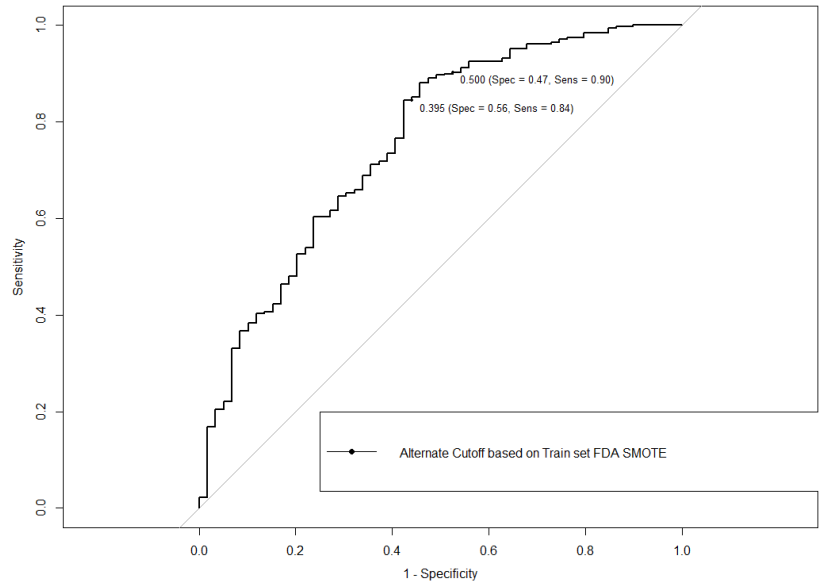
```
  threshold    accuracy specificity sensitivity         ppv
  0.5131750   0.7874659   0.6779661   0.8084416   0.9291045
[1] "is the same as"
[1] "F1 score = 0.783912478847743"
```



AlternateCutoffstrain(trainresultsSampling$fdaFitSMOTE , "FDA SMOTE",
testresultsSampling$fdaFitSMOTE)

Outcome:

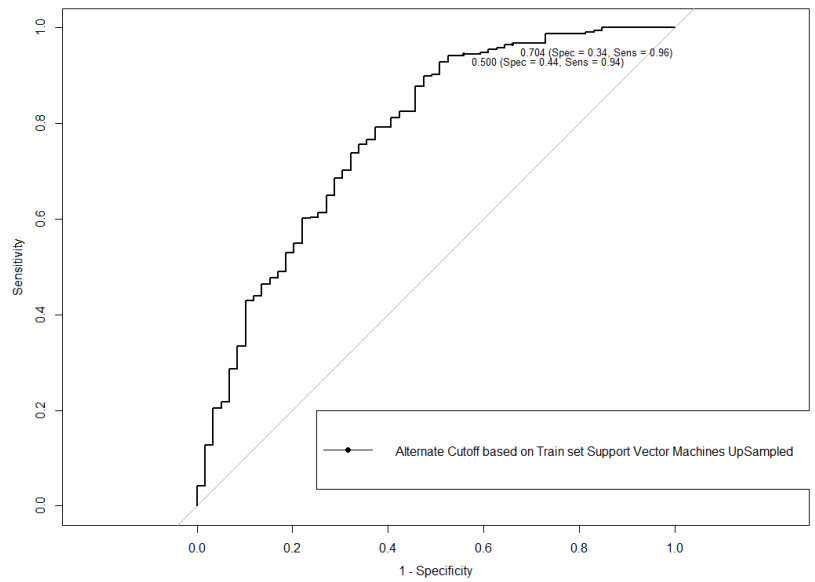
```
  threshold    accuracy specificity sensitivity         ppv
  0.3949082   0.7983651   0.5593220   0.8441558   0.9090909
[1] "is similar to"
[1] "F1 score = 0.692549842602308"
```



AlternateCutoffstrain(trainresultsSampling$svmFitUpSampled, "Support Vector Machines UpSampled", testresultsSampling$svmFitUpSampled)

Outcome:

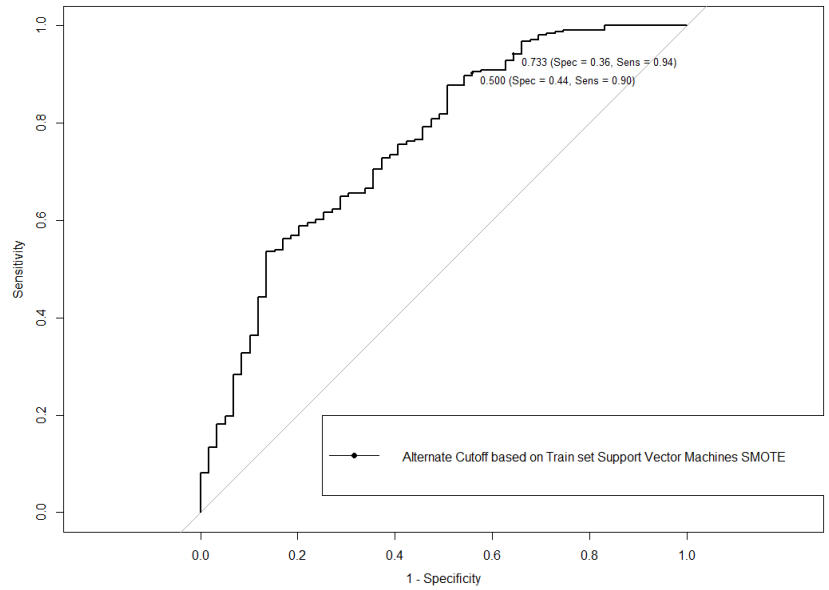
```
  threshold    accuracy specificity sensitivity         ppv
  0.7041408   0.8637602   0.3389831   0.9642857   0.8839286
         best            best
threshold "is similar to" "is the same as"
[1] "F1 score = 0.490038361588912"
```



AlternateCutoffstrain(trainresultsSampling$svmFitSMOTE, "Support Vector Machines SMOTE",
testresultsSampling$svmFitSMOTE)

Outcome:

```
  threshold    accuracy specificity sensitivity         ppv
  0.7329880   0.8474114   0.3559322   0.9415584   0.8841463
          best              best
threshold "is the same as" "is similar to"
[1] "F1 score = 0.507542295191266"
```

