Feature attribution in absenteeism prediction: A game-theoretic approach

by

Daan Slings (ANR: u186555)
Msc Tilburg University

A thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in Business Analytics and Operations Research

Tilburg School of Economics and Management
Tilburg University

Supervised by:
David Adewunmi (Crowe Foederer),
Leon Gerritsen (Crowe Foederer),
Pieter Kleer (Tilburg University)

Date: August 22, 2025

Abstract

Absenteeism negatively affects both employers and employees. This thesis, in collaboration with Crowe Foederer, develops a machine learning model to predict absenteeism, using HR data from Dutch employees and Dutch weather data. The primary model is a multiclass XGBoost classifier that predicts absenteeism in four predefined categories. The data processing and model development take the EU AI Act into account. Since the EU AI Act requires interpretable models, four game-theoretic feature attribution methods are compared: the Shapley value, the Banzhaf value, the nucleolus and the newly introduced Average Coalitional Surplus (ACS). The ACS yields an intuitive feature attribution method.

The XGBoost classifier achieves an accuracy of 88.50%, outperforming both a Support Vector Machine baseline (accuracy 86.66%) and a majority-class baseline that always predicts "not absent" (accuracy 86.12%). The feature attribution analysis shows that different game-theoretic concepts can yield different feature rankings, due to them having different objectives. However, the variable months in service is consistently distinguished as the most important predictor for absenteeism. The Shapley and Banzhaf values show the highest number of strong correlations between the normalised feature values and their attribution values, which is useful for interpretation. This thesis concludes with recommendations for future research and for Crowe Foederer regarding predicting absenteeism and applying game theory for feature attribution.

Contents

Chapter 1

Introduction

Absenteeism is a problem for both the employer and the employee. It leads to increased costs for the employer and may take a physical or mental toll on the employee. Both parties would benefit from intervening early and avoiding absenteeism. For example, (Lawrance et al., 2021) states that the annual cost of worker absenteeism in countries that are part of the Organisation for Economic Cooperation and Development (OECD) is estimated to be between 1.2 and 2% of their total GDP. Reducing absenteeism could also reduce employee turnover, according to (Lyons, 1972).

(Pauly et al., 2002) states that the productivity gains from reducing absenteeism by interventions or programs are likely to outweigh the wage of an employee. They also state that employees will likely benefit the most in the long run from such programs, and the employer will benefit the most in the short run.

Timely intervention for absenteeism requires first predicting absenteeism, such that interventions only occur if necessary. Thus, in this thesis a model will be developed to predict absenteeism. This model needs to be in line with the EU AI regulations (European Data Protection Supervisor, 2025) to be used. Many studies talk about both absenteeism and presenteeism; this thesis will only talk about absenteeism. According to (Sitarević et al., 2023), absenteeism is defined as employees who do not work when they are supposed to. Presenteeism refers to employees who go to work while sick (Maestas et al., 2020). Thus, there are both similarities and differences between absenteeism and presenteeism as pointed out by (Gosselin et al., 2012). The company for which this thesis is written specifically asked for a model which predicts absenteeism. There was also no data available that could measure presenteeism. Therefore, this thesis focuses exclusively on predicting absenteeism.

In this model, it is also important to know why someone is predicted to be more absent. According to (Kocakulah et al., 2016), better strategies can be implemented to reduce the costs related to absenteeism, and productivity can be increased if it is clear why an employee is absent. For this purpose, feature attribution can be used to identify variables contributing to an employee's predicted absenteeism, enabling more effective interventions.

One approach to apply feature attribution is through game theory. This has already been done in SHAP (Lundberg & Lee, 2017), which uses Shapley values to quantify feature contributions. We will use four game-theoretic concepts (the Shapley value, the Banzhaf value, the nucleolus and a newly introduced concept called Average Coalitional Surplus (ACS)) to investigate whether they lead to different feature attributions.

1.1 Problem definition

In binary classification for absenteeism, the goal is to predict whether someone will be absent (1) or not (0) within a certain timeframe. We are given n data points or observations: $D_i = (x_i, y_i)$ for $i = 1, .., n$, where $x_i = (x_{i1}, x_{i2}, ..., x_{ik}) \in \mathbb{R}^k$. So we have k features in our dataset, for example the contractual hours of an employee or the temperature last month. These features are going to be used to predict the label y_i, where $y_i \in \{0, 1\}$ (whether someone is absent or not).

The goal is to obtain a mapping function $f : \mathbb{R}^k \to \{0, 1\}$, which uses the k features of x_i and returns $\widehat{y_i}$, a prediction whether someone will be absent or not. Ideally, we want our mapping f to fit the data perfectly, i.e. $f(x_i) = y_i$ for all data points (x_i, y_i), however, this will not always be true. Instead it might be that $\widehat{y_i} = f(x_i)$ is not equal to y_i for a known data point (x_i, y_i).

This mapping function f can be generated using different techniques and models, for example a Neural Network, Support Vector Machine or Decision Tree. The mapping function is fitted by minimising a certain loss function $\mathcal{L}(f(x), y)$, which can also be called a measure of error. This mapping function then needs to be evaluated, this can for example be done by calculating the accuracy, which is the number of correct classifications divided by the total number of classifications and is a function of the data points D:

$$\text{Accuracy} = \frac{\sum_{i=1}^{n} |1 - \widehat{y_i} - y_i|}{n}.$$

The mapping f can also be evaluated in other ways. Other evaluation methods and model types will be discussed in Chapter 3. In this thesis a multiclass classification model with 4 classes is developed. Instead of absent or not, we will predict a predefined range of how many days someone is absent. This provides more insights that can be used in applying interventions for absenteeism (at Crowe Foederer).

While building a model for predicting absenteeism, it should be pointed out that absenteeism data is imbalanced. One has balanced data if the amount of positive and negative labels of the data is equal. Only a small part of the population is absent, as (Lawrance et al., 2021) also points out. Therefore, weights are used to address this imbalance (see Chapter 3.2.2).

1.2 Solution approaches

There are several machine learning models that are capable of predicting absenteeism. Some examples of this are: XGBoost, Support Vector Machines, or Neural Networks. This thesis uses XGBoost, which is introduced by (T. Chen & Guestrin, 2016). They state that it is fast and scalable while also having high performance. XGBoost has built-in regularisation and can handle non-linear relationships. XGBoost was also recommended by this thesis's supervisors at Crowe Foederer (Leon Gerritsen and David Adewunmi), who both have a lot of applied knowledge in the data science field. Therefore, we choose XGBoost over, for example, a Neural Network, a multinomial logistic regression or random forest.

Neural Networks generally perform worse than gradient boosted trees (such as XGBoost) on tabular data, which this thesis uses, according to (Grinsztajn et al., 2022). Similarly, (Wu & Weiland, 2024) shows an example where a logistic regression performs worse in predicting absenteeism than XGBoost, and (Björnfot & Fjelkestam, 2023) shows an example in which Random Forest performs worse than XGBoost in predicting absenteeism. Finally, a Support Vector Machine is used as a benchmark to compare the performance of the XGBoost model with another commonly used model for absenteeism prediction (see previous research in Chapter 2.1).

Bias-variance trade-off

When choosing a machine learning model, the bias-variance trade-off needs to be considered. (Ranglani, 2024) discusses this trade-off and explains that the bias measures a model's systematic deviation from the true data patterns, while the variance measures how sensitive a model's predictions are to changes in the data. In machine learning, minimising both the bias and the variance of a model simultaneously is difficult. This trade-off is important for generalisability because a low bias and high variance, or the other way around, results in a model with poor performance. XGBoost reduces the bias while only slightly increasing the variance according to (Ranglani, 2024), allowing it to generalise well.

Beyond the bias-variance trade-off, there are also legal and ethical considerations. The EU AI Act (European Data Protection Supervisor, 2025), was introduced in 2024. This imposes restrictions on what types of data can be used in machine learning. This leads to the following first research question:

1. *How well do machine learning models perform at predicting absenteeism while complying with the EU AI regulations?*

An important part of the EU AI Act is interpretability, a prediction needs to be explained. However, in machine learning there is a trade-off between accuracy and interpretability.

Accuracy vs interpretability

In machine learning accuracy and interpretability need to be balanced. Often, models with a higher accuracy have lower interpretability and vice versa. For example, boosting methods such as XGBoost

tend to have high accuracy, but in what way someone is classified is unknown or hard to find out. In contrast, a linear regression offers higher interpretability, at the cost of lower accuracy. By looking at the regression coefficients, one can find out how important a certain variable is for the prediction. This trade-off between accuracy and interpretability must also be considered in this thesis due to the nature of absenteeism. Absenteeism data is personal, and if choices within a company are based on a model which cannot be explained, a company might not want to use the obtained model. For complex models, feature attribution methods help improve the interpretability of models by showing how each feature contributes to a prediction; this can be done using game theory. The Shapley value has already been extensively researched for feature attribution (Lundberg & Lee, 2017). However, there are also many other game-theoretic concepts. This raises the following research questions:

2. *Do different game-theoretic solution concepts produce the same feature importance rankings?*
3. *For each game-theoretic concept, is there a relation between the feature value and its attribution value?*

Now that the solution approaches have been discussed, the contribution of this thesis will be shown.

1.3 Contribution of this thesis

The goal of this thesis is to predict absenteeism. This will be done on a large dataset containing HR data from approximately 10,000 employees across different companies.

As research question 1 shows, a central contribution of this thesis lies in developing a model that complies with the EU AI regulations. Predicting absenteeism under the EU AI regulation is a relatively new subject since the latest version (at the time of writing this thesis) of the EU AI regulation (European Data Protection Supervisor, 2025) was introduced in 2024.

In addition to evaluating model performance, this thesis also investigates various game-theoretic concepts as feature attribution methods by answering research questions 2 and 3. This is useful for black box models, in which we only know the input and output but do not know exactly how this output is obtained. Examples of black box models are XGBoost and other tree-based models, as (Sagi & Rokach, 2021) points out. Most of the current research on game-theoretic feature attribution focuses on the Shapley value (Shapley, 1952), for example (Lundberg & Lee, 2017), while little research has explored other game-theoretic methods for feature attribution, particularly in multiclass classification.

1.4 Ethics and machine learning

Ethics are an important factor in building models, machine learning, and artificial intelligence, especially if something as personal as HR data is used. If ethics are not taken into account, it may lead to discrimination or violate someone's right to privacy. Although some discriminatory variables can give insights, if they are used in the wrong way, it can have serious consequences according to (European Data Protection Supervisor, 2025). An example of this is the Dutch childcare benefits scandal (Toeslagen affaire), where people were classified as fraudsters while they were not. After it became clear that this happened, (Autoriteit Persoonsgegevens, 2020) searched for where it went wrong in the Belastingdienst (the Dutch tax department). There are several variables they found that the Belastingdienst should not have used. Firstly, the Belastingdienst processed whether someone had a double nationality; this variable was not necessary for the Toeslagen department (a branch of the tax department), and secondly, they used nationality as an indicator for risk. Due to some other mistakes, eventually, the variable nationality resulted in misclassifying innocent individuals as fraudsters. This example shows why it is important to think about the ethics of a (machine learning) model.

Another example of ethics and law is the story of "De hollende kleurling", which can be found in newspapers but is also discussed in (Doomen, 2010). It describes an instance in which a person was arrested for being in possession of drugs; however, the evidence was not obtained in accordance with the law, so he was let go. This story can be broadened to machine learning: Although a prediction may be correct, if it is not obtained the correct way by law, it may not be enforced or used.

Ethics are especially relevant for this thesis due to the use of HR data, which contains many personal details. Therefore, the data used will be anonymised, and irrelevant features will be excluded from

the dataset before training a model. Models that predict absenteeism must comply with the EU AI Act (European Data Protection Supervisor, 2025) if they are used. If a model is not for a company but purely for scientific purposes, it is excluded from the EU AI Act (European Data Protection Supervisor, 2025, Article 25). If a model to predict absenteeism is built to be used outside of research purposes, (European Data Protection Supervisor, 2025, Article 48) will classify it as a high-risk model due to the use of personal data collected by HR. Therefore, we must be extra careful while building a model. The most important factors to consider when building a model are: no discrimination, explainability, transparency, supervision, privacy and AI literacy. Discrimination will be avoided by excluding variables, such as age, gender and ethnicity. Explainability will be achieved by using game-theoretic feature attribution methods. Transparency is hard to implement in this thesis, but advice will be given on how to be transparent, for example, if a company uses one of the discussed models, it should let its employees know that their data is used to train a model to predict absenteeism and there is a model that predicts absenteeism. Next to that, all involved parties should know how the model is used, and what follow-up actions are possible. This thesis documents how the models are developed. Supervision can partially be implemented for this thesis, the developed models will be evaluated to see if their predictions are correct. If they are not, the model must be changed. The data quality must and will be assessed. If a model is implemented, the model should still be supervised, and no decision should be made solely based on a model. For privacy, some variables will be excluded and there could be aggregated over individuals. The data will also be anonymised, more about what is done with the data is in Chapter 5. Finally, AI literacy requires users to understand the model's predictions and its limitations. Reading this thesis will help develop AI literacy; however, additional knowledge and training are necessary and must be ensured throughout the organisation where the model is applied.

1.5 Outline of this thesis

First, this thesis will discuss previous research related to absenteeism and feature attribution using game theory in Chapter 2. Then we will discuss model types, data splitting and imbalanced data in Chapter 3. Chapter 4 will show four game-theoretic concepts and explain how to construct a game from a model and data that applies these concepts. In Chapter 5, we explain how the dataset for this thesis was constructed. Then Chapter 6 will discuss the experimental setup, followed by Chapter 7, which shows the results. These results will be summarised in Chapter 8. Chapter 9 discusses interesting opportunities for further research. Finally, Chapter 10 gives some recommendations for Crowe Foederer based on the results of this thesis.

Chapter 2

Previous research

In this chapter, previous research on absenteeism will be discussed. First, research regarding predicting absenteeism is discussed, followed by research that discusses what types of variables have an impact on absenteeism. In Appendix A possible interventions for absenteeism are also discussed.

2.1 Predicting absenteeism

This section reviews previous research on predicting absenteeism.

(Wahid et al., 2019) uses tree models to predict absenteeism. They use a Gradient Boosted Tree, Random Forest, Decision Tree and Tree Ensemble (ordered decreasingly in accuracy). What was especially useful for this thesis is how they split the amount of absenteeism into four classes: 0 hours absent, bigger than 0 hours and less than 8 hours absent, greater or equal to 8 hours and less than 40 hours absent, and greater or equal to 40 hours absent. Or equivalently, not absent, hours absent, days absent and weeks absent. This gave the idea to distribute the classes the following way while predicting absenteeism per month: not absent, absent for less than or equal to 7 days, absent for more than 7 days but less than the entire month, and absent the entire month (why is discussed in Chapter 6).

(Hoevenberg, 2021) is a thesis on predicting absenteeism; they used data from the company ProRail, which is a Dutch company. We will only be using data from Dutch employees (as explained in Chapter 5, therefore the results of (Hoevenberg, 2021) are also applicable to this thesis. They use the following models to predict absenteeism: Logistic Regression, Decision Tree, Gradient-Boosted Decision Tree and Multilayer Perceptron (which is a Neural Network). In their research, the gradient-boosted decision resulted in the highest accuracy. This also strengthens our choice of XGBoost as the main model in this thesis, because it also uses gradient tree boosting.

(Lawrance et al., 2021) illustrates what a next step can be after predicting absenteeism. They minimise a cost function for effective interventions on absenteeism. This is interesting because it shows the advantages of reducing absenteeism from a business perspective while still helping the employees. Their idea behind building a model to predict if an employee has a high risk of being absent is to reduce costs because the interventions can be personalised on an employee level. Although cost minimisation is not part of this thesis, (Lawrance et al., 2021) shows a valuable direction for future work at Crowe Foederer: developing a model that recommends interventions to reduce absenteeism. They have another useful recommendation, which is to include seasonal changes in their prediction model. Therefore, we will be using some variables that represent the season (some weather statistics and the month).

There are many other papers which have researched absenteeism prediction models. These papers are summarised in Table 2.1 below. The method which performed the best according to each paper is underlined. The best performance was based on accuracy, except for (Nath et al., 2022) (indicated with a footnote in the table). (Notenbomer et al., 2018) only used one method; therefore, no method is underlined. (Rajath & Pandita, 2022) was not publicly available, making it unclear which of their methods was the best. We included this paper since they also use XGBoost.

Study	Methods
(Ajmi, 2019)	Decision Tree, Logistic regression, Neural Network, Support Vector Machine
(Lumintu & Maududie, 2025)	CatBoost, Gradient Boosting, Random Forest
(Lima et al., 2020)	Long Short-Term Memory (Neural Network), Multilayer Perceptron, Recurrent Neural Network, Support Vector Machine
(Shah et al., 2020)	Decision Tree, Deep Neural Network, Random Forest, Support Vector Machine
(Nath et al., 2022)	Multi Layer Perceptron, Multinomial logistic regression, Random Forest, Support Vector Machine[1]
(Skorikov et al., 2020)	J48, KNN (Manhattan, Chebyshev and Euclidean), Naive Bayes, ZeroR
(Dogruyol & Sekeroglu, 2020)	Backpropagation Method-Based (Neural Network), Long-Short Term Memory (Neural Network), Radial-Basis Function Based (Neural Network)
(Notenbomer et al., 2018)	Logistic regression
(Rajath & Pandita, 2022)	Logistic regression, Random Forest, Support Vector Machine, XGBoost

Table 2.1: Other studies that predict absenteeism: the method each study reported as best is underlined.

2.2 Variables related to absenteeism

Some variables may be more relevant than others for predicting absenteeism. If we combine most of the variables that the papers in Table 2.1 use or recommend, we obtain the following list:

- family condition,
- number of children,
- marital status,
- relationship with peers and superiors,
- satisfaction with work,
- reason of absence,
- month of absence,
- day of the week,
- season,
- transportation expense,
- distance from residence to work,
- service time,
- age,
- average daily workload,
- hit target,
- disciplinary failure,
- education,
- son,
- social drinker,
- social smoker,
- (number of) pets,
- weight,
- height,
- body mass index,
- recent performance.

In addition to predicting absenteeism, some studies have identified variables related to absenteeism, which will be discussed in the remainder of this section. The seasonality of absenteeism is discussed in some of the research in this section. Some of the research also finds other features which suggest that the intensity of absenteeism may change from season to season. Furthermore, the seasonality may shift from year to year, due to for example weather conditions or different flu periods. First, we will discuss some research that shows that absenteeism changes year over year.

(Centraal Bureau voor de Statistiek, 2015) and (Centraal Bureau voor de Statistiek, 2025) show that sickness absence changes from year to year. Additionally, in (Centraal Bureau voor de Statistiek, 2025) it can even be seen that the absence does not always peak in the same quarter.

(Rijksinstituut voor Volksgezondheid en Milieu, 2025) shows a graph of the flu contaminations for different years. It is visible that there is seasonality, but it is clear that each season is not necessarily the same. For example, in some years there are more contaminations, or the flu season is shifted by a few weeks. This also has an impact on absenteeism, since (Fisman et al., 2024) found a relation between influenza and absenteeism.

The remaining papers and their variables related to absenteeism are shown in Table 2.2 below.

[1] Although they reported the Support Vector Machine as the best, their Random Forest model showed the highest accuracy.

Study	Variables
(Piha et al., 2012)	Individual income, job type, socioeconomic status
(Sakr et al., 2025)	Age, gender, married, low job grade, smoking
(Demou et al., 2018)	Age, gender, job type, mental health
(Miller, 2016)	Employee well being
(Hafner et al., 2015)	Bullying, chronic health condition, mental health
(Kocakulah et al., 2016)	Family, (financial) stress, illness, job satisfaction
(Schalk & Van Rijckevorsel, 2007)	Contract type, job type, workplace attitude
(Dionne & Dostie, 2007)	Compressed work week, part-time contract, shift work week, standard work hours, working from home possibility
(Coleman & Schaefer, 1990)	Season, weather
(Akyeampong, 2007)	Time of year
(Asghar et al., 2021)	Time of year, year
(Markham & Markham, 2005)	Air pressure, day of week, precipitation, season, snow, temperature, wind speed
(Burton et al., 2004)	Caregiver for ill dependent people
(Markussen & Røed, 2015)	Amount of daylight
(J. Shi & Skuterud, 2014)	Hourly pay, weather quality
(Caruso et al., 2004)	Overtime
(Mainar et al., 2017)	Permanent employment

Table 2.2: Studies and variables they found that are related to absenteeism

There are some small comments in addition to Table 2.2. Firstly, (Sakr et al., 2025) suggested: job type, job control, work time control, and job strain for feature research. Secondly, (J. Shi & Skuterud, 2014) measured the weather quality by making an index based on: temperature, relative humidity, precipitation and cloud cover.

A note on the coronavirus

Beyond the variables shown in Table 2.2, research also shows that the coronavirus caused more absenteeism (Goda & Soltas, 2023), (van Ballegooijen et al., 2021). Therefore, the years in which the coronavirus was present may need to be removed from the dataset. It could also be the case that absenteeism occurs more or less after the pandemic. In that case, splitting the data in before, during and after the pandemic might be the solution.

(Gielen & Ilham, 2024) states that the last coronavirus measures were dropped in March of 2022. We will assume this is the end of the pandemic.

2.3 Game theory and model interpretability

The EU AI Act (European Data Protection Supervisor, 2025) requires a model to be explainable and interpretable. Feature importance and feature attribution can help with this process. Feature importance shows how much each feature matters across the entire dataset, while feature attribution shows what the contribution of each feature is for individual predictions. Feature attribution can sometimes be aggregated to see what a feature's contribution is on the whole dataset (in SHAP for example). The Python package SHAP, introduced by (Lundberg, 2018), is designed to calculate feature importance and feature attribution based on Shapley values (Shapley, 1952). This is done by interpreting variables as players and calculating their coalitional values. A more detailed explanation of SHAP will be given in Chapter 4.4.3.

Another paper that investigates game theory for feature importance and attribution purposes is (Grigoryan, 2024). Their focus seems to be more towards feature importance. In addition to the Shapley value, they discuss other game-theoretic concepts to determine feature importance, such as:

- the nucleolus (Schmeidler, 1969),
- the Banzhaf power index (Banzhaf, 1965),
- Shapley-Shubik (Shapley & Shubik, 1954),

- constrained equal awards (O'Neill, 1982) (and others),
- constrained equal losses (O'Neill, 1982) (and others).

While this thesis focuses only on game-theoretic concepts, (Grigoryan, 2024) also analyses some non-game-theoretic concepts for feature importance, such as LIME (Ribeiro et al., 2016). (Grigoryan, 2024) explores all these feature attribution concepts for regression and binary classification tasks (using a logistic regression). However, this leaves out multiclass classification. This thesis will fill that gap by investigating various game-theoretic concepts, specifically in a multiclass setting.

In (Grigoryan, 2024), it is stated that it is hard to evaluate feature importance methods. Therefore, they introduce a new metric: Permutation Importance Evaluation (PRIME), which uses permutation tests and the Shannon entropy (Shannon, 1948). This is used to see how sensitive the final importance ordering of a game-theoretic concept is to change. However, this involves changing the data multiple times, which is not viable for this thesis since computing a fraction of the coalitional values already takes more than a day. Therefore, permuting the data multiple times and redoing these calculations takes too much time. They did this permuting of data to see whether a game-theoretic approach yielded consistent feature rankings. This thesis will compute the coalition values differently from (Grigoryan, 2024) (more on this in Chapter 4.4.1). Since this thesis calculates the coalitional values differently, the results can be entirely different, as (H. Chen et al., 2023) highlights.

In addition to feature importance (Grigoryan, 2024) is also interested in whether ties occurred in feature rankings or not. According to them, a large number of ties mean low discriminatory power and thus feature importance and feature attribution do not yield interpretable results. This happened, for example, in the constrained equal awards solution concept.

For each method, (Grigoryan, 2024) discusses what the strengths and limitations are. In their conclusions, they state that some methods are sensitive to correlations between the variables. For example, they state the nucleolus had less discriminatory power when the data was correlated. As shown in Chapter 5 in Figure 5.2, our data has several correlations that can be considered strong according to (Akoglu, 2018).

While (Grigoryan, 2024) provides an extensive comparison of several attribution methods, the paper does not discuss in much detail the direction of the feature attribution methods (whether a feature increases the predicted value, or probability in the case of classification, or decreases it). However, this aspect could yield relevant insights into model behaviour. Therefore, this thesis discusses the direction of the feature attribution methods that are used.

There are also many other papers which use some form of game theory for feature attribution or importance. To keep this chapter concise, a list of some relevant studies is provided here: (Zhou et al., 2024), (Condevaux et al., 2023), (Zhan & Kim, 2024), (Fang et al., 2016), (Dhamal et al., 2012), (Covert et al., 2020), (Benedek et al., 2020), (Il Idrissi et al., 2025), (Ribeiro et al., 2016), (Liu et al., 2024), and (Kulynych & Troncoso, 2017).

Finally, (Nauta et al., 2023) talks about explainable artificial intelligence (XAI) in general and identifies 12 properties. They use these properties to review the evaluation of 300 papers about XAI. Although this thesis does not specifically discuss these properties, (Nauta et al., 2023) should still be mentioned so that future research can evaluate XAI techniques the same way.

Chapter 3

Machine learning methodology

While predicting something using machine learning, there are two options: one can predict whether something belongs to a class, or one can predict the value of something. For absenteeism, we can try and predict a few things. Firstly, we can predict whether someone will be absent or not within a certain time frame, we can also add more layers to this and categorise between what amount of hours someone will be absent. For example, as discussed earlier (Wahid et al., 2019) has the following categories for absenteeism each month: 0 hours absent, between 0 and 8 hours absent, between 8 and 40 hours absent and finally more than 40 hours. Even more categories can be added, for example, we can also predict whether someone is $0, 1, 2, 3$, etcetera hours absent. The more categories there are, the more information the prediction contains. Instead of looking at categories, we can also predict a continuous value. So, for example, we try and predict how much time someone will exactly be absent, or what percentage of their contractual hours someone will be absent (for a chosen time frame). So we can build two types of models, a classification model or a regression model to predict absenteeism. Each model type has its own mathematical reasoning behind it. In this chapter, we will explain how each model type works.

Recall that we are given n data points or observations: $D_i = (x_i, y_i)$ for $i = 1, .., n$, where $x_i = (x_{i1}, x_{i2}, ..., x_{ik}) \in \mathbb{R}^k$. So we have k features in our dataset, which we are going to use to predict y_i, where y_i is a category or a real (possibly bounded) number depending on the model. During this thesis, if a function is a function of the data, it will be indicated with (D). This notation will only be used for the first few occurrences; from Chapter 3.1.2 onward, it will be omitted for simplicity.

3.1 Models

This section introduces different types of prediction models. First, we cover general types of prediction models and then move on to XGBoost (Extreme Gradient Boosting), a tree-based model. Finally, we will also briefly discuss the SVM (Support Vector Machine).

3.1.1 Regression model

For a regression model, to obtain a prediction we want a mapping function f, which takes x_i, from data point (x_i, y_i), and uses its k features to return a real number, which is a prediction of y_i. So $f : \mathbb{R}^k \to \mathbb{R}$. Ideally, we want our mapping f to fit the data perfectly, i.e., $f(x_i) = y_i$, however, this will not always be true. Instead, $f(x_i) = \widehat{y}_i$ where $\widehat{y}_i$ is the predicted value, which can either be y_i or something else. It could be that our prediction should be bounded by an upper bound U and a lower bound L. For example, someone cannot be absent for less than 0 days in a year or more than 365 days in a year (or 366 days in case of a leap year). This means that our mapping f should satisfy $L \leq \widehat{y}_i \leq U$.

Below is an example of what a regression mapping could look like. The data is generated by hand, but in such a way that it has some resemblance to the discussed literature in Chapter 2. Here, the predicted number of days absent should be bounded by 0 and 31.

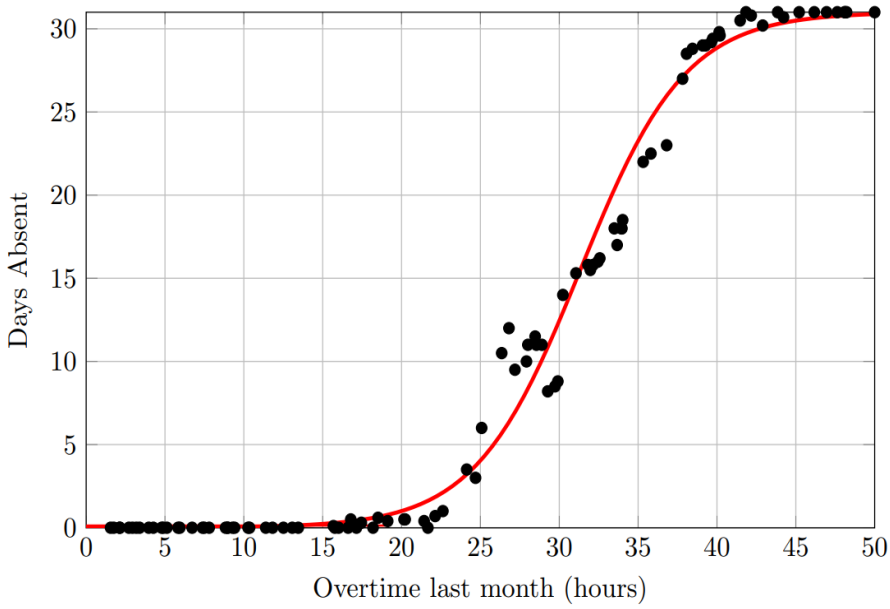

Figure 3.1: Logistic regression example: an example of how a regression model could look, with data generated by hand for illustrational purposes and fitted according to Equation 3.1.

Measures of error for regression models

To evaluate how well the obtained mapping (or model) is, measures of error can be used. While fitting a model, an error measure is (usually) minimised to find the best possible mapping. In a regression model, the error measure determines what the parameters of the regression function would be. For example, in a logistic regression function where

$$f(x) = \frac{M}{1 + e^{ax+b}}, \tag{3.1}$$

the a and b are determined by optimising over the measure of error (M is the maximum possible value and is determined beforehand).

A measure of error which would not make sense is $\sum_{i=1}^{n}(\widehat{y}_i - y_i)$ or $\sum_{i=1}^{n}(y_i - \widehat{y}_i)$ since in these measurements overshooting and undershooting would cancel each other out.

A better alternative is the absolute error measurement (also called mean absolute difference, or MAD in short). The MAD is a function of the data D, so $MAD = MAD(D)$, for $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ and is defined in the following way:

$$MAD = \frac{1}{n}\sum_{i=1}^{n}|\widehat{y}_i - y_i|.$$

Another good measure of error would be the mean squared error (MSE). Similar to the MAD the MSE is a function of the data D, so $MSE = MSE(D)$, for $D = \{(x_1, y_1),\ (x_2, y_2), \ldots, (x_n, y_n)\}$. The MSE is defined in the following way:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i - y_i)^2.$$

The root mean squared error is very similar to the mean squared error but is better comparable to the mean absolute difference due to taking the square root, and therefore the MAD and $RMSE$ are of the same order of magnitude. Again, $RMSE = RMSE(D)$ is a function of the data. Minimising over the MSE or $RMSE$ yields the same optimal f because the square root is monotonically increasing. The $RMSE$ is defined in the following way:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i - y_i)^2}.$$

Compared to the MAD, the $(R)MSE$ punishes predictions which are far from the truth, relatively speaking, due to the squaring. The $(R)MSE$ could therefore be preferred over the MAD. However, the MAD is a better measure of error if we want to minimise (in our case) the total number of days our model is off. Ideally, we want a model that minimises both the $(R)MSE$ and the MAD. In the case of absenteeism, if we want to predict the number of days someone will be absent, with the goal of intervening on time, then it does not matter if the model is off by 1 or 2 days. However, if the model is off by 30 days, it can be the case that the decision on whether intervening is necessary changes.

A mix of the previously stated measures of error could also give a better understanding of how a found model performs.

Performance metrics for regression models

For a regression model, the $(R)MSE$ and the MAD can also be used to evaluate how well a model performs on a validation and test dataset. The R^2, also called the coefficient of determination, may also be used as a performance metric. It explains how much of the variation of the dependent variable (absence in our case) is captured by our model. R^2 is a function of the data, so $R^2 = R^2(D)$, where $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$. Mathematically, R^2 is defined the following way:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \widehat{y_i})^2}{\sum_{i=1}^{n}(y_i - \overline{y_i})^2}, \text{ with } \overline{y_i} = \frac{\sum_{i=1}^{n} y_i}{n}.$$

It is clear that $R^2 \leq 1$. A higher R^2 implies a better model, since in that case the predictions $\widehat{y_i}$ are relatively close to the observations y_i compared to the mean of the observations $\overline{y_i}$. If $R^2 = 0$, the model performs the same as simply predicting the mean of the observations.

From now on, for easier notation, D will be omitted. However, if there is a function using $f(x) = \widehat{y}$ and y, it is a function of the data D.

3.1.2 Binary classification model

Instead of predicting how many days someone will be absent, we can also try to predict whether someone will be absent or not. This can, for example, be done by simply rounding a prediction from a regression model or by using a specific classification model.

First, recall the mathematical definition of binary classification from Chapter 1.1. In binary classification for absenteeism, we are going to predict whether someone will be absent (1) or not (0) within a certain timeframe. So the goal is to predict the label $y_i \in \{0, 1\}$ (whether someone is absent or not). Therefore, we want to obtain a mapping function f, which uses the k features of x_i and returns $\widehat{y_i}$, a prediction of whether someone will be absent or not. So $f : \mathbb{R}^k \to \{0, 1\}$. Recall that ideally $f(x_i) = y_i$, however again, this will not always be true. Instead, $f(x_i) = \widehat{y_i}$, a prediction of y_i. Below is an example of what a binary classification mapping could look like. The same data points were used as in Figure 3.1, however, every absence label was converted into 2 bins: absent (1) or not absent (0). The red line should represent a mapping which classifies observations as absent or not absent. We can for example first do a binary logistic regression as in Figure 3.2, and base our jump point on the turning point of the logistic regression curve.

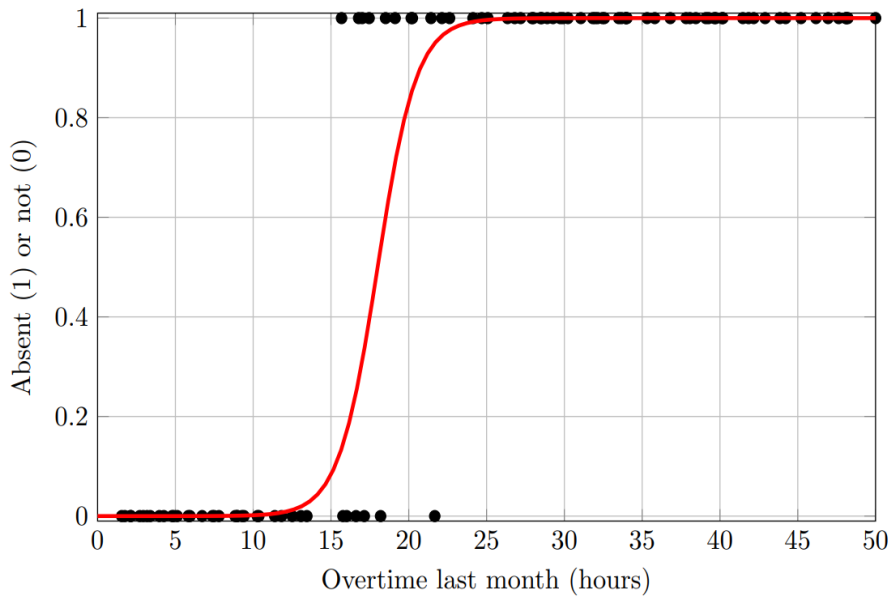

Figure 3.2: An example of what a logistic binary regression model could look like, with data generated by hand for illustrational purposes. Note that points overlap.

Figure 3.2 shows the probability that an employee is predicted as absent given the hours of overtime that the employee has. The turning point is at 18 hours of overtime. The predicted probability of being absent is here 0.5. So we can modify the regression from Figure 3.2 to make a binary classifier, which would then be:

$$\widehat{y_i} = \begin{cases} 1 \text{ if } x_i \geq \widehat{x^*}, \\ 0 \text{ if } x_i \ < \widehat{x^*}, \end{cases}$$

where $\widehat{x^*}$ is the solution to $f(\widehat{x^*}) = 0.5$, in our example $\widehat{x^*} = 18$. The obtained classifier is visualised in Figure 3.3.

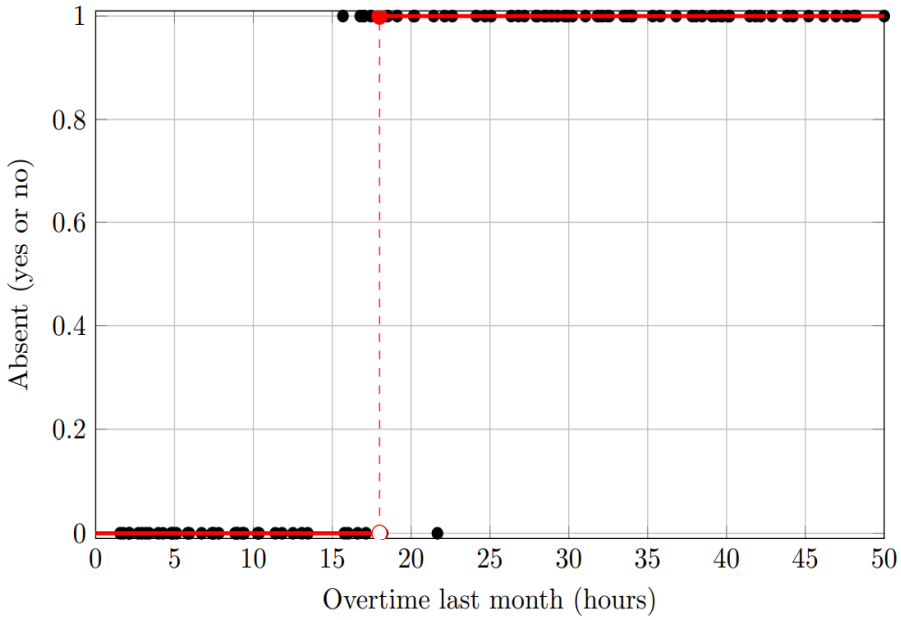

Figure 3.3: An example of what a binary classification model could look like, with data generated by hand for illustrational purposes. Note that points overlap.

One can choose to classify a prediction as positive (or 1 equivalently) when the prediction is greater or equal to 0.5 (as in Figure 3.3); however, this threshold can also be moved (as long as $0 \leq \text{threshold} \leq 1$). This moving of the threshold can be useful, especially in imbalanced datasets, where the model can have a bias towards the majority class according to (Johnson & Khoshgoftaar, 2019). For example, if $\widehat{y}_i = 0 \forall i$, the model predicts 0 (not absent) for every observation and is too conservative. In contrast, the model would be too generous if $\widehat{y}_i = 1 \, \forall \, i$, the model predicts 1 (absent) for every observation.

The mapping function f can be generated using different techniques and models, for example a Neural Network, Support Vector Machine or Decision Tree.

Measures of error for binary classification models

To evaluate how well the obtained mapping (or model) is, measures of error can be used. While fitting a model, an error measure is (usually) minimised to find the best possible mapping. In a binary classification model, the error measure would determine where the jump point is between predicting 0 and predicting 1.

A measure of error for binary classification is binary cross-entropy (or simply cross-entropy for multiclass classification, this will be discussed later). It is a widely accepted and effective measure of error according to (Boudiaf et al., 2020).