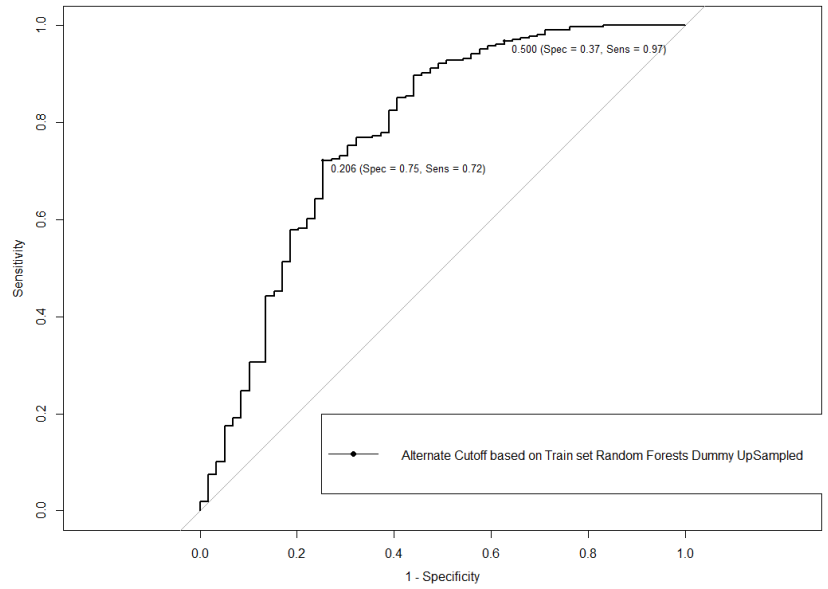


AlternateCutoffstrain(testresultsSampling$rfFitDummyUpSampled, "Random Forests Dummy UpSampled",
testresultsSampling$rfFitDummyUpSampled)

Outcome:

```
  threshold    accuracy specificity sensitivity         ppv
  0.2065000   0.7247956   0.7457627   0.7207792   0.9367089
```

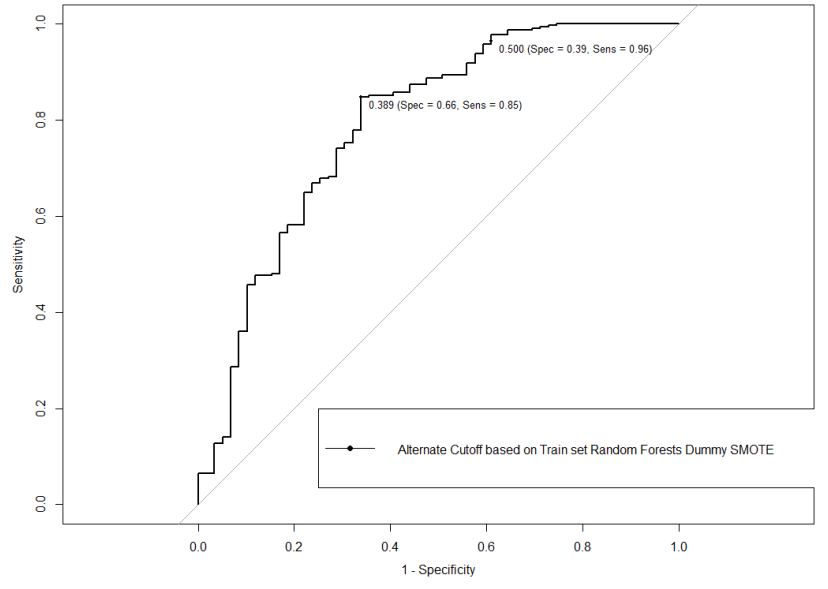
```
[1] "is the same as"
[1] "F1 score = 0.830400408059169"
```



AlternateCutoffstrain(testresultsSampling$rfFitDummySMOTE, "Random Forests Dummy SMOTE",
testresultsSampling$rfFitDummySMOTE)

Outcome:

```
  threshold    accuracy specificity sensitivity         ppv
  0.3885000   0.8174387   0.6610169   0.8474026   0.9288256
[1] "is the same as"
[1] "F1 score = 0.772365126337355"
```

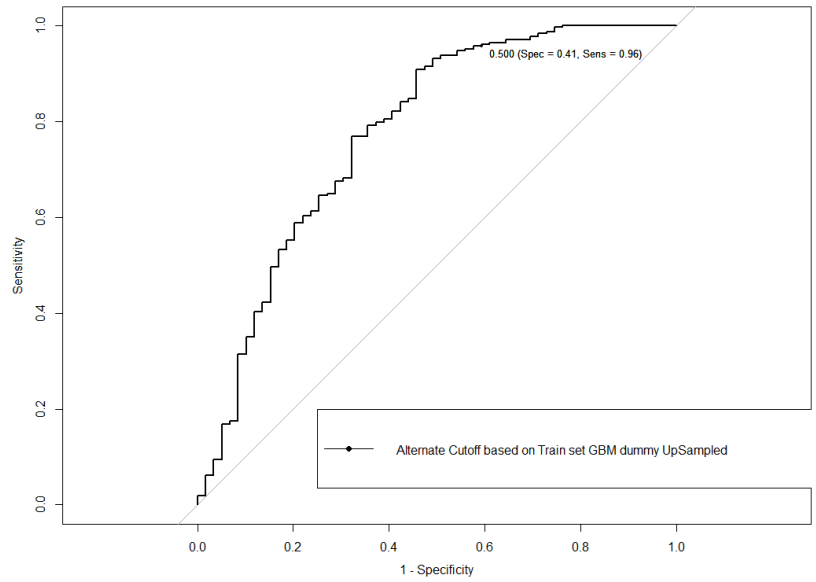


AlternateCutoffstrain(trainresultsSampling$gbmFitDummyUpSampled, "GBM dummy UpSampled",
testresultsSampling$gbmFitDummyUpSampled)

Outcome:

```
  threshold    accuracy specificity sensitivity         ppv
```

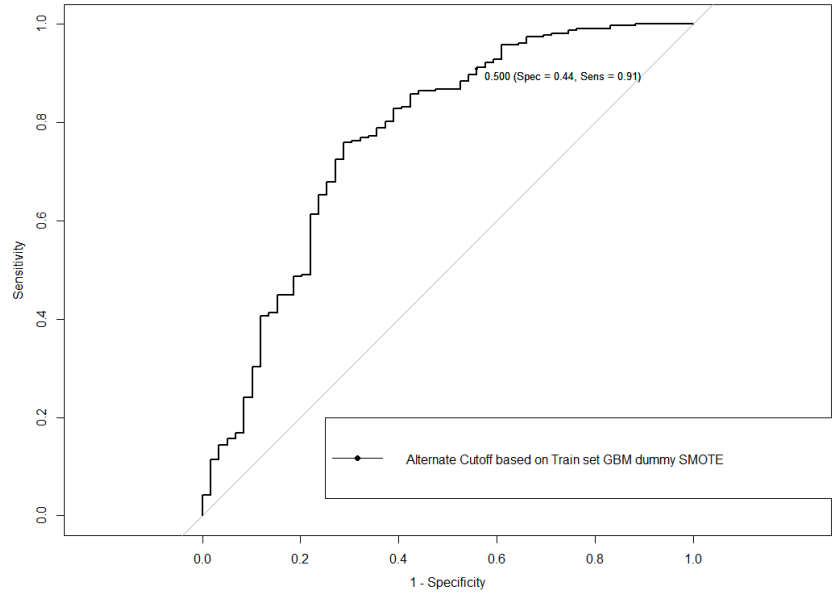
```
  0.5000121   0.8692098   0.4067797   0.9577922   0.8939394
[1] "is the same as"
[1] "F1 score = 0.559131293188549"
```



AlternateCutoffstrain(trainresultsSampling$gbmFitDummyUpSMOTE, "GBM dummy SMOTE",
testresultsSampling$gbmFitDummyUpSMOTE)

Outcome:

```
  threshold    accuracy specificity sensitivity         ppv
  0.5000006   0.8337875   0.4406780   0.9090909   0.8945687
[1] "is the same as"
[1] "F1 score = 0.590477735420553"
```



#Hypothesis testing

#----------

```
#Logistic Regression predictor significance and importance calculation
#-----------------------------------

#first make outcome variable a factor to be useful for classification.
HRdataDummyFullRankLowCorr$Attrition <-
as.factor(HRdataDummyFullRankLowCorr$Attrition)

#change the name of factor, so it can be computed.
HRdataDummyFullRankLowCorr$Attrition <-
make.names(HRdataDummyFullRankLowCorr$Attrition, unique = FALSE, allow_ = TRUE)

#Transform each X0 (= to no attrition) into X2 to make results more interpretable.
for(i in 1:length(HRdataDummyFullRankLowCorr)){
  if(HRdataDummyFullRankLowCorr[i, ] == "X0"){HRdataDummyFullRankLowCorr[i,
"Attrition"] <- "X2"}
}

#Transform each X0 (= to no attrition) into X2 to make results more interpretable.
for(i in 1:nrow(HRdataDummyFullRankLowCorr)){
  if(HRdataDummyFullRankLowCorr[i, "Attrition"] == "X0"){HRdataDummyFullRankLowCorr[i,
"Attrition"] <- "X2"}
}

#change levels of $attrition
HRdataDummyFullRankLowCorr$Attrition <-
factor(HRdataDummyFullRankLowCorr$Attrition)
levels(HRdataDummyFullRankLowCorr$Attrition)
```

Outcome:

```
[1] "X1" "X2"
```

```
#create dataset including years at company
## first delete total working years
dput(names(trainHRdataDummyFullRankLowCorr))

HRdataDummyFullRankLowCorr2 <- HRdataDummyFullRankLowCorr[ , c("Attrition", "Age",
"NumCompaniesWorked",
                                                               "YearsInCurrentRole", "YearsSinceLastPromotion",
"YearsWithCurrManager",
                                                               "TrainingTimesLastYear", "HourlyRate", "DailyRate",
"MonthlyRate",
                                                               "MonthlyIncome", "PercentSalaryHike",
"StockOptionLevel", "Education",
                                                               "DistanceFromHome", "JobInvolvement",
"PerformanceRating", "EnvironmentSatisfaction",
                                                               "JobSatisfaction", "RelationshipSatisfaction",
"WorkLifeBalance",
                                                               "Gender.Male", "MaritalStatus.Married",
"MaritalStatus.Single",
                                                               "OverTime.Yes", "EducationField.Life Sciences",
"EducationField.Marketing",
                                                               "EducationField.Medical", "EducationField.Other",
"EducationField.Technical Degree",
                                                               "BusinessTravel.Travel_Frequently",
"JobRole.Human Resources",
```

```
                                          "JobRole.Laboratory Technician",
"JobRole.Manager", "JobRole.Manufacturing Director",
                                          "JobRole.Research Director", "JobRole.Research
Scientist", "JobRole.Sales Executive",
                                          "JobRole.Sales Representative")]
HRdataDummyFullRankLowCorr2$YearsAtCompany <- HRdata$YearsAtCompany

# Next, Train the model
set.seed(1247)
logisticRegVarImp2 <- train(HRdataDummyFullRankLowCorr2[ , -1], y =
HRdataDummyFullRankLowCorr2$Attrition,  method = "glm",
                  preProc = c("BoxCox"),
                  metric = "ROC", trControl = ctrl)
logisticRegVarImp2
```

Outcome:

```
Generalized Linear Model

1470 samples
  39 predictor
   2 classes: 'X1', 'X2'

Pre-processing: Box-Cox transformation (13)
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 1323, 1323, 1322, 1324, 1323, 1323, ...
Resampling results:

  ROC        Sens       Spec
  0.8379086  0.4306522  0.9644781
```

```
#---------------------------------------------------------------------
#confusion matrix and roc curve for train set.
#The basic predict call evaluates new samples, and type = "prob" returns the class
probabilities.
TrainPredictlogisticRegVarImp <- predict(logisticRegVarImp,
HRdataDummyFullRankLowCorr[, -1], type = "prob")
TrainPredictlogisticRegVarImp$class <- predict(logisticRegVarImp,
HRdataDummyFullRankLowCorr[, -1])
TrainPredictlogisticRegVarImp$outcome <- HRdataDummyFullRankLowCorr$Attrition
TrainPredictlogisticRegVarImp$outcome <-
as.factor(TrainPredictlogisticRegVarImp$outcome)

#Confusion matrix for trainset:
cm <- confusionMatrix(data = TrainPredictlogisticRegVarImp$class, reference =
TrainPredictlogisticRegVarImp$outcome, positive = "X1")
cm
```

Outcome:

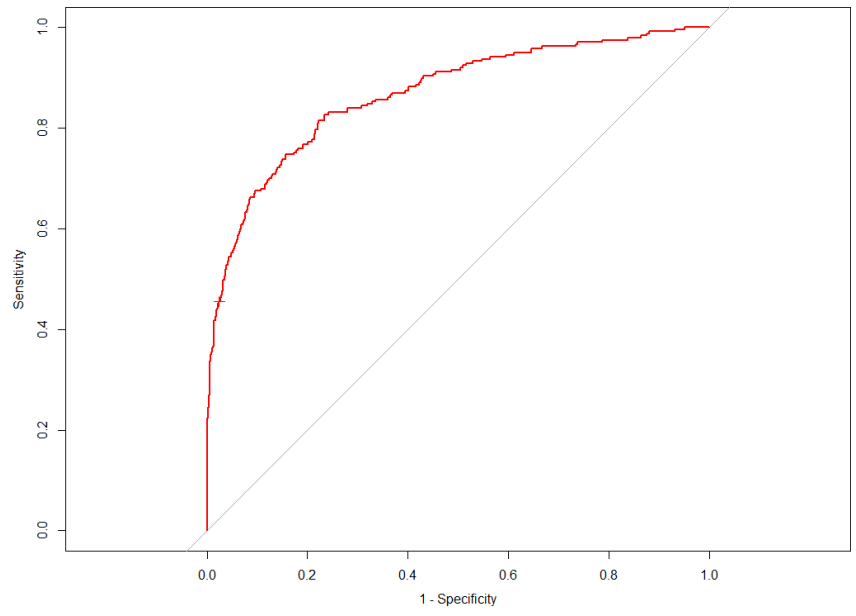
```
Confusion Matrix and Statistics

          Reference
Prediction   X1   X2
        X1  108   30
        X2  129 1203

               Accuracy : 0.8918
                 95% CI : (0.8748, 0.9073)
    No Information Rate : 0.8388
```

```
    P-Value [Acc > NIR] : 3.866e-09

                  Kappa : 0.5189
 Mcnemar's Test P-Value : 7.731e-15

            Sensitivity : 0.45570
            Specificity : 0.97567
         Pos Pred Value : 0.78261
         Neg Pred Value : 0.90315
             Prevalence : 0.16122
         Detection Rate : 0.07347
   Detection Prevalence : 0.09388
      Balanced Accuracy : 0.71568

       'Positive' Class : X1
```

```
#plot ROC curve and Area under the curve statistic
RocTrainlogisticRegVarImp <- roc(response = TrainPredictlogisticRegVarImp$outcome,
predictor = TrainPredictlogisticRegVarImp$X1, levels =
rev(levels(TrainPredictlogisticRegVarImp$outcome)))
plot(RocTrainlogisticRegVarImp, type = "s", print.thres = c(.5),
    print.thres.pch = 3,
    print.thres.pattern = "",
    print.thres.cex = 1.2,
    col = "red", legacy.axes = TRUE,
    print.thres.col = "red")
print(auc(RocTrain))
```

Outcome:

```
Area under the curve: 0.8656
```




```
logisticRegImp <- varImp(logisticRegVarImp, scale = FALSE)
logisticRegImp
```

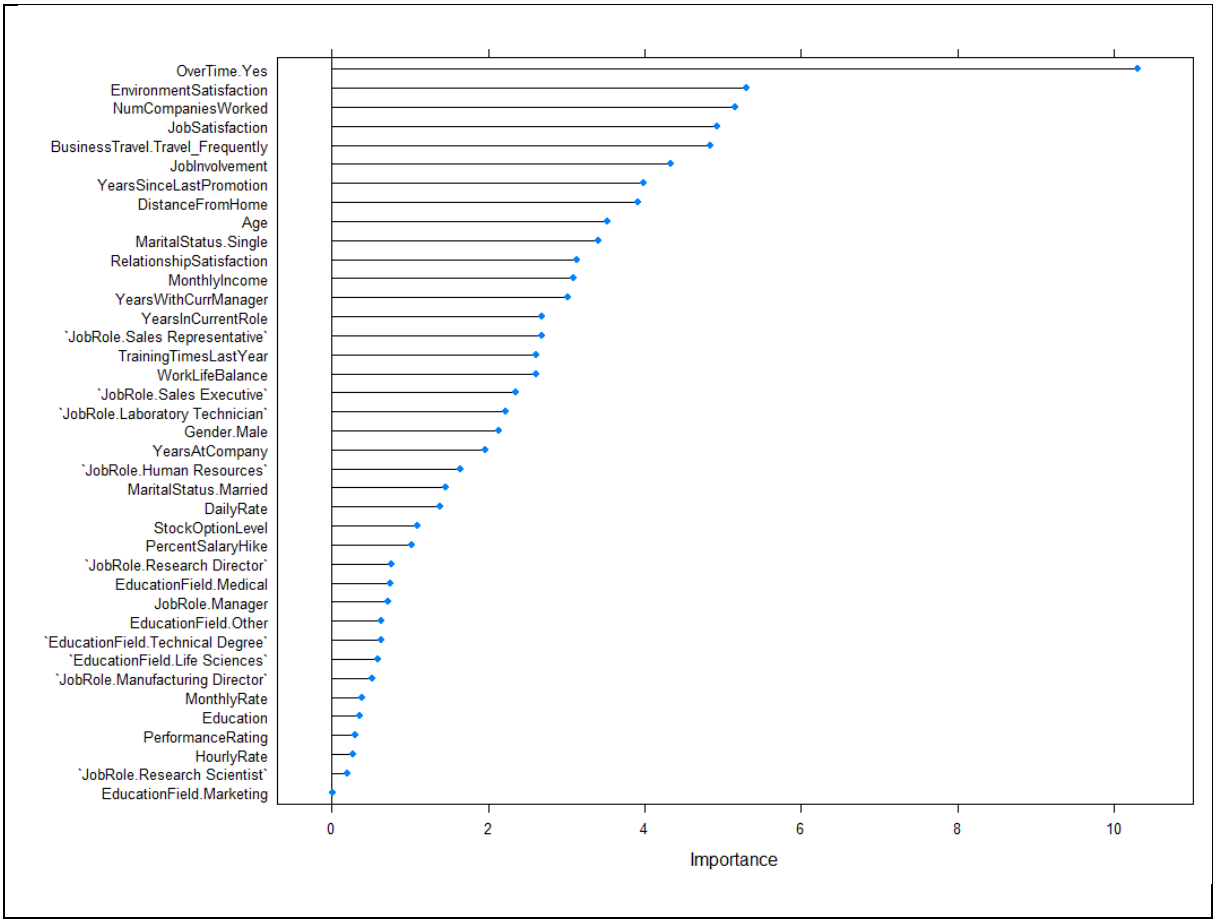
Outcome:

```
glm variable importance
```

```
  only 20 most important variables shown (out of 39)

                                Overall
OverTime.Yes                     10.295
EnvironmentSatisfaction           5.302
NumCompaniesWorked                5.161
JobSatisfaction                   4.918
BusinessTravel.Travel_Frequently  4.837
JobInvolvement                    4.334
YearsSinceLastPromotion           3.991
DistanceFromHome                  3.912
Age                               3.517
MaritalStatus.Single              3.403
RelationshipSatisfaction          3.138
MonthlyIncome                     3.090
YearsWithCurrManager              3.020
YearsInCurrentRole                2.685
`JobRole.Sales Representative`    2.681
TrainingTimesLastYear             2.618
WorkLifeBalance                   2.604
`JobRole.Sales Executive`         2.349
`JobRole.Laboratory Technician`   2.225
Gender.Male                       2.141
```

plot(logisticRegImp2)