Binary cross-entropy (BCE), also called logarithmic loss, is explained in (Ho & Wookey, 2019) and has the following mathematical definition:

$$\text{BCE} = -\frac{1}{n}\sum_{i=1}^{n}(y_i \log(q(\widehat{y}_i)) + (1 - y_i)\log(1 - q(\widehat{y}_i))).$$

Here $q(\widehat{y}_i)$ is the probability of a prediction being equal to 1, and is defined as:

$$q(\widehat{y}_i) = \mathbb{P}(\widehat{y}_i = 1).$$

The binary cross-entropy loss function can be modified to the weighted binary cross-entropy loss (WBCE), function to handle an imbalanced dataset. This would result in the following loss function:

$$\text{WBCE} = -\frac{1}{n}\sum_{i=1}^{n}(w_1 \cdot y_i \log(q(\widehat{y}_i)) + w_0 \cdot (1 - y_i)\log(1 - q(\widehat{y}_i))),$$

where $w_0 \geq 0$ is the weight associated with the observations in class 0, and $w_1 \geq 0$ is the weight associated with the observations in class 1. These weights need to be nonnegative, else incorrect classifications would be preferred over correct classifications. As explained before, the mapping f we obtain returns a probability distribution for each data point and classifies each data point, using a chosen general threshold. The threshold we choose for classifying a prediction as positive or negative does not matter in binary cross-entropy loss. $y_i \log(q(\widehat{y}_i)) > 0 \iff y_i = 1$ and $(1 - y_i)\log(1 - q(\widehat{y}_i))) > 0 \iff y_i = 0$. Since $0 \leq q(\widehat{y}_i) \leq 1$, we have $\log(q(\widehat{y}_i)) \leq 0$ and $\log(1 - q(\widehat{y}_i)) \leq 0$, by definition. To account for $y_i \log(q(\widehat{y}_i)) + (1 - y_i)\log(1 - q(\widehat{y}_i))$ always being negative, a minus is put in front of the summation, so we want to minimise the binary cross-entropy loss. So if $q(\widehat{y}_i)$ is close to y_i, the binary cross-entropy loss is low, and if it is far off, the loss becomes large very quickly, due to the exponential nature of the log function.

While (binary) cross-entropy is a common choice for a measure of error, other loss functions also exist, such as hinge loss (Cortes & Vapnik, 1995) (commonly used in Support Vector Machines). However, in this thesis, we focus on weighted (binary) cross-entropy, as it is a widely accepted and effective measure of error according to (Boudiaf et al., 2020). It also works well with XGBoost's framework.

Performance metrics for binary classification

For binary classification there are also ways to evaluate how well a model performs on a validation and test set. This can, for example, be done by calculating the accuracy, which is the number of correct classifications divided by the total number of classifications.

$$\text{Accuracy} = \frac{\sum_{i=1}^{n} |1 - \widehat{y}_i - y_i|}{n}$$

For a binary classification model, different types of calculations can be made using a confusion matrix. Before showing the confusion matrix for binary classification, we first explain the terminology. Firstly, Positive means label 1, and Negative means label 0. In a confusion matrix all possible types of (mis)classifications are stated. For binary classification there are 4 possibilities: True Positive, False Negative, False Positive and True Negative. Below is the confusion matrix, for binary classification:

Actual / Predicted	Positive (P)	Negative (N)
Positive (P)	True Positive (TP)	False Negative (FN)
Negative (N)	False Positive (FP)	True Negative (TN)

Table 3.1: Confusion matrix for a binary classification problem

We can also define these (mis)classifications mathematically in the following way:

$$y_i = \begin{cases} 1 \longrightarrow \text{(Positive observation)}, \\ 0 \longrightarrow \text{(Negative observation)}, \end{cases} \quad \widehat{y_i} = \begin{cases} 1 \longrightarrow \text{(Positive prediction)}, \\ 0 \longrightarrow \text{(Negative prediction)}, \end{cases}$$

$$\text{True Positive} = \sum_{i=1}^{n} y_i \times \widehat{y_i},$$

$$\text{False Positive} = \sum_{i=1}^{n} (1 - y_i) \times \widehat{y_i},$$

$$\text{True Negative} = \sum_{i=1}^{n} (1 - y_i) \times (1 - \widehat{y_i}),$$

$$\text{False Negative} = \sum_{i=1}^{n} y_i \times (1 - \widehat{y_i}).$$

Notice, that we can also define the formula for the accuracy the following way:

$$\text{Accuracy} = \frac{\text{True Positives (TP)} + \text{True Negative (TN)}}{\text{True Positives (TP)} + \text{False Positive (FP)} + \text{True Negative (TN)} + \text{False Negatives (FN)}}. \tag{3.2}$$

For binary classification, there are also other performance metrics. Firstly, there is precision, which is the fraction of the correctly predicted positives:

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}.$$

Secondly, there is recall (also called True Positive Rate, or hit rate), which is the fraction of how many of the actual positives were correctly predicted;

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}.$$

There is also the False Positive Rate, which is used for the ROC curve, which is explained later in this section. The False Positive Rate is the fraction of the Negatives that were predicted as Positive:

$$\frac{\text{False Positive (FP)}}{\text{False Positive (FP)} + \text{True Negative (TN)}}.$$

Finally, there is the F1-Score which is the harmonic mean[2] of the precision and recall:

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

[2]The Harmonic mean is defined the following way: $H(a, b) = \frac{2}{\frac{1}{a} + \frac{1}{b}}$.

It is important to use metrics other than accuracy, because there may be a relatively large number of False Positives or False Negatives, compared to other entries in the confusion matrix. When that happens it could be the case that the model needs to be reevaluated, or retrained with different parameters to resolve the imbalance. When only accuracy is used, these insights cannot be found. However, due to the preferences of Crowe Foederer, accuracy was used as performance metric in this thesis (see Chapter 6)

The AUC is the Area Under the ROC (Receiver Operating Characteristic) Curve and is another way to review how well a model performs. An AUC of 0.5 represents a guess without information, shown as the dotted line in Figure 3.4. This means that randomly selecting a class for each observation, for example by flipping a coin, would yield the same result. An AUC of 1 represents the theoretical best model, with a True Positive Rate of 1 and a False Positive Rate of 0, and thus the model predicts perfectly. In that case the ROC is completely in the top left. Below is an illustration where the blue line is an example of an ROC curve.

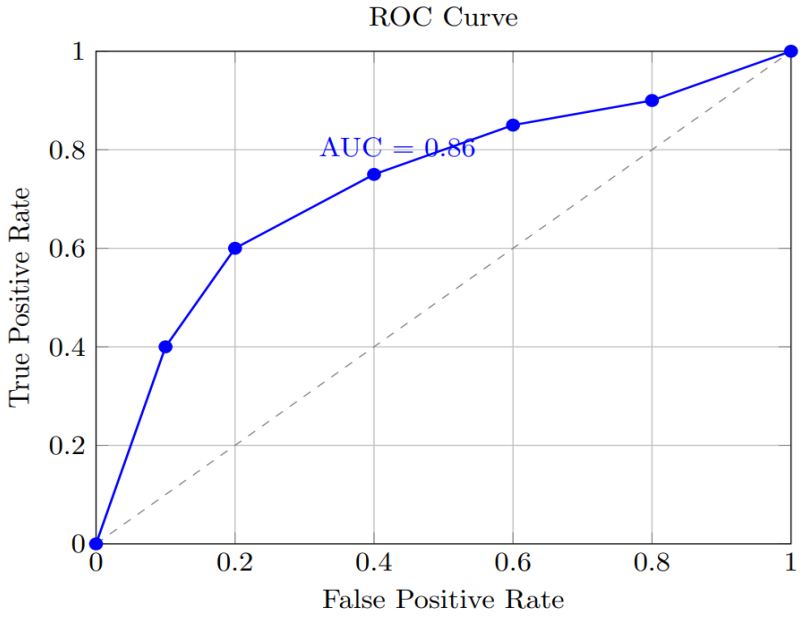

Figure 3.4: Receiver Operating Characteristic (ROC) Curve with AUC

In binary classification, a model returns one confusion matrix. The ROC curve is obtained by varying the classification threshold from 0 to 1 and for each instance calculating the True Positive Rate and False Positive rate. For the AUC, the area below the ROC curve is simply calculated. For a more detailed explanation of the ROC curve, see (Fawcett, 2006).

3.1.3 Multiclass classification model

We can expand binary classification into multiclass classification by making some changes. For a multiclass classification model, to obtain that prediction we want a mapping function f, which takes x_i, from data point (x_i, y_i), and uses its k features to classify it into one of the predefined number of categories C, the returned class is a prediction of y_i. So $f : \mathbb{R}^k \rightarrow \{1, ..., C\}$. Ideally, we want our mapping f to fit the data perfectly, i.e., $f(x_i) = y_i$, however, this will not always be true. Instead, $f(x_i) = \widehat{y}_i$ where $\widehat{y}_i$ is the predicted value, which can either be y_i or something else.

Below is an example of what a multiclass classification mapping could look like. The same data points were used as in Figure 3.1, however, every absence label was converted into 4 bins: never absent, absent more than 0 days and less or equal than 5 days, absent more than 5 days and less or equal than 30 days, and absent for 31 days. Each red square represents a bin.

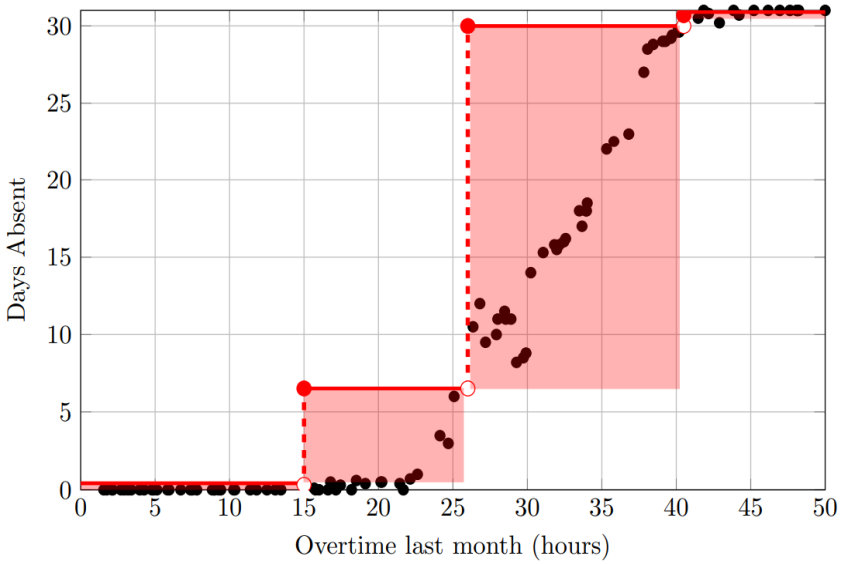

Figure 3.5: An example of what a multiclass classification model could look like, with data generated by hand for illustrational purposes.

Measures of error for multiclass classification models

To evaluate how well the obtained mapping (or model) is, measures of error can be used. While fitting a model, an error measure is (usually) minimised to find the best possible mapping. In a multiclass classification model, the error measure would determine where the jump points would be between the classes. As in the binary classification model, cross-entropy could again be used, which is again explained in (Ho & Wookey, 2019). Some slight adjustments must be made compared to binary cross-entropy. First, we need to define the true distribution of our observations $p_i(c)$, for class $c \in \{1, ..., C\}$:

$$p_i(c) = \begin{cases} 1 & \text{if } y_i = c, \\ 0 & \text{otherwise.} \end{cases}$$

After applying the mapping of a trained model, we obtain an estimated probability distribution for an observation $q_i(c)$, of which we select the class with highest probability, so $\widehat{y_i} = \max_{c \in \{1,...,C\}} q_i(c)$. p and q can be compared using the cross-entropy loss (CE), which is defined the following way:

$$\text{CE} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{c \in \{1,...,C\}} p_i(c) \log(q_i(c)).$$

This function is a generalisation of the binary cross-entropy loss function. $p_i(c)$ is equal to 1 only for one class per observation, $-\log(q_i(c))$ is exponentially decreasing in $q_i(c)$ (since $q_i(c)$ is a probability distribution and thus $0 \leq q_i(c) \leq 1$). By minimising the cross-entropy loss, the predicted probability of the correct class is maximised.

This cross-entropy loss function can be modified to include weights for different classes in case of an imbalanced dataset. This would result in the following loss function, called the weighted cross-entropy loss function (WCE):

$$\text{WCE} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{c \in \{1,...,C\}} w_c \cdot p_i(c) \log(q_i(c)), \tag{3.3}$$

where $w_c \in \mathbb{R}^+$ is the weight of class c, which is always nonnegative. Intuitively, the weights should even be positive, because a weight of 0 would just rule out a class. The weights also need to be nonnegative, because the loss function is minimised. A negative weight would lead to incorrect classifications.

Again, other measures of error exist, such as the hinge loss (Cortes & Vapnik, 1995) (commonly used in Support Vector Machines). In this thesis, we use the weighted cross-entropy loss as a measure of error, as it is both effective and widely used in multiclass classification with the XGBoost framework.

Performance metrics for multiclass classification

Other performance metrics, such as accuracy, precision, and others, which were discussed in Performance metrics for binary classification in Chapter 3.1.2, could also be used. However, as the number of classes grows, these metrics become less useful, since while evaluating a model, it is useful to see what types of misclassifications happen. Metrics like accuracy do not show this.

For a multiclass classification, the confusion matrix looks a bit different. Below is an example for 4 classes:

Actual / Predicted	Class 1	Class 2	Class 3	Class 4
Class 1	TP_1	$FP_{1,2}$	$FP_{1,3}$	$FP_{1,4}$
Class 2	$FN_{2,1}$	TP_2	$FP_{2,3}$	$FP_{2,4}$
Class 3	$FN_{3,1}$	$FN_{3,2}$	TP_3	$FP_{3,4}$
Class 4	$FN_{4,1}$	$FN_{4,2}$	$FN_{4,3}$	TP_4

Table 3.2: Confusion matrix for a 4-class classification problem. TP: True Positive, FP: False Positive, FN: False Negative.

3.1.4 XGBoost

In the original paper of (T. Chen & Guestrin, 2016), the XGBoost (which stands for Extreme Gradient Boosting) algorithm is introduced, they describe it as "a scalable end-to-end tree boosting system". Before discussing XGBoost, a short explanation will be given on tree-based models, gradients, boosting and Gradient Boosting, such that understanding XGBoost will become easier. An example of XGBoost will be given in the end.

Tree-based model

Before going into Gradient Boosting, we will first explain tree-based models. In a tree-based model, a class or value is predicted. This will be done by constructing a decision tree, which splits the data one or multiple times. The splits are performed hierarchically. The tree selects which splits to perform and the values at the terminal nodes (leaves) by minimising a loss function L.

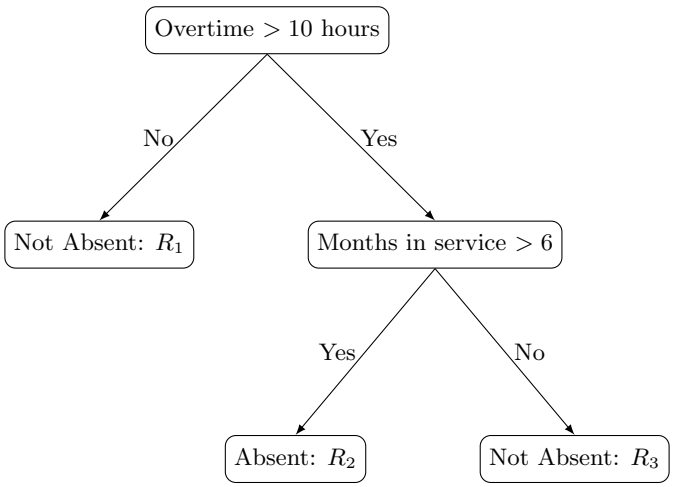

Figure 3.6: A decision tree splitting on Overtime and Months in service

The final nodes are terminal nodes; there are no splits after this. In our case, the terminal nodes determine whether an observation is predicted as absent or not. These terminal nodes can also be viewed as terminal regions, a visualisation is in the figure below.

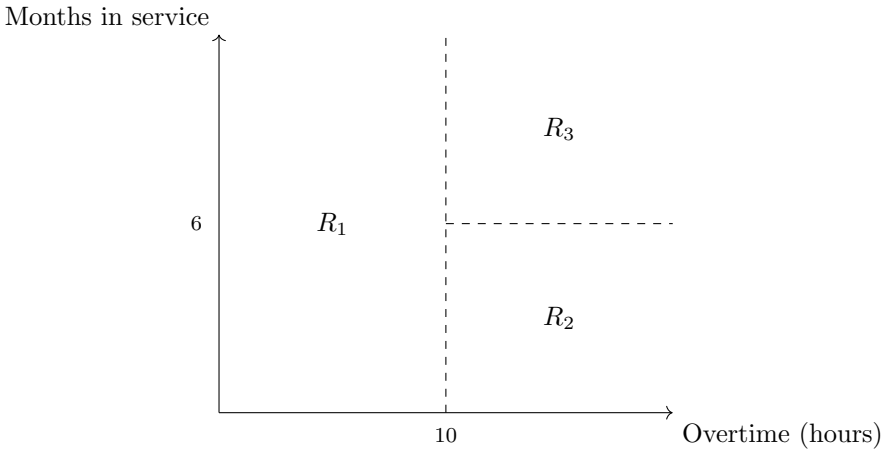

Figure 3.7: Terminal regions R_1, R_2, and R_3 based on tree splits

Another term used in tree-based models is depth, which is the number of layers a tree has (terminal nodes are not included in this calculation). So the example of Figure 3.6 has a depth of 2.

XGBoost has a specific algorithm to decide on which features and thresholds to split, and when to stop growing the tree. This algorithm (Algorithm 3) and an example (Example 3.1.1) will be discussed later in this section.

Boosting

It is important to explain boosting before going into Gradient Boosting. This is, among other things, explained in (Hastie et al., 2009), of which we follow the notation. The idea behind boosting is to combine multiple so-called weak learners. A weak learner can be seen as a simple model, for example a tree which tries to classify absence and only splits once based on overtime. Weak learners are allowed to be more complex. For example, they may have more splits on more variables and a depth greater than 1.

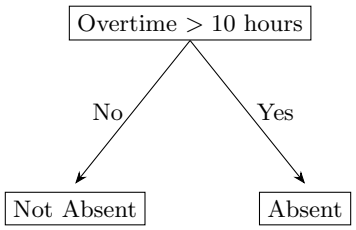

Figure 3.8: An example of a weak learner

Let $h_m(x)$ be a weak classifier, where $m = 1, ..., M$. This weak learner on its own does not yield very accurate predictions, however, combining multiple weak learners leads to better results. In boosting the strong learner is built by sequentially combining weak learners. A learning rate $\eta \in (0, 1]$ can be used to update the model more slowly, which helps prevent overfitting according to (Hastie et al., 2009). Overfitting occurs when a model fits the training data too closely, making it not generalisable to the validation or test set. This results in a high training performance but a low validation and test performance. In contrast, underfitting happens when a model fits the training data too loosely, which results in a low training, validation, and test performance.

Boosting begins with an initial model $H_0(x)$. This initial prediction depends on the boosting method and loss function L, some examples are $H_0(x) = 0$, or $H_0(x) = \bar{y}$. The model is usually iteratively updated in the following way:

$$H_m(x) = H_{m-1}(x) + \eta\alpha_m h_m(x). \tag{3.4}$$

The weight α_m and the weak learner $h_m(x)$ are calculated by the boosting algorithm.

In Gradient Boosting, each new weak learner is updated using the last weak learner and a fitted tree on the residuals of the last weak learner, as visualised below.

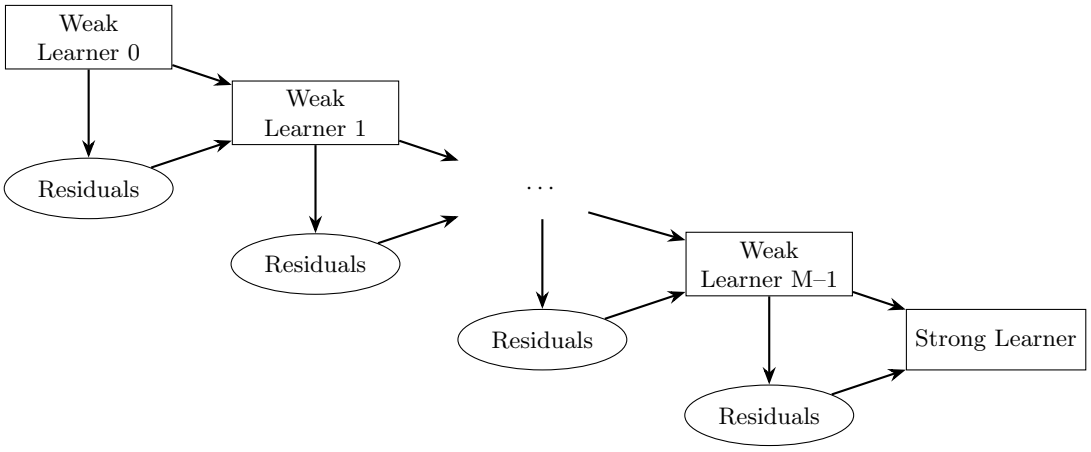

Figure 3.9: Boosting: sequential correction of residuals by weak learners

There are some models which only calculate the weight iteratively but do not have the same construction method as Equation 3.4, AdaBoost for example, as (Hastie et al., 2009) explains. In general, for boosting, the following equation holds:

$$H(x) = \sum_{m=1}^{M} \eta \alpha_m h_m(x).$$

Gradient descent
While fitting a model, a certain loss function L is minimised, such as $RMSE$, cross entropy etcetera:

$$f^* = \arg\min_f \mathcal{L}(y, f(x)).$$

This minimum is found using gradient descent. Gradient descent will now be briefly explained, for further explanation see (Boyd & Vandenberghe, 2004, Part III Algorithms) and (Ruder, 2016). In gradient descent, the gradient with respect to f is calculated, and new parameters are set using the gradient. Therefore, the loss function must be differentiable with respect to f for all possible combinations of $f(x)$ and y. There are different types of gradient descent possible, firstly, there can be differed in how much data is used to compute the gradient. Some examples of this are: stochastic gradient descent, batch gradient descent and mini-batch gradient descent. Secondly, different optimisation algorithms can be used, for example: steepest descent, Newton's method, momentum, Adagrad, RMSprop, Adam and many more. To make this thesis not too comprehensive only steepest descent will be explained, for a more in-depth explanation see (Boyd & Vandenberghe, 2004, Part III Algorithms) and (Ruder, 2016).

In steepest descent according to (Boyd & Vandenberghe, 2004, Chapter 9.4), we take the direction which decreases a function g by the most. Following the notation of (Boyd & Vandenberghe, 2004), the (unnormalised) steepest descent step is:

$$\Delta z_i = -\frac{\partial g(z_i)}{\partial z_i} \cdot e_i, \text{ where } e_i \text{ is the } i\text{-th standard basis vector.}$$

Where Δz_i is the descent step in coordinate i. Unnormalised means that the step size depends on the gradient size and not only on the sign (also called direction) of the gradient. In normalised steepest descent, the step size is set to 1 and depends only on the sign of the gradient. The steepest descent step for a loss function would be:

$$\Delta f(x_i) = -\frac{\partial \mathcal{L}(y_i, f(x_i))}{\partial f(x_i)}.$$

The steepest descent method will be used in Gradient Boosting.

Gradient Boosting

We will now discuss the Gradient Boosting algorithm as it is discussed in (Hastie et al., 2009), with some slight modifications. First, the general idea will be explained, and then the pseudo-code will be given. The idea behind Gradient Boosting is to initialise the model with an initial constant or class, calculate the residuals by optimising over the loss function using steepest descent via the gradient, fit a tree based on these residuals and update the model based on the terminal regions of the fitted tree on the residuals. The calculation of residuals and updating the models is repeated multiple times. See Figure 3.9 for a visualisation. In Figures 3.10 and 3.11 the updating of the model is visualised in a heatmap, where a darker colour means a higher value in the case of regression and a higher probability in the case of (binary) classification. (The terminal region heatmaps cannot be easily visualised in the case of multiclass classification.) The final model has a prediction for each possible feature combination. Figure 3.11 shows a strong learner.

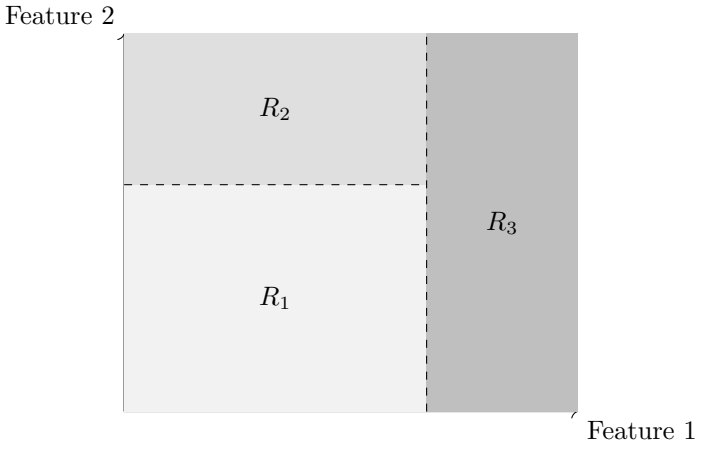

Figure 3.10: Heatmap of updated model, showing terminal regions R_1, R_2, and R_3 of the model fitted on residuals of initial model.

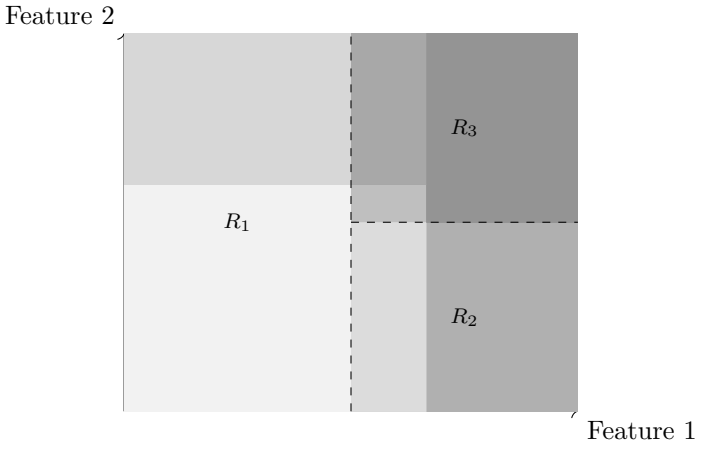

Figure 3.11: Heatmap of updated model, showing terminal regions R_1, R_2, and R_3 of the model fitted on residuals of previous model.

The negative gradients used in steepest descent are called pseudo-residuals in boosting. Pseudo-residuals are the negative gradient of the loss function with respect to the model's (current) prediction. They are called pseudo-residuals to show that they are similar to residuals (difference between y and $\widehat{y}$), but generalised to any differentiable loss function.

(Gradient) Boosting can have some disadvantages, as explained in (Hastie et al., 2009), including the risk of overfitting. Therefore, a learning rate η is introduced. The learning rate determines how much of

each newly generated tree, fitted on the residuals, contributes to the final model. By reducing the impact of each tree, the risk of overfitting is reduced as (Hastie et al., 2009) implies. A bigger tree size (also called depth) or a larger amount of trees M used in the boosting procedure could also lead to overfitting. However, increasing the tree size or the number of trees could reduce the risk of underfitting.

The advantage of boosting is that it reduces bias, with a relatively low increase in the variance according to (Ranglani, 2024).

Since the general idea behind Gradient Boosting has been explained, the algorithm can be made clearer via a pseudo-code. The first pseudo-code shows gradient tree boosting for regression tasks, the second pseudo-code shows gradient tree boosting for classification tasks. These algorithms follow the same Gradient Boosting logic that has been discussed.

ALGORITHM 1: Gradient Tree Boosting algorithm for regression tasks

Input : Training data $\{(x_i, y_i)\}_{i=1}^n$, differentiable loss function $\mathcal{L}(y, f(x))$, number of iterations M, learning rate η

Output: Final model $f^*(x) = f_M(x)$

Initialise $f_0(x) = \arg\min_\tau \sum_{i=1}^n \mathcal{L}(y_i, \tau)$
for $m = 1, \ldots, M$ **do**
  for $i = 1, \ldots, n$ **do**
    Compute pseudo-residual:
    $r_{im} = -\frac{\partial \mathcal{L}(y_i, f_{m-1}(x_i))}{\partial f_{m-1}(x_i)}$
  end
  Fit a regression tree to $\{(x_i, r_{im})\}_{i=1}^n$ to get terminal regions $R_{jm},\ j = 1, \ldots, J_m$
  for $j = 1, \ldots, J_m$ **do**
    Compute region-specific update:
    $\tau_{jm} = \arg\min_\tau \sum_{x_i \in R_{jm}} \mathcal{L}(y_i, f_{m-1}(x_i) + \tau)$
  end
  Update the model using indicator function $\mathbb{I}_{x \in R_{jm}}$ and region-specific update τ_{jm}:
  $f_m(x) = f_{m-1}(x) + \eta \sum_{j=1}^{J_m} \tau_{jm} \cdot \mathbb{I}_{x \in R_{jm}}$
end
return $f^*(x) = f_M(x)$

ALGORITHM 2: Gradient Tree Boosting algorithm for multi-class classification

Input : Training data $\{(x_i, y_i)\}_{i=1}^n$, number of classes C, number of iterations M, differentiable loss function $\mathcal{L}(y, f(x))$, learning rate η

Output: Additive model functions $f_c^*(x) = f_{c,M}(x)$ for $c = 1, \ldots, C$

Initialise $f_{c,0}(x) = 0$, for all $c = 1, \ldots, C$
for $m = 1$ **to** M **do**
  for $c = 1$ **to** C **do**
    for $i = 1$ **to** n **do**
      Compute negative gradient (pseudo-residual vector):
      $$r_{icm} = -\frac{\partial \mathcal{L}(y_i, f_{1,m-1}(x_i), \ldots, f_{C,m-1}(x_i))}{\partial f_{c,m-1}(x_i)}$$
    end
    Fit a regression tree to $\{(x_i, r_{icm})\}_{i=1}^n$ to obtain terminal regions R_{jcm}, $j = 1, \ldots, J_m$
    for $j = 1$ **to** J_m **do**
      Compute update in each region:
      $$\tau_{jcm} = \arg\min_\tau \sum_{x_i \in R_{jcm}} \mathcal{L}(y_i, f_{c,m-1}(x_i) + \tau \cdot \mathbf{e}_c)$$
      where $\mathbf{e}_c$ is the c-th standard basis vector
    end
    Update model using indicator function $\mathbb{I}_{x \in R_{jcm}}$ and region-specific update τ_{jcm}:
    $$f_{cm}(x) = f_{c,m-1}(x) + \eta \sum_{j=1}^{J_m} \tau_{jcm} \cdot \mathbb{I}_{x \in R_{jcm}}$$
  end
end
return $f_c^*(x) = f_{cM}(x)$ *for* $c = 1, \ldots, C$

XGBoost

XGBoost does some things differently compared to Gradient Boosting. Firstly, it uses a regularised objective function, which penalises more complex trees. Secondly, it uses the Hessian and gradient for faster convergence. It also implements a learning rate to prevent overfitting. Finally, it uses feature subsampling to avoid overfitting, each newly generated tree is only fitted on a prespecified percentage of all the features.

For the explanation of XGBoost notation of (T. Chen & Guestrin, 2016) will be followed with some minor adjustments. The regularised objective looks like this:

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} L(y_i, \widehat{y}_i) + \sum_{m=1}^{M} \Omega(f_m), \text{ where } \Omega(f_m) = \gamma T_m + \frac{1}{2}\lambda \|\beta_m\|^2.$$

L is the convex differentiable loss function and Ω is the part which regularises the objective $\mathcal{L}$ by penalizing more complex models. $T_m \in \mathbb{N}$ is the number of leaves of tree m, $\beta_m \in \mathbb{R}^{T_m}$ is the weight vector of the leaves of tree m. $\beta_{m,j}$ is the weight of leaf j on tree m. The tree below has $T_m = 3$ and $\beta_m = (-7, 2.25, 9.25)$. The leaf vector β_m shows by how much the terminal regions must be updated during the boosting procedure. Note that the tree is fitted on residuals, and thus indicates by how much the predictions should be adjusted. Therefore, it differs from Figures 3.6 and 3.8.

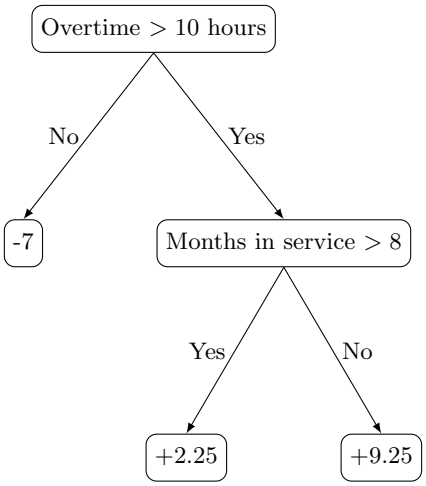

Figure 3.12: Example of XGBoost after an iteration

$\lambda \in \mathbb{R}^+$ and $\gamma \in \mathbb{R}^+$ are cost parameters which determine how much regularisation there is. A higher γ and λ penalise a complex tree structure more, and therefore avoid overfitting. γ penalises based on the number of leaves and thus the number of splits the tree has, and λ penalises by how much the residuals change in the leaves. The sum is taken over m, the trees. In short, the loss function minimises the total loss plus the complexity of the trees.

The reasoning behind the splitting of the decision tree which XGBoost uses, will be explained, similarly to (T. Chen & Guestrin, 2016). Since in boosting the model is updated sequentially, at iteration m, the following loss is obtained:

$$\mathcal{L}^{(m)} = \sum_{i=1}^{n} L\left(y_i, \widehat{y}_i^{(m-1)} + f_m(x_i)\right) + \Omega(f_m).$$

To quickly optimise over the objective, the second order Taylor series approximation is taken.

$$\mathcal{L}^{(m)} \approx \sum_{i=1}^{n} \left[L\left(y_i, \widehat{y}_i^{(m-1)}\right) + \frac{\partial L(y_i, \widehat{y}_i^{(m-1)})}{\partial \widehat{y}_i^{(m-1)}} f_m(x_i) + \frac{1}{2} \frac{\partial^2 L(y_i, \widehat{y}_i^{(m-1)})}{\partial^2 \widehat{y}_i^{(m-1)}} f_m^2(x_i) \right] + \Omega(f_m) \tag{3.5}$$

Then the constant terms are removed for a simplified objective. The term $L\left(y_i, \widehat{y}_i^{(m-1)}\right)$ is only affected by the previous models and not by the current tree that is fitted on the residuals. So therefore it is constant.

$$\widetilde{\mathcal{L}}^{(m)} \approx \sum_{i=1}^{n} \left[\frac{\partial L(y_i, \widehat{y}_i^{(m-1)})}{\partial \widehat{y}_i^{(m-1)}} f_m(x_i) + \frac{1}{2} \frac{\partial^2 L(y_i, \widehat{y}_i^{(m-1)})}{\partial^2 \widehat{y}_i^{(m-1)}} f_m^2(x_i) \right] + \Omega(f_m)$$

The function q is defined, which maps the observations x_i into the leaves j based on the tree structure. Each leaf has its own instance set I_j, which contains the observations assigned to leaf j:

$$I_j = \{\, i \mid q(x_i) = j \,\}.$$

$\Omega(f_m)$ is filled in to obtain:

$$\begin{aligned}
\widetilde{\mathcal{L}}^{(m)} &\approx \sum_{i=1}^{n} \left[\frac{\partial L(y_i, \widehat{y}_i^{(m-1)})}{\partial \widehat{y}_i^{(m-1)}} \beta_{m,j} + \frac{1}{2} \frac{\partial^2 L(y_i, \widehat{y}_i^{(m-1)})}{\partial^2 \widehat{y}_i^{(m-1)}} \beta_{m,j}^2 \right] + \gamma T_m + \frac{1}{2} \lambda \sum_{j=1}^{T_m} \beta_{m,j}^2 \\
&= \sum_{j=1}^{T_m} \left[\left(\sum_{i \in I_j} \frac{\partial L(y_i, \widehat{y}_i^{(m-1)})}{\partial \widehat{y}_i^{(m-1)}} \right) \beta_{m,j} + \frac{1}{2} \left(\sum_{i \in I_j} \frac{\partial^2 L(y_i, \widehat{y}_i^{(m-1)})}{\partial^2 \widehat{y}_i^{(m-1)}} + \lambda \right) \beta_{m,j}^2 \right] + \gamma T_m.
\end{aligned} \tag{3.6}$$

(T. Chen & Guestrin, 2016) then states that for a fixed $q(\boldsymbol{x})$ the optimal weight β_j^* of leaf j with respect to $\widetilde{\mathcal{L}}^{(m)}$ is the following:

$$\beta^*_{m,j} = -\frac{\sum_{i\in I_j} \frac{\partial L(y_i, \widehat{y}_i^{(m-1)})}{\partial \widehat{y}_i^{(m-1)}}}{\sum_{i\in I_j} \frac{\partial^2 L(y_i, \widehat{y}_i^{(m-1)})}{\partial^2 \widehat{y}_i^{(m-1)}} + \lambda}. \tag{3.7}$$

If the denominator of $\beta^*_{m,j}$ equals 0, the XGBoost package treats the split as invalid and does not perform it (as is shown in the source code (DMLC XGBoost Contributors, 2023, lines 123 – 137)). The optimal weight is obtained by taking the derivative of $\widetilde{\mathcal{L}}^{(m)}$ with respect to β_j and setting it to 0. This can be done since L is assumed to be a twice differentiable convex loss function. Meaning that

$$\frac{\partial^2 L(y_i, \widehat{y}_i^{(m-1)})}{\partial^2 \widehat{y}_i^{(m-1)}} \geq 0 \Longrightarrow \frac{\partial^2 L(y_i, \widehat{y}_i^{(m-1)})}{\partial^2 \widehat{y}_i^{(m-1)}} + \lambda \geq 0,$$

and thus $\widetilde{\mathcal{L}}^{(m)}$ is a convex quadratic function in β_j. Hence, setting the derivative of $\widetilde{\mathcal{L}}^{(m)}$ with respect to β_j to zero yields the optimal weight.
Putting the weight β^*_j of Equation 3.7 into Equation 3.6 results in:

$$\widetilde{\mathcal{L}}^{(m)}(q) = -\frac{1}{2}\sum_{j=1}^{T_m} \frac{\left(\sum\limits_{i\in I_j} \frac{\partial L(y_i, \widehat{y}_i^{(m-1)})}{\partial \widehat{y}_i^{(m-1)}}\right)^2}{\sum\limits_{i\in I_j} \frac{\partial^2 L(y_i, \widehat{y}_i^{(m-1)})}{\partial^2 \widehat{y}_i^{(m-1)}} + \lambda} + \gamma T_m.$$

Finally, a node is defined as $I = I_l \cup I_r$, where I_l is the left instances after the split and I_r is the right. This gives the following loss reduction after a split:

$$\mathcal{L}_{\text{split}} = \frac{1}{2}\left[\frac{\left(\sum\limits_{i\in I_l} \frac{\partial L(y_i, \widehat{y}_i^{(m-1)})}{\partial \widehat{y}_i^{(m-1)}}\right)^2}{\sum\limits_{i\in I_l} \frac{\partial^2 L(y_i, \widehat{y}_i^{(m-1)})}{\partial^2 \widehat{y}_i^{(m-1)}} + \lambda} + \frac{\left(\sum\limits_{i\in I_r} \frac{\partial L(y_i, \widehat{y}_i^{(m-1)})}{\partial \widehat{y}_i^{(m-1)}}\right)^2}{\sum\limits_{i\in I_r} \frac{\partial^2 L(y_i, \widehat{y}_i^{(m-1)})}{\partial^2 \widehat{y}_i^{(m-1)}} + \lambda} - \frac{\left(\sum\limits_{i\in I} \frac{\partial L(y_i, \widehat{y}_i^{(m-1)})}{\partial \widehat{y}_i^{(m-1)}}\right)^2}{\sum\limits_{i\in I} \frac{\partial^2 L(y_i, \widehat{y}_i^{(m-1)})}{\partial^2 \widehat{y}_i^{(m-1)}} + \lambda}\right] - \gamma. \tag{3.8}$$

γT becomes $-\gamma$, because before the split we have one leaf and after the split we have 2 leaves. At each iteration, the split with the highest loss reduction is selected, provided it exceeds the cost parameter γ. Below is the pseudo-code for selecting the best split, in which instance set I is a subset of the data, because after a split in a tree, a part goes to the left and a part towards the right.

Step 5 of Algorithm 3 says "in sorted", this is because the algorithm sorts the values of a feature increasingly, such that it can split based on the sorted index. Sorting by feature values ensures that the splits correspond to meaningful thresholds in the data rather than arbitrary indices.