Outcome:



#Check the variable significance for this model
summary.glm(logisticRegVarImp2$finalModel)

Outcome:

```
Call:
NULL

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.5508   0.0990   0.2543   0.4933   1.6533

Coefficients:
                                  Estimate Std. Error z value Pr(>|z|)
(Intercept)                     -2.462e+01  1.284e+01  -1.917 0.055257 .
Age                              7.463e-01  2.122e-01   3.517 0.000436 ***
NumCompaniesWorked              -1.945e-01  3.769e-02  -5.161 2.46e-07 ***
YearsInCurrentRole               1.199e-01  4.466e-02   2.685 0.007245 **
YearsSinceLastPromotion         -1.667e-01  4.178e-02  -3.991 6.58e-05 ***
YearsWithCurrManager             1.391e-01  4.607e-02   3.020 0.002527 **
TrainingTimesLastYear            1.909e-01  7.293e-02   2.618 0.008850 **
HourlyRate                      -1.161e-03  4.412e-03  -0.263 0.792471
DailyRate                        2.161e-03  1.558e-03   1.387 0.165538
MonthlyRate                     -8.248e-05  2.124e-04  -0.388 0.697791
MonthlyIncome                    9.037e-01  2.924e-01   3.090 0.001999 **
PercentSalaryHike                1.813e+01  1.787e+01   1.015 0.310243
StockOptionLevel                 1.706e-01  1.553e-01   1.099 0.271832
Education                       -2.566e-02  7.290e-02  -0.352 0.724799
DistanceFromHome                -3.464e-01  8.854e-02  -3.912 9.14e-05 ***
JobInvolvement                   3.452e-01  7.965e-02   4.334 1.46e-05 ***
PerformanceRating               -9.942e-02  3.326e-01  -0.299 0.764985
EnvironmentSatisfaction          4.390e-01  8.280e-02   5.302 1.15e-07 ***
JobSatisfaction                  4.009e-01  8.153e-02   4.918 8.75e-07 ***
RelationshipSatisfaction         2.588e-01  8.247e-02   3.138 0.001699 **
WorkLifeBalance                  1.885e-01  7.240e-02   2.604 0.009208 **
Gender.Male                     -3.949e-01  1.845e-01  -2.141 0.032271 *
MaritalStatus.Married           -3.873e-01  2.675e-01  -1.448 0.147677
MaritalStatus.Single            -1.173e+00  3.448e-01  -3.403 0.000666 ***
OverTime.Yes                    -1.993e+00  1.936e-01 -10.295  < 2e-16 ***
`EducationField.Life Sciences`   4.417e-01  7.612e-01   0.580 0.561736
EducationField.Marketing         1.200e-02  8.064e-01   0.015 0.988125
EducationField.Medical           5.682e-01  7.590e-01   0.749 0.454068
EducationField.Other             5.276e-01  8.292e-01   0.636 0.524581
`EducationField.Technical Degree`-4.903e-01  7.836e-01  -0.626 0.531522
BusinessTravel.Travel_Frequently -1.020e+00  2.109e-01  -4.837 1.32e-06 ***
`JobRole.Human Resources`        -1.120e+00  6.848e-01  -1.635 0.101972
`JobRole.Laboratory Technician`  -1.068e+00  4.799e-01  -2.225 0.026107 *
JobRole.Manager                  -4.927e-01  6.811e-01  -0.723 0.469377
`JobRole.Manufacturing Director` -2.737e-01  5.355e-01  -0.511 0.609300
`JobRole.Research Director`       7.230e-01  9.455e-01   0.765 0.444497
`JobRole.Research Scientist`     -9.713e-02  4.924e-01  -0.197 0.843634
`JobRole.Sales Executive`        -1.050e+00  4.469e-01  -2.349 0.018812 *
`JobRole.Sales Representative`   -1.481e+00  5.524e-01  -2.681 0.007338 **
YearsAtCompany                   -6.665e-02  3.410e-02  -1.954 0.050655 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1298.58  on 1469  degrees of freedom
Residual deviance:  862.52  on 1430  degrees of freedom
AIC: 942.52

Number of Fisher Scoring iterations: 7
```

# exponentiated coefficients, indicating the odds change of one step in variable x
exp(coef(logisticRegVarImp$finalModel))

Outcome:

```
                  (Intercept)                           Age             NumCompaniesWorked
                 2.039295e-11                  2.109233e+00                  8.232488e-01
            YearsInCurrentRole          YearsSinceLastPromotion          YearsWithCurrManager
                 1.127425e+00                  8.464197e-01                  1.149291e+00
         TrainingTimesLastYear                    HourlyRate                     DailyRate
                 1.210352e+00                  9.988399e-01                  1.002163e+00
                   MonthlyRate                 MonthlyIncome             PercentSalaryHike
                 9.999175e-01                  2.468710e+00                  7.503338e+07
               StockOptionLevel                     Education             DistanceFromHome
                 1.186075e+00                  9.746627e-01                  7.072241e-01
                JobInvolvement             PerformanceRating       EnvironmentSatisfaction
                 1.412258e+00                  9.053622e-01                  1.551084e+00
              JobSatisfaction       RelationshipSatisfaction                WorkLifeBalance
                 1.493239e+00                  1.295417e+00                  1.207495e+00
                  Gender.Male          MaritalStatus.Married           MaritalStatus.Single
                 6.737289e-01                  6.789110e-01                  3.093547e-01
                  OverTime.Yes   `EducationField.Life Sciences`       EducationField.Marketing
                 1.362764e-01                  1.555331e+00                  1.012075e+00
       EducationField.Medical         EducationField.Other `EducationField.Technical Degree`
                 1.765103e+00                  1.694859e+00                  6.124698e-01
BusinessTravel.Travel_Frequently      `JobRole.Human Resources`   `JobRole.Laboratory Technician`
                 3.605459e-01                  3.263085e-01                  3.438094e-01
               JobRole.Manager `JobRole.Manufacturing Director`       `JobRole.Research Director`
                 6.109514e-01                  7.605850e-01                  2.060522e+00
   `JobRole.Research Scientist`       `JobRole.Sales Executive`    `JobRole.Sales Representative`
                 9.074396e-01                  3.500024e-01                  2.273715e-01
              YearsAtCompany
                 9.355257e-01
```

#----------

#Goodness Of Fit

#----------

# Based on the deviances a p value for the model can be calculated
# H0 = Logistic regression model provides an adequate fit for the data
pvalue <- 1- pchisq(1298.58 - 862.52, df = (1469-1430))
pvalue

Outcome:

```
[1] 0
```

p=0, so evidence to reject the null hypothesis.