ALGORITHM 3: Exact greedy split finding

Input : Labels and predictions $\{(y_i, \widehat{y}_i)\}_{i=1}^{n}$, instance set I, number of features K, split cost parameter γ, weight cost parameter λ, twice differentiable loss function $L(y_i, \widehat{y}_i)$

Output: Best feature and split point

$G = \sum_{i \in I} \frac{\partial L(y_i, \widehat{y}_i)}{\partial \widehat{y}_i}, \quad H = \sum_{i \in I} \frac{\partial^2 L(y_i, \widehat{y}_i)}{\partial^2 \widehat{y}_i}$;
gain $= 0$;
for $k = 1, \ldots, K$ **do**
  $G_l = 0, \quad H_l = 0$;
  for j in sorted $(I$, by $x_{jk})$ **do**
    $G_l = G_l + \frac{\partial L(y_j, \widehat{y}_j)}{\partial \widehat{y}_j}, \quad H_l = H_l + \frac{\partial^2 L(y_j, \widehat{y}_j)}{\partial^2 \widehat{y}_j}$;
    $G_r = G - G_l, \quad H_r = H - H_l$;
    gain $= \max\left(\text{gain}, \frac{1}{2} \cdot \left(\frac{G_l^2}{H_l + \lambda} + \frac{G_r^2}{H_r + \lambda} - \frac{G^2}{H + \lambda}\right)\right)$;
  end
end
if gain $> \gamma$ **then**
  return Best split with maximum score
else
  return No good split found
end

After this algorithm the optimal leaf weight $\beta_{m,j}$ of leaf j in tree m is calculated according to Equation 3.7. This results in:

$$\beta_l = -\frac{G_l}{H_l + \lambda}, \tag{3.9}$$

$$\beta_r = -\frac{G_r}{H_r + \lambda}. \tag{3.10}$$

If weights are used to deal with imbalanced classes, then $L(y_i, \widehat{y}_i)$ is replaced by $w_i \cdot L(y_i, \widehat{y}_i)$, where w_i represents the weight of observation i, which is class-specific. More regarding imbalanced data is in Chapter 3.2.2.

Algorithm 3 is a greedy algorithm, meaning that it goes through all possibilities, which is slow according to (T. Chen & Guestrin, 2016). Therefore, there is also a faster approximation algorithm, which groups data points together in quantiles (or bins). So while checking where to split, for each feature, all feature-specific bins are checked, instead of all possible values. These splits can be proposed for each new tree (locally) or all trees at the same time (globally). The pseudo-code for the approximate split finding algorithm can be found in Algorithm 5 in Appendix B.

To propose the candidate split points, the weighted quantile sketch algorithm is used. The paper also discusses how, for sparse data, the splits are found (which is not applicable to the data used for this thesis). More information regarding the weighted quantile sketch algorithm and the sparsity-aware split finding can be found in (T. Chen & Guestrin, 2016). These concepts will not be explained, such that this thesis does not become overly detailed and lengthy.

The approximation algorithm (Algorithm 5) and the weighted quantile sketch algorithm both allow for less memory usage and faster tree generation, without losing accuracy according to (T. Chen & Guestrin, 2016). So these algorithms are both used to predict absenteeism in this thesis, instead of the greedy algorithm (Algorithm 3).

XGBoost and multiclass classification

For multiclass classification prediction $\widehat{y}_i$ becomes a vector with a discrete probability distribution and label y_i becomes a standard basis vector, which represents the true distribution. So if, for example, a model gives the following prediction: $\widehat{y}_1 = [0.25, 0.1, 0.3, 0.35]$ that means that according to the model, there is a 25% chance that y_1 is from class 0, a 10% chance that y_1 is from class 1, etcetera. The final prediction would be class 3 since this has the highest probability. If y_1 would belong to class 2 then it should have the following form: $y_1 = [0, 0, 1, 0]$ (the true distribution).

XGBoost and Gradient Boosting
There are some differences between XGBoost and the Gradient Boosting algorithms discussed earlier (Algorithm 1 and 2 in Chapter 3.1.4). Firstly, Gradient Boosting calculates the pseudo-residuals and a tree is fitted on those, minimising the loss through steepest descent. After the tree is fitted, the region-specific update $\tau_{j(c)m}$ is calculated. In contrast, XGBoost fits a tree to minimise Equation 3.5, using a specific formula for the optimal weight (Equation 3.7). Additionally, XGBoost uses regularisation.

XGBoost also has several other hyperparameters, which have not been discussed yet. We quickly go over the ones we use:

- Maximum tree depth, a positive integer limiting the number of layers in each tree. This helps prevent overfitting by limiting the complexity of a tree.
- Subsample, a percentage which determines how much of the data is used to grow a tree in XGBoost. This helps prevent overfitting by restricting the amount of data for each tree.
- Colsample is similar to subsample. It determines what fraction of features is used to grow a tree, again to help avoid overfitting. Colsample can be applied per tree, per level or per node. In this thesis, colsample is applied only per tree.
- Minimum child weight specifies the minimal sum of Hessians $H = H_l + H_r$ required for a split to be valid. This helps avoid overfitting by indirectly restricting the minimal loss reduction needed for a split.

More hyperparameters of XGBoost can be found in the documentation (XGBoost developers, 2022).

Example 3.1.1 In this example a few iterations of XGBoost will be shown, using Algorithm 3 (exact greedy split finding). The goal is to predict how many days someone is absent, so for each observation i we want to predict days absent current month (y_i), using overtime hours last month ($x_{i,\text{Overtime}}$) and months in service ($x_{i,\text{Months in service}}$). Consider the following data:

	Days absent current month	**Overtime hours last month**	**Months in service**
Person 1	14	15	8
Person 2	0	0	16
Person 3	2	10	24
Person 4	30	20	32

Table 3.3: Data XGBoost iteration example

The following (hyper)parameters are chosen:

- number of features $K = 2$
- split loss cost $\gamma = 0$ (default of XGBoost)
- L_2 regularisation cost $\lambda = 1$ (default of XGBoost)
- number of iterations (or trees generated) M
- learning rate $\eta = 0.1$
- maximum tree depth is 2

Consider the following squared error loss function:

$$L(y, \widehat{y}) = \sum_{i=1}^{n} L(y_i, \widehat{y_i}) = \sum_{i=1}^{n} \frac{1}{2}(y_i - \widehat{y_i})^2,$$

with the following partial derivative with respect to $\widehat{y_i}$:

$$\frac{\partial L(y_i, \widehat{y_i})}{\partial \widehat{y_i}} = -(y_i - \widehat{y_i}) = \widehat{y_i} - y_i,$$

and the following second-order partial derivative with respect to $\widehat{y_i}$:

$$\frac{\partial^2 L(y_j, \widehat{y_j})}{\partial^2 \widehat{y_j}} = 1.$$

The first step in performing XGBoost is to come up with an initial constant prediction in which we minimise the loss function (see Step 1 Algorithm 1). So

$$\arg\min_\tau \sum_{i=1}^{n} \frac{1}{2}(y_i - \tau)^2 = \arg\min_\tau \frac{1}{2}(14-\tau)^2 + \frac{1}{2}(0-\tau)^2 + \frac{1}{2}(2-\tau)^2 + \frac{1}{2}(30-\tau)^2.$$

This is minimised by setting the partial derivative with respect to τ equal to 0, yielding:

$$(14-\tau) + (0-\tau) + (2-\tau) + (30-\tau) = 0.$$

This results in $\tau = 11.5$, the average of y_i. So $f_0(x) = 11.5$. Now we are going fit the first tree using Algorithm 3 we have

$$G = \sum_{i \in I} \frac{\partial L(y_i, \widehat{y}_i)}{\partial \widehat{y}_i} = 4 \cdot 11.5 - 14 - 0 - 2 - 30 = 0, \quad H = \sum_{i \in I} \frac{\partial^2 L(y_i, \widehat{y}_i)}{\partial^2 \widehat{y}_i} = 4.$$

For the first split, we will go through all possible split options for all features. We will start with the feature overtime. First we sort the possible split options with respect to the feature overtime to obtain:

Person	**Overtime**	**Gradient** $(\widehat{y}_i - y_i)$	**Hessian**	G_l	G_r	H_l	H_r
2	0	11.5	1	11.5	-11.5	1	3
3	10	9.5	1	21	-21	2	2
1	15	-2.5	1	18.5	-18.5	3	1
4	20	-18.5	1	n.a.	n.a.	n.a.	n.a.
Total		0	4				

Table 3.4: Calculations used to evaluate possible first split on overtime of XGBoost example.

Then the gain can be calculated for each possible split, using:

$$\text{gain}_{i,\text{feature}} = \frac{1}{2} \cdot \left(\frac{G_l^2}{H_l + \lambda} + \frac{G_r^2}{H_r + \lambda} - \frac{G^2}{H + \lambda} \right).$$

This results in the following calculations:

$$\text{gain}_{1,\text{Overtime}} = \frac{1}{2} \cdot \left(\frac{11.5^2}{1+1} + \frac{(-11.5)^2}{3+1} - \frac{0^2}{4+1} \right) = 49.59375,$$

$$\text{gain}_{2,\text{Overtime}} = \frac{1}{2} \cdot \left(\frac{21^2}{2+1} + \frac{(-21)^2}{2+1} - \frac{0^2}{4+1} \right) = 147,$$

$$\text{gain}_{3,\text{Overtime}} = \frac{1}{2} \cdot \left(\frac{18.5^2}{3+1} + \frac{(-18.5)^2}{1+1} - \frac{0^2}{4+1} \right) = 128.34375.$$

Now the same is done for the feature Months in service.

Person	**Months**	**Gradient** $(\widehat{y}_i - y_i)$	**Hessian**	G_l	G_r	H_l	H_r
1	8	-2.5	1	-2.5	2.5	1	3
2	16	11.5	1	9	-9	2	2
3	24	9.5	1	18.5	-18.5	3	1
4	32	-18.5	1	n.a.	n.a.	n.a.	n.a.
Total		0	4				

Table 3.5: Calculations used to evaluate possible first split on months in service of XGBoost example.

$$\text{gain}_{1,\text{Months in service}} = \frac{1}{2} \cdot \left(\frac{(-2.5)^2}{1+1} + \frac{(-2.5)^2}{3+1} - \frac{0^2}{4+1} \right) = 2.34375,$$

$$\text{gain}_{2,\text{Months in service}} = \frac{1}{2} \cdot \left(\frac{9^2}{2+1} + \frac{(-9)^2}{2+1} - \frac{0^2}{4+1} \right) = 27,$$

$$\text{gain}_{3,\text{Months in service}} = \frac{1}{2} \cdot \left(\frac{18.5^2}{3+1} + \frac{(-18.5)^2}{1+1} - \frac{0^2}{4+1} \right) = 128.34375.$$

Then the maximum gain is chosen as a split, which is $\text{gain}_{2,\text{overtime}} = 147$. The weights for the nodes need to be calculated. Using Equations 3.9 and 3.10, the following weights are obtained:

$$\beta_l = -\frac{21}{2+1} = -7, \quad \beta_r = -\frac{-21}{2+1} = 7.$$

So the following split is obtained:

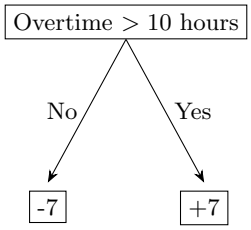

Figure 3.13: First split of XGBoost example

This puts person 2 and 3 in the left node and person 1 and 4 in the right node. We will look for further splits, starting with the left node.

First, the new G and H must be calculated for the left node.

$$G = \sum_{i \in I} \frac{\partial L(y_i, \widehat{y}_i)}{\partial \widehat{y}_i} = 2 \cdot 11.5 - 0 - 2 = 21, \quad H = \sum_{i \in I} \frac{\partial^2 L(y_i, \widehat{y}_i)}{\partial^2 \widehat{y}_i} = 2$$

Person	**Overtime**	**Gradient** $(\widehat{y}_i - y_i)$	**Hessian**	G_l	G_r	H_l	H_r
2	0	11.5	1	11.5	9.5	1	1
3	10	9.5	1	n.a.	n.a.	n.a.	n.a.
Total		21	2				

Table 3.6: Calculations used to evaluate possible second split on overtime for left node of XGBoost example.

$$\text{gain}_{1,\text{Overtime}} = \frac{1}{2} \cdot \left(\frac{11.5^2}{1+1} + \frac{9.5^2}{1+1} - \frac{21^2}{2+1} \right) = -17.875.$$

Now try to split on the other feature on the left leaf.

Person	**Months**	**Gradient** $(\widehat{y}_i - y_i)$	**Hessian**	G_l	G_r	H_l	H_r
2	16	11.5	1	11.5	9.5	1	1
3	24	9.5	1	n.a.	n.a.	n.a.	n.a.
Total		21	2				

Table 3.7: Calculations used to evaluate possible second split on months in service for left node of XGBoost example.

$$\text{gain}_{1,\text{Months in service}} = \frac{1}{2} \cdot \left(\frac{11.5^2}{1+1} + \frac{9.5^2}{1+1} - \frac{21^2}{2+1} \right) = -17.875.$$

Both gains on the left leaf are negative so we will not split any further. Even though another split would seem reasonable, the regularisation parameter λ makes sure not to split here. 0 and 2 days absent already are relatively close to each other. Now try to split on the right leaf on overtime.

Person	**Overtime**	**Gradient** $(\widehat{y}_i - y_i)$	**Hessian**	G_l	G_r	H_l	H_r
1	15	-2.5	1	-2.5	-18.5	1	1
4	20	-18.5	1	n.a.	n.a.	n.a.	n.a.
Total		-21	2				

Table 3.8: Calculations used to evaluate possible second split on overtime for right node of XGBoost example.

$$\text{gain}_{1,\text{Overtime}} = \frac{1}{2} \cdot \left(\frac{(-2.5)^2}{1+1} + \frac{(-18.5)^2}{1+1} - \frac{(-21)^2}{2+1} \right) = 27.25.$$

Next, we consider months in service on the right node.

Person	**Months**	**Gradient** $(\widehat{y}_i - y_i)$	**Hessian**	G_l	G_r	H_l	H_r
1	8	-2.5	1	-2.5	-18.5	1	1
4	32	-18.5	1	n.a.	n.a.	n.a.	n.a.
Total		-21	2				

Table 3.9: Calculations used to evaluate possible second split on months in service for right node of XGBoost example.

$$\text{gain}_{1,\text{Months in service}} = \frac{1}{2} \cdot \left(\frac{(-2.5)^2}{1+1} + \frac{(-18.5)^2}{1+1} - \frac{(-21)^2}{2+1} \right) = 27.25.$$

Both the gains are the same, so on which feature is split does not matter. For this example, we split on months in service. First, the weights must be calculated:

$$\beta_l = -\frac{-2.5}{1+1} = 2.25, \quad \beta_r = -\frac{-18.5}{1+1} = 9.25.$$

We update the regions the following way:

$$R_1 : 11.5 + 0.1 \cdot -7 = 10.8,$$
$$R_2 : 11.5 + 0.1 \cdot 9.25 = 12.425,$$
$$R_3 : 11.5 + 0.1 \cdot 2.25 = 11.725.$$

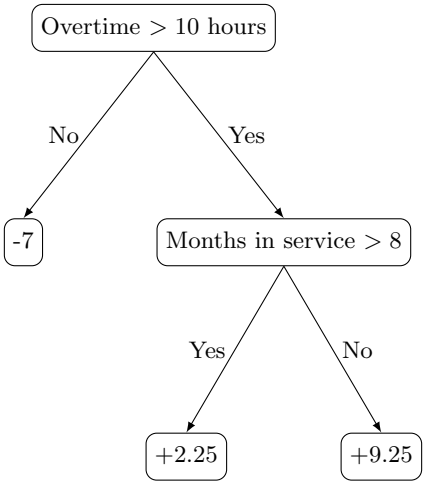

Figure 3.14: Second split of XGBoost example

The tree above f_1 we are going to add to the original model:

$$f_1 = f_0 + \eta \sum_{j=1}^{J_1} \tau_{j1} \cdot \mathbb{I}_{x \in R_{j1}}.$$

The figure below shows how this looks in the feature space.

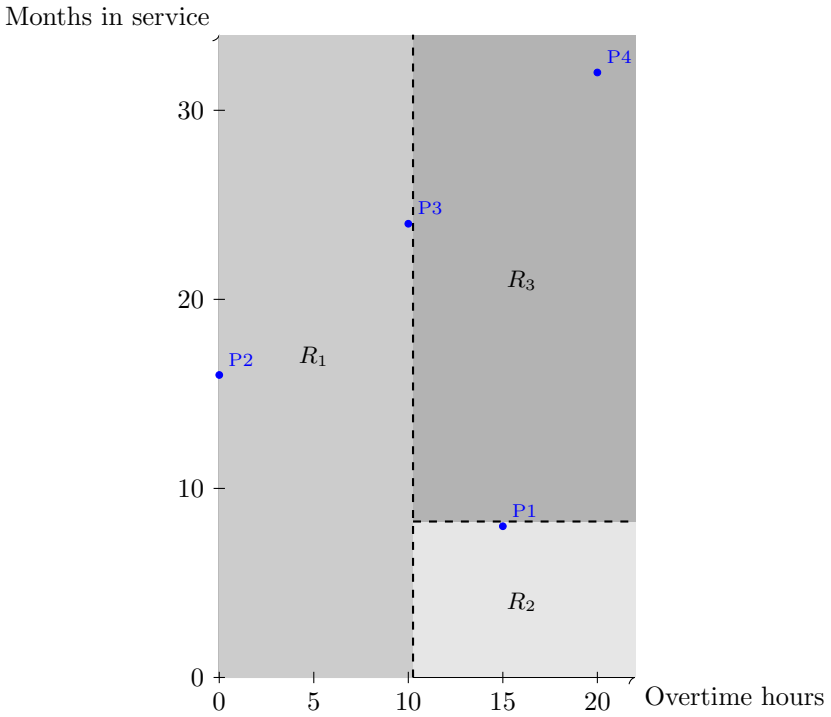

Figure 3.15: Updated model

So person 1 is in region 2, person 2 and 3 are in region 1, and person 4 is in region 3. These new predictions are now used for the next iteration of the XGBoost algorithm. ▲[3]

3.1.5 Support Vector Machine (SVM)

This subsection contains an explanation of the Support Vector Machine (Cortes & Vapnik, 1995). This explanation will be kept brief since the main focus of this thesis is XGBoost, and the SVM is mainly for validation. A more in-depth explanation can be found in (Hastie et al., 2009). For classification tasks, an SVM tries to find the optimal hyperplane to separate the data D into the predetermined classes, by minimising over parameters $\mathbf{w} \in \mathbb{R}^K$ and $b \in \mathbb{R}$, which define the separation hyperplane. (Recall that $k = 1, ..., K$ are indices for the features.) This is done by solving a Quadratic Program (QP). Usually, perfect separation is not possible; therefore, some new parameters are introduced. Let $\tau \in \mathbb{R}^+$ be a cost parameter, which allows for some misclassifications (using slack variable $\xi_i \in \mathbb{R}^+$) to happen but still penalises them. The QP for an SVM, which does binary classification using a linear hyperplane, looks as follows.

$$\begin{aligned}
\min_{\mathbf{w},b} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + \tau \sum_{i=1}^{n} \xi_i & \\
\text{s.t.} \quad & y_i \left(x_i^\top \mathbf{w} + b\right) \geq 1 - \xi_i & \forall\ i = 1, ..., n \\
& \xi_i \geq 0 & \forall\ i = 1, ..., n \\
& y_i \in \{-1, 1\} & \forall\ i = 1, ..., n
\end{aligned}$$

[3] In this thesis, the symbol ▲ indicates the end of an example or definition.

Note that for the SVM, the labels $y_i = 0$ are changed to $y_i = -1$ to satisfy the QP formulation above.[4] Once the optimal hyperplane parameters $\mathbf{w}$ and b are obtained from the QP, a test observation x_i is classified using the obtained separator the following way:

$$\hat{y}_i = \begin{cases} 1 & \text{if } \mathbf{w}^\top x_i + b > 0, \\ -1 & \text{else.} \end{cases}$$

For multiclass classification, the QP is modified slightly. For each class $c \in \{1, \ldots, C\}$, the goal is to find the separator which distinguishes class c from all other classes. This results in the following QP:

$$\begin{aligned} \min_{\mathbf{w}^{(c)}, b^{(c)}} \quad & \frac{1}{2}\|\mathbf{w}^{(c)}\|^2 + \tau \sum_{i=1}^{n} \xi_i^{(c)} & \\ \text{s.t.} \quad & y_i^{(c)} \left(x_i^\top \mathbf{w}^{(c)} + b^{(c)} \right) \geq 1 - \xi_i^{(c)} & \forall\ i = 1, ..., n \\ & \xi_i^{(c)} \geq 0 & \forall\ i = 1, ..., n \\ & y_i^{(c)} \in \{-1, 1\} & \forall\ i = 1, ..., n. \end{aligned}$$

Again, after solving the QP and obtaining the optimal hyperplane parameters $\mathbf{w}^{(c)}$ and $b^{(c)}$, for each class $c \in \{1, ..., C\}$, a test observation x_i is classified using the obtained separators the following way:

$$\hat{y}_i = \arg \max_{c \in \{1, \ldots, C\}} \left(x_i^\top \mathbf{w}^{(c)} + b^{(c)} \right).$$

While the QPs above use a linear hyperplane, SVMs can also be used to find non-linear separators, for example, a hyperplane of polynomial degree $d \in \mathbb{N}^+$. This can be done using a kernel $\kappa(x_i, x_j) = \langle \psi(x_i), \psi(x_j) \rangle$, where $\psi(x_i)$ maps the original input x_i into a feature space of another dimension where the data may be more easily separable by a linear hyperplane. In the QPs above, x_i is simply replaced by $\psi(x_i)$. Therefore, the linear kernel would be $\kappa(x_i, x_j) = x_i^\top x_j$. The kernel can be modified using the kernel coefficient γ and a coefficient term $p \in \mathbb{R}$.
This thesis uses the following kernels:

- Radial Basis Function (rbf) kernel: $\kappa(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$.
- Polynomial kernel: $\kappa(x_i, x_j) = (\gamma x_i^T x_j + p)^d$.
- Sigmoid kernel: $\kappa(x_i, x_j) = \tanh(\gamma x_i^T x_j + p)$, with $\tanh = \dfrac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$.

It should be noted that various computational tricks exist that involve rewriting the QP to make it easier to solve. However, they are beyond the scope of this thesis, since the SVM is purely for validation.

3.2 Data splitting

Now that the model methodology has been discussed, we move to the methodology regarding the data. First, we describe how the data is split into training, validation, and test sets. Then, we discuss how imbalanced data is handled.

3.2.1 Train, validate, and test split

The data will be split into a train, validation and test set. The training data will be used to train the models, the validation data will be used to select a model and its hyperparameters. The validation set helps to check whether the model is under- or overfitting. The test (also called hold out) set will be used to determine model performance on data not used in training or validation. This gives a realistic evaluation of how well the model performs in practice.

We choose to not randomly sample to avoid the model already knowing that a certain period is a flu season period for example and therefore using that in the test data. This way, the model is similar to

[4] This can be done using the mapping $y_i^{\text{new}} = 2y_i^{\text{old}} - 1$, for binary classification.

how it would be used in the real world. If we were to randomly sample, test results might indicate strong performance, while the model's real-world performance could be worse, as discussed by (Tashman, 2000). This decision was based on an internal discussion at Crowe Foederer. The following example may explain it more clearly:

Example 3.2.1 As stated before, sick leave absence is seasonal, but the intensity of absenteeism differs from year to year, and the peak can also shift (see Chapter 2.2). Suppose absence within a given year follows the blue curve shown in Figure 3.16 during a year, and month 7 be the month we aim to predict.

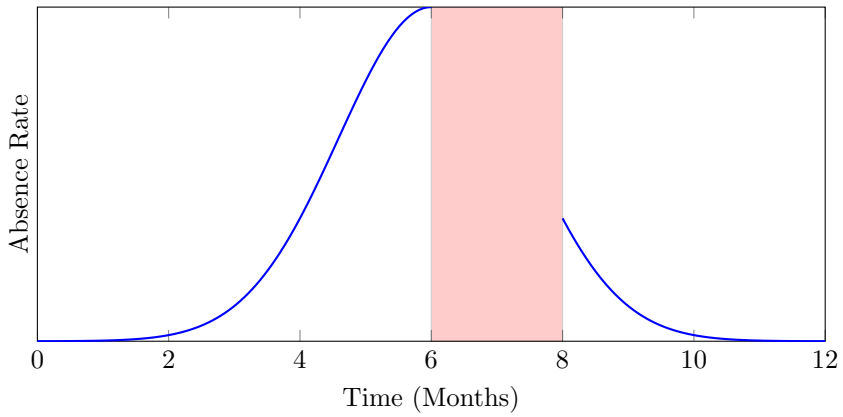

Figure 3.16: Absence peaks during flu season make random sampling across months misleading.

Month 8 provides information which would not be available in the real world. This makes the prediction easier since it can be relatively certain that the absence of month 7 will be somewhere between that of month 6 and month 8, allowing the model to simply interpolate. ▲

Randomly sampling amongst observations goes one step further, as it results in already giving a model close to the exact distribution of absence. To address this, we use a validation and testing method that guarantees the model trains only on data before the validation and test period. One possible approach is called the sliding (or rolling) window method, which is also used by (Zupančič & Panov, 2024) to predict absenteeism.

Sliding window method

In the sliding window method, which is for example explained in (Zhan & Kim, 2024), a fixed training duration is selected. This window will be shifted (or slid) forward with a chosen time step. Each time the window shifts a new model will be fitted on the training data within the window. Each fitted model can therefore be different. However, for each fitted model the same hyperparameters are used.

The validation period for each window, must come (immediately or with a delay) after the training period and will be shifted using the same chosen time step. As (Zupančič & Panov, 2024) explains, there can be one or multiple validation periods. While choosing a model and optimising the hyperparameters, the model is selected with the best average validation performance over all the possible windows.

The sliding window will now be defined in a more mathematical way:

- let $t \in \{\mathbb{Z}_+\}$ be the time period index (months in our case),
- let N be the number of periods we want to use for training,
- let V be the number of periods we want to use for validation,
- let M be the number of windows we want to use.

So the first training set contains periods $t = 1$ to $t = N$. The validation period can consist of one or more periods and is after the training period, so the first validation set would consist of periods $t = N + 1$ to $t = N + V$. For the next training and validation set, we will shift one period forward (as can be seen in figure 3.17) to obtain $t = 2$ to $t = N + 1$ for training, and $t = N + 2$ to $t = N + V + 1$ for validation. This process is repeated until there are M of these windows, as is illustrated in 3.17. Thus, for each window $m = 1, 2, \ldots, M$, the training and validation periods are defined the following way:

$$\text{Training period}(m): \quad t = \{m, ..., m + N - 1\},$$

$$\text{Validation period}(m): \quad t = \{m + N, ..., m + N + V - 1\}.$$

Test (also called hold-out) samples are also necessary to check the final model performance on unseen data. So we do not want to validate on a month that we will also test on. They must not overlap; otherwise, there has already been optimised over a test month due to it being used in validation, as (Raykar & Saha, 2015) shows. Unfortunately, there were only 2 years of post-corona data available (accounting for the variable absence last year not being affected by corona). Therefore, after an internal discussion at Crowe Foederer, the choice was made to alternate between validation and test months. The main reason for this is to give them equal weight. (Raykar & Saha, 2015) also suggests to give the validation and test set equal weight. Below is a visualisation.

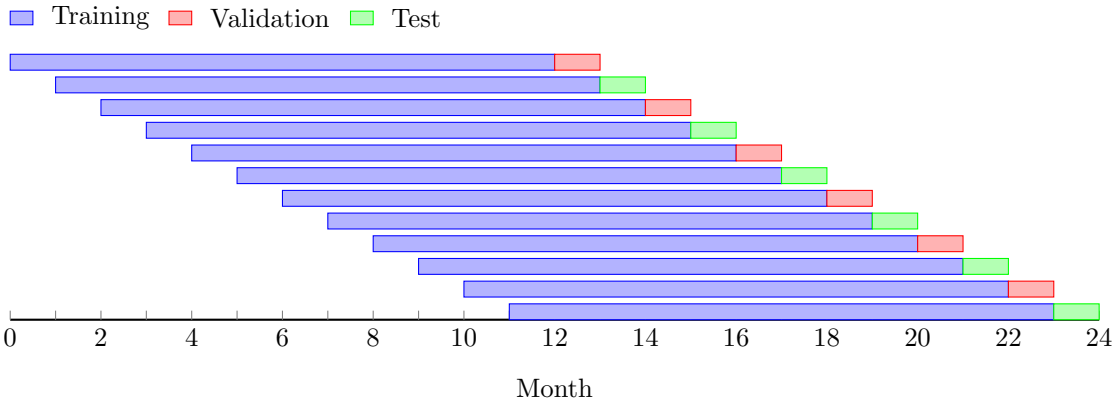

Figure 3.17: The rolling window method, with $N = 12$, $V = 1$, $M = 12$: each row shows a 12-month training set followed by a validation or test month.

3.2.2 (Im)balanced data

In the created dataset, there is an imbalance. Approximately 13.12% of all rows in the dataset have absence current month greater than 0. The remaining rows all have absence current month equal to 0. This means that if we were to do binary classification, always predicting 0 would result in an accuracy of 86.88%. But we want our model to be able to classify when someone is absent and not rely on the case that a lot of times someone is not absent. For the test and validation sets, however, we keep the original imbalance to reflect a real-world scenario. Thus, only the training data is balanced. This decision is based on an internal decision at Crowe Foederer.

Assigning different weights to certain classes can solve this problem of imbalance, as (Bakırarar & Elhan, 2023) shows. (Chawla et al., 2002) introduced SMOTE (Synthetic Minority Over-sampling Technique), which, as the name suggests, oversamples the minority classes. In (Chawla et al., 2002), using SMOTE gave better results than oversampling the minority class, and a combination of SMOTE and undersampling performed better than just undersampling. Their explanation of why SMOTE returned better results than oversampling the minority class was that using SMOTE returned bigger decision regions, which could generalise better, since there were more different minority samples. (Anand et al., 2010) discusses that both oversampling and undersampling have advantages and disadvantages. For undersampling, a disadvantage is the loss of data which can be learned from. A disadvantage of oversampling is that, often, synthetic data is generated, which can introduce potentially new errors due to the nature of synthetic data. Intuitively, oversampling (without generating synthetic data) and using weights should yield the same result, oversampling introduces only a random element. After an internal discussion at Crowe Foederer, the choice was made to use weights, since the random element of SMOTE does some things in the background, making it hard to see what actually happens.

These weights can be chosen in different ways as (Bakırarar & Elhan, 2023) describes. After an internal discussion at Crowe Foederer, the choice was made to choose the weights inversely to their frequency. This way each class is treated equally. This is also one of the methods described in (Bakırarar & Elhan, 2023). So we have the following formula:

$$w_c = \frac{100 \cdot n}{\sum_{i=1}^{n} \mathbb{I}_{y_i=c}}, \text{ where } \mathbb{I}_{y_i=c} \text{ is the indicator function equal to 1 if } y_i = c \text{ and 0 else.} \tag{3.11}$$

Chapter 4

Game theory methodology

In this chapter, the game-theoretic concepts which are used in this thesis are explained. More details of some of the game-theoretic aspects of this chapter (such as transferable utility games) can be found in (Quant et al., 2003). After the game-theoretic concepts have been discussed, with examples, the solutions of these examples will be compared. Finally, this chapter will talk about how game theory can be used for feature attribution. Further on in this thesis game-theoretical concepts will be used to study feature attribution in Chapter 7.2.

Before going into game-theoretic concepts, we first need to define a transferable utility game (or TU game in short). In a TU game, a value needs to be allocated amongst all $N = \{1, .., n\}$ players.

Definition 4.0.1 A transferable utility game is defined by player set $N = \{1, ..., n\}$ and value function $v : 2^N \to \mathbb{R}$, which assigns a value to each possible coalition $S \subseteq N$ (a subset of all the players)[5]. Such that each coalition has a value $v(S)$. ▲

The value of the coalition N, which is the coalition of all the players, needs to be allocated amongst all the players. The values of the other coalitions can be used to determine this allocation. Now that a TU game has been defined, we will show an example of a TU game.

Example 4.0.1 Table 4.1 shows an example of a transferable utility game with 3 players.

S	$\emptyset$	{1}	{2}	{3}	{1,2}	{1,3}	{2,3}	N
$v(S)$	0	0	10	0	20	10	30	40

Table 4.1: A game-theoretic example ▲

Definition 4.0.2 Let a be a function that maps a transferable utility game $v \in TU^N$ to a vector $(a_1, \ldots, a_n) \in \mathbb{R}^N$ where a_i is the allocation that player $i \in N$ receives.[6] TU^N is the domain of transferable utility games with player set N. Thus, we obtain the following mapping: $a : TU^N \to \mathbb{R}^N$. ▲

Solution method a determines the allocation of $v(N)$ for any transferable utility game v. There are different allocations possible; for example, everything can be divided equally among the players, so

$$a_i(v) = \frac{v(N)}{n}.$$

In this thesis, we focus on four specific solution concepts: the Shapley value, the Banzhaf value, the nucleolus and a newly defined solution concept.

4.1 Properties in transferable utility games

Now we will discuss some properties which a transferable utility game can satisfy:

- Monotonicity property: As the coalition size increases, the coalitional value should also increase,

$$v(S) \leq v(T) \iff S \subseteq T. \tag{4.1}$$

[5] There are 2^N coalitions possible.

[6] The mathematically correct way would be to write $\mathbb{R}^{|N|}$, but the use of $\mathbb{R}^N$ is a more common notation.

- Additivity property: The coalitional value of the union of two disjoint sets is equivalent to the coalitional value of those two sets added up,

$$v(S \cup T) = v(S) + v(T) \quad \forall \text{ disjoint } S, T. \tag{4.2}$$

- Superadditivity property: The coalitional value of the union of the two disjoint sets is greater or equal than the sum of the coalitional values of the disjoint sets,

$$v(S \cup T) \geq v(S) + v(T) \quad \forall \text{ disjoint } S, T. \tag{4.3}$$

The games obtained to apply feature attribution in Chapter 7 do not satisfy monotonicity and (super)additivity (Equations 4.1, 4.2 and 4.3). Therefore, certain game-theoretic solutions are eliminated, such as the core (Gillies, 1953).

Now that we have discussed some of the properties games can satisfy we move on to the properties game-theoretic solution concepts $a : TU^N \to \mathbb{R}^N$ can satisfy. Each property can be satisfied by a game-theoretic concept, or not. It should be noted that we will only use the properties relevant to this thesis, and there are also other properties. The properties are used to evaluate and compare game-theoretic solution concepts. We consider the following desirable properties that are commonly used in transferable utility games:

- Efficiency property: The value of the grand coalition is divided among all players,

$$\sum_{i \in N} a_i(v) = v(N). \tag{4.4}$$

  This ensures that the full value of the grand coalition is allocated.
- Individual rationality property: Each player should receive at least what they would get on their own,

$$a_i(v) \geq v(i) \quad \forall i \in N. \tag{4.5}$$

  This ensures that joining a coalition is beneficial.
- Additivity property: If two games are added, and the solution concept is calculated, it is the same as calculating the solution concept for those games and adding up the values of the solution concepts,

$$a(v + w) = a(v) + a(w) \quad \forall \text{ games } v, w. \tag{4.6}$$

  This allows consistent results when combining games.
- Symmetry property: If two players are identical (so switching those players yields the same coalitional value for every coalition), then they get the same in the final allocation,

$$a_i(v) = a_j(v) \iff v(S \cup \{i\}) = v(S \cup \{j\}) \quad \forall S \subseteq N \setminus \{i, j\}. \tag{4.7}$$

  This ensures that players who contribute equally receive the same payoffs.
- Dummy property: If a player contributes no additional value to any coalition, they receive zero in the final allocation,

$$v(S \cup \{i\}) = v(S) \quad \forall S \subseteq N \setminus \{i\} \implies a_i(v) = 0. \tag{4.8}$$

  This avoids assigning value to players who do not contribute to any coalition.
- Homogeneity property: Multiplying a game by a scalar results in the same solution as multiplying the solution by the same scalar,

$$a(\alpha v) = \alpha a(v) \quad \forall \ \alpha \in \mathbb{R}. \tag{4.9}$$

  This keeps the solution proportional when the game is scaled.

The imputation set is the set for which both Equations 4.4 and 4.5 hold. If no solution exists which satisfies both equations, then the imputation set is empty.

This thesis discusses four game-theoretic concepts: the Shapley value (Shapley, 1952), the Banzhaf value (Banzhaf, 1965), the nucleolus (Schmeidler, 1969), and the newly introduced ACS concept (see Chapter 4.3). All these game-theoretic concepts satisfy some of the properties discussed above, which are summarised in Table 4.2 below. The satisfied properties are discussed in the original papers of each

concept, with some exceptions. For the nucleolus, the properties are also partially discussed in (Iñarra et al., 2020), and for the Banzhaf value, they are discussed in (Haimanko, 2019). Finally, proofs for the properties of the ACS concept are given in Chapter 4.3.

Property	**Shapley Value**	**Banzhaf Value**	**Nucleolus**	**ACS Concept**
Efficiency	Yes	No	Yes	No
Individual rationality	No	No	Yes*	No
Additivity	Yes	Yes	No	Yes
Symmetry	Yes	Yes	Yes	Yes
Dummy Property	Yes	Yes	Yes	No
Homogeneity	Yes	Yes	Yes	Yes

*The nucleolus satisfies efficiency only when the sum of the singleton coalition values does not exceed the value of the grand coalition, i.e., $\sum_{i \in N} v(\{i\}) \leq v(N)$.

Table 4.2: Satisfied properties of each game-theoretic concept

4.2 Solution concepts

Now that TU games and the relevant properties of games and solution concepts have been introduced, we will discuss specific solution concepts.

4.2.1 Shapley value

In (Shapley, 1952), the Shapley value is introduced. The Shapley value is defined as the average of a player's marginal contributions across all permutations of players.

Definition 4.2.1 Given a game $v \in TU^N$, the Shapley value for player i is defined as:

$$\Phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \cdot [v(S \cup \{i\}) - v(S)].$$

$|N|!$ denotes the total number of permutations. $|S|!(|N|-|S|-1)!$ counts all permutations of players where the first $|S|$ elements are the players in S (in any order), followed by player i, and then the remaining $|N|-|S|-1$ elements are the players in $N \setminus (S \cup \{i\})$ (again in any order). Thus, the fraction in Definition 4.2.1 gives the probability of player i joining a coalition immediately after the players in S have joined. ▲

Example 4.2.1 Here is how the Shapley value would be calculated for Example 4.0.1:

$$\begin{aligned}
\Phi_1(v) &= \sum_{S \subseteq \{2,3\}} \frac{|S|!(2 - |S|)!}{3!} \cdot [v(S \cup \{1\}) - v(S)] \\
&= \frac{0! \cdot 2!}{6} \cdot (v(\{1\}) - v(\emptyset)) + \frac{1! \cdot 1!}{6} \cdot (v(\{1,2\}) - v(\{2\})) \\
&\quad + \frac{1! \cdot 1!}{6} \cdot (v(\{1,3\}) - v(\{3\})) + \frac{2! \cdot 0!}{6} \cdot (v(\{1,2,3\}) - v(\{2,3\})) \\
&= \frac{1}{3} \cdot 0 + \frac{1}{6} \cdot 10 + \frac{1}{6} \cdot 10 + \frac{1}{3} \cdot 10 \\
&= 0 + \frac{10}{6} + \frac{10}{6} + \frac{10}{3} \\
&= \frac{10 + 10 + 20}{6} = \frac{40}{6} = 6\frac{2}{3}.
\end{aligned}$$

Similar calculations are performed for player 2 and 3 to obtain $\Phi(v) = (6\frac{2}{3}, 21\frac{2}{3}, 11\frac{2}{3})$ as the Shapley allocation vector. ▲

Although in Table 4.2 it is stated that the Shapley value is efficient with respect to $v(N)$, this is only the case if $v(\emptyset) = 0$, which is often assumed. For our games in Chapter 7, this is not the case. If $v(\emptyset) \neq 0$, then the Shapley value is efficient with respect to $v(N) - v(\emptyset)$. While analysing the results in Chapter 7, it became clear that $v(N) - v(\emptyset) < 0$ is possible. This leads in our case, once, to all negative contributions, which is unusual. However, theoretically, it is possible. The following example will illustrate this.

Example 4.2.2 Consider the following 2-player TU game:

S	$\emptyset$	$\{1\}$	$\{2\}$	N
$v(S)$	3	2	1	0

Table 4.3: TU game example to illustrate efficiency condition Shapley

The TU game above results in the following Shapley values $\Phi(v) = (-1, -2)$, which are both negative and sum up to $v(N) - v(\emptyset)$. ▲

4.2.2 Banzhaf value

The Banzhaf value is similar to the Shapley value in the way that it measures the marginal contribution. The Banzhaf value originates from the Banzhaf power index, introduced by (Banzhaf, 1965). The Banzhaf power index is for simple games with a yes or no outcome (values 1 or 0 for every coalition); this is not the type of game we will investigate. Luckily, the Banzhaf value is applicable to any TU game. Where the Shapley value calculates the average marginal contribution over all permutations of players, the Banzhaf value calculates the marginal contribution averaged over the number of coalitions that exclude player i ($2^{|N|-1}$), which is what we sum over. This results in the following definition, in which we follow the notation of (Haimanko, 2019).

Definition 4.2.2 Given a game $v \in TU^N$, the Banzhaf value for player i is:

$$\beta_i(v) = \frac{1}{2^{|N|-1}} \sum_{S \subseteq N \setminus \{i\}} \left[v(S \cup \{i\}) - v(S) \right]. \qquad \blacktriangle$$

Example 4.2.3 We will now show how the Banzhaf value would be calculated given the TU game of Example 4.0.1.

$$\begin{aligned}
\beta_1(v) &= \frac{1}{2^{3-1}} \sum_{S \subseteq \{2,3\}} \big(v(S \cup \{1\}) - v(S) \big) \\
&= \frac{1}{4} \Big[(v(\{1\}) - v(\emptyset)) + (v(\{1,2\}) - v(\{2\})) \\
&\quad + (v(\{1,3\}) - v(\{3\})) + (v(\{1,2,3\}) - v(\{2,3\})) \Big] \\
&= \frac{1}{4}(0 + 10 + 10 + 10) = 7\frac{1}{2}
\end{aligned}$$

Similar calculations are performed for $\beta_2(v)$ and $\beta_3(v)$ to obtain the following solution vector $\beta(v) = (7\frac{1}{2}, 22\frac{1}{2}, 12\frac{1}{2})$. ▲

In short, the Banzhaf value is the marginal contribution, averaged over all coalitions that do not contain a player.

4.2.3 Nucleolus

The nucleolus was introduced by (Schmeidler, 1969). The goal of the nucleolus is to minimise dissatisfaction between coalitions. To explain this further, the excess of a coalition $E(S, a(v))$, for a given solution concept $a(v)$ and coalitions $S \subseteq N$, is first defined (using similar notation as (Borm, 2024)).

Definition 4.2.3 Given a game $v \in TU^N$, coalition $S \subseteq N$ and proposed solution $a(v)$, let the excess $E(S, a(v))$ be defined the following way:

$$E(S, a(v)) = v(S) - \sum_{i \in S} a_i(v). \qquad \blacktriangle$$

This excess is minimised by finding the lexicographical minimum. To do this, all excesses are ordered in non-increasing order, resulting in a vector $\theta(a(v)) \in \mathbb{R}^{2^{|N|}}$. First, the biggest excess of $\theta(a(v))$ is minimised, then the second biggest and so on until a unique solution is obtained (if it exists). The excesses of the empty coalition and grand coalition do not change, so they do not need to be taken into account. This will be illustrated in the example below, which shows how excesses are calculated and compared for different proposed allocations.

Example 4.2.4 In this example, the calculation of excesses and how they are compared will be shown. We still have the same transferable utility game from Example 4.0.1. 2 possible solutions are proposed: $a^1(v) = (10, 20, 10)$ and $a^2(v) = (7, 22, 11)$.

S	$\emptyset$	{1}	{2}	{3}	{1,2}	{1,3}	{2,3}	N
$v(S)$	0	0	10	0	20	10	30	40
$E(S, a^1(v))$	0	-10	-10	-10	-10	-10	0	0
$E(S, a^2(v))$	0	-7	-12	-11	-9	-8	-3	0

Table 4.4: Excess calculation example

We obtain the following excess vector in decreasing order: $\theta(a^1(v)) = (0, 0, 0, -10, -10, -10, -10, -10)$ and $\theta(a^2(v)) = (0, 0, -3, -7, -8, -9, -11, -12)$. $\theta(a^2(v))$ is lexicographically smaller than $\theta(a^1(v))$, because the the first and second entries are identical and the third entry has $-3 < 0$. ▲

As discussed before, if the imputation set is non-empty, then the nucleolus satisfies individual rationality, which is the case in Example 4.0.1 (see Equation 4.5).

The example above illustrates that the excess of the empty coalition is always equal to the value of the empty coalition, since

$$E(\emptyset, a(v)) = v(\emptyset) - \sum_{i \in \emptyset} a_i(v) = v(\emptyset) - 0 = v(\emptyset).$$

Due to the efficiency, the excess of the grand coalition is always 0, since

$$E(N, a(v)) = v(N) - \sum_{i \in N} a_i(v) = v(N) - v(N) = 0.$$

Thus, both the empty and grand coalition are not taken into account while minimising the excesses lexicographically.

To find the nucleolus a series of linear programs (LPs) can be solved. Each of the linear programs is used to find the maximum excess and minimises it. Once the maximum excess is found and minimised, it is fixed, the next LP then minimises the second largest excess. After the second largest excess is found, minimised and fixed the next LP goes to the next biggest excess. This iterative process is repeated until the nucleolus is found. It should be emphasised that at each iteration the found excess is fixed.

We will now give the LP formulation at iteration k, similar to (Guajardo & Jörnsten, 2014).

Definition 4.2.4 Given a game $v \in TU^N$, let ϵ_k be the k-th largest excess, which is calculated (and thus the k-th highest excess). Let T_j for $j = 1, ..., k-1$ be the set which contains the equations and excesses $\epsilon_j^\star$ that are fixed in iteration $j = 1, \ldots, k-1$. So let

$$T_j = \left\{ S \subset N, S \neq \emptyset \;\middle|\; v(S) - \sum_{i \in S} a_i(v) = \epsilon_j^\star \right\} \text{ for } j = 1, ..., k.$$

Then we obtain the following LP at iteration k.

$$
\begin{aligned}
\max_{\epsilon_k,\, a(v)} \quad & \epsilon_k \\
\text{s.t.} \quad & \sum_{i \in N} a_i(v) = v(N) \quad \text{(efficiency)} \\
& v(S) - \sum_{i \in S} a_i(v) \geq \epsilon_k \quad \forall S \subseteq N, \quad S \notin \bigcup_{j=1}^{k-1} T_j \\
& v(S) - \sum_{i \in S} a_i(v) = \epsilon_j^\star \quad \forall S \in T_j, \quad j = 1, \ldots, k-1 \\
& \epsilon_k \in \mathbb{R}, a_i \in \mathbb{R}
\end{aligned}
$$
▲

This primal is used to find the maximal excess, it could be that multiple equations are equal to the excess, depending on a solution $a(v)$. The dual is used to identify which coalition constraints in the primal should be fixed.

We will now provide the dual at iteration 1, similar to (Guajardo & Jörnsten, 2014), in which our goal is to find a weight vector $b \in \mathbb{R}^{2^{|N|}-1}$, that maximises the objective value. b_S is the weight for coalition S, and $K \subseteq 2^N \backslash \{\emptyset\}$ is the set which contains all non-empty subsets of N.

Definition 4.2.5 Let $b \in \mathbb{R}^{2^{|N|}-1}$, then given Definition 4.2.4 the dual at iteration $k = 1$ is defined as follows.

$$
\begin{aligned}
\max_{b} \quad & \sum_{S \in K} b_S \cdot v(S) \\
\text{s.t.} \quad & \sum_{S \in K,\, i \in S} b_S = 0 \quad \forall i \in N \\
& \sum_{S \in K \backslash N} b_S = 1 \\
& b_S \in \mathbb{R}^+ \quad \forall S \in K \setminus N, \quad b_N \in \mathbb{R}
\end{aligned}
$$
▲

We will fix the equations of the coalitions in the primal, of which the dual has positive b_S as is also explained by (Guajardo & Jörnsten, 2014). This follows from the fact that when the optimal values of the primal and dual are equivalent, then the positive dual variables correspond to the binding constraints in the primal (Hillier & Lieberman, 2010). (Staudacher et al., 2019) is an R package which does these calculations of the nucleolus automatically (see (Staudacher & Anwander, 2019) for the documentation). We will use their package throughout this thesis. After contacting the creators of the package, it became clear that the calculations done by the package also iteratively solve the series of primal LPs with their corresponding duals as discussed above.

Example 4.2.5 Now, the calculation of the nucleolus for Example 4.0.1 will be shown. If we fill in the LP from Definition 4.2.4 with Example 4.0.1, we obtain the following:

$$
\begin{aligned}
\max_{\epsilon_1, a_1(v), a_2(v), a_3(v)} \quad & \epsilon_1, \\
\text{s.t.} \quad & a_1(v) + a_2(v) + a_3(v) = 40, \\
& -a_1(v) \geq \epsilon_1, \\
& 10 - a_2(v) \geq \epsilon_1, \\
& -a_3(v) \geq \epsilon_1, \\
& 20 - (a_1(v) + a_2(v)) \geq \epsilon_1, \\
& 10 - (a_1(v) + a_3(v)) \geq \epsilon_1, \\
& 30 - (a_2(v) + a_3(v)) \geq \epsilon_1.
\end{aligned}
$$

Solving the equations above yields $\epsilon_1 = -5$, there are multiple allocations $a(v)$ possible that result in this maximum epsilon. By solving the dual (Definition 4.2.5) we identify the coalitions with positive

dual weights, which correspond to the equations that need to be fixed. In this example, we find that coalitions $\{1\}$ and $\{2,3\}$ with corresponding weights $b_{\{1\}} = \frac{1}{2}, b_{\{2,3\}} = \frac{1}{2}$ need to be fixed. Together with $b_{\{1,2,3\}} = -1$, this reproduces the primal objective value of -5. Thus, by fixing the equations of coalitions $\{1\}$ and $\{2,3\}$ we obtain $a_1(v) = 5$ and $a_2(v) + a_3(v) = 35$, which ensures that the efficiency constraint is satisfied automatically. Note that the excesses for the empty set and grand coalition are always 0 by definition. Now we move on to the next LP, where we fix $a_1(v) = 5$ and $a_2(v) + a_3(v) = 35$:

$$\begin{aligned} \max_{\epsilon_2, a_1(v), a_2(v), a_3(v)} \quad & \epsilon_2, \\ \text{s.t.} \quad & a_2(v) + a_3(v) = 35, \\ & a_1(v) = 5, \\ & 10 - a_2(v) \geq \epsilon_2, \\ & -a_3(v) \geq \epsilon_2, \\ & 20 - (a_1(v) + a_2(v)) \geq \epsilon_2, \\ & 10 - (a_1(v) + a_3(v)) \geq \epsilon_2. \end{aligned}$$

Continuing the series of LPs leads to the solution $a(v) = (5, 22\frac{1}{2}, 12\frac{1}{2})$ (this solution can be verified by the Kohlnberg criterion (Kohlberg, 1971), but this is out of scope for this thesis). Observe that this solution satisfies individual rationality, meaning that the imputation set for the game in Example 4.0.1 is non-empty. If the imputation set of Example 4.0.1 had been empty, then the obtained solution would not have satisfied individual rationality. ▲

If the imputation set is non-empty, then the nucleolus lies inside the imputation set. If the individual rationality constraint conflicts with the efficiency constraint, then (as mentioned earlier) the imputation set is empty, and the solution will not satisfy individual rationality. This is the case for the results of the XGBoost model explained in Chapter 7.

4.3 Averaged Coalitional Surplus (ACS)

Although we discussed the Shapley value, Banzhaf value and nucleolus, many other concepts are not applicable for feature attribution, because the imputation set is empty given the coalition values, $v(S)$, of the XGBoost model explained in Chapter 7. (Some of the concepts will be discussed later in this section, with a further explanation of why they are not applicable.) Therefore, we propose our own game-theoretic concept designed for feature attribution that can handle an empty imputation set. The idea is to check for each player what all their coalitional values are, and how far they are from the value of the grand coalition. Since we want this concept for feature attribution, we are interested in how far the coalitional values of a feature are from the value of the grand coalition, rather than dividing the value of the grand coalition amongst all features, which is usually done in TU games. This way, we can see how much a feature's coalitional values align with the prediction of the full model or not, as will be discussed later in this section. If all coalitional values of a player are close to the value of the grand coalition, it indicates that this player is more important because they behave similarly to the set of all features together. We can also check whether these coalitional values are higher or lower than the value of the grand coalition. Looking at every single coalitional value that contains a specific player i gives a large number of data points. To make this information interpretable, we want to summarise it such that we get a value for each player, which states how far on average a player is from the value of the grand coalition. To understand how to summarise this information, it is important to know what a coalitional value depends on (especially in the context of feature attribution).

The value of a coalition in feature attribution depends on which players are involved and how many players are in the coalition, since larger coalitions mean that more information is available (from a feature attribution perspective, where players are variables). Each coalition size can be seen as a different level of information available. We want to take into account that the number of coalitions containing player i and of size k differs for each coalition size $k = 1, ..., |N|$ (this is visualised in Figure 4.1 below).

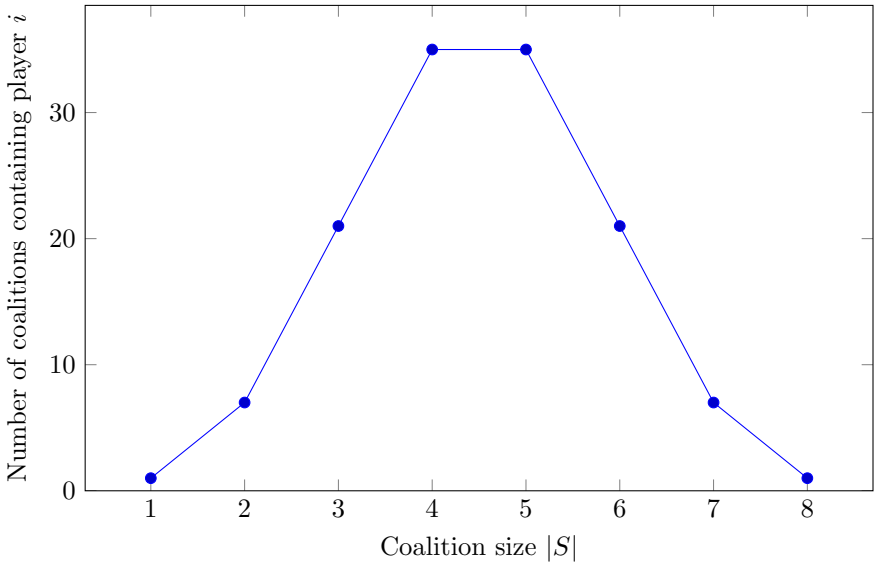

Figure 4.1: A graph of how many coalitions there are for every size of coalitions for $|N| = 8$ players.

To get a balanced overview of a player's contribution, we treat each coalition size equally. This avoids overrepresenting coalition sizes that occur more frequently (due to the binomial distribution of coalition counts), ensuring that each level of information (coalition size) contributes equally. This is important for feature attribution because we do not want a feature attribution method to be biased towards a certain information level.

What we can therefore do is, for each player, for each coalition size $k = |S|$, take the average over $v(S)$:

$$w_i^k(v) = \sum_{i \in S, |S|=k, S \subset N} v(S) \cdot \binom{|N|-1}{k-1}^{-1} \text{ for } k = 1, \ldots, |N|.$$

This results in $|N|$ points $w_i^1(v), \ldots, w_i^{|N|}(v)$ for each player i. Since the value of the grand coalition is the same for everyone, we exclude it from our calculation to better highlight what an individual player can obtain on their own, making it easier to compare to the value of the grand coalition later on. This leaves $|N| - 1$ averages per player. To reduce these $|N| - 1$ points to a single value for each player, we again take the average. So we obtain:

$$w_i(v) = \frac{1}{|N|-1} \sum_{k=1}^{|N|-1} \sum_{i \in S, |S|=k, S \subset N} v(S) \cdot \binom{|N|-1}{k-1}^{-1}.$$

Finally, to see how each feature behaves compared to the value of the grand coalition, we subtract $v(N)$ from the value we obtained. This results in the following definition.

Definition 4.3.1 Given a game $v \in TU^N$, let the ACS solution concept for player i be defined the following way:

$$ACS_i(v) = \left(\frac{1}{|N|-1} \sum_{k=1}^{|N|-1} \sum_{i \in S, |S|=k, S \subset N} v(S) \cdot \binom{|N|-1}{k-1}^{-1} \right) - v(N) \text{ for } i = 1, ..., n. \quad \blacktriangle$$

This lets us see, on average, how far a coalitional value containing player i tends to be off the value of the grand coalition $v(N)$ and what direction this deviation is (positive or negative). A positive value means that a player on average scores higher than the grand coalitional value. A value close to 0 means that the difference between the value of the grand coalition is close to the average coalitional value of player i. In the case of feature attribution, a closer value to 0 means that a feature is more important, since its coalitional values on average lie close to the value of the grand coalition.

This solution is not efficient (so the values of the solution do not add up to the value of the grand coalition), thus it would not be great in a classical TU game setting, in which the goal is to allocate the value of the grand coalition amongst all the players. It does explain however, how far and in which direction the coalitional value of each player tends to be from the value of the grand coalition. This is perfect for feature attribution. Methods such as SHAP (Lundberg, 2018), (Lundberg & Lee, 2017) and LIME (Ribeiro et al., 2016) are well-known feature attribution methods which are also able to interpret directionality. The interpretation of the direction for the ACS concept is the following: If ACS allocation for a player is positive, it means that a player has coalitional values (on average) above the value of the grand coalition. In the case of feature attribution, this intuitively means that a feature tends to overpredict the desired value. The reverse holds for a negative ACS value for a player. While direction shows whether a feature tends to have higher or lower coalitional values than the grand coalition, the magnitude shows whether a feature tends to predict coalitional values similar to the value of the grand coalition or not. A small absolute value means that the coalitional values of a player (on average) lie close to the value of the grand coalition, and thus it aligns more with the value of the grand coalition. This means that such a player supports the overall outcome (the value of the grand coalition). For this concept, such a player is considered more important than a player with a higher absolute value whose coalitional values, on average, lie further from the value of the grand coalition. We can order the variables from the lowest absolute value (which means the lowest absolute difference compared to the grand coalition) up to the highest absolute value (which means the highest absolute difference compared to the value of the grand coalition). This way, the variables are ordered from most important to least important. The combination of the interpretation behind the direction and magnitude makes the concept intuitive and easy to understand, even for people with less understanding of game theory.

Example 4.3.1 We will now work out the calculations of the ACS for Example 4.0.1 given Definition 4.3.1:

$$ACS_1(v) = \frac{1}{2}\left(\frac{0}{1} + \frac{1}{2}\cdot(20+10)\right) - 40 = -32\frac{1}{2},$$

$$ACS_2(v) = \frac{1}{2}\left(\frac{10}{1} + \frac{1}{2}\cdot(20+30)\right) - 40 = -22\frac{1}{2},$$

$$ACS_3(v) = \frac{1}{2}\left(\frac{0}{1} + \frac{1}{2}\cdot(10+30)\right) - 40 = -30.$$

Thus $ACS(v) = (-32\frac{1}{2}, -22\frac{1}{2}, -30)$ would be our solution, which does not satisfy efficiency. The coalitional values of player 1 are the furthest from the value of the grand coalition, and the coalitional values of player 2 are the closest to the value of the grand coalition. This example shows that each player has the same sign and thus all coalitional values, containing a fixed player, are on average below the value of the grand coalition. This can also be seen in the coalitional values of Example 4.0.1. ▲

The example above highlights that the ASC solution could entirely be negative (or positive). This means that on average the coalitional values of every player are below (or above) the value of the grand coalition.

Proof properties ACS concept

Now that we have seen an example of how the ACS concept is computed, we can demonstrate why each property discussed in Chapter 4.1 is (or is not) satisfied:

- Efficiency: Example 4.3.1 shows that the ACS concept does not satisfy efficiency. Since

$$ACS_1(v) + ACS_2(v)_+ ACS_3(v) = -32\frac{1}{2} - 22\frac{1}{2} - 30 = -85 \neq 40 = v(N).$$

- Individual rationality: Example 4.3.1 shows that the ACS concept also does not satisfy individual rationality. Since

$$ACS_1(v) = -32\frac{1}{2} < 0 = v(\{1\}),$$

$$ACS_2(v) = -22\frac{1}{2} < 10 = v(\{2\}),$$

$$ACS_3(v) = -30 < 0 = v(\{3\}).$$

- Additivity is satisfied by the ACS concept. For TU games v and w, we will show that $ACS_i(v+w) = ACS_i(v) + ACS_i(w) \quad \forall\, i = 1, ..., n$

$$\begin{aligned} ACS_i(v+w) &= \left(\frac{1}{|N|-1} \sum_{k=1}^{|N|-1} \sum_{i \in S,\, |S|=k,\, S \subset N} (v(S) + w(S)) \cdot \binom{|N|-1}{k-1}^{-1} \right) - (v(N) + w(N)) \\ &= \left[\left(\frac{1}{|N|-1} \sum_{k=1}^{|N|-1} \sum_{i \in S,\, |S|=k,\, S \subset N} v(S) \cdot \binom{|N|-1}{k-1}^{-1} \right) - v(N) \right] \\ &\quad + \left[\left(\frac{1}{|N|-1} \sum_{k=1}^{|N|-1} \sum_{i \in S,\, |S|=k,\, S \subset N} w(S) \cdot \binom{|N|-1}{k-1}^{-1} \right) - w(N) \right] \\ &= ACS_i(v) + ACS_i(w). \end{aligned}$$

  This holds due to the linearity of the sum.
- Symmetry is also satisfied, if player i and j all have the same coalitional values, they will clearly also get the same allocation in the ACS concept according to Definition 4.3.1.
- The dummy property is not satisfied by the ACS concept, for this let us consider the following 2-player TU game:

S	$\emptyset$	$\{1\}$	$\{2\}$	N
$v(S)$	0	0	1	1

Table 4.5: Example TU game to show ACS does not satisfy the dummy property

  This results in $ACS_1(v) = 0 + \frac{1}{2} - 1 = -\frac{1}{2} \neq 0$.
- Homogeneity is satisfied by the ACS concept. Let $\alpha \in \mathbb{R}$, then

$$\begin{aligned} ACS_i(\alpha v) &= \left(\frac{1}{|N|-1} \sum_{k=1}^{|N|-1} \sum_{i \in S,\, |S|=k,\, S \subset N} \alpha v(S) \cdot \binom{|N|-1}{k-1}^{-1} \right) - \alpha v(N) \\ &= \alpha \cdot \left[\left(\frac{1}{|N|-1} \sum_{k=1}^{|N|-1} \sum_{i \in S,\, |S|=k,\, S \subset N} v(S) \cdot \binom{|N|-1}{k-1}^{-1} \right) - v(N) \right] \\ &= \alpha \cdot ACS_i(v). \end{aligned}$$

  This holds again due to the linearity of the sum.

Now that we have proven what properties the ACS concept satisfies, as shown in Table 4.2, we can move on to comparing the ACS concept to other solution concepts and reflect on some of these properties. In Chapter 4.4.2, we will explain why certain properties are desirable (or not) for feature attribution.

The ACS concept compared to other solution concepts

Compared to other game-theoretic concepts, the ACS concept has some advantages that will now be discussed:

- Firstly, the ACS solution (from Definition 4.3.1) also exists for games with an empty imputation set and thus for games which have:

$$\sum_{i=1}^{n} v(\{i\}) > v(N). \tag{4.10}$$

  A lot of other game-theoretic concepts do not exist if Equation 4.10 holds. For example, solutions like Constrained Equal Awards, Constrained Equal losses (O'Neill, 1982), the Talmud principle (Aumann & Maschler, 1985) and the core (Gillies, 1953) are not defined if Equation 4.10 holds.

- In addition to the last point, in a TU game, in general, coalitional values do not exceed the value of the grand coalition. However, for feature attribution of a model, this can happen. The ACS concept is designed to be able to handle cases where coalitional values are greater than the value of the grand coalition. It does so by looking at the (weighted) average coalitional values for each player and comparing them to the value of the grand coalition. This way, ACS shows whether a player tends to have coalitional values higher or lower than the value of the grand coalition. Thus, if coalitional values of a player are mostly above the value of the grand coalition, the solution in the ACS concept will show this by having a positive value.
- Another advantage of the ACS concept is that it can deal with missing coalitional values by simply excluding them from the average (the weighting term needs to be adjusted in that case, by subtracting the amount of missing coalitions in the denominator inside the sum in Definition 4.3.1). In contrast, the methods shown in this thesis to calculate the Shapley value, the Banzhaf value, and the nucleolus (Definitions 4.2.1 - 4.2.5) require all coalitional values $v(S)$ to be known. This problem can be solved by imputing the missing coalitional values, as explained in (Černý, 2022). However, incorrect imputations may have a substantial impact on the resulting allocation.
- As discussed before, the ACS concept is simple and intuitive. This becomes clearer when compared to the Shapley or Banzhaf value, which are based on all possible permutations of players, or compared to the nucleolus which minimises the maximum excess.
- Finally, calculating the ACS concept is slightly easier than computing the Shapley and Banzhaf value, and much easier compared to the nucleolus, which requires solving a series of linear programs.

Now that some advantages have been discussed, we will show how the ACS concept differs from previously discussed game-theoretic concepts.

The ACS solution concept is the closest related to the Shapley and Banzhaf value, with which we will make a comparison. The main difference is that the Banzhaf value and Shapley value calculate an average of the marginal contribution, where the ACS concept calculates just an average of the coalitional values and subtracts the value of the grand coalition. The ACS solution looks at all the coalitional values which contain player i and therefore looks at the behaviour of player i across all coalitions. The Shapley and Banzhaf value look at what happens if player i joins a coalition compared to the coalition without player i. So the ACS concept captures what the average effect is of a player being present in a coalition, while the Banzhaf and Shapley value specifically focus on the average marginal contribution.

The ACS concept is relatively simple compared to marginal contribution concepts such as the Shapley value and the Banzhaf value, since it just takes the average 2 times over coalitional values and compares it to the value of the grand coalition. This makes it easier to interpret for people who do not understand game theory. The ACS concept also uses different coalition size weighting compared to the Shapley and Banzhaf value. It is clear that the weighting from the Banzhaf value is different from the weighting of the ACS concept[7]:

$$\frac{1}{2^{|N|-1}} \neq \frac{1}{|N|-1} \cdot \binom{|N|-1}{|S|-1}^{-1}.$$

The difference between the weighting of the Shapley value and the ACS may be less obvious:

$$\frac{|S|!(|N|-|S|-1)!}{|N|!} = \frac{|S|}{|N|} \cdot (|N|-|S|-1) \cdot \frac{(|S|-1)!(|N|-1-|S|-1)!}{(|N|-1)!}$$

$$= \frac{|S| \cdot (|N|-1-|S|)}{|N|} \cdot \binom{|N|-1}{|S|-1}^{-1} \neq \frac{1}{|N|-1} \cdot \binom{|N|-1}{|S|-1}^{-1}.$$

The Shapley value calculates the marginal contribution averaged over all possible permutations of players and the Banzhaf value calculates the marginal contribution averaged over all coalitions a player is in. The ACS concept treats each coalition size equally. In feature attribution, as the coalition size grows, more information becomes available. Thus, by treating each coalition size equally, the ACS concept captures how a feature contributes to a prediction across varying levels of information available. As explained earlier, this gives a balanced overview of a feature's influence on a prediction, because it prevents more occurring information levels (coalition sizes) from dominating the weighting.

[7]Note that $|S| = k$ in Definition 4.3.1.

There are also some disadvantages to the ACS concept. Firstly, as stated before, it is not efficient, so Equation 4.4 does not hold. This differs from the Shapley value and nucleolus, which do satisfy efficiency. If the goal was to distribute $v(N)$ among players, this would be a problem (which normalisation could solve). However, the ACS concept is constructed for feature attribution, in which efficiency is not necessarily required, since the order, magnitude and sign already provide enough information.

Secondly, the ACS concept does not satisfy the dummy property (Equation 4.8), unlike the Shapley value, the Banzhaf value and the nucleolus. This property is often considered desirable because it aligns with the intuition that players receive nothing if they contribute nothing. However, the ACS concept is designed to measure how far a player's coalitional values are on average from the value of the grand coalition. This means that players who contribute nothing will, on average, be further from the value of the grand coalition. Therefore, whether a player contributes less or more compared to another player is still preserved. Only the situation where a player contributes nothing cannot be seen in the ACS concept. In practice for feature attribution, it is unlikely that a feature (player) will contribute nothing. Therefore, this limitation is not problematic for feature attribution.

Overall, the ACS solution concept may be less practical in a classical game-theoretic setting. However, its simplicity and design make it a useful alternative for feature attribution.

4.3.1 Comparison of solutions of examples

Now we will compare all of the solutions of Examples 4.2.1, 4.2.3, 4.2.5 and 4.3.1. Recall that we have the following solutions for Example 4.0.1:

- Shapley $\Phi(v) = (6\frac{2}{3}, 21\frac{2}{3}, 11\frac{2}{3})$,
- Banzhaf $\beta(v) = (7\frac{1}{2}, 22\frac{1}{2}, 12\frac{1}{2})$,
- nucleolus $a(v) = (5, 22\frac{1}{2}, 12\frac{1}{2})$,
- ACS concept $ACS(v) = (-32\frac{1}{2}, -22\frac{1}{2}, -30)$.

For the Shapley value, Banzhaf value and nucleolus, we can see that player 2 receives the highest payoff, and player 1 the lowest. Ranking players from highest value (most important) to the lowest value (least important) results in the following player order: 2,3,1. We can see that these three solution concepts return relatively similar results. Firstly, note that

$$\beta(v) = \Phi(v) + (\frac{5}{6}, \frac{5}{6}, \frac{5}{6}).$$

Secondly, the Banzhaf value and nucleolus only differ in the first element by a value of 2, besides that, they are the same.

For the ACS concept, we also see that player 2 receives the highest payoff, while player 1 the lowest. However, recall that for the ACS concept, players are ranked differently. If we want to rank the players from most important to least important, we need to order them from lowest absolute value to highest absolute value. Using this method, the player ranking is the same: 2,3,1. Therefore, all the concepts rank the players the same in terms of player importance. The ACS concept tells us that, on average, the coalitional values of each player are below the value of the grand coalition. As discussed before, we can see that for Example 4.0.1 the Shapley value and nucleolus are efficient, and the Banzhaf value and ACS concept are not. In this example, the ACS concept is the only solution concept that produces a negative payoff vector.

4.4 Practical uses of game theory for feature attribution

Now that the game-theoretic concepts have been discussed, the next step is to explain how to use them for feature attribution. First, the coalitional values need to be obtained from the model and data.

4.4.1 From data to $v(S)$

The coalitional values are computed using an expectation, but first, some terminology needs to be introduced.

We are given a model f, training data and test data. We want to compute the coalitional values of the test data, given the model that has been trained on the training data. First, recall that data points are denoted $D_i = (x_i, y_i)$ for $i = 1, ..., n$, and we have train, validate and test data points. Thus D_{train} are all the points in the training dataset. For this dataset we will define an empirical distribution $\mathcal{D}_{\text{train}}$ from which we can sample observations:

$$(x_{\text{train}}, y_{\text{train}}) \sim \mathcal{D}_{\text{train}}.$$

Now for each coalition $S \subseteq N$, we can split the features based on whether they belong to coalition S or not:

$$x_i = [x_i^S, x_i^{N \setminus S}] \quad \forall\ i = 1, ..., n.$$

Thus each observation x_i consists of features which are in the coalition S and features which are not in the coalition S.

Now suppose we want to calculate the coalitional values for a test observation x_i, we condition on the known features in coalition S, i.e. x_i^S.

This leads to the following definition of a coalitional value $v_i(S)$ which is stated in (Lundberg et al., 2018).

Definition 4.4.1 Given a model f, training dataset $(x_{\text{train}}, y_{\text{train}})$ which is distributed according to empirical distribution $\mathcal{D}_{\text{train}}$, test observation x_i and coalition $S \subseteq N$, let coalitional value $v_i(S)$ be calculated the following way:

$$v_i(S) \approx \mathbb{E}_{x_{\text{train}} \sim \mathcal{D}_{\text{train}}}[f(x_{\text{train}}) \mid x_{\text{train}}^S = x_i^S]. \qquad \blacktriangle$$

The interpretation of Definition 4.4.1 is that for a test observation i, we fix the feature values in coalition S, x_i^S. For the unknown features not in S, we take the expectation over their possible values, weighted according to their distribution in the training data.

While there is a method to calculate the expectation from Definition 4.4.1, which requires reconstructing data points using all training observations for each test observation, this is computationally heavy. For this thesis, it was unfortunately too slow. The method is explained in Appendix D for those who are interested, but it is not relevant for the continuation of this thesis. Since XGBoost is used to predict absenteeism in this thesis, another algorithm can be used. (Lundberg et al., 2018) introduces an algorithm to calculate $v(S)$ for a tree-based learner, which makes use of the tree structure.

The idea is to compute this expectation efficiently by traversing the tree. At splits on features in S, the path is determined by the feature value of observation x_i. At splits on features in $N \setminus S$, the weighted average of both branches is taken using the proportion of training data going down each branch. This way, we can effectively estimate the expectation in Definition 4.4.1. In short, we fix the features we know, and take the expectation over those we do not know.

Before we give the algorithm, we will introduce its parameters in the table below.

Parameter	**Description**
v_j	Prediction value at node j (only relevant if j is a leaf).
d_j	Feature index used for the split at node j. So if a node splits on Overtime, and Overtime is the second feature, then $d_j = 2$. This is undefined at leaf nodes.
t_j	Threshold for splitting on feature d_j at node j. If the observation has $x_{d_j} \leq t_j$, we follow the left child; otherwise, we go right.
a_j	Index of the left child of node j.
b_j	Index of the right child of node j.
r_j	Proportion of training data reaching node j. These proportions are needed for when the path of an observation depends on the coalition.
w_j	Weight assigned to node j. This weight is needed to keep track of how much of the observation's path goes through node j during the algorithm.

Table 4.6: Parameters used in the tree-based coalitional value algorithm

Each of the parameters from Table 4.6 can be grouped into a vector over all nodes in the tree. Since there are J nodes in a tree, we define the vectors $v, d, t, a, b, r, w \in \mathbb{R}^J$ that are all indexed by subscript $j = 1, ..., J$. Note that these vectors can depend on observations. We will refer to the tree structure as (v, d, t, a, b, r). Note that w is not in the tree structure, because it is updated during the algorithm. The subscript j simply refers to node j in the tree. The algorithm tracks observations through the tree, using the recursive function $G(j, w_j)$ (where j and w_j are the same as defined above). The tree starts at the top of the tree $G(1, 1)$, node 1 with weight 1 and works its way down. At each level we update j and w_j in $G(j, w_j)$. The nodes j increase from left to right, from top to bottom (this will be visualised in the coming example).

The pseudo-code for the algorithm is shown below.

ALGORITHM 4: Estimating $\mathbb{E}_{x_{\text{train}} \sim \mathcal{D}_{\text{train}}}[f(x_{\text{train}}) \mid x_{\text{train}}^S = x_i^S]$

Input: Instance x_i, subset of features $S \subseteq N$, tree structure $\{v, a, b, t, r, d\}$
Output: Estimate of $\mathbb{E}_{x_{\text{train}} \sim \mathcal{D}_{\text{train}}}[f(x_{\text{train}}) \mid x_{\text{train}}^S = x_i^S]$

Procedure *G(j, w_j)*:
  if v_j is leaf **then**
    return $w_j \cdot v_j$;
  else if $d_j \in S$ **then**
    if $x_{d_j} \leq t_j$ **then**
      return $\mathrm{G}(a_j, w_j)$;
    end
    else
      return $\mathrm{G}(b_j, w_j)$;
    end
  else
    return $\mathrm{G}(a_j, w_j \frac{r_{a_j}}{r_j}) + \mathrm{G}(b_j, w_j \frac{r_{b_j}}{r_j})$;
  end
return $\mathrm{G}(1, 1)$;

In the following example we will show how Algorithm 4 works.

Example 4.4.1 In this example, we try to predict the amount of days absent using the overtime someone had last month and the number of months someone has been in service. Consider the following tree, which also shows what percentage of the training data is allocated to the left or right at each split.

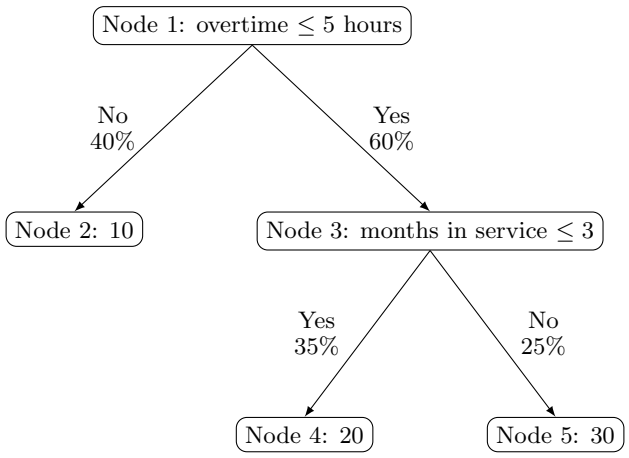

Figure 4.2: Example tree explanation from data to $v(S)$

Consider the following test observation:

Amount of days absent	Overtime	Months in service
15	6	2

Table 4.7: Test observation for example of coalitional value calculation

Now the goal is to estimate the coalitional values. First, we start with the empty coalition, so we have no data to use. We get the following calculation:

$$G(1,1) = G(2,0.4) + G(3,0.6) = 0.4 \cdot 10 + G(4, 0.6 \cdot \frac{0.35}{0.6}) + G(5, 0.6 \cdot \frac{0.25}{0.6}) = 4 + 0.35 \cdot 20 + 0.25 \cdot 30 = 18\frac{1}{2}.$$

Now we want to determine $v(\{\text{Overtime}\})$. Thus, we only use that our test observation has 6 hours of overtime.

$$G(1,1) = G(3,1) = G(4, \frac{35}{60}) + G(5, \frac{25}{60}) = 20 \cdot \frac{35}{60} + 30 \cdot \frac{25}{60} = 24\frac{1}{6}$$

Next up is $v(\{\text{Months in service}\})$:

$$G(1,1) = G(2,0.4) + G(3,0.6) = 0.4 \cdot 10 + G(4,0.6) = 4 + 0.6 \cdot 30 = 16.$$

Finally, $v(N)$:

$$G(1,1) = G(3,1) = G(4,1) = 20.$$

This results in the following game:

S	$\emptyset$	{Overtime}	{Months in service}	N
$v(S)$	18.5	$24\frac{1}{6}$	16	20

Table 4.8: Coalitional values estimated from the tree ▲

If there are multiple observations, then this process is repeated multiple times in order to get a game for each observation.

Implementation details for computing $v(S)$ with XGBoost in multiclass settings

Since XGBoost returns a model which consists of multiple trees, Algorithm 4 is repeated for each tree and the outcome is multiplied by the learning rate to get the coalitional values of the whole XGBoost model.

Furthermore, since we calculate coalitional values for multiclass classification, with four classes, our model returns a probability distribution for each observation. This means that we can calculate Definition 4.4.1 for each class and each observation. Therefore, for each observation i, we get coalitional values $v_i^c(S)$, where $c \in \{0, ..., 3\}$ represents the class (the class segmentation is discussed in Chapter 6). The game-theoretic attribution values are calculated using these coalitional values. This results in a game-theoretic attribution value for each class, feature, and observation.

Motivation behind method of determining $v(S)$

We use Algorithm 4 to determine $v(S)$ over the approach proposed by (Grigoryan, 2024) for various reasons. Firstly, the way from (Lundberg et al., 2018) does not require retraining the model for different subsets of features, while most of the methods of (Grigoryan, 2024) do. We would argue that retraining a model on a subset of features does not capture how the model with all the features handles that specific subset of features. Because it is very likely that the models parameters will change. Besides that, there is also a very high chance that a different set of hyperparameters is optimal, and thus comparing different coalitional values becomes difficult. (Lundberg et al., 2018) captures how each subset of feature interacts with the final model. Secondly, there already exists more research on determining $v(S)$ similar to (Lundberg et al., 2018), (Lundberg & Lee, 2017) and others, which means it is theoretically more grounded. Thirdly, (Grigoryan, 2024) only gives an intuition why they chose their way of determining $v(S)$. Calculating $v(S)$ using Definition 4.4.1 is in line with the definition of a coalitional value. Because

only the known information, the value for coalition S is calculated, given x^S. The expectation makes sure we only use information from the coalition. This all combined yields a close approximation of the coalitional value.

(Grigoryan, 2024) uses evaluation metrics such as the R^2 for example for the coalitional values. However, we are interested in knowing what the impact of a feature is on the final prediction of an observation (locally) or on the whole dataset (globally), instead of knowing what the impact is on evaluation metrics. This is also more in line with the EU AI Act (European Data Protection Supervisor, 2025), according to which predictions must be explained.

Some of the methods which do not require retraining to obtain $v(S)$ in (Grigoryan, 2024), are for voting games, which are not applicable in this thesis.

Finally, some methods for conflicting claims games, as described in (Grigoryan, 2024), do not use coalitional values. This excludes feature interactions because those methods only consider an individual feature compared to all features. Therefore, the approach of (Lundberg et al., 2018), which uses coalitional values and captures feature interactions, is more suitable for this thesis.

In conclusion, the method from (Lundberg et al., 2018) is used to estimate $v(S)$, since it reflects model behaviour better than the methods from (Grigoryan, 2024), which have shortcomings or are not applicable in the scope of this thesis.

4.4.2 Desirable properties of game-theoretic feature attribution methods

In Chapter 4.1, several properties were discussed that a game-theoretic solution can satisfy. This section explains why these properties are desirable in the context of feature attribution:

- Efficiency property (Equation 4.4): In feature attribution, it is important to fully explain the prediction by attributing every part of it to one or more features. If some of the prediction remains unexplained, it is unclear which features are responsible for that portion, making the explanation incomplete and less useful. The efficiency property guarantees that the value of the grand coalition (the prediction) is fully distributed among all features (players). Therefore, methods satisfying efficiency provide a complete overview of how features contribute to the prediction. However, the usefulness of this explanation depends on the accuracy of the model. Explaining an inaccurate prediction is unnecessary, this holds in general for feature attribution and is discussed at the beginning of Chapter 7.2.
- Individual rationality property (Equation 4.5): This ensures that every player receives at least what they can obtain on their own. In feature attribution, this property is less desirable, since a feature can either have a positive or a negative impact on a prediction. Since our coalitional values are probabilities (and thus non-negative), enforcing individual rationality would only allow for positive attributions. However, its explanation was included in Chapter 4.1 to explain the imputation set.
- Additivity and homogeneity properties (Equations 4.6 and 4.9): These properties allow for more efficient computation when averaging feature attributions over a dataset. This is because adding up all the games, dividing by the number of games, and calculating the game-theoretic value yields the same result as calculating the value for each observation and then taking the average.
- Symmetry property (Equation 4.7): If features have the same impact on a model, symmetry ensures they receive the same attribution value. This allows for fair comparison of the attribution values between features.
- Dummy property (Equation 4.8): If a feature does not contribute to the model, its attribution value is zero. This is desirable because it clearly distinguishes between contributing features and non-contributing features. However, in a model with well-selected variables, it is unlikely that a feature makes no contribution to a prediction.

4.4.3 Interpretation using Shapley values

(Lundberg, 2018) and (Lundberg & Lee, 2017) give a good explanation of what the SHAP package does. These explanations will be summarised in this subsection.

SHAP (SHapley Additive exPlanations) uses Shapley values, a game-theoretic concept which is explained in Chapter 4.2, to determine what the impact of a variable is on the final prediction. A part of cooperative game theory focuses on the allocation of revenue to players, as (Borm, 2024) explains. This

can also be of use in predictive models, by interpreting the prediction as the revenue and the variables as the players. The question then becomes how did each variable contribute to a prediction?

Although it seems like SHAP has no use case in a simple interpretable model, it can still be useful because interpreting coefficients alone can disregard the scale of the input features. For example, a coefficient can be relatively small compared with others, but if the feature values associated with this coefficient are extremely high, the corresponding feature has a big impact on the final prediction, even though it initially seems otherwise.

(Lundberg & Lee, 2017) states that SHAP calculates the Shapley values from the conditional expectation function of the model. This conditional expectation is similar to the one in Definition 4.4.1. They use these conditional expectations to see how a feature impacts the expectation of a prediction. By taking, for example, the difference between $\mathbb{E}[f(x)]$ and $\mathbb{E}[f(x) \mid x_1 = z_1]$, the impact of feature x_1 being equal to value z_1 can be determined. Computing the Shapley value is challenging, thus, they are estimated. By combining Shapley values with other explanation methods such as LIME and DeepLIFT, the approximations for the SHAP values can be made even faster. Since SHAP uses approximation techniques to obtain the Shapley values, we will not be using this package during this thesis. This explanation is included for the sake of completeness.

SHAP calculates the feature impact on a prediction locally (per observation). To see the global feature impact on predictions in general, we can take the average over multiple absolute values calculated by SHAP of multiple observations, as (Molnar, 2020, Section 18 SHAP) explains.

While using SHAP, it is important to keep in mind that it does not necessarily find causal relationships, it could also be that there is only correlation, as (Dillon et al., 2018) points out.

Optimised SHAP

Calculating Shapley values can be computationally heavy, therefore there are faster algorithms within the SHAP package for specific models. Deep-SHAP for example is a more efficient approach for Deep Neural Networks, as (Lundberg & Lee, 2017) explains. (Lundberg et al., 2018) explains how this is done for tree-based models such as XGBoost.

Chapter 5

Data

In this section, it is explained how the dataset is constructed. The choice was made to predict absence per month per employee, this decision is based on an internal discussion at Crowe Foederer. There was already aggregated data available for employee absence per month. Predicting on a monthly basis whether someone is absent is also a convenient time frame for an intervention.

The data used was the intersection of available and usable data, variables from the literature (see Chapter 2.2) that are used to predict or are related to absenteeism, and variables allowed by the EU AI Act (European Data Protection Supervisor, 2025).

Below in Table 5.1 is a short explanation of each feature in the dataset and its possible values. The features Function, Hours, Country and Container were only used to construct other features and not used while predicting. Function consisted of 2672 different possible entries, of which a lot were similar but not identical, so after an internal discussion at Crowe Foederer, Function was not used in prediction. The variable Hours was not used for prediction since it can be derived from FTE and FTE Hours. Container was not used as a feature for prediction, since it has nothing to do with absence. The variable Country was not used while predicting because it is assumed that everyone in the dataset used for prediction is from the Netherlands.

Features	**Domain**
Absence of current month	$0, 1, 2, 3, ...,$ maximum number of days current month
Year	2023, 2024 or 2025
Month sine	$\sin(2\pi \cdot \frac{m}{12})$ for $m = 1, \ldots, 12$
Month Cosine	$\cos(2\pi \cdot \frac{m}{12})$ for $m = 1, \ldots, 12$
Container	$\mathbb{Z}_+$, where each integer represents an anonymised company or group of companies
Function	$\mathbb{Z}_+$, where each integer represents an anonymised job title an employee has
Hours	$\mathbb{R}^+$
FTE	$\mathbb{R}^+$
FTE Hours	$0, \frac{1}{2}, 1, \frac{3}{2}, 2, \ldots$
Overtime hours	$\mathbb{R}^+$
Months in service	$\mathbb{N} \cup 0$
Absence last month	-1 if employee had no contract at that time, else integer in $\{0, 1, 2, 3, ...,$ maximum number of days in last month$\}$
Absence last year	-1 if employee had no contract at that time, else integer in $\{0, 1, 2, 3, ...$maximum number of days in month last year$\}$
Average windspeed of last month	Average float of daily average of windspeed which was measured in 0.1 m/s
Average temperature of last month	Average float of daily average of temperature which was measured in 0.1 degrees Celsius
Average precipitation of last month	Average float of daily average of precipitation which was measured in 0.1 mm (rounded to 0 if smaller than 0.05 mm)
Average air pressure of last month	Average float of daily average air pressure at sea level which was measured in 0.1 hPa
Average cloud coverage of last month	Average float of daily average of cloud coverage which was measured in octants, bigger or equal than 0 and smaller or equal than 9
Average temperature of current month	Average float of daily average of temperature which was measured in 0.1 degrees Celsius
Average precipitation of current month	Average float of daily average of precipitation which was measured in 0.1 mm (rounded to 0 if smaller than 0.05 mm)

Table 5.1: All the features and their possible values

5.1 Data and the EU AI Act

As was discussed in Chapter 1.4, the EU AI Act (European Data Protection Supervisor, 2025) rules out the use of variables which can, for example, discriminate against certain groups. Variables such as Children, Marital Status, Woman and (anonymised) EmployeeCode should be treated with caution, since they could allow a model to discriminate against certain groups. Therefore, they were excluded in this thesis, even though previous research does suggest there is a relation. This decision was based on an internal discussion at Crowe Foederer, a discussion with the TiSEM (Tilburg School of Economics and Management) Institutional Review Board, and a discussion between Pieter Kleer (this thesis's supervisor) and Cristian Dobre (MSc thesis coordinator of Business Analytics and Operations Research).

5.2 Structure of dataframe

Now the structure of the dataframe we use for prediction will be shown. It should be noted that only for the following table, the notation x_{ik} is changed to x_i^k (observation i and feature k) to improve readability.

	Feature 1	**Feature 2**	**Feature 3**	**Feature 4**	$\cdots$	**Feature N**
Employee 1, Month 1	$x_{1,1}^1$	$x_{1,1}^2$	$x_{1,1}^3$	x_{11}^4	$\cdots$	x_{11}^N
Employee 1, Month 2	$x_{1,2}^1$	$x_{1,2}^2$	$x_{1,2}^3$	$x_{1,2}^4$	$\cdots$	$x_{1,2}^N$
Employee 1, Month 3	$x_{1,3}^1$	$x_{1,3}^2$	$x_{1,3}^3$	x_{13}^4	$\cdots$	$x_{1,3}^N$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
Employee 1, Month 24	$x_{1,24}^1$	$x_{1,24}^2$	$x_{1,24}^3$	$x_{1,24}^4$	$\cdots$	$x_{1,24}^N$
Employee 2, Month 1	$x_{2,1}^1$	$x_{2,1}^2$	$x_{2,1}^3$	$x_{2,1}^4$	$\cdots$	$x_{2,1}^5$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$

Table 5.2: Visualisation of the dataset: Each row is an employee-month combination.

For every month an employee has a contract within the period April 2023 and March 2025, there is a row within the dataset. So if an employee has a contract for 3 months within the period April 2023 and March 2025 he has 3 rows in the dataset. The period April 2023, March 2025 was selected, such that each month occurs an equal amount of times in the dataset. The starting date was chosen as April 2023, such that the variable absence last year was not affected by corona. Since absence was increased by the coronavirus (see Chapter 2.2). This choice was made after an internal discussion at Crowe Foederer. After all data processing and cleaning steps, the final dataset consists of 86,323 observations. With the general structure of the dataframe discussed, the following section describes the construction of the dataframe.

5.3 Construction of dataframe

The dataframe, which will be used to predict absenteeism, was constructed using different dataframes. The data in each dataframe consists of data from multiple companies. The following dataframes were used:

- Dataframe 1 contains employee details,
- Dataframe 2 contains contract details,
- Dataframe 3 contains contract changes,
- Dataframe 4 contains sick leave,
- Dataframe 5 contains overtime
- Dataframe 6 contains weather data.

First, all unnecessary features in each dataset were dropped. Then all privacy-sensitive features were anonymised, such as EmployeeCode and ContractID, so that they could not be linked back to the employees. This was done by assigning a random number to each unique entry. Employees were dropped if they worked at companies which did not register sick leave.

Using the contract data a row was created in the new dataset for each month an employee has a contract within the period April 2023 and March 2025. The features from Table 5.2 were extracted from the original 6 datasets. Those features were put into the dataset by matching ContractID, EmployeeCode, year and month.

Finally, some anomalies were fixed, missing values were imputed and some changes were made to some of the variables. How some of the missing values were imputed can be found in the next subsection. For information on what types of missing values there are and how to deal with them, see (Bennett, 2001).

5.3.1 Data cleaning and choices regarding variables

Only employees who lived in the Netherlands were used, such that the weather features could be used. Employees who lived in the Netherlands also represented the largest portion of the dataset. The missing values for country were imputed the following way: per container[8] the Netherlands percentage was calculated (excluding unknown countries), so

$$\text{NL percentage} = \frac{\text{Number of employees from the Netherlands}}{\text{Number of employees for which country is known}}.$$

The results of these calculations are visualised in Figure 5.1. The employees with unknown countries were excluded if the NL percentage was lower than 0.85 else they were assumed to be living in the Netherlands. This choice was made since the NL percentage values were either below 0.7 or above 0.85. The whole process of how to deal with unknown countries was based on an internal discussion at Crowe Foederer.

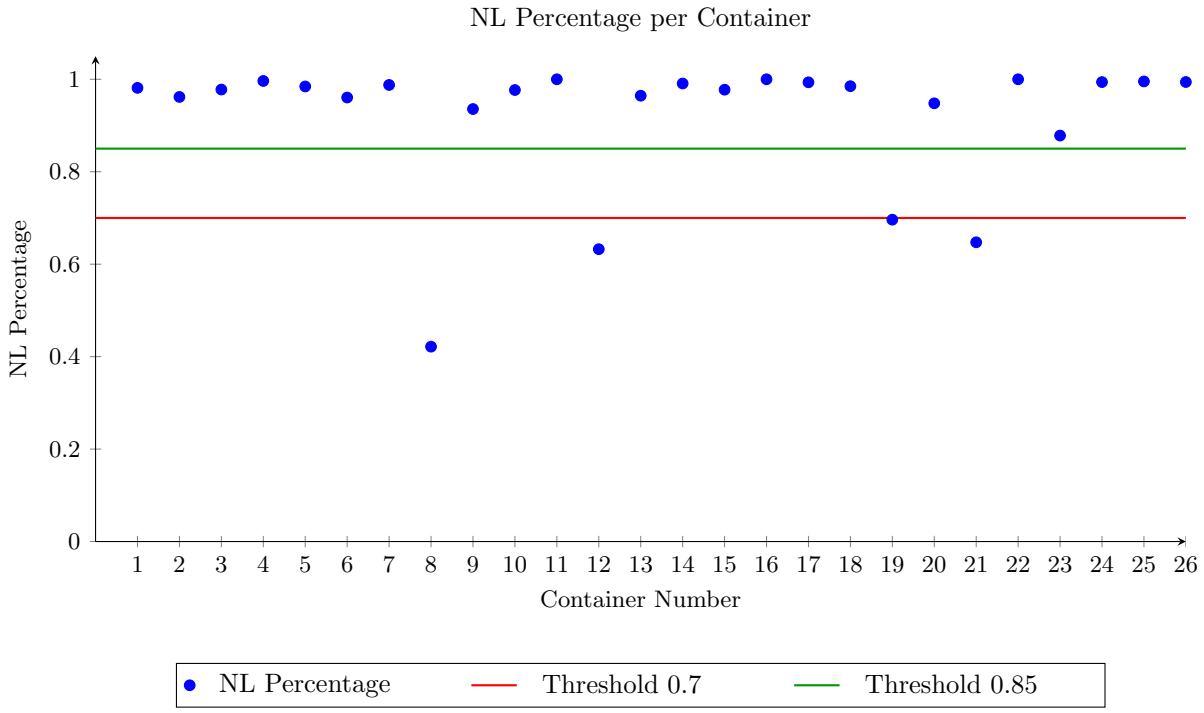

Figure 5.1: Visualisation of exclusion based on NL Percentage

The variables regarding the weather were based on De Bilt, which (Koninklijk Nederlands Meteorologisch Instituut, 2023) states is representative of weather in the Netherlands due to its central location. For these variables, daily weather data from (Koninklijk Nederlands Meteorologisch Instituut, 2025) was used. The feature precipitation was modified, because it contained -1 if the precipitation on a day was lower than 0.05 mm. These values were converted into 0 since precipitation was measured in 0.1 mm. After this conversion, all the features were averaged over time such that they turned into monthly averages.

[8] A container can be seen as a company or a group of companies.

The months were transformed in a cyclical way using the sine and cosine as is recommended by (Mahajan et al., 2021). The following formulas were used:

$$\text{Month sine} = \sin(2\pi \cdot \frac{m}{12}) \text{ for } m = 1, \ldots, 12,$$

$$\text{Month cosine} = \cos(2\pi \cdot \frac{m}{12}) \text{ for } m = 1, \ldots, 12.$$

This way, (Month sine, Month cosine) goes in a circle as Month is varied from 0 up to 12 (January up to and including December).

The feature FTE Hours was created by using the features FTE and Hours, where FTE represents the percentage an employee is working of their contractual full-time equivalent hours (FTE Hours). The following calculation was made:

$$\text{FTE Hours} = \begin{cases} \dfrac{\text{Hours}}{\text{FTE}} & \text{if FTE} > 0, \\ 0 & \text{if FTE} = 0. \end{cases}$$

If FTE Hours was missing, the mode was taken from contracts with the same function. If FTE Hours was still missing after taking the mode, it was set to 40 by default, since a 40-hour work week was the mode of the whole dataset.

The only features with missing values after all the modifications were Absence last month and Absence last year; these instances occurred if an employee had no contract at that time. After an internal discussion at Crowe Foederer, these missing values were replaced by -1. This was done after the correlation matrix (Figure 5.2) was generated.

Correlation matrix: Time dependency and employee dependency

Some features depend on the specific month (and year), such as the weather, some features depend on employee specifics, and some depend on both. The distinction between features that depend on time and features that depend on EmployeeCode can also clearly be seen in Figure 5.2. Two blocks can be distinguished. The first block, which is more EmployeeCode dependent, consists of:

- FTE,
- FTE Hours,
- Months in service,
- Absence current month,
- Absence last month,
- Absence last year.

The second block, which is more time-dependent, consists of:

- Year,
- Month sine,
- Month cosine,
- Average daylight minutes of current month,
- Average windspeed of last month,
- Average temperature of last month,
- Average precipitation of last month,
- Average air pressure of last month,
- Average cloud coverage of last month,
- Average temperature of current month,
- Average precipitation of current month.

Overtime last month does not fall in any of those two blocks.

The variable average monthly daylight minutes was dropped due to having a perfect negative correlation with month cosine. Intuitively this makes sense, since the average monthly daylight minutes are the same each year.

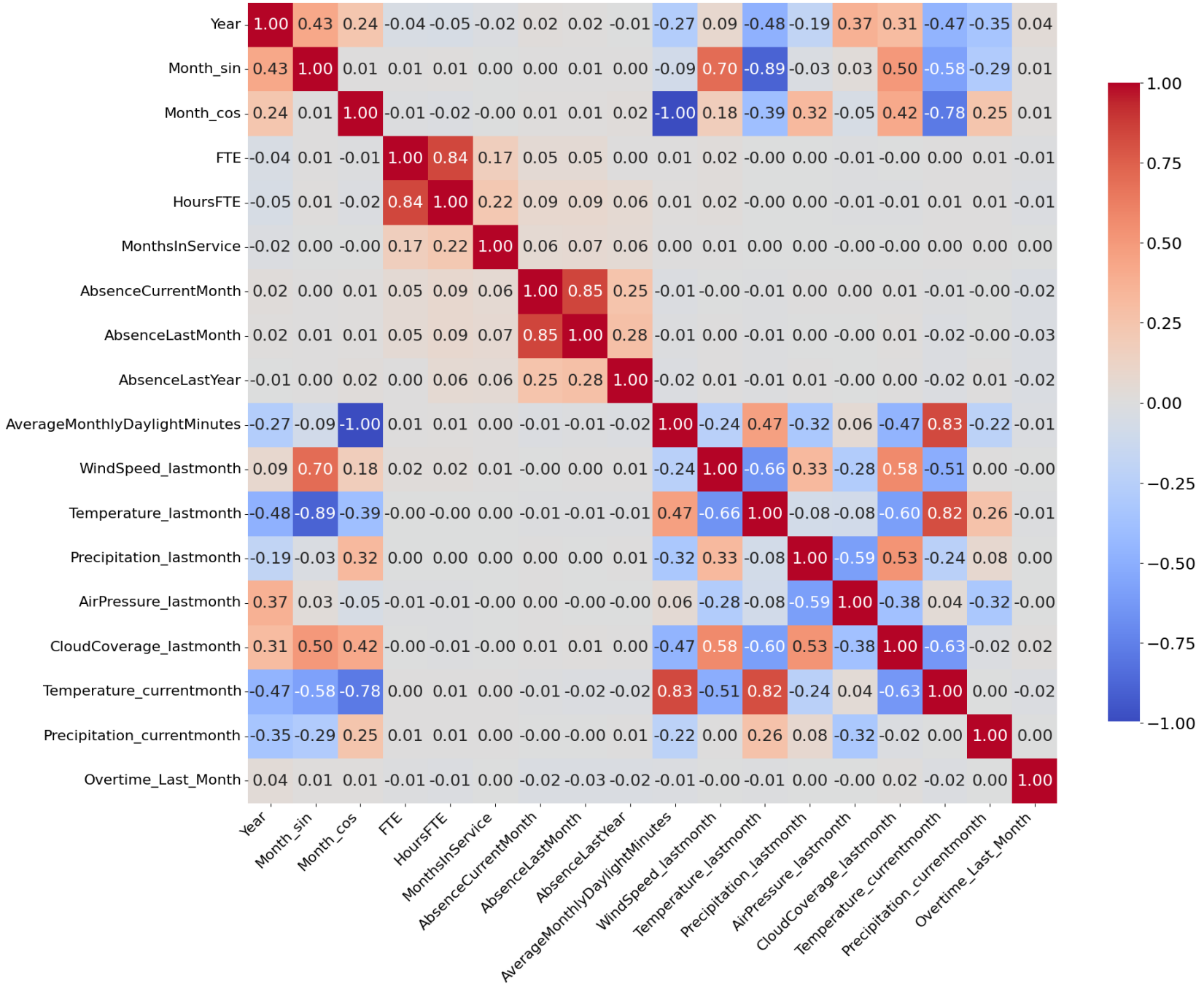

Figure 5.2: Correlation matrix of all the features

Further data cleaning decisions
In addition to data preparation discussed above, there were also some specific decisions made. For the sake of completeness, they are listed below:

- If a contract had no contract changes in FTE Hours or FTE and those values were missing in the initial contract, then those rows were removed.
- All employees who did not have a start date for their contract were excluded, because it is impossible to know during which months they were employed.
- In the case of multiple contracts in a month, the contract with the most recent start date will be used for that month.
- Employees who were missing in any of the used datasets were removed.
- Employees associated with containers which did not contain any absence were excluded from the dataset. The same holds for employees associated with containers which did not contain any overtime.
- The variable overtime was used despite reducing the size of the dataset from 145,933 observations to 86,323 observations, as it increased the validation accuracy and a custom metric (the custom metric is defined at Equation 6.1 in Chapter 6.1).
- There is a slight increase in observations as time goes on (see Figure F.1 in Appendix F). After an internal discussion at Crowe Foederer, it was determined that this is not a problem.

5.4 Selection of time-dependent variables

When building a predictive model for absenteeism, it is important to consider which variables are available at the time of prediction. Some features may only be available with a delay, some can be forecasted, and others may not be available at all. This section explains the reasoning behind the chosen time-dependent features.

Why variables of last month and last year were used
While predicting absenteeism, it is most useful to use variables which are known at the time of predicting, such as the contractual hours an employee has. However, some variables are not known at the time of prediction, for example, what the average monthly cloud coverage will be in the current month. In such cases, the last known value of the variable (from the previous month in our case) is used, as it is the closest available data point and could be argued to still influence absenteeism in the current period. This decision was based on an internal discussion at Crowe Foederer.

Why only some weather features of the current month were used
To include some weather features for the current month, we use temperature and precipitation predicted one month ahead. Some sources, such as (Weer33, 2025), provide such predictions. There are also other sources which give a prediction for the temperature and an indication of the precipitation (such as (AccuWeather, 2025) and (The weather channel, 2025)). No accessible sources could be found that predict other weather features a month into the future, and thus only temperature and precipitation for the current month were used. There are some sources (such as (Weather Atlas, 2025) and (Weer1.com, 2025)) which give the historical mean of weather features for given months. However, this is just the same as giving the month (which is already in the data). It is unfortunately not clear how accurate these predictions are a month ahead and with which models these predictions are generated. There is some previous research on how accurate weather predictions are (see for example (Mishra et al., 2018), (Zheng et al., 2017), (Brown, 2014), (Fan & van den Dool, 2011) and (Rodwell & Doblas-Reyes, 2006)). However, the question then becomes whether those papers use the same models as the sources that predict the weather a month ahead. They probably do not use the same models since (Mishra et al., 2018), (Zheng et al., 2017), (Fan & van den Dool, 2011), (Rodwell & Doblas-Reyes, 2006) predict for multiple countries, and (Brown, 2014) predicts for the United States of America. One could argue that, averaged over a month, the errors are reduced because over- and underestimations could cancel each other out. However, whether this effect occurs depends on the accuracy and potential biases of the weather forecasting model. An internal discussion at Crowe Foederer resulted in the inclusion of the features temperature current month and precipitation current month. This solution was the best balance between using the weather of the current month for prediction and realistically obtainable features. It should be noted that for

training, validating and testing the true monthly averages were picked for temperature current month and precipitation current month, because old predictions were not available. Further research is necessary to check what the effect of that decision is.

5.5 Normalisation and standardisation

(Hastie et al., 2009) explains that tree-based learners are invariant to strictly monotone transformations. Since they split created splits based on the order of the possible values of the features. Thus normalisation and standardisation is not necessary for the XGBoost model. It was used for regression models, the SVM. For the normalisation and standardisation, we use formulas which are presented in the documentation of the skicit-learn Python package (see (scikit-learn developers, 2025a) and (scikit-learn developers, 2025b)). For normalisation, we used the following formula:

$$x'_{ij} = \frac{x_{ij} - x_{\min,j}}{x_{\max,j} - x_{\min,j}}. \tag{5.1}$$

Where $x_{\min,j}$ and $x_{\max,j}$ are defined the following way:

$$x_{\min,j} = \min_{1 \leq i \leq n} x_{ij} \quad \text{and} \quad x_{\max,j} = \max_{1 \leq i \leq n} x_{ij}.$$

For standardisation, we used the following formula:

$$x'_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}. \tag{5.2}$$

Where the mean μ_j and the standard deviation σ_j are defined the following way:

$$\mu_j = \frac{1}{n} \sum_{i=1}^{n} x_{ij} \quad \text{and} \quad \sigma_j = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_{ij} - \mu_j)^2}.$$

Chapter 6

Experimental setup

In this chapter, we describe the experimental setup. First, we explain why we have chosen four classes and accuracy as the performance metric, then we will discuss hyperparameter optimisation, and finally feature importance.

6.1 Class segmentation and performance metric

The chosen model is a multiclass classification model, which would be the most interesting to research. This and how the classes are segmented were determined after an internal discussion at Crowe Foederer. Based on (Wahid et al., 2019), the classes were segmented in the following way:

- Class 0 if days absent is equal to 0,
- Class 1 if days absent is bigger than 0 and less or equal than 7 days,
- Class 2 if days absent is bigger than 7 days and less or equal than the whole month,
- Class 3 if days absent is equal to the whole month.

This decision was based on the distribution of the data. Class 0 appeared 86.03% of the time in our selected data. In Figure 6.1 below is the distribution of the number of days an employee is absent in a month, given that he is absent that month, so $(y_i \mid y_i > 0)$. It can clearly be observed that the whole month absent should be its own class, and in frequency, there is also a big decrease after one week; therefore, it is chosen as its own class.

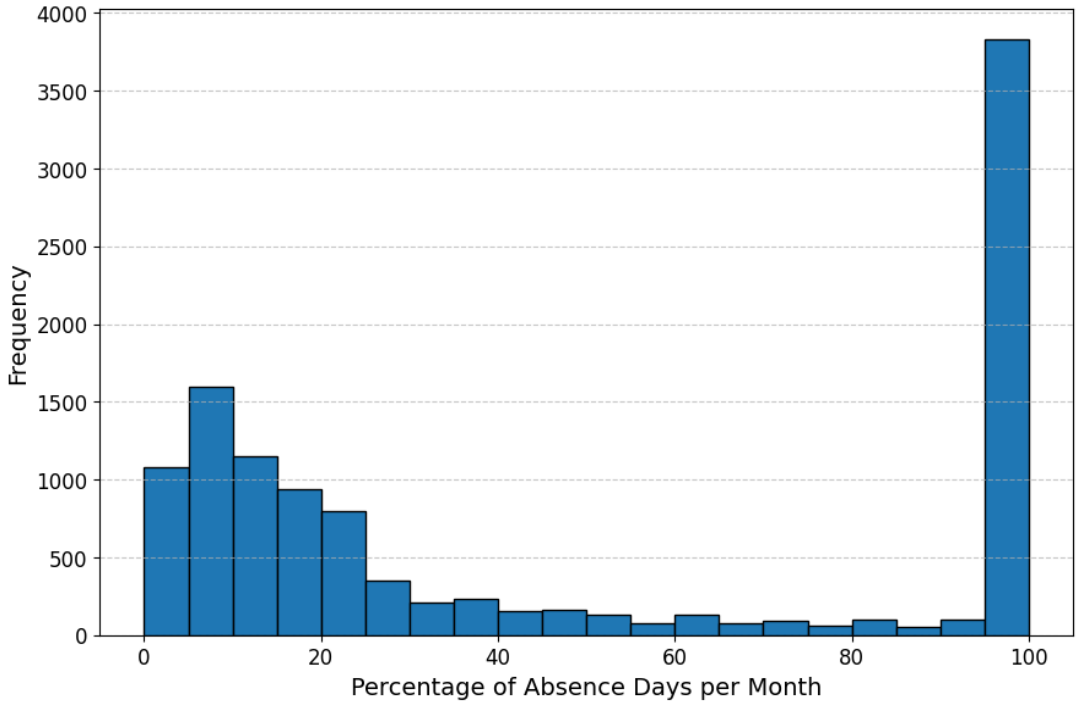

Figure 6.1: Frequency of amount of days sick per month (excluding zero) normalised over months (and accounting for leap years) and put into bins of 5 percent.

Performance metric

After an internal discussion at Crowe Foederer it became clear that they wanted a model with a high accuracy (see Equation 3.2), that would rather predict conservatively than overestimate the amount of absence (and thus the class). If we have the following confusion matrix:

Actual / Predicted	Class 0	Class 1	Class 2	Class 3
Class 0	TP_0	$\mathrm{FP}_{0,1}$	$\mathrm{FP}_{0,2}$	$\mathrm{FP}_{0,3}$
Class 1	$\mathrm{FN}_{1,0}$	TP_1	$\mathrm{FP}_{1,2}$	$\mathrm{FP}_{1,3}$
Class 2	$\mathrm{FN}_{2,0}$	$\mathrm{FN}_{2,1}$	TP_2	$\mathrm{FP}_{2,3}$
Class 3	$\mathrm{FN}_{3,0}$	$\mathrm{FN}_{3,1}$	$\mathrm{FN}_{3,2}$	TP_3

Table 6.1: Confusion matrix for a 4-class classification problem. TP: True Positive, FP: False Positive, FN: False Negative.

Then maximising the amount of conservative predictions implies minimising our false positive predictions. We do this by maximising the fraction of true positives over the true positives and false positives. This leads to the following custom metric:

$$\text{Custom metric} = \frac{\sum_{i=0}^{3} TP_i}{\sum_{i=0}^{3}(TP_i + \sum_{j=1}^{3} FP_{i,j})}. \tag{6.1}$$

After running experiments, it became clear that the model with the highest accuracy (see Equation 3.2) was also the model with the highest custom metric (see Equation 6.1). This is probably due to a high correlation between the two metrics. For example, for one grid search there was a correlation of 0.9832 between the metrics. So the goal became to find the model with the highest accuracy, averaged across all validation sets.

6.2 Hyperparameter optimisation

The performance of the multiclass classification XGBoost model will be compared to an SVM multiclassification model, always predicting 0 and an easier regression model, whose predictions are rounded to classes. The easier regression model was scrapped because it returned results with an accuracy lower than 20%. For this model we tried a linear regression, a piecewise linear regression, and a logistic regression (so a regression fit on the function of Equation 3.1), all with normalised (Equation 5.1), standardised (Equation 5.2) and untransformed data. For the XGBoost model and the SVM, we will run grid searches to determine their optimal hyperparameters.

There are several methods to determine the "best" hyperparameters, for example, grid search, random search or Bayesian search ((Shekhar et al., 2021) compares some of these methods). After an internal discussion at Crowe Foederer the choice was made to perform grid searches.

(XGBoost developers, 2022) talks about parameter optimisation being hard. All the possible hyperparameters can be found in the XGBoost documentation (see (XGBoost developers, 2022)).

XGBoost Grid search

Grid searches were performed. The hyperparameters that were optimised, their ranges, and their optimal values can be found in the table below.

Hyperparameter	**Search Range**	**Optimal Value**
Amount of estimators M	[100, 6000]	2800
Learning rate η	[0.01, 0.4]	0.07
Maximum depth	[6, 50]	30
Subsample	[0.7, 1]	0.95
Colsample (per tree)	[0.7, 1]	0.7
γ	[0, 1]	0
λ	[0.5, 5]	1
Minimum child weight	[1, 5]	1

Table 6.2: Grid search ranges and optimal hyperparameters for the multiclass XGBoost model

For all the other hyperparameters, the default values were used. Around each best hyperparameter combination, additional searching was done. This also means that the grid was sometimes extended (but never outside of the specified range above).

The loss function is the weighted cross-entropy function with weights inversely proportional to how often a class occurred in the training data (Equations 3.3 and 3.11). As objective, the multi softprob was chosen; this uses the cross entropy loss as (Mao et al., 2023) states. We chose this loss because it is widely accepted and an effective measure of error according to (Boudiaf et al., 2020).

SVM Grid search

The hyperparameters, parameter ranges and the optimal values can be found in the table below. We ran grids for the untransformed data, normalised (Equation 5.1) data and standardised data (Equation 5.2). The optimal hyperparameters, together with the untransformed data (non-normalised and non-standardised) returned the best results for the validation sets. We again applied weights to address the class imbalance. All possible hyperparameters can be found in the documentation (scikit-learn developers, 2025c). For all non-specified hyperparameters, the default values were used.

Hyperparameter	Search Range	Optimal Value
Penalty parameter τ	$[0.01,\ 100]$	11
Kernel type κ	{rbf, poly, sigmoid}	rbf
Kernel coefficient γ	$[0.0001,\ 0.1] \cup \{\text{scale},\ \text{auto}\}$	0.0005
Polynomial degree d	{2, 3, 4, 5}	–*
Coefficient term p	{0, 0.5, 1, 5}	–**

*Only applicable for the `poly` kernel, not used for the optimal `rbf` or `sigmoid` kernels.
**Applicable for the `poly` and `sigmoid` kernels, not used for the optimal `rbf` kernel.

Table 6.3: Grid search ranges and optimal hyperparameters for the multiclass SVM model

6.3 Feature importance

Since the model is trained on 16 features it would result in $2^N = 2^{16} = 65{,}536$ possible coalitions. These calculations with an optimised algorithm on a computer with 128GB of memory would take around 60 days. To limit energy consumption, we chose only 10 features, which brought the runtime down to roughly a day. Therefore, we look at the feature importance to a select 10 features for the game-theoretic interpretation. The concept of interpreting the model using game theory can be generalised to more features, but for this thesis, it would unfortunately take too much time. After an internal discussion at Crowe Foederer, the decision was made to do feature attribution on the eight most important features according to the feature importance of XGBoost and the two least important features according to the feature importance of XGBoost. Crowe Foederer was more interested in the highly ranked features importance (according to the XGBoost feature importance). After an internal discussion at Crowe Foederer the two lowest ranked features were also included, such that we can see what the difference is from a game-theoretic perspective of highly ranked features compared to lowly ranked features (again, according to the XGBoost feature importance).

In (XGBoost developers, 2022) it is explained how the feature importance calculation works of the XGBoost package for multiclass classification. The feature importance (for a tree model, which is used in this thesis) can be calculated in different ways within the XGBoost package:

- Gain, the average gain of splits which use that feature (gain is calculated in Equation 3.8).
- Weight, the number of times that feature appears in a tree.
- Cover, the average of the number of samples affected by a split on that feature, averaged over all the splits which use that feature.
- Total gain, the gain of that feature over all splits.
- Total cover, the total cover over all splits.

We used total gain, since this shows overall how much a feature added to the loss reduction while training. Since we want to know what the impact of a feature is on the whole model, we chose total gain instead of gain (which is averaged over the splits a feature is used in). Figure 6.2 shows the total gain for each feature, for the model with the optimal hyperparameters (which is only trained on datasets which were used for validation). A higher total gain means a higher loss reduction by using this feature in splits and thus a more important feature.

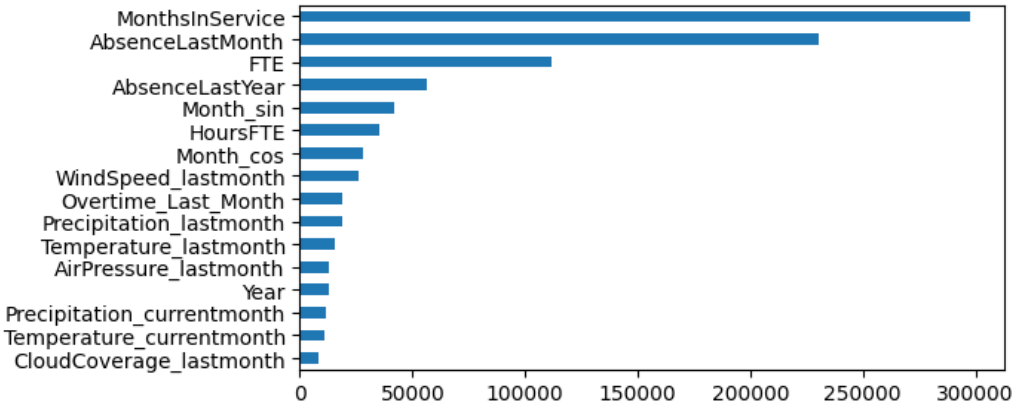

Figure 6.2: Total gain of XGBoost after training, used for feature importance. On the y-axis the features and on the x-axis the total gain.

So in the end we used the following features for feature attribution:

- Months in service,
- Absence last month,
- FTE,
- Hours FTE,
- Absence last year,
- Month sin,
- Month cos,
- Wind speed last month,
- Temperature current month,
- Cloud coverage last month.

To validate this choice of parameters, we wanted the coalitional values of the 10 features above, averaged over all observations for each class, to be close to the value of the grand coalition, also averaged over all observations for each class. Multi softprob returns a probability distribution; thus we need to compare two distributions.

Coalition	**Class 0**	**Class 1**	**Class 2**	**Class 3**
$\boldsymbol{v}$**(selected 10 features)**	0.925614244	0.021554954	0.004194128	0.048636675
$\boldsymbol{v(N)}$	0.939739713	0.010516263	0.001632298	0.048111727

Table 6.4: Comparison of v(selected 10 features) and $v(N)$

To compare v(selected 10 features) and $v(N)$, we will use the total variation distance, which is for example explained in (Bhattacharyya et al., 2023). Following their notation, the total variation distance $d_{\mathrm{TV}}(P, Q)$ between two discrete distributions P and Q is defined as:

$$d_{\mathrm{TV}}(P, Q) = \frac{1}{2} \sum_{x \in D} |P(x) - Q(x)|. \tag{6.2}$$

The distance between distributions P and Q is low if $d_{\mathrm{TV}}(P, Q)$ is close to 0, the distance is high if $d_{\mathrm{TV}}(P, Q)$ is close to 1.

So we have, $P = v$(selected 10 features) and $Q = v(N)$, which means P and Q are the following:

$$P = \{0.925614244, 0.021554954, 0.004194128, 0.048636675\},$$

$$Q = \{0.939739713, 0.010516263, 0.001632298, 0.048111727\}.$$

By plugging P and Q into Equation 6.2 we obtain:

$$
\begin{aligned}
d_{\mathrm{TV}}(P,Q) &= \frac{1}{2}\left(|0.925614244 - 0.939739713| + |0.021554954 - 0.010516263|\right. \\
&\quad \left.+|0.004194128 - 0.001632298| + |0.048636675 - 0.048111727|\right) \\
&= \frac{1}{2}\left(0.014125469 + 0.011038691 + 0.002561830 + 0.000524948\right) \\
&= \frac{1}{2} \cdot 0.028250938 \\
&= 0.014125469.
\end{aligned}
$$

This is close to 0, and thus the distance between the distributions is low. Therefore, using the selected 10 features captures the most important information of the model. Hence, using feature attribution for the selected 10 features gives us relevant information on how those features influence the whole model.

Chapter 7

Results

In this chapter, we discuss the results of our models. First, we show the performance of the prediction models, followed by the game-theoretic feature attribution.

7.1 Model performance

Now we will discuss the results of the XGBoost model and SVM (using the found hyperparameters as discussed in Chapter 6.2). Recall from Chapter 3.2.2 that the test data is not balanced to reflect a real-world scenario.

	XGBoost	**SVM**	**Always Predict 0**
Average Accuracy	0.8850	0.8666	0.8612
Average Custom Score	0.9715	0.9707	1.0000

Table 7.1: The average accuracy and average custom score, averaged over the 6 datasets and 6 models, of the XGBoost model, SVM, and always predicting 0.

We can clearly see that the XGBoost model outperforms the SVM both by means of the accuracy (Equation 3.2) and the custom score (Equation 6.1). Both the XGBoost model and the SVM score better in terms of accuracy than always predicting 0. (We do not need to look at the average custom score of always predicting 0, because it is always 1 by definition in that case).

As has been discussed earlier in Chapter 6.1, while discussing the custom metric, Crowe Foederer would rather have a model that underpredicts (so predict a lower class for an observation than it is) than a model which overpredicts (so predict a higher class for an observation than it is), in the case of absenteeism. Since the custom score is not 1, one should consider the risk and determine whether the improved accuracy of XGBoost is worth implementing, compared to always predicting 0 (or not implementing a model at all). In this thesis, a decision cannot be made as to whether it is "the best decision" to implement the XGBoost model, because that depends on the preferences of Crowe Foederer.

Now we will specifically show the XGBoost confusion matrix for each test set. Note that we retrain each time we run a new test set as is shown in Figure 3.17.

True\Pred	**Class 0**	**Class 1**	**Class 2**	**Class 3**
Class 0	3207	16	1	18
Class 1	180	3	2	15
Class 2	49	2	1	12
Class 3	15	1	1	152

Table 7.2: Confusion matrix for the first test set

True\Pred	Class 0	Class 1	Class 2	Class 3
Class 0	3170	117	9	20
Class 1	159	13	1	12
Class 2	52	5	2	14
Class 3	20	0	3	149

Table 7.3: Confusion matrix for the second test set

True\Pred	Class 0	Class 1	Class 2	Class 3
Class 0	3304	1	2	12
Class 1	220	0	0	9
Class 2	64	0	0	27
Class 3	15	1	0	153

Table 7.4: Confusion matrix for the third test set

True\Pred	Class 0	Class 1	Class 2	Class 3
Class 0	3241	50	2	15
Class 1	256	10	3	6
Class 2	49	7	2	15
Class 3	19	1	3	159

Table 7.5: Confusion matrix for the fourth test set

True\Pred	Class 0	Class 1	Class 2	Class 3
Class 0	3120	53	8	9
Class 1	365	7	3	9
Class 2	82	1	2	17
Class 3	8	1	4	152

Table 7.6: Confusion matrix for the fifth test set

True\Pred	Class 0	Class 1	Class 2	Class 3
Class 0	3111	70	3	9
Class 1	302	17	2	13
Class 2	100	3	0	15
Class 3	36	5	0	133

Table 7.7: Confusion matrix for the sixth test set

It becomes clear that the XGBoost models can predict classes 0 and 3 relatively well, but are poor at predicting classes 1 and 2. Even though weights were used to deal with the class imbalance, this probably was not enough. It could be that a different weight distribution would have yielded better results. It could also be the case that no clear relation can be found in the used features and classes 1 and 2.

7.2 Feature attribution

Now we use the game theory discussed in Chapter 4 to interpret the behaviour of the 6 XGBoost models we have. We will average our results in this section over those 6 models, to get a general overview on the impact of the features in our models (and not for a specific month). Some of the plots were put in Appendix E for improved readability.

Feature attribution assumption
Before showing the feature attribution results, we first need to go over one assumption that is needed for feature attribution. In feature attribution, the goal is to explain a model's prediction, such that we know why a certain prediction is made (as stated by (Molnar, 2020) and others). By stating this, we implicitly assume that predictions are correct. However, in our case, we can see in the confusion matrices in Tables 7.2 - 7.7, that this is definitely not the case for predictions for classes 1 and 2. Since almost no predictions for those classes are correct, feature attribution for these classes does not make sense, because explaining incorrect predictions does not give useful information. Therefore, for classes 1 and 2, we will not show any feature attribution results. However, if some unexpected behaviour occurs, then we may briefly discuss it.

7.2.1 Average feature attribution

Feature attribution can be done for each (test) observation. However, we cannot interpret the feature attribution of all test observations. So we interpret only the average feature attribution. We do this by first calculating the coalitional values for every observation, then calculating the game-theoretic values for every test observation, and finally taking the average over all those observations per game-theoretic value. These game-theoretic attributions will be visualised in bar charts. In these bar charts, the values should not be interpreted directly, since some methods are efficient, and others are not. To be able to compare different figures with each other, we scaled the average attribution, such that the average attribution values always lie within $[-1, 1]$. We did this scaling by dividing the average attribution by the maximum of the absolute value of the minimum average feature attribution and the maximum average feature attribution. The minima and maxima were calculated over all k features, and to each feature, the same scaling was applied. The formula used for the scaling looks the following way:

$$x_k^{\text{scaled}} = \frac{x_k}{\max\left(\left|\min_k x_k\right|, \left|\max_k x_k\right|\right)}. \tag{7.1}$$

Normally, while interpreting these attribution values, this scaling would not be applied. This scaling is only used to better compare attribution values of different methods to each other.

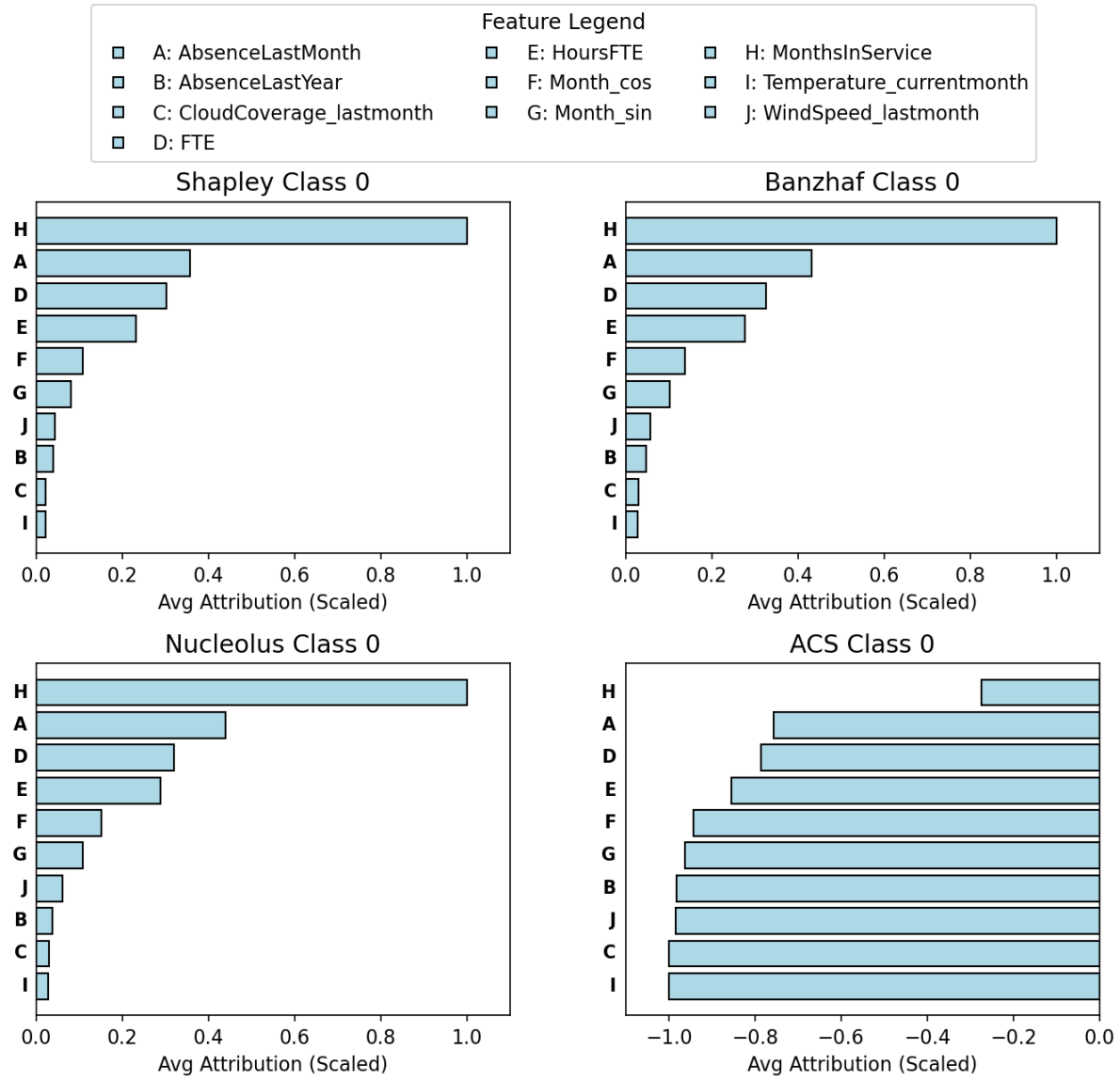

Figure 7.1: Feature attribution (scaled to be within $[-1, 1]$) averaged over all samples all four game-theoretic concepts for class 0.

We can see that the Shapley, Banzhaf and nucleolus average feature attributions are close to identical (see Figure 7.1). They also look very similar to the ACS concept, if one looks at what part of a bar is not colored. However, the absence last year and wind speed last month are flipped rank wise.

Looking at the bars for class 3 in Figure E.1 (see Appendix E), one can see that they all differ. The only common thing is that the feature months in service always has a positive feature attribution value for this class. These differences in attributions arise because each game-theoretic concept has a different objective.

General interpretation of each attribution method

Since the goal of using various game-theoretic methods is to see how they differ, we will now discuss how to interpret the bars in general for each game-theoretic method and not go into too much detail. Later on, we will discuss rankings and beeswarm plots, which show how various game-theoretic methods give similar or different solutions.

Before diving into individual methods, one general observation can be made. As shown in Figures 7.1 and E.1, all feature attribution values can be either positive or negative (in all methods). Now that this

overall result has been discussed, we will move on to method-specific general interpretations.

Interpreting the bars for the ACS solution is relatively easy. We can see how far a feature is on average from the value of the grand coalition, and whether this is positive or negative. If it is negative, then on average (with respect to coalitional size, as discussed before) the coalitional values are below the value of the grand coalition; if it is positive, then on average (again with respect to coalitional size) the coalitional values are above the value of the grand coalition. A larger absolute value of the ACS for a feature indicates that, on average, the coalitional values that contain that feature are further from the value of the grand coalition. Therefore, such a feature disagrees more with the value of the grand coalition, compared to a feature which has a lower ACS value. It should be noted that the ACS concept does not take the value of the empty coalition into account.

Interpreting the bars for the nucleolus is difficult, since the nucleolus minimises the maximum excess over all subsets of features, which can also be seen as minimising the maximum dissatisfaction over all subsets of features. A larger absolute attribution value indicates a larger impact of a feature. A positive nucleolus value means that a feature attributes positively to coalitions, and a negative value means that a feature attributes negatively to coalitions. An advantage of the nucleolus could be that it is efficient, so it distributes the value of the grand coalition amongst all features, so a prediction can be decomposed into a fraction per feature, which represents the attribution. Similarly to the ACS concept, the nucleolus also does not take the value of the empty coalition into account.

The interpretation of the Banzhaf and Shapley values of the bars is relatively easy. They represent the marginal contribution. The advantage of the Shapley value over the Banzhaf value is that the Shapley value is efficient (with respect to $v(N)-v(\emptyset)$ see Chapter 4.2). A larger absolute attribution value means a larger impact for both concepts. A positive attribution value means that a feature has a weighted positive marginal contribution, again for both concepts, while for a negative attribution value, the opposite holds.

We observed one inconsistency: the Shapley and Banzhaf values are all negative for class 2. This is due to the nature of the Shapley and Banzhaf value. As stated before, the Shapley value is efficient not to $v(N)$, but to $v(N) - v(\emptyset)$. $v(\emptyset)$ represents the average prediction (in this case for class 2) across the training data (see Chapter 4.4.1). This, combined with a decrease in coalitional values as the coalition size increases (which is the case for class 2), results in negative marginal contributions. This makes interpreting these values less intuitive, since we stated that the Shapley and Banzhaf values give a weighted average of marginal contributions, which is rarely all negative. However, all negative marginal contributions mean that all features decrease the probability of class 2 being predicted. Although it aligns with the mathematical definition of the Shapley and Banzhaf value, it does not give the intuitive interpretation that the Shapley and Banzhaf value usually have. Since usually at least one feature has a positive marginal contribution on a class (for the Shapley value, see the examples in (Lundberg, 2018) and (Lundberg et al., 2018)). However due to the nature of our TU game all negative marginal contributions are possible.

The Shapley value, Banzhaf value and ACS solution are relatively easy to interpret. However, the Shapley and Banzhaf value both show a case where all their values are negative, which leads to a less intuitive interpretation, since often at least one feature has a positive impact on a class. So if the goal is intuitive interpretability, using the ACS solution is recommended. Using Shapley values is recommended if a more researched game-theoretic approach for feature attribution is required, besides that there already exists a package SHAP which requires less coding and is therefore easier to use. If one does not want to look at marginal contributions and does not want to use the ACS, the nucleolus gives a different perspective; however, the interpretation is less intuitive.

Now that the results for each method have been discussed generally, we would like to interpret the bars of all bar charts. Since it is difficult to interpret all bar charts at once, we will look at the rankings that appear from each bar chart. (Note that for the ACS concept, a smaller bar indicates a higher importance, as has been discussed in Chapter 4.3.) We will compare these rankings to each other, but also to the XGBoost feature importance ranking. The XGBoost feature importance ranking is again obtained by looking at the total gain per feature, as was previously done for the validation datasets in Figure 6.2. However, this time only the training datasets associated with testing were used (see Figure 3.17). For the selected 10 features, the order is exactly the same as in Figure 6.2.

Method	ACS	Banzhaf	Nucleolus	Shapley	ACS	Banzhaf	Nucleolus	Shapley	XGBoost
Class	0	0	0	0	3	3	3	3	-
Months in service	1	1	1	1	1	1	2	1	1
Absence last month	2	2	2	2	9	7	1	3	2
FTE	3	3	3	3	2	4	4	2	3
Hours FTE	4	4	4	4	3	6	8	4	6
Month cos	5	5	5	5	6	9	10	8	7
Month sin	6	6	6	6	5	5	5	7	5
Wind speed last month	8	7	7	7	7	10	7	9	8
Absence last year	7	8	8	8	10	2	3	5	4
Cloud coverage last month	9	9	9	9	4	3	9	6	10
Temperature current month	10	10	10	10	8	8	6	10	9

Table 7.8: Feature ranking of all feature attribution methods for each class, and XGBoost feature importance, where 1 is the most important feature and 10 is the least important feature.

Permutation swap distance	ACS Class 0	Banzhaf Class 0	Nucleolus Class 0	Shapley Class 0	ACS Class 3	Banzhaf Class 3	Nucleolus Class 3	Shapley Class 3	XGBoost
ACS Class 0	0	1	1	1	14	20	14	7	8
Banzhaf Class 0	1	0	0	0	13	19	15	8	7
Nucleolus Class 0	1	0	0	0	13	19	15	8	7
Shapley Class 0	1	0	0	0	13	19	15	8	7
ACS Class 3	14	13	13	13	0	18	26	17	12
Banzhaf Class 3	20	19	19	19	18	0	24	21	20
Nucleolus Class 3	14	15	15	15	26	24	0	13	20
Shapley Class 3	7	8	8	8	17	21	13	0	13
XGBoost	8	7	7	7	12	20	20	13	0
Sum	66	63	63	63	126	160	142	95	94

Table 7.9: The permutation swap distance of the rankings between all methods and classes, including XGBoost feature importance

To interpret the rankings in Table 7.8, we will calculate the Kendall tau rank distance (introduced by (Kendall, 1938), which can also be called the bubble sort distance. It calculates for 2 rankings the number of pairs that are in a different order (or, in bubble sort terms, the minimal number of swaps of the adjacent ranked elements needed to obtain the other ranking).

The Kendall tau rank distance can be seen in the symmetric Table 7.9. Note that the maximum distance is 45. We can use these distances to compare the feature rankings between different methods for the same class (and XGBoost), as well as within the same method for different classes (and XGBoost). It can be seen that the rankings for the features in class 0 are equivalent, with the exception of one swap

for the ACS concept. It stands out that this is the only class where all the methods are so similar. Of the rankings of class 3, the ranking based on the Shapley value is the closest to the other rankings. The ranking of class 3 that uses the Banzhaf value is the furthest from all other rankings.

In Table 7.8, it can be seen that the rankings of features for class 3 differ between methods. With the exception of the feature months in service being the (second) most important feature.

Looking at the ranks in Table 7.8, we can see that for each method, the rankings of features differ between classes. Among the game-theoretic concepts, the Shapley value shows the smallest distance in feature importance ranking between classes 0 and 3, in contrast to the Banzhaf value, which shows the largest distance. This indicates that each class has a different ranking of feature importance per method. Thus, for each class, different features can be seen as less or more important. Meaning that the combination of features is necessary to make good predictions for all classes. There is one clear pattern between the ranked features: the feature months in service always ranked first, with one exception. This is similar to the feature importance of XGBoost (see Figure 6.2) in which months in service is ranked most important.

It can be seen that no feature attribution method for any of the classes aligns exactly with the feature importance of XGBoost itself (which is visualised in Figure 6.2). We can also see that for class 0 the bottom-ranked features in the XGBoost feature importance are also the bottom features for the methods of class 0. For the methods of class 3, one of those features is always ranked number 6 or lower. This shows that, while looking at feature attribution, one should not look only at the most important features, but also at others.

7.2.2 Relation between feature value and feature attribution values

As stated before feature attribution can also be applied to single observations, but examining many observations can be overwhelming. Besides looking at the average feature attributions, we can also generate beeswarm plots. These plots show for each feature, for each observation, what the attribution value is (on the x-axis) and what the normalised feature value is (shown in heat). This gives an idea of what the relation is between the feature values, and their associated game-theoretic solution values. For this process, the feature values were normalised (as in Equation 5.1), such that the heat of the beeswarm plot is not skewed towards certain variables. (The attribution values were not scaled as in Equation 7.1.) A high normalised feature value corresponds to a dark red colour, and a low normalised feature value corresponds to a dark blue colour. Figure 7.2 shows the beeswarm plot for the ACS concept for class 0, the other beeswarm plots (Figures E.2 - E.8) for class 0 and class 3 can be found in Appendix E.

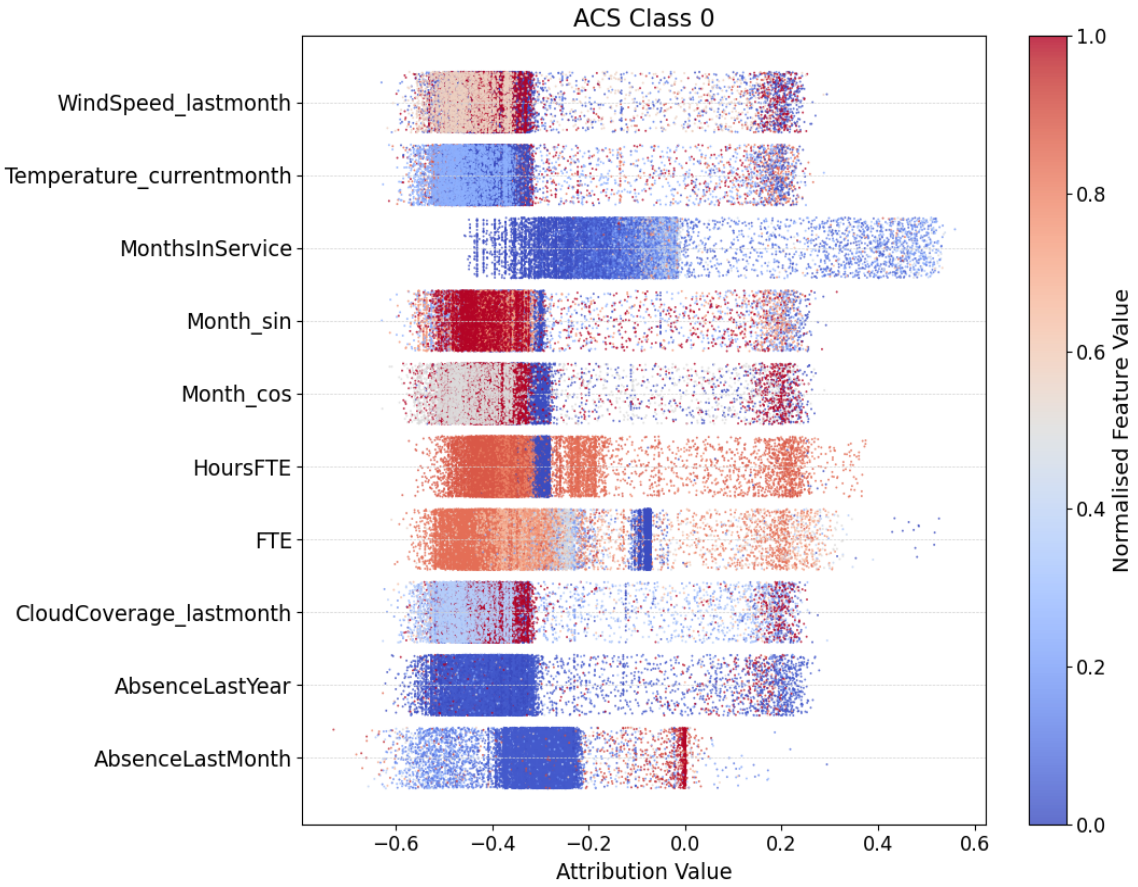

Figure 7.2: Beeswarm plot for the ACS solution of class 0, which shows on the x-axis the feature attribution for each observation, on the left y-axis the features and on the right y-axis the normalised feature value.

We would like to distinguish a relation between the feature value and its game-theoretic value. Any attribution method which has this relation can be considered more interpretable, since we can link the attribution value back to the feature value. We can find this relation for example by analysing the correlation between the normalised feature value and the corresponding feature attribution value. This way, we can link the feature attribution values back to the normalised data. According to (Akoglu, 2018) a (Pearson) correlation can be considered strong if it is higher than 0.7. Tables E.1 -E.8 show the correlations between the normalised feature value and the feature attribution value. Besides that, the mean and variance of each normalised feature value and of each feature attribution value are also included in those tables to provide some more background on the distribution of the normalised feature values and feature attribution values. Table 7.10 shows the number of times the absolute value of the correlation between the attribution value and the normalised feature value was greater or equal than 0.7, for each attribution method, for classes 0 and 3.

	Class 0	**Class 3**
ACS	0	1
Banzhaf	3	1
Nucleolus	2	0
Shapley	3	1

Table 7.10: Number of times the absolute value of the correlation between the attribution value and the normalised feature value was greater or equal than 0.7

The Banzhaf and Shapley methods have the highest number of strong correlations. So the Shapley and

Banzhaf value are the best in generating attribution values that link back to the feature values. This could make them more preferred over the other concepts.

We will now list which variables had strong correlations, the signs of those correlations and the importance rank of those variables (see Table 7.8):

- ACS class 3: Absence last month (negative, rank 9).
- Banzhaf class 0: Absence last month (negative, rank 2), FTE (negative, rank 3), Month cos (negative, rank 5).
- Banzhaf class 3: Absence last month (positive, rank 7).
- Nucleolus class 0: Absence last month (negative rank 2), FTE (negative, rank 3).
- Shapley class 0: Absence last month (negative, rank 2), FTE (negative, rank 3), Month cos (negative, rank 5).
- Shapley class 3: Absence last month (positive, rank 3).

For the Banzhaf value, nucleolus and Shapley value, when these strong correlations occur, they imply that a higher absence last month results in a lower feature attribution value for class 0 and a higher feature attribution value for class 3. This means that if someone is more absent in the last month, it increases the probability of them being absent again (ceteris paribus) in the average of our XGBoost models. This seems logical, firstly, because being absent once might lead to an absence that covers the end of a month and the beginning of the next. Secondly, because if someone is sick for an entire month for example due to a burnout, there is a high chance that he remains sick the next month. Similarly, the strong correlations imply that a higher FTE decreases the attribution value in class 0. This means that, for the Banzhaf value, nucleolus and Shapley value, a higher FTE decreases the probability of class (not absent) being predicted (ceteris paribus) in the average of our XGBoost models. Which can also be logically explained. Firstly, if someone has a lower FTE there is a lower chance that he is absent during work hours. Secondly, as is highlighted by papers in the literature review (Chapter 2.2), a higher FTE could lead to more stress and therefore to absence.

It should be noted that the Month cos feature represents a cosine transformation of the month (as discussed in Chapter 5). Because of this transformation, its values are not easy to interpret directly. While we can see correlations, understanding exactly how Month cos affects predictions is less straightforward than for the other features.

For the ACS concept, the only strong correlation that occurs is negative for absence last month. This implies that a higher absence last month decreases the feature attribution value of the ACS concept. If the ACS values decrease then the coalitional values decrease compared to the value of the grand coalition. This means that if someone is more absent in the last month, it decreases the probability of them not being absent this month (ceteris paribus) in the average of our XGBoost models. The same reasoning as above can be used to explain why this is logical.

There is no clear relation between the feature rank and the correlation.

Chapter 8

Conclusions and discussion

This thesis used an XGBoost multiclassification model to predict absenteeism. For this prediction model compliance with the EU AI regulations (European Data Protection Supervisor, 2025) was kept in mind. It introduced a new game-theoretic concept designed for feature attribution, the average coalitional surplus (ACS). The ACS concept is an intuitive approach that measures, for each observation, how far coalitional values are, on average, from the value of the grand coalition (the prediction of a test observation using all features). Finally, the ACS concept, Banzhaf value, nucleolus and Shapley value were all used for feature attribution to see whether they would yield different feature attributions, and if so, how they would differ. To investigate this, the research questions from Chapter 1 will now be answered.

1. *How well do machine learning models perform at predicting absenteeism while complying with the EU AI regulations?*
   The variable selection was done in such a way that the models are compliant with the EU AI regulations, if implemented properly. The resulting XGBoost model for the multiclassification task of predicting absenteeism achieved a higher accuracy (0.8850) than the SVM model (0.8666) or always predicting class 0 (0.8612). Although the XGBoost model was rarely able to predict classes 1 and 2 correctly, it was able to predict classes 0 and 3 correctly relatively often. Despite the improvement over the baseline, the gain of only 2.38 percentage points in accuracy is relatively small. This may not justify the effort of selecting variables and building a complex model such as XGBoost. It may therefore also not be worth allocating resources to implementing this model. Alternative approaches could potentially yield better results. For example, as discussed in Chapter 2.1, Neural Networks performed well compared to other methods in absenteeism prediction. Additionally, exploring different sets of variables could further improve the performance.

Now for the feature attribution using game theory. First, feature attribution assumes that a prediction is true. Since in our case the XGBoost model (often) fails to predict classes 1 and 2 correctly, interpreting these predictions does not provide useful information. So in general, we only want to explain predictions if they are often correct, because only in that case are the explanations meaningful for understanding the model's "decisions".

2. *Do different game-theoretic solution concepts produce the same feature importance rankings?*
   Each feature attribution method yields the same ranking for class 0 (with the exception of one swap for the ACS concept compared to the other methods), but for class 3 all concepts result in different rankings compared to each other and to class 0. Despite these differences, the feature months in service is consistently ranked first or second across all concepts for both classes 0 and 3. The differences in feature rankings are due to each game-theoretic concept having its own objective. The rankings of the feature attribution methods did not follow the ranking of the XGBoost feature importance. Therefore, when doing feature attribution, all features need to be taken into account, not just a few of the most important features. Otherwise, valuable information may be missed.

   A noteworthy observation that became clear during the comparison of the different feature attribution methods is that both the Shapley value and Banzhaf value showed a situation in which the attribution values of all features were negative. This gives a less intuitive interpretation, since all negative marginal contributions rarely appear in examples. However, it is mathematically possible for all marginal contributions to be negative.

3. *For each game-theoretic concept, is there a relation between the feature value and its attribution value?*
   To answer this research question, we looked for strong correlations between the feature attribution values and the normalised feature values. The Shapley and Banzhaf methods showed the highest number of strong correlations. The variable months in service showed the highest number of strong correlations, followed by FTE, and finally month cosinus. For months in service and FTE, the strong correlations yielded a logical relation between those features and their feature values.

This thesis discussed game-theoretic feature attribution methods for multiclass classification. It showed that new feature attribution methods can provide additional insights. There is no clear feature ranking if different classes are compared, or when different game-theoretic concepts are compared for class 3. Except for the feature months in service, which is almost always ranked first, with only one instance where it is ranked second. While the initial goal of this thesis was to find the best game-theoretic feature attribution method, it became clear that the results of a feature attribution method depend on the objective of the game-theoretic concept. However, if one game-theoretic concept had to be recommended for feature attribution, it would be the Shapley value because it has already been researched extensively and has applicable packages for machine learning models. The Shapley value (together with the Banzhaf value) also returned the highest number of strong correlations between a feature's normalised feature value and its game-theoretic value.

Chapter 9

Recommendations for future work

Several parts of this thesis could be explored further in future research, either by investigating them in more depth or by testing alternative approaches. We will first discuss opportunities related to predicting absenteeism, followed by opportunities regarding game-theoretic feature attribution.

Firstly, due to the inclusion of weather data, only employees who live in the Netherlands were used. Further research could expand this to include more countries, or see whether there is different behaviour for absenteeism in different countries.

Secondly, further research could investigate how including other variables, which were discussed in Chapter 2.2, would affect the model accuracy. Including other variables could lead to an increase in accuracy of classes 1 and 2. An example of an interesting variable to include would be job type (physical or not). Additionally, models such as Neural Networks or different weights to deal with class imbalance could be explored to potentially improve performance.

It would also be interesting to investigate how interventions impact absenteeism, or to develop a model that selects an intervention method based, for example, on feature attribution.

Further research could also look into how the EU AI Act impacts the predictive power of models that predict absenteeism. For example, by comparing a model that follows the EU AI guidelines and a model that does not. (However, a model that does not follow the EU AI Act (European Data Protection Supervisor, 2025) may only be used for research purposes, since this is allowed by the EU AI Act. Implementing such a model is not allowed.)

With regard to game theory for feature attribution, it could be interesting to expand to even more game-theoretic concepts, for example, by including the compromise value (Borm et al., 1992). However, when using other game-theoretic concepts, it is important to check whether they are applicable to the TU game based on a model's coalitional values. The ACS concept could also be further expanded or modified.

Feature research could also look into what game-theoretic concepts tend to produce more feature attribution values that are outliers. This can, for example, be done by comparing the variances of feature attribution values.

Extending the research of (Černý, 2022) and finding more approximation methods for the calculation of the coalitional values would also be very useful. The number of coalitions grows exponentially, and thus it is hard to analyse more complex models with many features. (The package SHAP (Lundberg, 2018) already approximates the Shapley values, but does not approximate the coalitional values.)

Finally, it would be interesting to investigate feature attribution for all features (instead of only the 10 that were selected). This is particularly relevant, as it became clear that the bottom-ranked features of the XGBoost feature importance were not always in the bottom rankings of the feature attribution methods.

Chapter 10

Recommendations for Crowe Foederer

This thesis can be used to predict absenteeism and build a robust model to account for spikes in absenteeism based on the predictions. However, this thesis is intended as research on absenteeism and should not be used as the sole basis for staff decisions. Models like absenteeism should comply with the relevant regulations, such as the EU AI Act (European Data Protection Supervisor, 2025). While this thesis has made an effort to consider regulatory requirements, compliance is not guaranteed, as no legal experts were consulted during its writing process.

Moreover, ethical and legal considerations must be taken into account when deploying models with HR data. We recommend keeping a close eye on machine learning models that use HR data as is required by (European Data Protection Supervisor, 2025). All the variables used must be checked to see whether they are related to a non-used variable that introduces discrimination (such as age, gender, ethnicity, etc.). With these considerations in mind, we can now move on to the findings of this thesis.

The results show that the model can predict when someone is never absent or absent the entire month, but it fails to predict classes in between.

To interpret the predictions to comply with the EU AI ACT (European Data Protection Supervisor, 2025), feature attribution methods are recommended. Before doing feature attribution, we would recommend thinking about what the goal of feature attribution is (finding the marginal contribution, finding the average difference from the value of the grand coalition or minimising the maximum dissatisfaction of all coalitions, or possibly something else). Among the tested methods, the Shapley and Banzhaf values had the highest number of strong correlations between feature values and their attribution values. Therefore, using one of those methods can give the clearest relations between feature values and predictions. Since the Shapley value already has an easily implementable package (SHAP (Lundberg, 2018)), and has already been extensively researched for feature attribution, we recommend using the Shapley value for feature attribution. It is important to note, however, that feature attributions do not imply causation as (Lundberg, 2018) explains.

Chapter 11

AI usage

Throughout this thesis, the following generative AIs were used: Grammarly for correcting grammar and spelling mistakes and enhancing readability, and ChatGPT for removing errors from code and brainstorming.

References

AccuWeather. (2025). *Weer utrecht per maand.* Retrieved from https://www.accuweather.com/nl/nl/de-bilt/250542/july-weather/250542?year=2025 (Accessed: 2025-06-10)

Ajmi, S. Q. (2019, 12). "Predicting absenteeism at work using machine learning algorithms. *Muthanna Journal of Pure Science*, *7*(1), 1–12. Retrieved from https://doi.org/10.52113/2/07.01.2020/1-12

Akoglu, H. (2018). User's guide to correlation coefficients. *Turkish Journal of Emergency Medicine*, *18*(3), 91-93. Retrieved from https://doi.org/10.1016/j.tjem.2018.08.001

Akyeampong, E. B. (2007). Trends and seasonality in absenteeism. *Perspectives on Labour and Income*, *19*(3), 13–15. Retrieved from https://www.proquest.com/docview/213990800?fromopenview=true&pq-origsite=gscholar&sourcetype=Scholarly$\%$20Journals

Anand, A., Pugalenthi, G., Fogel, G. B., & Suganthan, P. N. (2010, 4). An approach for classification of highly imbalanced data using weighting and undersampling. *Amino Acids*, *39*(5), 1385–1391. Retrieved from https://doi.org/10.1007/s00726-010-0595-2

Asghar, Z. B., Wankhade, P., Bell, F., Sanderson, K., Hird, K., Phung, V.-H., & Siriwardena, A. N. (2021, 9). Trends, variations and prediction of staff sickness absence rates among NHS ambulance services in England: a time series study. *BMJ Open*, *11*(9), e053885. Retrieved from https://doi.org/10.1136/bmjopen-2021-053885

Aumann, R. J., & Maschler, M. (1985). Game theoretic analysis of a bankruptcy problem from the talmud. *Journal of Economic Theory*, *36*(2), 195-213. Retrieved from https://doi.org/10.1016/0022-0531(85)90102-4

Autoriteit Persoonsgegevens. (2020). *Belatingdienst/ toeslagen: De verwerking van de nationaliteit van aanvragers van kinderopvangtoeslag* (Tech. Rep.). Autoriteit Persoonsgegevens. Retrieved from https://www.autoriteitpersoonsgegevens.nl/uploads/imported/onderzoek_belastingdienst_kinderopvangtoeslag.pdf

Bain, L., & Engelhardt, M. (1992). *Introduction to probability and mathematical statistics*. Brooks/Cole Cengage Learning. Retrieved from https://books.google.nl/books?id=MkFRIAAACAAJ

Bakırarar, B., & Elhan, A. H. (2023, 01). Class weighting technique to deal with imbalanced class problem in machine learning: Methodological research. *Turkiye Klinikleri Journal of Biostatistics*, *15*, 19-29. Retrieved from https://doi.org/10.5336/biostatic.2022-93961

Banzhaf, J. (1965). Weighted voting doesn't work: A mathematical analysis. *Rutgers Law Review*, *19*(2), 317-343.

Benedek, M., Fliege, J., & Nguyen, T.-D. (2020, 6). Finding and verifying the nucleolus of cooperative games. *Mathematical Programming*, *190*(1-2), 135–170. Retrieved from https://doi.org/10.1007/s10107-020-01527-9

Bennett, D. A. (2001, 10). How can I deal with missing data in my study? *Australian and New Zealand Journal of Public Health*, *25*(5), 464–469. Retrieved from https://doi.org/10.1111/j.1467-842x.2001.tb00294.x

Bhattacharyya, A., Gayen, S., Meel, K. S., Myrisiotis, D., Pavan, A., & Vinodchandran, N. V. (2023, 8). On approximating total variation distance. In E. Elkind (Ed.), *Proceedings of the thirty-second international joint conference on artificial intelligence, IJCAI-23* (pp. 3479–3487). International Joint Conferences on Artificial Intelligence Organization. Retrieved from https://doi.org/10.24963/ijcai.2023/387 (Main Track)

Bhui, K. S., Dinos, S., Stansfeld, S. A., & White, P. D. (2012, 2). A Synthesis of the Evidence for Managing Stress at Work: A review of the reviews reporting on anxiety, depression, and absenteeism. *Journal*

of Environmental and Public Health, *2012*, 1–21. Retrieved from https://doi.org/10.1155/2012/515874

Björnfot, A., & Fjelkestam, S. (2023). *Predicting Short-term Absences of a Railway Crew using Historical Data* (Master's thesis, KTH Royal Institute of Technology). Retrieved from https://kth.diva-portal.org/smash/record.jsf?pid=diva2%3A1761858

Borm, P. (2024). *Games, cooperative behaviour and economics.* Retrieved from https://tilburguniversity.instructure.com/courses/13236/files/2505569?module_item_id=552129 (Course reader for course: Games and Cooperative Behavior.)

Borm, P., Keiding, H., McLean, R. P., Oortwijn, S., & Tijs, S. (1992, 6). The compromise value for NTU-games. *International Journal of Game Theory*, *21*(2), 175–189. Retrieved from https://doi.org/10.1007/bf01245460

Boudiaf, M., Rony, J., Masud, Z. I., Granger, E., Pedersoli, M., Piantanida, P., & Ben Ayed, I. (2020). *A unifying mutual information view of metric learning: Cross-entropy vs. pairwise losses.* Retrieved from https://doi.org/10.48550/arXiv.2003.08983

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization.* Cambridge, UK: Cambridge University Press. Retrieved from https://doi.org/10.1017/cbo9780511804441 (Seventh printing with corrections, 2009)

Brown, J. D. (2014, 3). *Verification of temperature, precipitation and streamflow forecasts from the hydrologic ensemble forecast service (hefs) of the u.s. national weather service: an evaluation of the medium-range forecasts with forcing inputs from ncep's global ensemble forecast system (gefs) and a comparison to the frozen version of ncep's global forecast system (gfs)* (Tech. Rep. Nos. Agreement 2013-09, Deliverable No. 2, Phase III, Final). Hydrologic Solutions Limited, prepared for U.S. National Weather Service Office of Hydrologic Development. Retrieved from https://www.weather.gov/media/owp/oh/hrl/docs/Contract_2013-09-HEFS_Deliverable_02_Phase_III_report_FINAL.pdf

Burton, W. N., Chen, C.-Y., Conti, D. J., Pransky, G., & Edington, D. W. (2004, 10). Caregiving for Ill Dependents and Its Association with Employee Health Risks and Productivity. *Journal of Occupational and Environmental Medicine*, *46*(10), 1048–1056. Retrieved from https://doi.org/10.1097/01.jom.0000141830.72507.32

Caruso, C. C., Hitchcock, E. M., Dick, R. B., Russo, J. M., & Schmit, J. M. (2004, 4). *Overtime and extended work shifts: recent findings on illnesses, injuries and health behaviors.* (Tech. Rep. No. 2004-143). CDC: Centers for Disease Control and Prevention. Retrieved from https://doi.org/10.26616/nioshpub2004143

Centraal Bureau voor de Statistiek. (2015, 3). *Sickness absence rate dips to lowest level since 1996.* Retrieved from https://www.cbs.nl/en-gb/news/2015/13/sickness-absence-rate-dips-to-lowest-level-since-1996 (Accessed: 2025-04-30)

Centraal Bureau voor de Statistiek. (2025). *Ziekteverzuim.* Retrieved from https://www.cbs.nl/nl-nl/visualisaties/dashboard-arbeidsmarkt/werkenden/ziekteverzuim (Accessed: 2025-04-30)

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002, 6). SMOTE: Synthetic Minority Over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. Retrieved from https://doi.org/10.1613/jair.953

Chen, H., Covert, I. C., Lundberg, S. M., & Lee, S.-I. (2023, 5). Algorithms to estimate Shapley value feature attributions. *Nature Machine Intelligence*, *5*(6), 590–601. Retrieved from https://doi.org/10.1038/s42256-023-00657-x

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (p. 785–794). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2939672.2939785

Coleman, D. F., & Schaefer, N. V. (1990, 12). Weather and absenteeism. *Canadian Journal of Administrative Sciences / Revue Canadienne des Sciences de l Administration*, *7*(4), 35–42. Retrieved from https://doi.org/10.1111/j.1936-4490.1990.tb00540.x

Condevaux, C., Harispe, S., & Mussard, S. (2023). Fair and efficient alternatives to shapley-based attribution methods. In M.-R. Amini, S. Canu, A. Fischer, T. Guns, P. Kralj Novak, & G. Tsoumakas (Eds.), *Machine learning and knowledge discovery in databases* (pp. 309–324). Cham: Springer International Publishing. Retrieved from https://doi.org/10.1007/978-3-031-26387-3_19

Cortes, C., & Vapnik, V. (1995, 9). Support-vector networks. *Machine Learning*, *20*(3), 273–297. Retrieved from https://doi.org/10.1007/bf00994018

Covert, I., Lundberg, S., & Lee, S.-I. (2020, 11). *Explaining by removing: A Unified framework for model explanation.* Retrieved from https://doi.org/10.48550/arXiv.2011.14878

Demou, E., Smith, S., Bhaskar, A., Mackay, D. F., Brown, J., Hunt, K., ... Macdonald, E. B. (2018, 1). Evaluating sickness absence duration by musculoskeletal and mental health issues: a retrospective cohort study of Scottish healthcare workers. *BMJ Open*, *8*(1), e018085. Retrieved from https://doi.org/10.1136/bmjopen-2017-018085

Dhamal, S., Bhat, S., Anoop, K., & Embar, V. R. (2012, 1). *Pattern Clustering using Cooperative Game Theory.* Retrieved from https://doi.org/10.48550/arXiv.1201.0461

Dillon, E., LaRiviere, J., Lundberg, S., Roth, J., & Syrgkanis, V. (2018). *Be careful when interpreting predictive models in search of causal insights.* Retrieved from https://shap.readthedocs.io/en/latest/example_notebooks/overviews/Be%20careful%20when%20interpreting%20predictive%20models%20in%20search%20of%20causal%20insights.html#Be-careful-when-interpreting-predictive-models-in-search-of-causal%C2%A0insightsl

Dionne, G., & Dostie, B. (2007). New evidence on the determinants of absenteeism using linked employer-employee data. *Industrial and Labor Relations Review*, *61*(1), 108–120. Retrieved from http://www.jstor.org/stable/25249126

DMLC XGBoost Contributors. (2023). *Xgboost source code (expand entry h).* Retrieved from https://github.com/dmlc/xgboost/blob/master/src/tree/hist/expand_entry.h (Accessed: 2025-06-24)

Dogruyol, K., & Sekeroglu, B. (2020). Absenteeism prediction: A comparative study using machine learning models. In R. A. Aliev, J. Kacprzyk, W. Pedrycz, M. Jamshidi, M. B. Babanli, & F. M. Sadikoglu (Eds.), *10th international conference on theory and application of soft computing, computing with words and perceptions - icsccw-2019* (pp. 728–734). Cham: Springer International Publishing. Retrieved from https://doi.org/10.1007/978-3-030-35249-3_94 (Advances in Intelligent Systems and Computing, vol 1095. Springer, Cham.)

Doomen, J. (2010). *De hollende kleurling: het nederlandse strafrecht in tien verhalen.* Atlas Contact. Retrieved from https://books.google.nl/books?id=R9AvAgAAQBAJ

European Data Protection Supervisor. (2025). *Ai act regulation (eu) 2024/1689 – regulation (eu) 2024/1689 of the european parliament and of the council of 13 june 2024 laying down harmonised rules on artificial intelligence and amending regulations (ec) no 300/2008, (eu) no 167/2013, (eu) no 168/2013, (eu) 2018/858, (eu) 2018/1139 and (eu) 2019/2144 and directives 2014/90/eu, (eu) 2016/797 and (eu) 2020/1828 (artificial intelligence act) (text with eea relevance).* Publications Office of the European Union. Retrieved from https://data.europa.eu/doi/10.2804/4225375

Fan, Y., & van den Dool, H. (2011, 6). Bias correction and forecast skill of NCEP GFS ensemble Week-1 and Week-2 precipitation, 2-m surface air temperature, and soil moisture forecasts. *Weather and Forecasting*, *26*(3), 355–370. Retrieved from https://doi.org/10.1175/waf-d-10-05028.1

Fang, Q., Li, B., Sun, X., Zhang, J., & Zhang, J. (2016, 1). Computing the least-core and nucleolus for threshold cardinality matching games. *Theoretical Computer Science*, *609*, 500-510. Retrieved from https://doi.org/10.1016/j.tcs.2015.11.015

Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters*, *27*(8), 861-874. Retrieved from https://doi.org/10.1016/j.patrec.2005.10.010 (ROC Analysis in Pattern Recognition)

Fisman, D., Postma, M., Levin, M. J., & Mould-Quevedo, J. (2024, 12). Absenteeism and productivity loss due to influenza or influenza-like illness in adults in Europe and North America. *Diseases*, *12*(12), 331. Retrieved from https://doi.org/10.3390/diseases12120331

Gielen, W., & Ilham, M. (2024, 7). *Ziekteverzuim naar bedrijfstak: ontwikkelingen en verschillen.* Retrieved from https://www.cbs.nl/nl-nl/longread/statistische-trends/2024/ziekteverzuim-naar-bedrijfstak-ontwikkelingen-en-verschillen?onepage=true

Gillies, D. B. (1953). *Some theorems on n-person games* (Unpublished doctoral dissertation). Princeton University, Princeton, New Jersey.

Goda, G. S., & Soltas, E. J. (2023, 4). The impacts of Covid-19 absences on workers. *Journal of Public Economics*, *222*, 104889. Retrieved from https://doi.org/10.1016/j.jpubeco.2023.104889

Gosselin, E., Lemyre, L., & Corneil, W. (2012, 12). Presenteeism and absenteeism: Differentiated understanding of related phenomena. *Journal of Occupational Health Psychology*, *18*(1), 75–86.

Retrieved from https://doi.org/10.1037/a0030932
Grigoryan, G. (2024). *Explainable artificial intelligence: Methods and evaluation* (Ph.D. Dissertation, Old Dominion University, Norfolk, VA, USA). Retrieved from https://doi.org/10.25777/xy0g-x273 (Engineering Management & Systems Engineering)
Grinsztajn, L., Oyallon, E., & Varoquaux, G. (2022, 7). *Why do tree-based models still outperform deep learning on tabular data?* Retrieved from https://doi.org/10.48550/arXiv.2207.08815
Guajardo, M., & Jörnsten, K. (2014, 4). *Common mistakes in computing the nucleolus* (Tech. Rep.). Retrieved from https://openaccess.nhh.no/nhh-xmlui/bitstream/handle/11250/194983/1514.pdf?sequence=1
Hafner, M., Van Stolk, C., Saunders, C. L., Krapels, J., & Baruch, B. (2015, 1). *Health, wellbeing and productivity in the workplace.* RAND. Retrieved from https://doi.org/10.7249/RR1084 (Retrieved 07-02-2025)
Haimanko, O. (2019, 1). Composition independence in compound games: a characterization of the Banzhaf power index and the Banzhaf value. *International Journal of Game Theory*, *48*(3), 755–768. Retrieved from https://doi.org/10.1007/s00182-019-00660-w
Hassink, W. (2018, 1). How to reduce workplace absenteeism. *IZA World of Labor*. Retrieved from https://doi.org/10.15185/izawol.447
Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer New York, NY. Retrieved from https://doi.org/10.1007/978-0-387-84858-7
Hillier, F. S., & Lieberman, G. J. (2010). *Introduction to operations research* (9th ed.). McGraw-Hill.
Ho, Y., & Wookey, S. (2019, 12). The Real-World-Weight Cross-Entropy Loss Function: Modeling the costs of mislabeling. *IEEE Access*, *8*, 4806–4813. Retrieved from https://doi.org/10.1109/access.2019.2962617
Hoevenberg, M. (2021). *Predicting workplace absenteeism by use of personal and employee data* (Master's thesis). Tilburg University, Tilburg. (Available at https://arno.uvt.nl/show.cgi?fid=160975)
Il Idrissi, M., Fernandes Machado, A., Gallic, E., & Charpentier, A. (2025). *Unveil sources of uncertainty: Feature contribution to conformal prediction intervals.* Retrieved from https://doi.org/10.48550/arXiv.2505.13118
Iñarra, E., Serrano, R., & Shimomura, K.-I. (2020). The nucleolus, the kernel, and the bargaining set: an update. *Revue économique*, *71*(2), 225–266. Retrieved from https://doi.org/10.3917/reco.712.0225
Johnson, J. M., & Khoshgoftaar, T. M. (2019, 3). Survey on deep learning with class imbalance. *Journal Of Big Data*, *6*(27). Retrieved from https://doi.org/10.1186/s40537-019-0192-5
Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, *30*(1/2), 81–93. Retrieved from https://doi.org/10.2307/2332226
Kocakulah, M. C., Kelley, A. G., Mitchell, K. M., & Ruggieri, M. P. (2016, 5). Absenteeism Problems and costs: Causes, effects and cures. *International Business & Economics Research Journal (IBER)*, *15*(3), 89–96. Retrieved from https://doi.org/10.19030/iber.v15i3.9673
Kohlberg, E. (1971, 1). On the Nucleolus of a Characteristic Function Game. *SIAM Journal on Applied Mathematics*, *20*(1), 62–66. Retrieved from https://doi.org/10.1137/0120009
Koninklijk Nederlands Meteorologisch Instituut. (2023, 6). *Veelgestelde vragen over landelijk gemiddelden.* Retrieved from https://www.knmi.nl/kennis-en-datacentrum/achtergrond/veelgestelde-vragen-over-landelijk-gemiddelden (Accessed: 2025-04-15)
Koninklijk Nederlands Meteorologisch Instituut. (2025). *Daggegevens van het weer in nederland.* Retrieved from https://www.knmi.nl/nederland-nu/klimatologie/daggegevens (Accessed: 2025-04-15)
Kulynych, B., & Troncoso, C. (2017, 11). *Feature Importance Scores and Lossless Feature Pruning Using Banzhaf Power Indices.* Retrieved from https://doi.org/10.48550/arXiv.1711.04992
Lawrance, N., Petrides, G., & Guerry, M.-A. (2021, 2). Predicting employee absenteeism for cost effective interventions. *Decision Support Systems*, *147*, 113539. Retrieved from https://doi.org/10.1016/j.dss.2021.113539
Lima, E., Vieira, T., & De Barros Costa, E. (2020, 3). Evaluating deep models for absenteeism prediction of public security agents. *Applied Soft Computing*, *91*, 106236. Retrieved from https://doi.org/10.1016/j.asoc.2020.106236

Liu, Y., Witter, R. T., Korn, F., Alrashed, T., Paparas, D., & Freire, J. (2024, 10). *Kernel Banzhaf: A Fast and Robust Estimator for Banzhaf Values.* Retrieved from https://doi.org/10.48550/arXiv.2410.08336

Lumintu, I., & Maududie, A. (2025, 1). Supporting Sustainable Workforce Management for Worker Illness Absence Through Predictive Analytics. *Engineering Proceedings*, *84*(1), 17. Retrieved from https://doi.org/10.3390/engproc2025084017

Lundberg, S. M. (2018). *An introduction to explainable AI with Shapley values — SHAP latest documentation.* Retrieved from https://shap.readthedocs.io/en/latest/example_notebooks/overviews/An%20introduction%20to%20explainable%20AI%20with%20Shapley%20values.html

Lundberg, S. M., Erion, G. G., & Lee, S.-I. (2018, 1). *Consistent individualized feature attribution for tree ensembles.* Retrieved from https://doi.org/10.48550/arxiv.1802.03888

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems*, *30*, 4765–4774. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf (Can also be retrieved from https://doi.org/10.48550/arXiv.1705.07874 with some very minor differences.)

Lyons, T. F. (1972, 6). Turnover and absenteeism: A review of relationships and shared correlates. *Personnel Psychology*, *25*(2), 271–281. Retrieved from https://doi.org/10.1111/j.1744-6570.1972.tb01103.x

Maestas, N. A., Mullen, K. J., & Rennane, S. (2020, 6). Absenteeism and presenteeism among American workers. *Journal of Disability Policy Studies*, *32*(1), 13–23. Retrieved from https://doi.org/10.1177/1044207320933211

Mahajan, T., Singh, G., & Bruns, G. (2021, 3). An experimental assessment of treatments for cyclical data. In *Proceedings of the 2021 computer science conference for csu undergraduates.* Retrieved from https://cscsu-conference.github.io/index.html

Mainar, I. G., Green, C. P., & Paniagua, M. N. (2017, 6). The Effect of Permanent Employment on Absenteeism: Evidence from Labor Reform in Spain. *ILR Review*, *71*(2), 525–549. Retrieved from https://doi.org/10.1177/0019793917717226

Mao, A., Mohri, M., & Zhong, Y. (2023). *Cross-Entropy Loss Functions: Theoretical analysis and applications.* Retrieved from https://doi.org/10.48550/arXiv.2304.07288

Markham, S. E., & Markham, I. S. (2005, 1). Biometeorological effects on worker absenteeism. *International Journal of Biometeorology*, *49*(5), 317–324. Retrieved from https://doi.org/10.1007/s00484-004-0246-y

Markussen, S., & Røed, K. (2015, 1). Daylight and absenteeism – Evidence from Norway. *Economics & Human Biology*, *16*, 73–80. Retrieved from https://doi.org/10.1016/j.ehb.2014.01.002

Miller, J. (2016, 9). The well-being and productivity link: a significant opportunity for research-into-practice. *Journal of Organizational Effectiveness People and Performance*, *3*(3), 289–311. Retrieved from https://doi.org/10.1108/joepp-07-2016-0042

Mishra, N., Prodhomme, C., & Guemas, V. (2018, 8). Multi-model skill assessment of seasonal temperature and precipitation forecasts over Europe. *Climate Dynamics*, *52*(7-8), 4207–4225. Retrieved from https://doi.org/10.1007/s00382-018-4404-z

Molnar, C. (2020). *Interpretable machine learning: A guide for making black box models explainable.* Leanpub. Retrieved from https://books.google.nl/books?id=jBm3DwAAQBAJ

Nath, G., Wang, Y., Coursey, A., Saha, K. K., Prabhu, S., & Sengupta, S. (2022, 6). Incorporating a machine learning model into a web-based administrative decision support tool for predicting workplace absenteeism. *Information*, *13*(7). Retrieved from https://doi.org/10.3390/info13070320

Nauta, M., Trienes, J., Pathak, S., Nguyen, E., Peters, M., Schmitt, Y., ... Seifert, C. (2023, 7). From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai. *ACM Comput. Surv.*, *55*(13s). Retrieved from https://doi.org/10.1145/3583558

Notenbomer, A., van Rhenen, W., Groothoff, J. W., & Roelen, C. A. M. (2018, 11). Predicting long-term sickness absence among employees with frequent sickness absence. *International Archives of Occupational and Environmental Health*, *92*(4), 501–511. Retrieved from https://doi.org/10.1007/s00420-018-1384-6

Nunes, A. P., Richmond, M. K., Pampel, F. C., & Wood, R. C. (2017, 10). The effect of employee assistance services on reductions in employee absenteeism. *Journal of Business and Psychology*,

33, 699–709. Retrieved from https://doi.org/10.1007/s10869-017-9518-5

Olsen, L. H. B., Glad, I. K., Jullum, M., & Aas, K. (2024, 3). A comparative study of methods for estimating model-agnostic Shapley value explanations. *Data Mining and Knowledge Discovery*, *38*, 1782–1829. Retrieved from https://doi.org/10.1007/s10618-024-01016-z

O'Neill, B. (1982). A problem of rights arbitration from the talmud. *Mathematical Social Sciences*, *2*(4), 345-371. Retrieved from https://doi.org/10.1016/0165-4896(82)90029-4

Pauly, M. V., Nicholson, S., Xu, J., Polsky, D., Danzon, P. M., Murray, J. F., & Berger, M. L. (2002, 3). A general model of the impact of absenteeism on employers and employees. *Health Economics*, *11*(3), 221–231. Retrieved from https://doi.org/10.1002/hec.648

Piha, K., Laaksonen, M., Martikainen, P., Rahkonen, O., & Lahelma, E. (2012, 12). Socio-economic and occupational determinants of work injury absence. *European Journal of Public Health*, *23*(4), 693–698. Retrieved from https://doi.org/10.1093/eurpub/cks162

Quant, M., Borm, P., Reijnierse, J., & van Velzen, S. (2003). *Compromise stable tu-games* (Discussion Paper No. 2003-55). CentER for Economic Research, Tilburg University. Retrieved from https://research.tilburguniversity.edu/en/publications/compromise-stable-tu-games (Pagination: 23. JEL Classification: C71)

Rajath, J., & Pandita, D. (2022, 10). Machine learning as a data science tool to predict absenteeism for factory workers. In *2021 international conference on data analytics for business and industry (icdabi)* (pp. 261–265). Retrieved from https://doi.org/10.1109/icdabi56818.2022.10041623

Ranglani, H. (2024, 12). Empirical analysis of the Bias-Variance tradeoff across machine learning models. *Machine Learning and Applications: An International Journal (MLAIJ)*, *11*(4). Retrieved from https://doi.org/10.2139/ssrn.5086450

Raykar, V. C., & Saha, A. (2015). Data split strategies for evolving predictive models. In A. Appice, P. Rodrigues, V. Santos Costa, C. Soares, J. Gama, & A. Jorge (Eds.), *Machine learning and knowledge discovery in databases* (Vol. 9284, pp. 3–19). Springer, Cham. Retrieved from https://doi.org/10.1007/978-3-319-23528-8_1

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). *"Why Should I Trust You?": Explaining the Predictions of Any Classifier.* Retrieved from https://doi.org/10.48550/arXiv.1602.04938

Rijksinstituut voor Volksgezondheid en Milieu. (2025). *Feiten en cijfers griep.* Retrieved from https://www.rivm.nl/griep-griepprik/feiten-en-cijfers (Accessed: 2025-04-30)

Rodwell, M. J., & Doblas-Reyes, F. J. (2006, 12). Medium-Range, monthly, and seasonal prediction for Europe and the use of forecast information. *Journal of Climate*, *19*(23), 6025–6046. Retrieved from https://doi.org/10.1175/jcli3944.1

Ruder, S. (2016). *An overview of gradient descent optimization algorithms.* Retrieved from https://doi.org/10.48550/arXiv.1609.04747

Sagi, O., & Rokach, L. (2021). Approximating xgboost with an interpretable decision tree. *Information Sciences*, *572*, 522-542. Retrieved from https://doi.org/10.1016/j.ins.2021.05.055

Sakr, C. J., Fakih, L. M., Musharrafieh, U. M., Khairallah, G. M., Makki, M. H., Doudakian, R. M., ... Rahme, D. V. (2025, 1). Absenteeism among healthcare workers: job grade and other factors that matter in sickness absence. *International Journal of Environmental Research and Public Health*, *22*(1), 127. Retrieved from https://doi.org/10.3390/ijerph22010127

Schalk, R., & Van Rijckevorsel, A. (2007, 11). Factors influencing absenteeism and intention to leave in a call centre. *New Technology Work and Employment*, *22*(3), 260–274. Retrieved from https://doi.org/10.1111/j.1468-005x.2007.00198.x

Schmeidler, D. (1969). The nucleolus of a characteristic function game. *SIAM Journal on Applied Mathematics*, *17*(6), 1163–1170. Retrieved 2025-06-12, from https://doi.org/10.1137/0117107

scikit-learn developers. (2025a). *sklearn.preprocessing.minmaxscaler — scikit-learn documentation.* Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html (Accessed: 2025-07-18)

scikit-learn developers. (2025b). *sklearn.preprocessing.standardscaler — scikit-learn documentation.* Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html (Accessed: 2025-07-18)

scikit-learn developers. (2025c). *Svc — scikit-learn 1.7.1 documentation.* https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. (Accessed: 2025-07-21)

Shah, S. A. A., Uddin, I., Aziz, F., Ahmad, S., Al-Khasawneh, M. A., & Sharaf, M. (2020, 2). An enhanced deep neural network for predicting workplace absenteeism. *Complexity*, *2020*, 1–12. Retrieved from

https://doi.org/10.1155/2020/5843932
Shannon, C. E. (1948, 7). A mathematical theory of communication. *Bell System Technical Journal*, *27*(3), 379–423. Retrieved from https://doi.org/10.1002/j.1538-7305.1948.tb01338.x
Shapley, L. S. (1952). *A value for N-Person games.* Retrieved from https://doi.org/10.7249/p0295
Shapley, L. S., & Shubik, M. (1954, 9). A method for evaluating the distribution of power in a committee system. *American Political Science Review*, *48*(3), 787–792. Retrieved from https://doi.org/10.2307/1951053
Shekhar, S., Bansode, A., & Salim, A. (2021). A comparative study of hyper-parameter optimization tools. In *2021 ieee asia-pacific conference on computer science and data engineering (csde)* (p. 1-6). Retrieved from https://doi.org/10.1109/CSDE53843.2021.9718485
Shi, J., & Skuterud, M. (2014, 6). GONE FISHING! REPORTED SICKNESS ABSENTEEISM AND THE WEATHER. *Economic Inquiry*, *53*(1), 388–405. Retrieved from https://doi.org/10.1111/ecin.12109
Shi, Z., & O'Brien, W. (2019). Development and implementation of automated fault detection and diagnostics for building systems: A review. *Automation in Construction*, *104*, 215-229. Retrieved from https://doi.org/10.1016/j.autcon.2019.04.002
Sitarević, A., Tomašević, A. N., Sofić, A., Banjac, N., & Novaković, N. (2023, 4). The Psychosocial Model of Absenteeism: Transition from 4.0 to 5.0. *Behavioral Sciences*, *13*(4), 332. Retrieved from https://doi.org/10.3390/bs13040332
Skorikov, M., Hussain, M. A., Khan, M. R., Akbar, M. K., Momen, S., Mohammed, N., & Nashin, T. (2020). Prediction of absenteeism at work using data mining techniques. In *2020 5th international conference on information technology research (icitr)* (p. 1-6). Retrieved from https://doi.org/10.1109/ICITR51448.2020.9310913
Staudacher, J., & Anwander, J. (2019). *Using the r package coopgame for the analysis, solution and visualization of cooperative games with transferable utility.* Retrieved from https://cran.r-project.org/web/packages/CoopGame/vignettes/UsingCoopGame.pdf (R Vignette)
Staudacher, J., Anwander, J., Tiukkel, A., Maerz, M., Mueller, F., Gebele, D., ... Cyl, N. (2019, 3). *CoopGame: Important concepts of Cooperative Game Theory.* Retrieved from https://doi.org/10.32614/cran.package.coopgame
Tashman, L. J. (2000, 10). Out-of-sample tests of forecasting accuracy: an analysis and review. *International Journal of Forecasting*, *16*(4), 437–450. Retrieved from https://doi.org/10.1016/s0169-2070(00)00065-0
The weather channel. (2025). *Weersverwachting voor een maand- de bilt, utrecht.* Retrieved from https://weather.com/nl-NL/weer/monthly/l/a9d99021cdc93deb638663bb52edbabf2a609e61b3b2ecfde9c8ac49f36cd88e (Accessed: 2025-06-10)
van Ballegooijen, H., Goossens, L., Bruin, R. H., Michels, R., & Krol, M. (2021, 3). Concerns, quality of life, access to care and productivity of the general population during the first 8 weeks of the coronavirus lockdown in Belgium and the Netherlands. *BMC Health Services Research*, *21*(1). Retrieved from https://doi.org/10.1186/s12913-021-06240-7
Wahid, Z., Satter, A. K. M. Z., Al Imran, A., & Bhuiyan, T. (2019). Predicting absenteeism at work using tree-based learners. In *Proceedings of the 3rd international conference on machine learning and soft computing* (p. 7–11). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3310986.3310994
Weather Atlas. (2025). *July weather forecast de bilt, netherlands.* Retrieved from https://www.weather-atlas.com/en/netherlands/de-bilt-weather-july (Accessed: 2025-06-10)
Weer1.com. (2025). *Het weer in nederland in juli 2025.* Retrieved from https://www.weer1.com/europe/netherlands?page=month&month=July (Accessed: 2025-06-10)
Weer33. (2025). *Weer utrecht per maand.* Retrieved from https://www.weer33.nl/utrecht (Accessed: 2025-06-10)
Wu, T., & Weiland, C. (2024, 11). *Leveraging modern machine learning to improve early warning systems and reduce chronic absenteeism in early childhood* (No. 24-1081). Retrieved from https://doi.org/10.26300/xxvz-cv94
XGBoost developers. (2022). *Xgboost documentation — xgboost 3.0.2.* https://xgboost.readthedocs.io/en/release_3.0.0/. (Accessed: 2025-05-30)

Zhan, Z., & Kim, S.-K. (2024, 7). Versatile time-window sliding machine learning techniques for stock market forecasting. *Artificial Intelligence Review*, *57*(209). Retrieved from https://doi.org/10.1007/s10462-024-10851-x

Zheng, W., Ek, M., Mitchell, K., Wei, H., & Meng, J. (2017, 7). Improving the stable surface layer in the NCEP Global Forecast system. *Monthly Weather Review*, *145*(10), 3969–3987. Retrieved from https://doi.org/10.1175/mwr-d-16-0438.1

Zhou, H., Ren, J., Deng, H., Cheng, X., Zhang, J., & Zhang, Q. (2024, 1). Interpretability of neural networks based on game-theoretic interactions. *Machine Inteligence Research*, *21*(4), 718–739. Retrieved from https://doi.org/10.1007/s11633-023-1419-7

Zupančič, P., & Panov, P. (2024, 8). Predicting Employee Absence from Historical Absence Profiles with Machine Learning. *Applied Sciences*, *14*(16), 7037. Retrieved from https://doi.org/10.3390/app14167037

Černý, M. (2022, 12). *Approximations of solution concepts of cooperative games.* Retrieved from https://doi.org/10.48550/arxiv.2212.04748

Appendix

A Possible interventions for absenteeism

(Kocakulah et al., 2016) has a study which also gives an idea for possible interventions (also known as Employee Assistance Programs, or EAP in short) to reduce absenteeism. They reference another paper, that states six traits of highly successful EAPs:

- Provide counselling for issues such as mental health, alcohol or substance abuse.
- Maintain ongoing communication to encourage employees to use EAP services.
- Offer workshops covering topics such as nutrition, diet, quitting smoking, exercise and stress management.
- Provide supervisory training and management consultations.
- Provide resources or give referrals for personal or work-related issues.
- Offer legal and financial support services.

However, they also explain that they are expensive, and it is difficult to know how much they reduce absenteeism. If an intervention could be tailored to the employee's needs or recommended based on the employee's needs, it would increase the performance of the intervention. However, this is difficult due to the employee's privacy.

(Nunes et al., 2017) shows that absenteeism, specifically sick leave usage, for employees with access to EAP was lower compared to the employees without EAP access. However, they had only 290 employees to test it on. They also state that EAP services are more effective for employees who have moderate absenteeism levels than for those with chronic absenteeism.

(Hassink, 2018) gives different intervention methods to reduce workplace absenteeism. However, in that study, it is also argued that one should watch out and make the interventions not in such a way that employees force themselves to work when they are too sick to work. The proposed programs can be divided into a few categories. In the first category, the employee gets a positive reward for being less absent or having a higher performance, or they could be penalised for being more absent. However, this may push employees to work, even when they are sick. The second category has to do with improving working conditions and the employee's mental and physical health. The third category is loyalty. The final category is faster recovery; for example, returning partially to work. In (Hassink, 2018), the second and third categories are grouped together.

(Bhui et al., 2012) found that absenteeism is reduced by physical activities as an organisational intervention.

B Approximate split finding algorithm for XGBoost

ALGORITHM 5: Approximate split finding

Input : Labels and predictions $\{(y_i, \widehat{y}_i)\}_{i=1}^{n}$, instance set I, number of features K, split cost parameter γ, weight cost parameter λ, twice differentiable loss function $L(y_i, \widehat{y}_i)$, number of split bins V

Output: Best approximate split

for $k = 1, \ldots, K$ **do**
  Propose split points $S_k = \{s_{k1}, \ldots, s_{kV}\}$ by percentiles on feature k;
end
for $k = 1, \ldots, K$ **do**
  for $v = 1, \ldots, V$ **do**
    $G_{kv} = \sum_{j:x_{jk} > s_{kv}} \frac{\partial L(y_j, \widehat{y}_j)}{\partial \widehat{y}_j}$;
    $H_{kv} = \sum_{j:x_{jk} > s_{kv}} \frac{\partial^2 L(y_j, \widehat{y}_j)}{\partial^2 \widehat{y}_j}$;
  end
end
Use same gain computation as Algorithm 3 to select best candidate split, but G_l is replaced with G_{kv} and H_l with H_{kv};
if gain $> \gamma$ **then**
  return Best feature and split among the proposed candidates
else
  return No good split found
end

The proposal split S_k mentioned in Algorithm 5 above, can be done globally (per tree), or locally (per split).

C Proof of non-empty imputation set in bankruptcy games

We will now prove by contradiction that a bankruptcy game written as a TU game with n players requires a non-empty imputation set. A bankruptcy game has a claim vector $c \in \mathbb{R}_+^n$, which consists of the claim of each player who wants a part of an estate $E \in \mathbb{R}^{++}$. The goal is to divide E amongst all players equally. Note that for a bankruptcy game we have:

$$v(S) = \max\left(0, E - \sum_{j \neq i} c_j\right),$$

and

$$\sum_{i=1}^{n} c_i > E.$$

Now our proof by contradiction will show the following:

$$\sum_{i=1}^{n} v(\{i\}) \leq E = v(N).$$

Assume, that

$$\sum_{i=1}^{n} v(\{i\}) > E. \tag{C.1}$$

We can rewrite the value of the singleton coalition the following way:

$$v(\{i\}) = \max\left(0, c_i - \left(\sum_{j=1}^{n} c_j - E\right)\right) \quad \forall\ i = 1, ..., n.$$

We now define the indicator function:

$$\mathbf{1}_{\{c_i > \sum_{j=1}^n c_j - E\}} = \begin{cases} 1 & \text{if } c_i > \sum_{j=1}^n c_j - E, \\ 0 & \text{else.} \end{cases}$$

Then, if we plug everything in including Equation C.1 we obtain:

$$\sum_{i=1}^n v(\{i\}) = \sum_{i=1}^n \left(c_i - \left(\sum_{j=1}^n c_j - E \right) \right) \mathbf{1}_{\{c_i > \sum_{j=1}^n c_j - E\}}$$

$$= \sum_{i=1}^n c_i \mathbf{1}_{\{c_i > \sum c_j - E\}} - \left(\sum_{j=1}^n c_j - E \right) \sum_{i=1}^n \mathbf{1}_{\{c_i > \sum c_j - E\}} > E$$

$$\iff \sum_{i=1}^n c_i \mathbf{1}_{\{c_i > \sum c_j - E\}} > E + \left(\sum c_j - E \right) \sum_{i=1}^n \mathbf{1}_{\{c_i > \sum c_j - E\}}. \tag{C.2}$$

But for each i with $\mathbf{1}_{\{c_i > \sum c_j - E\}} = 1$, we have:

$$c_i > \sum c_j - E.$$

Thus,

$$\sum_{i=1}^n c_i \mathbf{1}_{\{c_i > \sum c_j - E\}} > \left(\sum c_j - E \right) \sum_{i=1}^n \mathbf{1}_{\{c_i > \sum c_j - E\}}. \tag{C.3}$$

Subtracting Equation C.2 from Equation C.3 yields:

$$0 > E,$$

which is a contradiction since $E > 0$ by definition.
Therefore, the assumption of Equation C.1 is false, and

$$\sum_{i=1}^n v(\{i\}) \leq E = v(N).$$

D From data to $v(S)$, other approach

Recall from Chapter 4.4.1, that we want to estimate $v(S)$ using the following equation:

$$v_i(S) \approx \mathbb{E}_{x_{\text{train}} \sim \mathcal{D}_{\text{train}}}[f(x_{\text{train}}) \mid x_{\text{train}}^S = x_i^S]. \tag{D.1}$$

Where x_i is the observation which is being explained, and $x_i = [x_i^S, x_i^{N \setminus S}]$, so x_i consists of the features that are in the coalition S and the features that are not in the coalition S.

To calculate the expectation in Equation D.1, we construct new inputs. For each observation in x_{train}, we replace the features in coalition S with those of the observation that is being explained, x_i^S, while keeping the remaining features $x_{\text{train}}^{N \setminus S}$ unchanged. This way, new data points are generated that combine x_i^S with the distribution of the remaining features in the training data. We then use the model f to generate predictions for all the synthetic observations and take the average of the predictions. By the law of large numbers (see (Bain & Engelhardt, 1992)), this average approximates the expectation in Equation D.1.

Example D.1 The process above will now be illustrated with an example. Assume we have the following training data for a model, where for each observation i we want to predict days absent current month y_i using overtime hours last month x_{i1} and months in service x_{i2}:

	Days absent current month (y_i)	**Overtime hours last month (x_i^1)**	**Months in service (x_i^2)**
Person 1	14	15	8
Person 2	0	0	16
Person 3	2	10	24
Person 4	30	20	32

Table D.1: Training data example going from data to coalition values

We have the following test observation:

	Days absent current month (y_i)	**Overtime hours last month (x_i^1)**	**Months in service (x_i^2)**
Person 5	5	12	10

Table D.2: Test observation example going from data to coalitional values

Now assume we want to calculate $v_5(\{\text{Overtime hours last month}\})$, we would do the following:

	Predicted days absent current month ($\widehat{y}_i$)	**Overtime hours last month (x_i^1)**	**Months in service (x_i^2)**
Person 1	8	12	8
Person 2	0	12	16
Person 3	5	12	24
Person 4	25	12	32

Table D.3: All prediction possibilities with x_5^{overtime} fixed

We then average over all predictions to obtain $v_5(\{\text{Overtime hours last month}\})$:

$$v_5(\{\text{Overtime hours last month}\}) = \frac{1}{4}\sum_{i=1}^{4} y_{i5} = \frac{1}{4}\cdot(8+0+5+25) = 9\frac{1}{2}. \qquad \blacktriangle$$

E Other plots of results

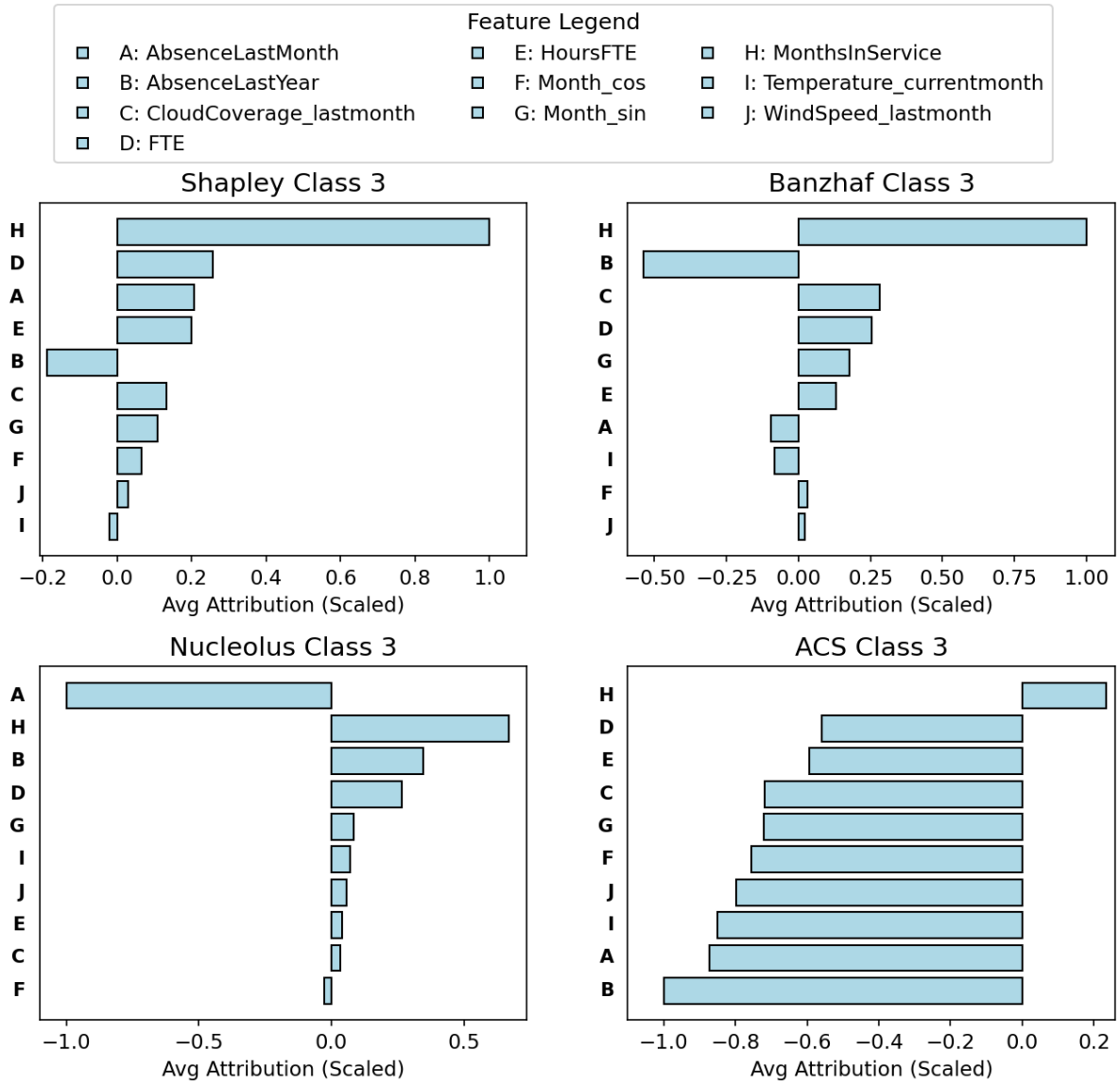

Figure E.1: Feature attribution (scaled to be within $[-1, 1]$) averaged over all samples all four game-theoretic concepts for class 3.

Feature	Attribution mean	Attribution variance	Mean normalised feature value	Variance normalised feature value	Correlation attribution and normalised feature value
Absence last month	-0.29388	0.00856	0.09223	0.04438	0.46534
Absence last year	-0.38199	0.01898	0.06229	0.02931	0.18943
Cloud coverage last month	-0.38866	0.01921	0.41330	0.10925	0.03500
FTE	-0.30568	0.02896	0.64007	0.08507	-0.60359
HoursFTE	-0.33211	0.01926	0.75173	0.08847	-0.12410
Month cos	-0.36684	0.01934	0.50568	0.16608	-0.17123
Month sin	-0.37397	0.01898	0.49949	0.12543	-0.08314
Months in service	-0.10651	0.02253	0.14483	0.02807	0.34876
Temperature current month	-0.38877	0.01925	0.52178	0.14159	0.01361
Wind speed last month	-0.38294	0.01911	0.44020	0.16958	-0.01673

Table E.1: Summary statistics of beeswarm plot of the ACS concept for class 0 in Figure 7.2

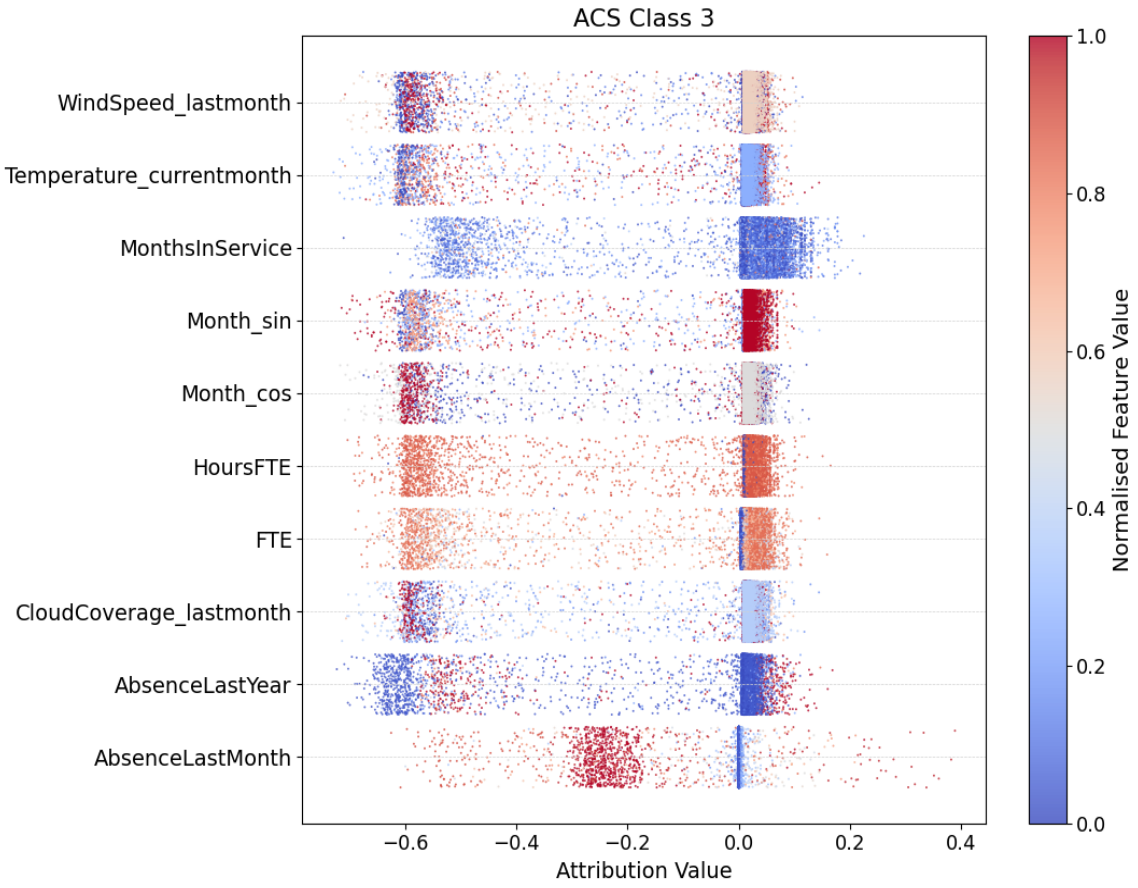

Figure E.2: Beeswarm plot of the ACS solution for class 3, which shows on the x-axis the feature attribution for each observation, on the left y-axis the features and on the right y-axis the normalised feature value.

Feature	Attribution mean	Attribution variance	Mean normalised feature value	Variance normalised feature value	Correlation attribution and normalised feature value
Absence last month	-0.01171	0.00385	0.09223	0.04438	-0.76973
Absence last year	-0.01342	0.01467	0.06229	0.02931	-0.19776
Cloud coverage last month	-0.00965	0.01446	0.41330	0.10925	-0.00868
FTE	-0.00749	0.01381	0.64007	0.08507	-0.03519
HoursFTE	-0.00797	0.01426	0.75173	0.08847	-0.06533
Month cos	-0.01014	0.01460	0.50568	0.16608	-0.04379
Month sin	-0.00967	0.01442	0.49949	0.12543	0.02514
Months in service	0.00315	0.01177	0.14483	0.02807	-0.07810
Temperature current month	-0.01141	0.01486	0.52178	0.14159	0.03409
Wind speed last month	-0.01072	0.01470	0.44020	0.16958	0.01510

Table E.2: Summary statistics of beeswarm plot of the ACS concept for class 3 in Figure E.2

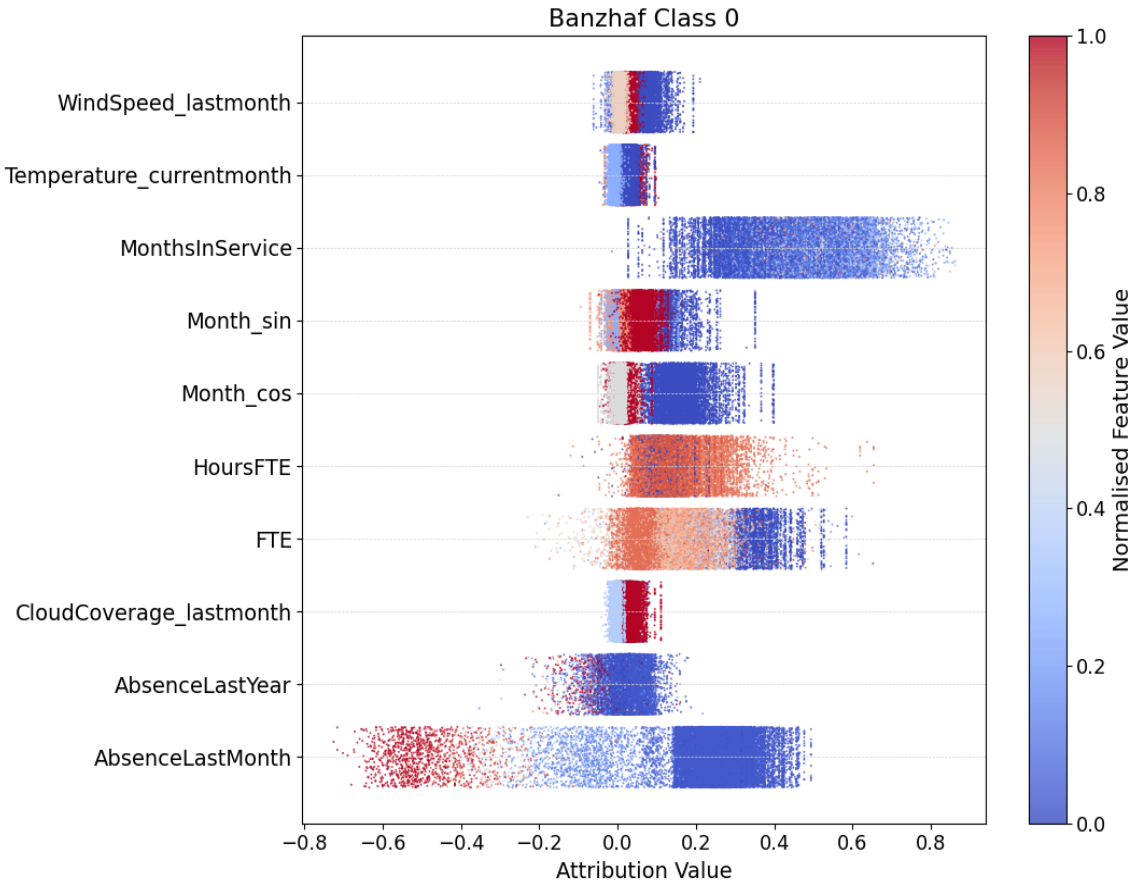

Figure E.3: Beeswarm plot of the Banzhaf value for class 0, which shows on the x-axis the feature attribution for each observation, on the left y-axis the features and on the right y-axis the normalised feature value.

Feature	Attribution mean	Attribution variance	Mean normalised feature value	Variance normalised feature value	Correlation attribution and normalised feature value
Absence last month	0.19587	0.03850	0.09223	0.04438	-0.89166
Absence last year	0.02122	0.00126	0.06229	0.02931	-0.36469
Cloud coverage last month	0.01359	0.00030	0.41330	0.10925	0.30403
FTE	0.14730	0.01457	0.64007	0.08507	-0.80429
HoursFTE	0.12566	0.00393	0.75173	0.08847	-0.00241
Month cos	0.06255	0.00610	0.50568	0.16608	-0.77358
Month sin	0.04590	0.00241	0.49949	0.12543	-0.36956
Months in service	0.45326	0.01790	0.14483	0.02807	0.33968
Temperature current month	0.01268	0.00036	0.52178	0.14159	0.20552
Wind speed last month	0.02606	0.00091	0.44020	0.16958	-0.25284

Table E.3: Summary statistics of beeswarm plot of the Banzhaf value for class 0 in Figure E.3

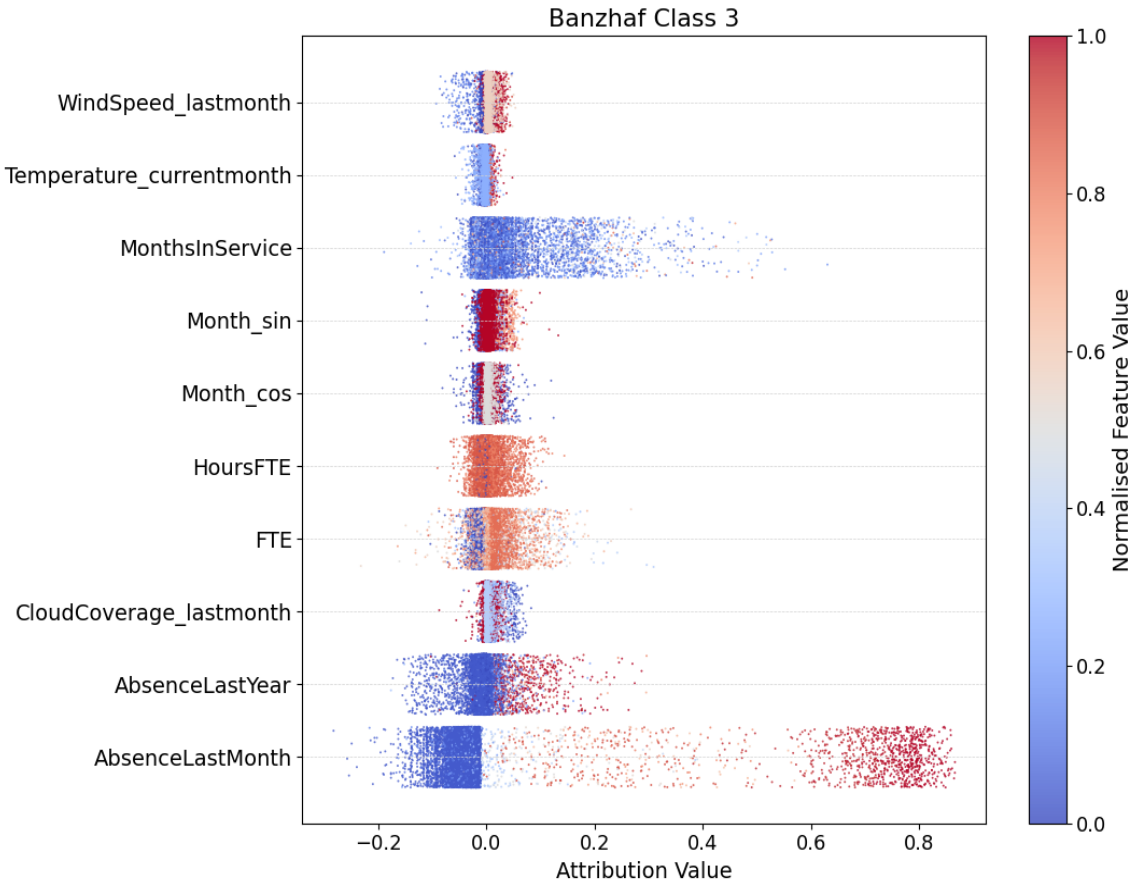

Figure E.4: Beeswarm plot of the Banzhaf value for class 3, which shows on the x-axis the feature attribution for each observation, on the left y-axis the features and on the right y-axis the normalised feature value.

Feature	Attribution mean	Attribution variance	Mean normalised feature value	Variance normalised feature value	Correlation attribution and normalised feature value
Absence last month	-0.00086	0.02382	0.09223	0.04438	0.91900
Absence last year	-0.00495	0.00048	0.06229	0.02931	0.55904
Cloud coverage last month	0.00260	0.00004	0.41330	0.10925	-0.12395
FTE	0.00233	0.00037	0.64007	0.08507	0.18987
HoursFTE	0.00119	0.00013	0.75173	0.08847	0.08737
Month cos	0.00029	0.00004	0.50568	0.16608	0.15134
Month sin	0.00163	0.00006	0.49949	0.12543	0.34247
Months in service	0.00918	0.00233	0.14483	0.02807	0.06505
Temperature current month	-0.00076	0.00001	0.52178	0.14159	0.49758
Wind speed last month	0.00020	0.00004	0.44020	0.16958	0.37795

Table E.4: Summary statistics of beeswarm plot of the Banzhaf value for class 3 in Figure E.4

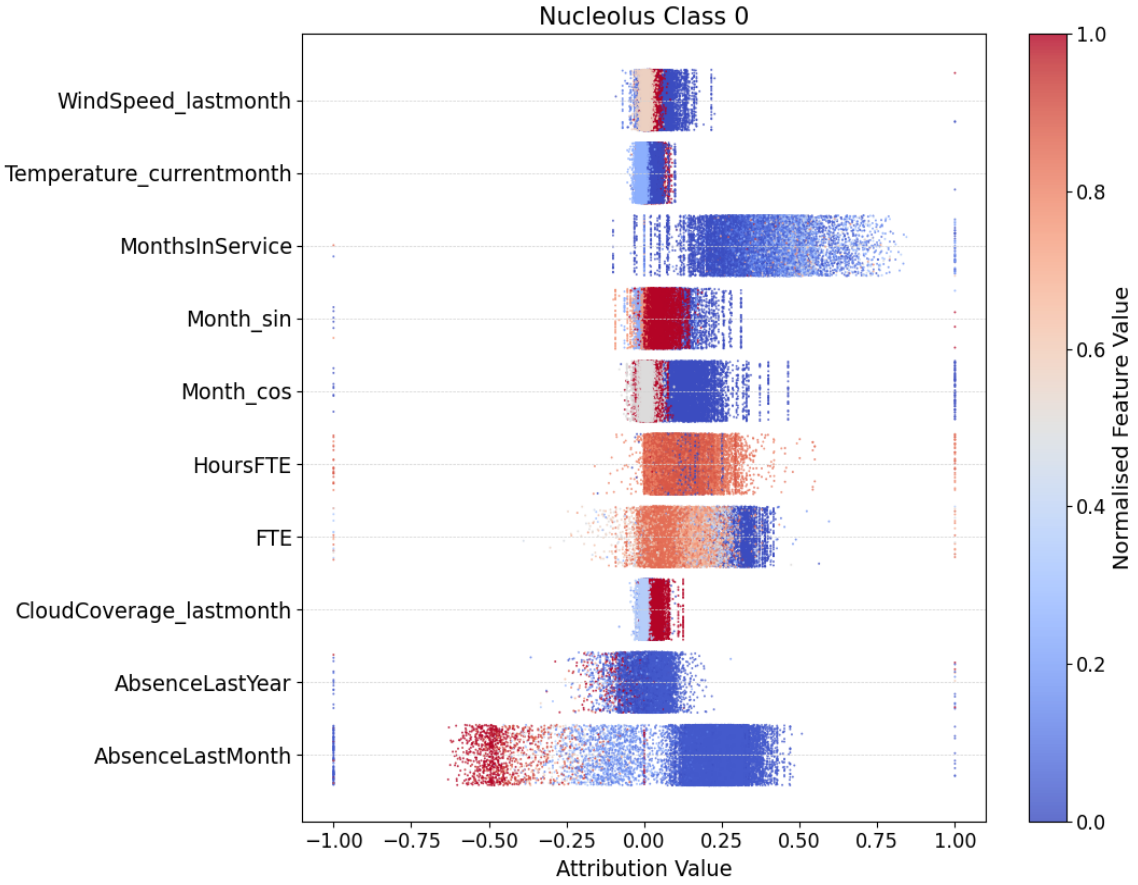

Figure E.5: Beeswarm plot of the nucleolus for class 0, which shows on the x-axis the feature attribution for each observation, on the left y-axis the features and on the right y-axis the normalised feature value.

Feature	Attribution mean	Attribution variance	Mean normalised feature value	Variance normalised feature value	Correlation attribution and normalised feature value
Absence last month	0.16460	0.04134	0.09201	0.04422	-0.76971
Absence last year	0.01389	0.00292	0.06223	0.02927	-0.22729
Cloud coverage last month	0.01147	0.00029	0.41324	0.10929	0.29817
FTE	0.12001	0.01256	0.63993	0.08514	-0.73300
HoursFTE	0.10849	0.00631	0.75156	0.08858	-0.08355
Month cos	0.05703	0.00841	0.50601	0.16610	-0.62989
Month sin	0.04073	0.00311	0.49931	0.12541	-0.29035
Months in service	0.37487	0.01452	0.14477	0.02808	0.36833
Temperature current month	0.01062	0.00043	0.52158	0.14165	0.19342
Wind speed last month	0.02272	0.00102	0.43994	0.16958	-0.24669

Table E.5: Summary statistics of beeswarm plot of the nucleolus for class 0 in Figure E.5

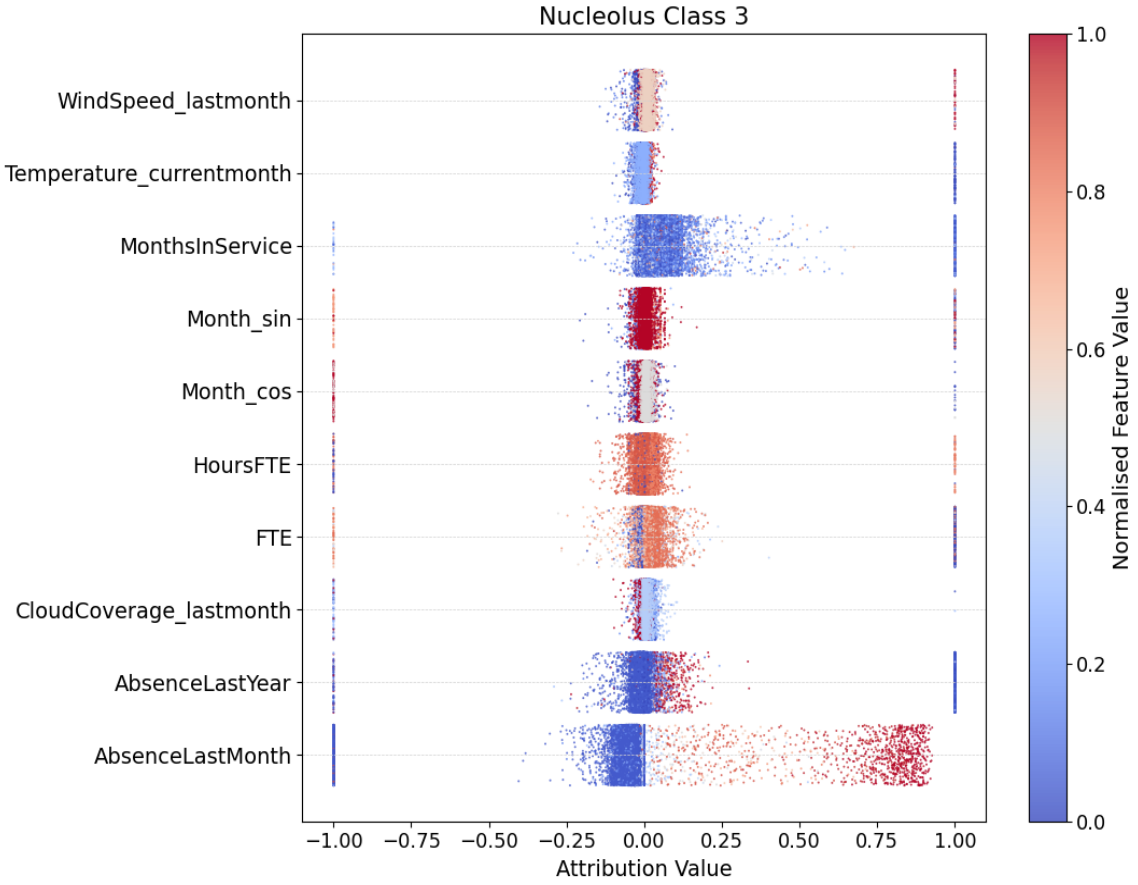

Figure E.6: Beeswarm plot of the nucleolus for class 3, which shows on the x-axis the feature attribution for each observation, on the left y-axis the features and on the right y-axis the normalised feature value.

Feature	Attribution mean	Attribution variance	Mean normalised feature value	Variance normalised feature value	Correlation attribution and normalised feature value
Absence last month	-0.07538	0.09271	0.09358	0.04522	0.45969
Absence last year	0.02617	0.03986	0.06292	0.02969	-0.00671
Cloud coverage last month	0.00246	0.00406	0.41158	0.10866	-0.00920
FTE	0.02000	0.02102	0.64498	0.08222	-0.18840
HoursFTE	0.00317	0.00620	0.75799	0.08438	0.05225
Month cos	-0.00196	0.00580	0.50240	0.16577	-0.06520
Month sin	0.00638	0.00838	0.49849	0.12568	-0.02324
Months in service	0.05056	0.02821	0.14631	0.02820	-0.00752
Temperature current month	0.00535	0.00528	0.52512	0.14107	-0.03441
Wind speed last month	0.00439	0.00239	0.43940	0.16935	0.10916

Table E.6: Summary statistics of beeswarm plot of the nucleolus for class 3 in Figure E.6

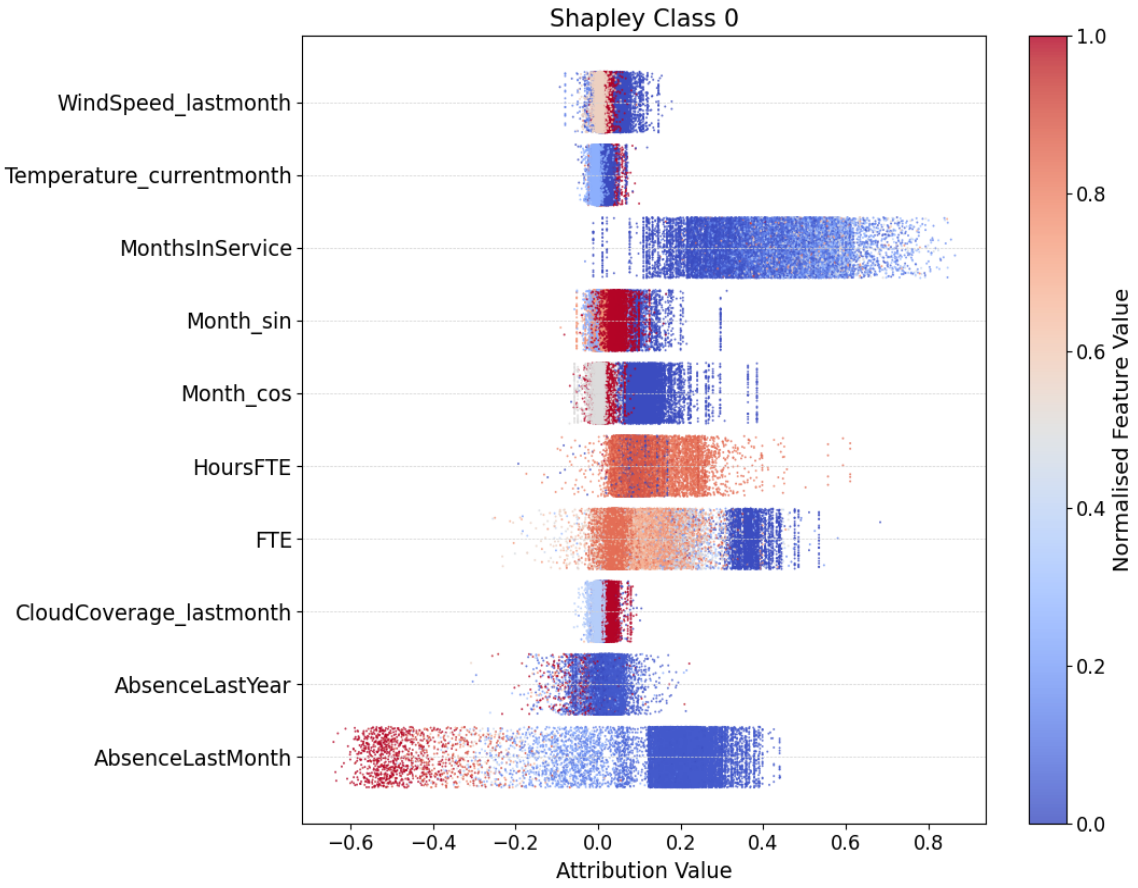

Figure E.7: Beeswarm plot of the Shapley value for class 0, which shows on the x-axis the feature attribution for each observation, on the left y-axis the features and on the right y-axis the normalised feature value.

Feature	Attribution mean	Attribution variance	Mean normalised feature value	Variance normalised feature value	Correlation attribution and normalised feature value
Absence last month	0.14879	0.02859	0.09223	0.04438	-0.90539
Absence last year	0.01648	0.00080	0.06229	0.02931	-0.32798
Cloud coverage last month	0.00938	0.00017	0.41330	0.10925	0.27397
FTE	0.12557	0.01418	0.64007	0.08507	-0.85988
HoursFTE	0.09624	0.00285	0.75173	0.08847	-0.00596
Month cos	0.04530	0.00346	0.50568	0.16608	-0.74642
Month sin	0.03325	0.00131	0.49949	0.12543	-0.35531
Months in service	0.41554	0.01566	0.14483	0.02807	0.39944
Temperature current month	0.00883	0.00019	0.52178	0.14159	0.20310
Wind speed last month	0.01831	0.00048	0.44020	0.16958	-0.24453

Table E.7: Summary statistics of beeswarm plot of the Shapley value for class 0 in Figure E.7

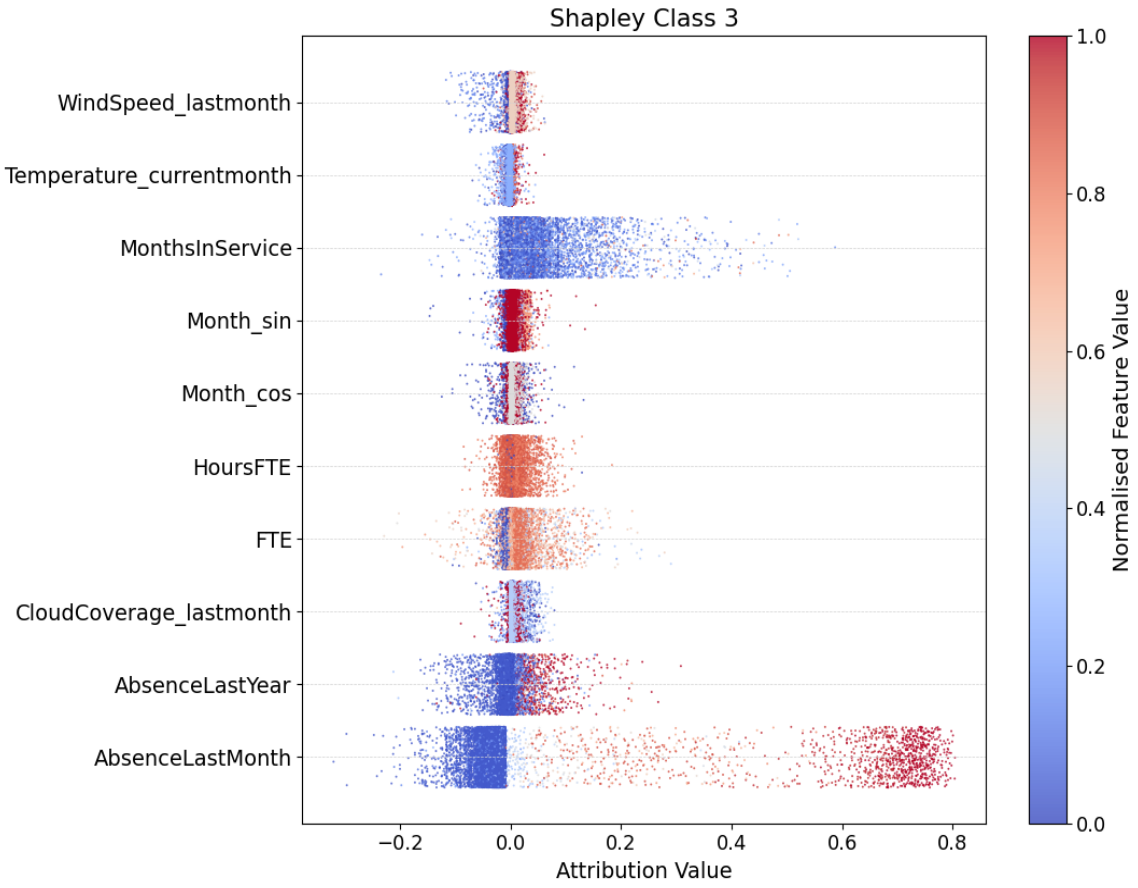

Figure E.8: Beeswarm plot of the Shapley value for class 3, which shows on the x-axis the feature attribution for each observation, on the left y-axis the features and on the right y-axis the normalised feature value.

Feature	Attribution mean	Attribution variance	Mean normalised feature value	Variance normalised feature value	Correlation attribution and normalised feature value
Absence last month	0.00372	0.02072	0.09223	0.04438	0.91859
Absence last year	-0.00338	0.00032	0.06229	0.02931	0.52290
Cloud coverage last month	0.00238	0.00003	0.41330	0.10925	-0.13004
FTE	0.00462	0.00029	0.64007	0.08507	0.12289
HoursFTE	0.00359	0.00009	0.75173	0.08847	0.06548
Month cos	0.00116	0.00003	0.50568	0.16608	-0.01831
Month sin	0.00193	0.00004	0.49949	0.12543	0.32641
Months in service	0.01793	0.00195	0.14483	0.02807	0.06774
Temperature current month	-0.00038	0.00001	0.52178	0.14159	0.47603
Wind speed last month	0.00053	0.00004	0.44020	0.16958	0.31500

Table E.8: Summary statistics of beeswarm plot of the Shapley value for class 3 in Figure E.8

F Data insights

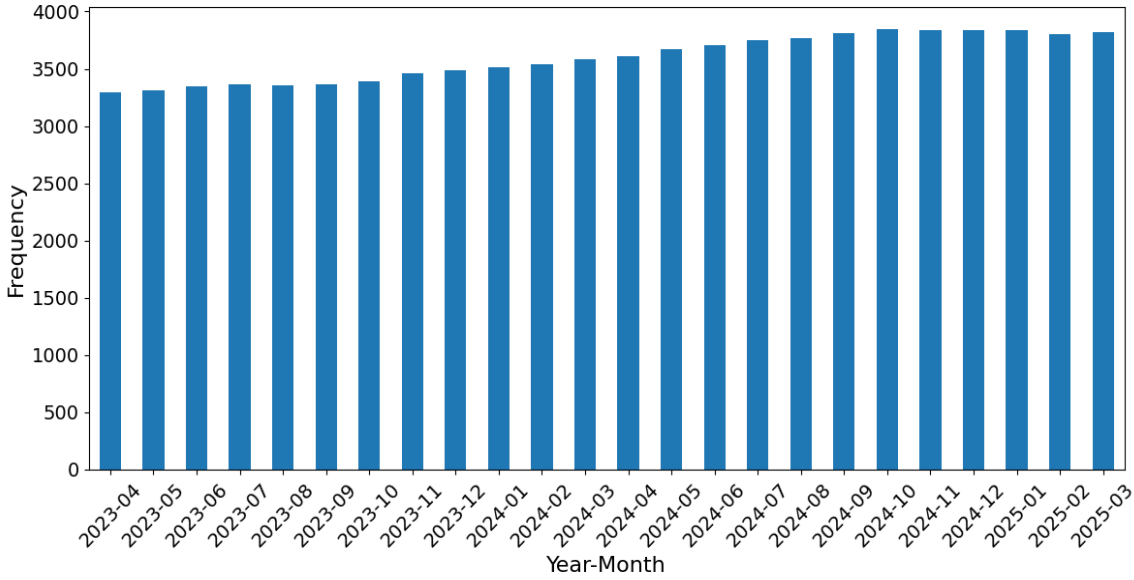

Figure F.1: Amount of observations per year-month combination

From August 2015 onward there is sick leave every month, this is not the case before that date, so we will use data from August 2015 onward. We see that the aggregated sick leave per month is increasing, but this is probably due to this company obtaining more data over the years.

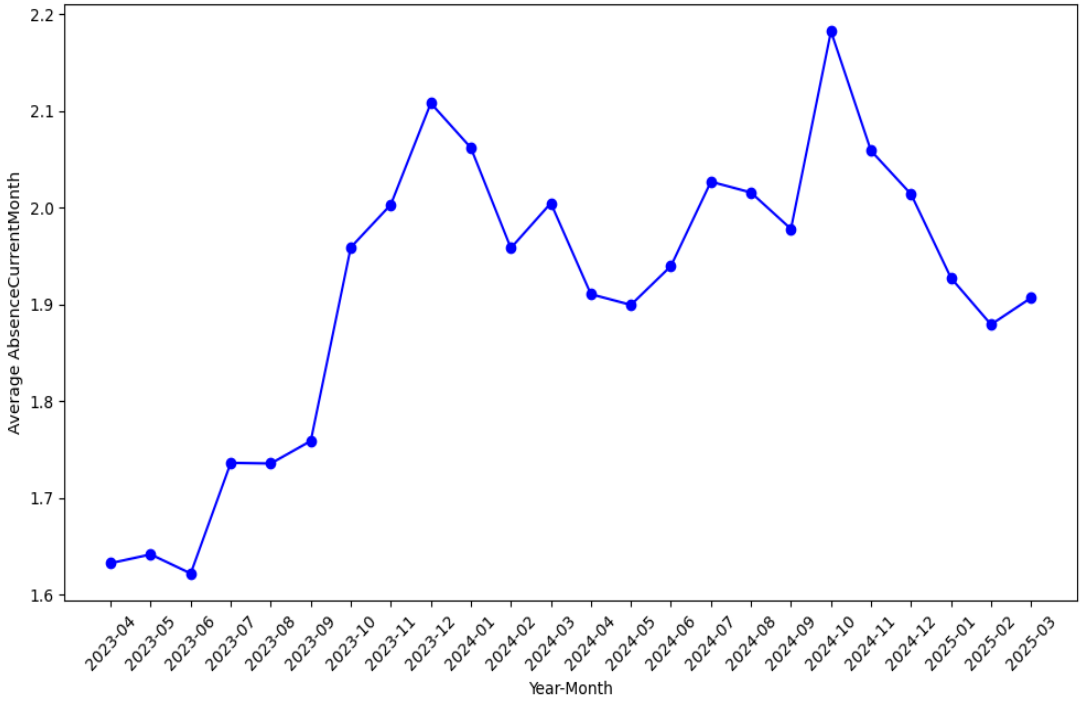

Figure F.2: Average absence per employee per month

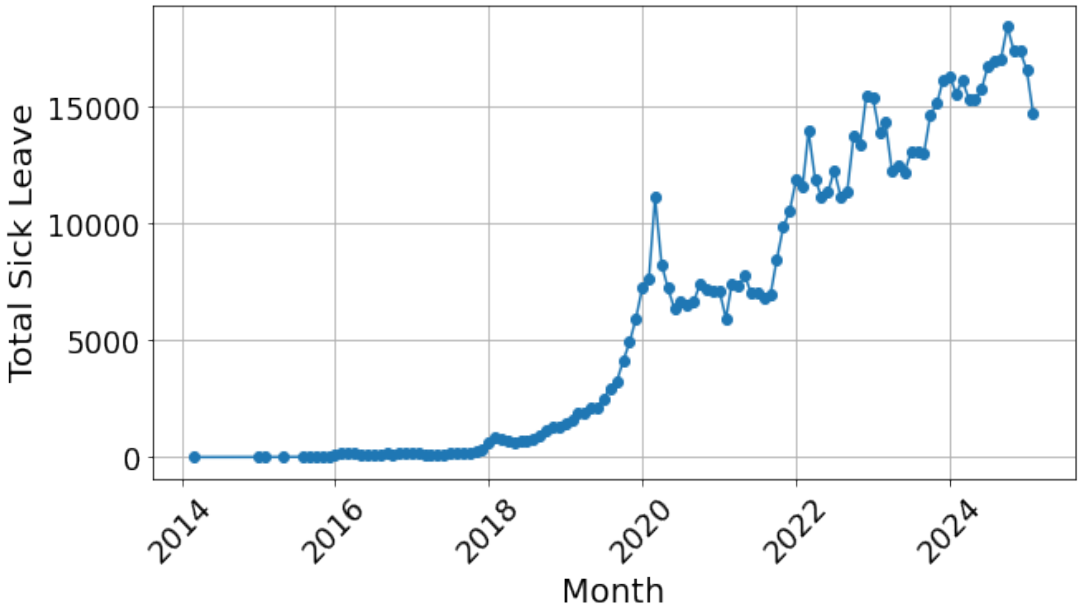

Figure F.3: Days of sick leave per month aggregated over the employees

Days	28 days	29 days	30 days	31 days
1	0.0357	0.0345	0.0333	0.0323
2	0.0714	0.0689	0.0667	0.0645
3	0.1071	0.1034	0.1	0.0968
4	0.1429	0.1379	0.1333	0.1290
5	0.1786	0.1724	0.1667	0.1613
6	0.2143	0.2069	0.2	0.1935
7	0.25	0.2414	0.2333	0.2258
8	0.2857	0.2759	0.2667	0.2581
9	0.3214	0.3103	0.3	0.2903
10	0.3571	0.3448	0.3333	0.3226
11	0.3929	0.3793	0.3667	0.3548
12	0.4286	0.4138	0.4	0.3871
13	0.4643	0.4483	0.4333	0.4194
14	0.5	0.4828	0.4667	0.4516
15	0.5357	0.5172	0.5	0.4839
16	0.5714	0.5517	0.5333	0.5161
17	0.6071	0.5862	0.5667	0.5484
18	0.6429	0.6207	0.6	0.5806
19	0.6786	0.6552	0.6333	0.6129
20	0.7143	0.6897	0.6667	0.6452
21	0.75	0.7241	0.7	0.6774
22	0.7857	0.7586	0.7333	0.7097
23	0.8214	0.7931	0.7667	0.7419
24	0.8571	0.8276	0.8	0.7742
25	0.8929	0.8621	0.8333	0.8065
26	0.9286	0.8966	0.8667	0.8387
27	0.9643	0.9311	0.9	0.8709
28	1	0.9655	0.9333	0.9032
29		1	0.9667	0.9355
30			1	0.9677
31				1

Table F.1: Fraction that decides in which bin an absence of a certain number of days, in a month with a certain number of days, is put in the histogram

Country	Number of employees
The Netherlands	12,513
NaN	1256
The United States	497
Germany	407
France	387
Belgium	300
Japan	132
Spain	122
Uruguay	78
South Africa	60
Singapore	48
Canada	43
Bangladesh	37
China	30
Poland	28
Switzerland	19
United Kingdom of Great Britain and Northern Ireland	16
Hungary	15
Australia	14
Italy	10
Austria	9
Brazil	6
Sweden	6
Curacauo	6
United Arab Emirates	5
Denmark	2
Ireland	2
India	2
Turkey	2
Bonaire, Sint Eustatius, and Saba	1
Mexico	1
Portugal	1
Malaysia	1
South Korea	1

Table F.2: Number of employees working in each country of dataset before data reduction