#Next, goodness of fit test Hosmer-Lemeshow
hoslem.test(as.numeric(TrainPredictlogisticRegVarImp2$class),as.numeric(TrainPredictlogisticRegVarImp2$outcome), g = 10)

Outcome:

```
        Hosmer and Lemeshow goodness of fit (GOF) test

data:  as.numeric(TrainPredictlogisticRegVarImp2$class), as.numeric(TrainPredictlogisticRegVarImp2$outcome)
X-squared = -4.3229, df = 8, p-value = 1
```

#Identify the box-cox transformed variables.

logisticRegVarImp$preProcess$bc

Outcome:

```
Box-Cox Transformation

1470 data points used to estimate Lambda
```

```
Input data summary:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  18.00   30.00   36.00   36.92   43.00   60.00

Largest/Smallest: 3.33
Sample Skewness: 0.412

Estimated Lambda: 0.2


$HourlyRate
Box-Cox Transformation

1470 data points used to estimate Lambda

Input data summary:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  30.00   48.00   66.00   65.89   83.75  100.00

Largest/Smallest: 3.33
Sample Skewness: -0.0322

Estimated Lambda: 0.8
With fudge factor, no transformation is applied


$DailyRate
Box-Cox Transformation

1470 data points used to estimate Lambda

Input data summary:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  102.0   465.0   802.0   802.5  1157.0  1499.0

Largest/Smallest: 14.7
Sample Skewness: -0.00351

Estimated Lambda: 0.7


$MonthlyRate
Box-Cox Transformation

1470 data points used to estimate Lambda

Input data summary:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2094    8047   14240   14310   20460   27000

Largest/Smallest: 12.9
Sample Skewness: 0.0185

Estimated Lambda: 0.7


$MonthlyIncome
Box-Cox Transformation

1470 data points used to estimate Lambda
```

```
Input data summary:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1009    2911    4919    6503    8379   20000

Largest/Smallest: 19.8
Sample Skewness: 1.37

Estimated Lambda: -0.2
With fudge factor, Lambda = 0 will be used for transformations


$PercentSalaryHike
Box-Cox Transformation

1470 data points used to estimate Lambda

Input data summary:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  11.00   12.00   14.00   15.21   18.00   25.00

Largest/Smallest: 2.27
Sample Skewness: 0.819

Estimated Lambda: -1.3


$Education
Box-Cox Transformation

1470 data points used to estimate Lambda

Input data summary:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   2.913   4.000   5.000

Largest/Smallest: 5
Sample Skewness: -0.289

Estimated Lambda: 1.2


$DistanceFromHome
Box-Cox Transformation

1470 data points used to estimate Lambda

Input data summary:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   7.000   9.193  14.000  29.000

Largest/Smallest: 29
Sample Skewness: 0.956

Estimated Lambda: 0.1
With fudge factor, Lambda = 0 will be used for transformations


$JobInvolvement
Box-Cox Transformation

1470 data points used to estimate Lambda
```

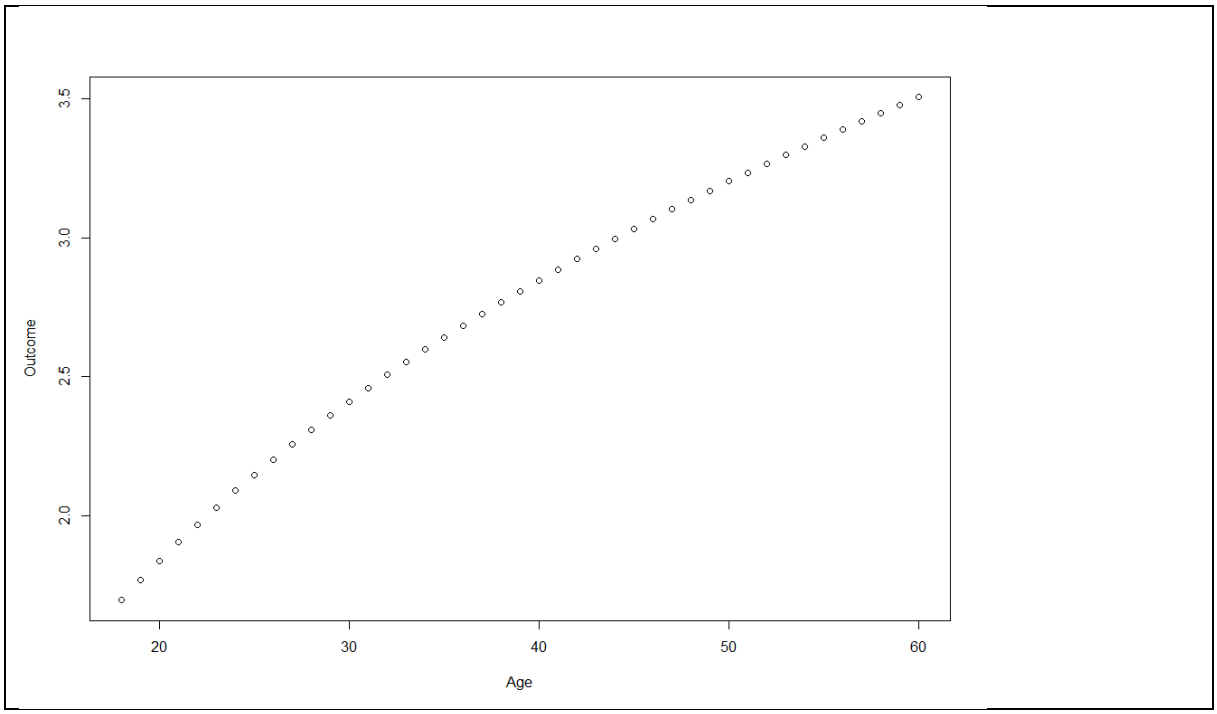
```
Input data summary:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00    2.00    3.00    2.73    3.00    4.00

Largest/Smallest: 4
Sample Skewness: -0.497

Estimated Lambda: 1.5


$EnvironmentSatisfaction
Box-Cox Transformation

1470 data points used to estimate Lambda

Input data summary:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   2.722   4.000   4.000

Largest/Smallest: 4
Sample Skewness: -0.321

Estimated Lambda: 1.1
With fudge factor, no transformation is applied


$JobSatisfaction
Box-Cox Transformation

1470 data points used to estimate Lambda

Input data summary:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   2.729   4.000   4.000

Largest/Smallest: 4
Sample Skewness: -0.329

Estimated Lambda: 1.1
With fudge factor, no transformation is applied


$RelationshipSatisfaction
Box-Cox Transformation

1470 data points used to estimate Lambda

Input data summary:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   2.712   4.000   4.000

Largest/Smallest: 4
Sample Skewness: -0.302

Estimated Lambda: 1.1
With fudge factor, no transformation is applied


$WorkLifeBalance
Box-Cox Transformation

1470 data points used to estimate Lambda
```

```
Input data summary:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   2.761   3.000   4.000

Largest/Smallest: 4
Sample Skewness: -0.551

Estimated Lambda: 1.6
```

```
#Create plots for the significant box-cox transformed predictors
#-------------------------------------------------------------

# first identify the mean of each predictor
means <- colMeans(HRdataDummyFullRankLowCorr2[,-1])
means <- as.data.frame(t(means))
means
# next, get estimates for model
model <- logisticRegVarImp2$finalModel$coefficients
model <- as.data.frame(t(model))
model
str(model)
#transform box-cox transformed variables in means
means$Age<- ((means$Age^0.2)-1)/0.2
means$DailyRate <- ((means$DailyRate^0.7)-1)/0.7
means$MonthlyRate <- ((means$MonthlyRate^0.7)-1)/0.7
means$MonthlyIncome <- log(means$MonthlyIncome)
means$PercentSalaryHike <- ((means$PercentSalaryHike^(-1.3))-1)/(-1.3)
means$Education <- ((means$Education^1.2)-1)/1.2
means$DistanceFromHome <- log(means$DistanceFromHome)
means$JobInvolvement <- ((means$JobInvolvement^1.5)-1)/1.5
means$WorkLifeBalance <- ((means$WorkLifeBalance^1.6)-1)/1.6

#create intercept in means table and right place in model
means$`(Intercept)` <- 1
means
str(means)
means <- means[, c("(Intercept)", "Age", "NumCompaniesWorked",
            "YearsInCurrentRole", "YearsSinceLastPromotion", "YearsWithCurrManager",
            "TrainingTimesLastYear", "HourlyRate", "DailyRate", "MonthlyRate",
            "MonthlyIncome", "PercentSalaryHike", "StockOptionLevel", "Education",
            "DistanceFromHome", "JobInvolvement", "PerformanceRating", "EnvironmentSatisfaction"
,
            "JobSatisfaction", "RelationshipSatisfaction", "WorkLifeBalance",
            "Gender.Male", "MaritalStatus.Married", "MaritalStatus.Single",
            "OverTime.Yes", "EducationField.Life Sciences", "EducationField.Marketing",
            "EducationField.Medical", "EducationField.Other", "EducationField.Technical Degree",
            "BusinessTravel.Travel_Frequently", "JobRole.Human Resources",
            "JobRole.Laboratory Technician", "JobRole.Manager", "JobRole.Manufacturing Director",
            "JobRole.Research Director", "JobRole.Research Scientist", "JobRole.Sales Executive",
            "JobRole.Sales Representative", "YearsAtCompany")]
```

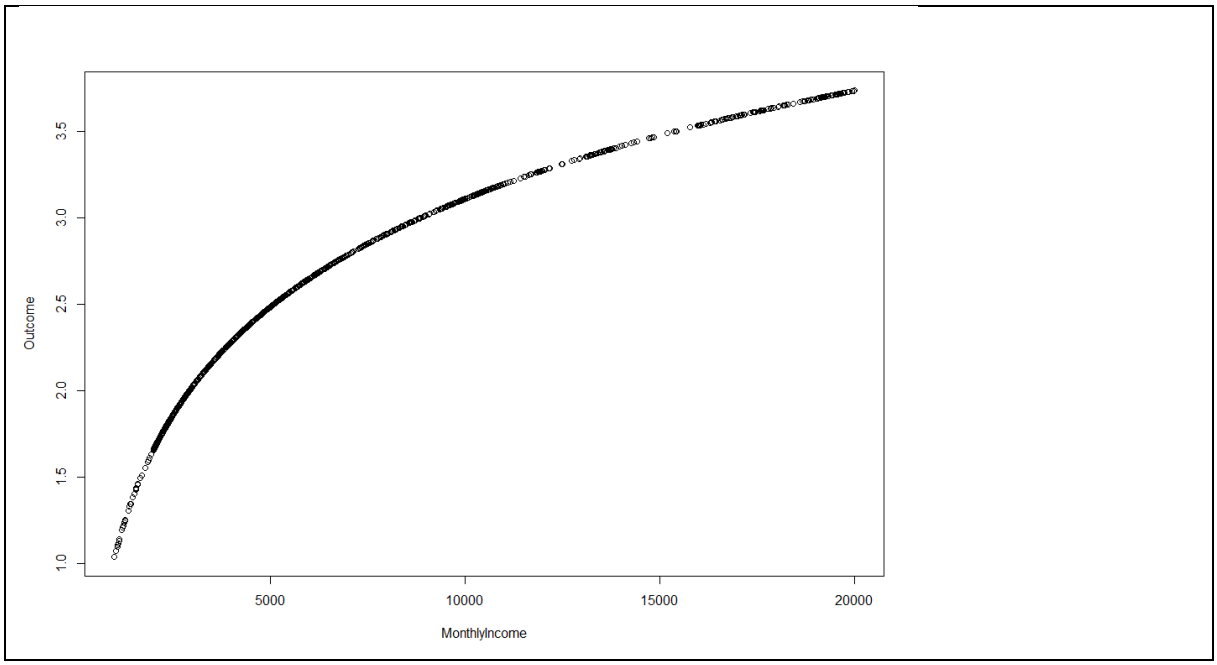
```
#calculate model average outcome
modelaverageoutcome <- means
modelaverageoutcome <- modelaverageoutcome*model

#plot Age against outcome
plot(HRdataDummyFullRankLowCorr2$Age,
    (sum(modelaverageoutcome[, -2])+
      model$Age*(((HRdataDummyFullRankLowCorr2$Age^0.2)-1)/0.2)),
    type = "p",
    xlab = "Age",
    ylab = "Outcome")
```

Outcome:



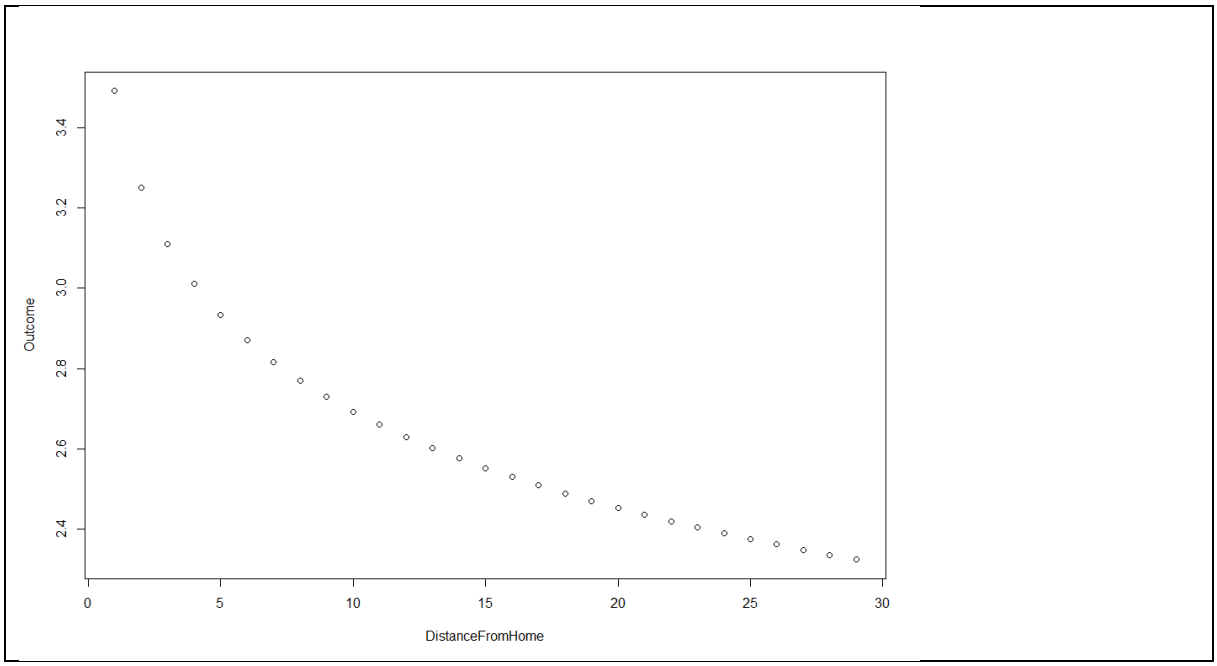

```
#plot Monthly Income against outcome
plot(HRdataDummyFullRankLowCorr2$MonthlyIncome,
    (sum(modelaverageoutcome)-modelaverageoutcome$MonthlyIncome+
      model$MonthlyIncome*log(HRdataDummyFullRankLowCorr2$MonthlyIncome)),
    type = "p",
    xlab = "MonthlyIncome",
    ylab = "Outcome"
)
```

Outcome:




```
#plot Distance from home against outcome
plot(HRdataDummyFullRankLowCorr2$DistanceFromHome,
    (sum(modelaverageoutcome)-modelaverageoutcome$DistanceFromHome+

model$DistanceFromHome*log(HRdataDummyFullRankLowCorr2$DistanceFromHome)),
    type = "p",
    xlab = "DistanceFromHome",
    ylab = "Outcome"
)
```

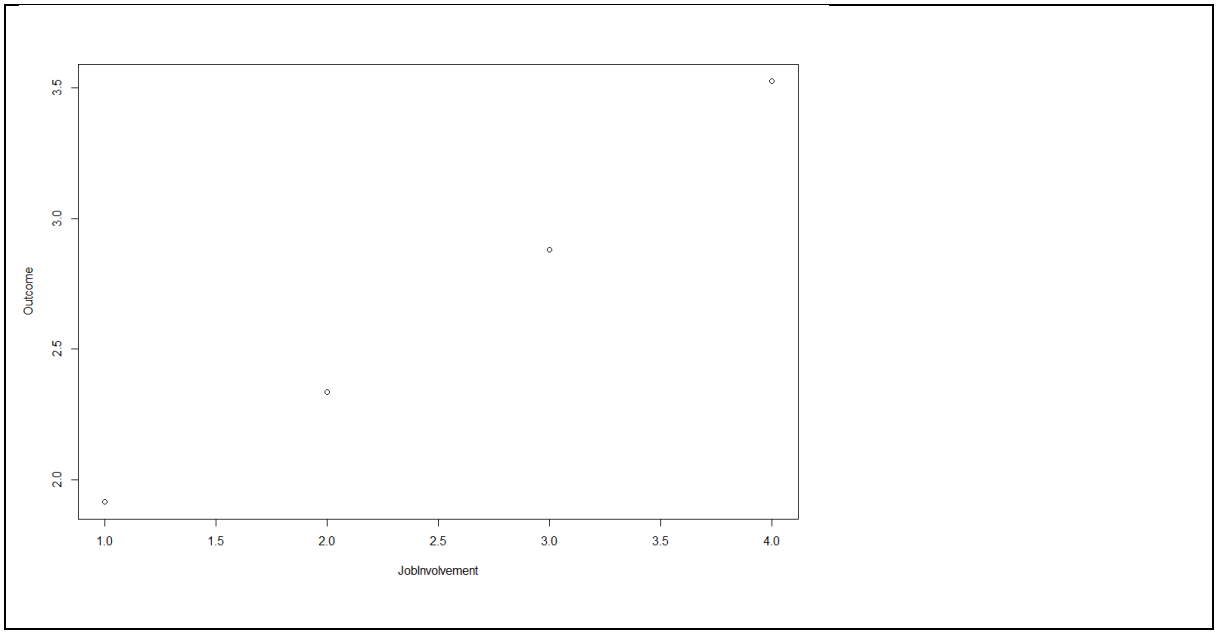
Outcome:




```
#plot Job involvement against outcome
plot(HRdataDummyFullRankLowCorr2$JobInvolvement,
    (sum(modelaverageoutcome)-modelaverageoutcome$JobInvolvement+
```

```
      model$JobInvolvement*(((HRdataDummyFullRankLowCorr2$JobInvolvement^1.5)-
1)/1.5)),
    type = "p",
    xlab = "JobInvolvement",
    ylab = "Outcome"
)
```

Outcome:




```
#plot Work life balance against outcome
plot(HRdataDummyFullRankLowCorr2$WorkLifeBalance,
    (sum(modelaverageoutcome)-modelaverageoutcome$WorkLifeBalance+
      model$WorkLifeBalance*(((HRdataDummyFullRankLowCorr2$WorkLifeBalance^1.6)-
1)/1.6)),
    type = "p",
    xlab = "WorkLifeBalance",
    ylab = "Outcome"
)
```

Outcome: