Predicting Real Estate Price Using Text Mining

Automated Real Estate Description Analysis

Dick Stevens

ANR: 640003

HAIT Master Thesis

Tilburg University School of Humanities

Department of Communication and Information Sciences

Tilburg center for Cognition and Communication (TiCC)

Tilburg, The Netherlands

July 2014

Supervisor:

Dr. S. Wubben

Second Reader:

Dr. M.M. van Zaanen

Tags:

text mining, data mining, house prices, selling price, asking price, price fluctuation, automated description analysis, unigrams, bigrams, stemming, regression, classification

Abstract

This study describes two methods for real estate price prediction using text mining: classification and regression. Both methods use stemmed n-grams (unigrams and bigrams) derived from real estate description to train different machine learning techniques. Real estate price predictions involve the following pricing indicators: selling price, asking price and price fluctuation. In our classification experiment we succeeded to predict all pricing indicators significantly above baseline with a maximum increase of 38% (F^1 = .652). Our regression experiment shows that the SGD classifier performs best for all pricing indicators with highest performance achieved for predicting selling price (R^2 = .303). Both can be seen as respectable results given the complex nature of the task.

Our results indicate that performance increases when we enlarge the total amount of unigrams and bigrams used in our price prediction model. For the prediction of selling price we see a significant logarithmic increase, which means that the positive effect of adding more n-grams to the model reduces over time.

A predictive system for real estate price using text mining reveals to be highly challenging and becomes increasingly complex for larger datasets. Extracting predictive n-grams from real estate description can be a unique and meaningful addition to the standard hedonic pricing models widely used for the prediction of real estate price.

Acknowledgements

I remember my first day at the Tilburg University like it was yesterday. Traveling from Utrecht by train with nothing but a notebook and a pen. The necessary books were rented that day at the campus library.

Attending the classes at the university, I noticed the broad and interesting possibilities of data mining. This was almost directly followed by the notion that only a few took real advantage of this way of exploring valuable data. Classes such as Programming in Python, Information Search, Retrieval & Recommendation and Data Mining taught me the different approaches in data mining. Not an easy matter at first, but the enthusiastic approach of the professors and assistant professors made every student go that extra mile in understanding this interesting topic. When I went to study in California, I enrolled for Natural Language Programming and Artificial Intelligent Programming. All were valuable experiences and one by one essential for writing the most important piece of research in my time as a scholar: this Master thesis.

I would like to thank Assistant Professor Sander Wubben for supervising this thesis. Despite his full agenda, he always managed to find time to meet at the university or send detailed feedback by email. This gave me the advice and personal guidance needed to finish this thesis.

Also, I would like to thank my girlfriend, Lize, for her patience and understanding. My parents for the unconditional support only a parent can give and my friends for the words of wisdom and, of course, words of foolishness.

While looking back with a smile, it is now time to look forward to the next exciting chapter of my life.

Dick Stevens

Table of Contents

1 Introduction

Over the past 35 years, a vast amount of knowledge has been accumulated on text mining for Information Retrieval (IR). Using automated text mining algorithms to discover knowledge from natural language texts provides numerous challenges but also offer unique possibilities. One of the most natural forms of storing information is in the form of natural language texts (Tan, 1999). This can be easily interpreted by a human but it is still a great challenge for computers to derive meaning from this data. However, computers do offer an important advantage over human capabilities: computing power. This means that computers can find patterns, which are non-trivial recurrences, within data faster and more accurate than their human counterpart, but this can only be done if the structure of the data is known. Natural language does contain implicit grammatical structure (Rajman & Basancon, 1997), but these structures are deeply complex and vary across different languages.

Although there are no global algorithms to this day that automatically process all natural language texts, researchers keep improving text mining techniques and also show promising results and unique approaches of application. Yu, Wang and Lai (2005) describe a forecasting system for crude oil prices using text mining. Their results reveal that valid patterns were found in their natural language texts and that their approach outperforms other forecasting models. This suggests that their system can be suitable to a wide range of prediction problems. In a similar vein, Mittermayer (2004) describes a predictive system based on text mining techniques. This system, named NewsCATS (News Categorization and Trading System) was implemented to immediately predict trends in stock prices after the publication of press releases. Most of these studies predict a certain price range, which is known as classification, and not an exact price, known as regression. Researchers Ghani and Simmons (2004) conclude that the classification approach is less complex than the regression approach and will therefor result in better performance.

Other than price prediction, text mining techniques have also proven to be useful in predicting sentiment. Since the sale of online products is becoming increasingly popular, the amount of customer reviews also have grown rapidly (Hu & Liu, 2004). Determining whether a review conveys a positive or a negative opinion by an automatic system has proven to be remarkably useful in areas such as e-commerce.

Another area in which price prediction is especially important is the real estate area. Real estate agents, but also researchers, generally focus on regular real estate characteristics and location in order to make predictions on price. This is also known as the hedonic pricing method (Peterson, 2008). For example, two bathrooms, a large living room and a two-car garage can be attractive features for potential buyers of real estate property (Haurin, 1988). These standardized characteristics can be compared with each other in order to sale predictions, but the description of the house, which can also contain valuable information, cannot be compared in such a way. This is due to the form of the description: natural language text.

Inspired by the before mentioned studies on predictive systems using text mining, this study will explore the possibility of finding salient patterns within real estate descriptions. For this purpose, we will analyze a large real estate database. This database was assembled in November 2012 by Nanne van Noord: a PhD student from Tilburg University in the Netherlands. It contains data of more than 250,000 houses on sale in the United States. Using text mining techniques we will automatically process and analyze real estate descriptions in order to make predictions regarding sale. This will include selling price, asking price and price fluctuation.

1.1 Problem Statement

The general and standardized real estate characteristics are often listed separately from the asking price and general description. Because these characteristics are separately listed in a structured way, they can be easily compared across the whole range of potential houses. Because every house also has its own unique characteristics, such as a particular view or type of sink, house sellers can provide a summary of all the important features of the house in the description. All given real estate features can be considered by the potential buyers, but it is nearly impossible to provide an automated comparison on all variables due to the large diversity. This is also true in the other direction: house sellers have to make an estimation of the value based on its features in comparison to the current market price of similar houses. The diversity of features makes it challenging to estimate an adequate market price.

Apart from providing a summary of the important features of the house, the house description is also a means of raising curiosity in the reader, or in other words to persuade the person. It is possible that there are certain word sequences in the natural language text that seduce potential buyers more than others. Therefore, there might be a relation between the language used in the description and the price of the property. This comparison does not focus primarily on the house characteristics, but on all words within the description. For example, a description with the word ‘highly’ can outperform one with the word ‘very’ looking at price fluctuation: the difference between real estate asking- and selling price. This can mean that the word ‘highly’ is commonly seen in descriptions that show an increase in real estate price while the word ‘very’ generally leads to a decrease in price. In addition, we can also find words that are distinctive for a certain range in selling- or asking price, thus can be used for prediction tasks. Hence, we have determined three pricing indicators that will be meaningful to predict: *selling price, asking price* and *price fluctuation.* To shed light on the predictive capability of the descriptions we propose the following research questions:

RQ1: To what extend can real estate descriptions predict selling price?
RQ2: To what extend can real estate descriptions predict asking price?
RQ3: To what extend can real estate descriptions predict price fluctuation?

Since we cannot predict the outcomes of this study we do not separate the house characteristic features from other content. We are interested in predicting pricing indicators based on description and want to include all its content.

Our additional research questions are based on findings in previous research on price prediction. These studies have shown that predicting a price range usually results in a higher performance over predicting exact prices (Ghani & Simmons, 2004). In addition, positive relations have been found between the size of the training sample and performance (Zhang, Jin, Yang & Hauptmann, 2003). More training samples generally lead to a higher performance due to the potential refinements in the model.
Also, we can see that previous research use a wide range of different machine learning algorithms. Researchers like Tan (1999) and Frank, Trigg, Holmes and Witten (1999) do not use one single algorithm in their experiment, but a variety. This reduces the chance of choosing an ill suited algorithm and allows comparison between algorithm performances. We propose our additional research questions:

RQ4: What is the influence of training set size on performance?

RQ5: What is the influence of attribute variable amount on performance?

Finally, to the best of our knowledge, no studies have been done on predicting real estate sale using text mining on its description. However, research has been done on real estate price prediction based on house characteristics: the hedonic pricing model (Dubin, 1998; Peterson, 2008; Nguyen & Cripps, 2001). Further elaboration on these studies can be found in section 2.8. We have constructed the following hypotheses relating to all research questions:

H1: Classification will result in a better overall performance than regression

H2: Using more attribute variables will generally result in better performance

H3: Increasing the amount of training samples will result in higher performance

2 Theoretical Framework

2.1 Data Mining

Data mining is part of the process known as knowledge discovery in databases (KDD). KDD concerns the understanding of useful and structural patterns in data, thus referring to the overall procedure of information discovery from data (Rajman & Basancon, 1997). These data driven discoveries became increasingly popular in the domains of machine learning and artificial intelligence (Fayyad, Piatetsky-Shapiro & Smyth, 1996). Within these domains a variety of specific algorithms are used in order to extract patterns from data. This data represents a set of facts, for example cases in a database, and a generated model that applies to a subset in the data is the extracted pattern. Thus, a pattern is a description of the non-trivial structure in the data, thus, is can be generalized to new data examples in order to make predictions (Witten, Frank & Hall, 2011). The extracting of these patterns by applying specific algorithms is data mining.

Data mining algorithms to represent a model exist in a wide variety. Popular algorithms include: decision trees, linear- and nonlinear regression and classification, instance-based methods and probabilistic dependency models (Witten, Frank & Hall, 2011; Ye, 2013). Important to emphasize is that each technique generally suits some problems better than others. These techniques will be explained further in section 2.5.

Data mining algorithms learn from the data. It learns to predict the required output from the given input. This type of learning does not concern the committing of the data to the machines' (computer) memory, but it learns to change its model in order to perform better in the future (Witten, Frank & Hall, 2011). This type of learning is also referred as training.

The input on which the algorithm is trained is represented by two types of features: the attribute variables and the target variables. Values from the attribute variables, also known as the independent variables, provide the basis for determining the target variables, the dependent variable that needs to be predicted. Variables in the data can have categorical values (e.g. *sex* = 'female') or numeric values (e.g. *age* = 29). This information is listed in a structured manner to let the algorithm learn from the data. An example of learning from structured data can be seen in Table 1 and Table 2. An attribute-value database is provided (Table 1) and induced rules (Table 2) on the structured data are formed.

Table 1. Potential Customer Table

Person	Age	Sex	Income	Customer
Ann	29	F	100,000	Yes
Joan	53	F	1,000,000	Yes
Jack	50	M	200,000	No
Bob	25	M	20,000	Yes
Rob	30	M	100,000	Yes
Sam	31	M	100,000	No
Dave	21	M	90,000	Yes

Table 2. Induced Rules

If Income(Person) >= 100,000 **and** Sex(Person) = Female **then** Potential-Customer(Person)
If Age(Person) < 31 **and** Sex(Person) = Male **then** Potential-Customer(Person)

These rules can only be formed using the predefined structure of this relational database. The fields in the database table are well defined and there are clear values (*age*, *sex*, *income* and *customer*) paired with the attributes (*person*).

Data mining techniques can discover knowledge from data using a variety of methods. These methods can be divided in two groups: classification and regression (Ye 2013). A method is chosen depending on the required type of prediction. Classification (see section 2.3) learns to map, or classifies, an item in the data into one of several predefined classes while regressions (see section 2.4) maps the item to a continuous numerical prediction (Fayyad, Piatetsky-Shapiro & Smyth, 1996).

2.2 Text Mining

Text mining is the discovery of knowledge from natural language texts. These databases contain text documents that are analyzed using a variety of algorithms to find non-trivial patterns or knowledge in the data. As previously mentioned, information is most commonly stored as *text*. Due to this quantity, it is believed that text mining even has a higher commercial potential than data mining (Tan, 1999).

Text mining is a multidisciplinary field that involves text analysis, categorization, information extraction and machine learning and is often viewed as an extension to data mining (Tan, 1999). Although data mining and text mining do have overlap in their domains, text mining is inherently different from data mining due to the nature of the data it processes.

The form, or structure of the data, is highly important to understand the difference between text mining and data mining. Standard data mining techniques are primarily concerned with the processing of databases with a structured form (Rajman & Basancon, 1997). This in comparison with text mining, which is dedicated to process unstructured data, namely textual data, and automatically extract its information. Textual information is not structured as neatly as the attribute-value database seen in Table 1. The structure of natural texts offers great challenges. Text data is believed to be unstructured and fuzzy in nature and it is therefore a much more complex task to find valid and useful patterns in the data in an automated way (Tan, 1999).

Even though text is seen as unstructured, it does contain an implicit grammatical structure and patterns (Rajman & Basancon, 1997). This implicit structure is discovered and used with a variety in specialized text mining techniques. Natural Language Processing, a field that involves the understanding of natural language input, can be used to pre-process the textual data. A simple example of rules that can be induced using text mining based learning is shown in Table 3 and Table 4: the corpus (Table 3) is analyzed to induce rules (Table 4) on the meaning of the word ‘bank’.

Table 3. Corpus

Sentence = “Across the water, you can see a boat on the bank.”
Sentence = “On the bank, it was filled with trash carried by the water.”
Sentence = “When the robbers came into the bank, they demanded money.”
Sentence = “I went into the bank to deposit money that I earned this month.”

Table 4. Induced Rules

If Sentence **includes** ‘bank’ **and** ‘water’ **then** ‘bank’ = ground bounding waters
If Sentence **includes** ‘bank’ **and** ‘money’ **then** ‘bank’ = financial institution
If Sentence **includes** ‘bank’ **and** *preposition* = ‘on’ **then** ‘bank’ = ground bounding waters
If Sentence **includes** ‘bank’ **and** *preposition* = ‘into’ **then** ‘bank’ = financial institution

Text mining can be of high practical use and can fill the gap that conventional data mining has created in the discovery of knowledge in databases. However, due to the ambiguous and non-universal nature of natural language texts it is impossible to create all-embracing and nonexclusive algorithms (Rajman & Basancon, 1997). The rules induced shown in Table 4 are based on a small and consistent corpus but will incorrectly predict the meaning of the word ‘bank’ as a financial institution in sentences as: “Walking over the bank of the river, I found a briefcase with money.” In this sentence, the word ‘bank’ refers to ground bounding waters. These rules can be extended so more cases can be predicted correctly. However, highly accurate classification involves high training cost and can rapidly lead to incomprehensible models (Ifrim, Bakir & Weikum, 2008).

2.3 Classification

Conducting analyses on text can be done using a variety of approaches. This depends greatly on the type of insight that is required. Text classification, also known as text categorization, is a vital element of text mining where natural language texts are automatically assigned to predefined groups or categories (Tan, Wang & Lee, 2002). Category assignment is based on the content of the text and helps in retrieving relevant information.

A well-known example of text classification is categorizing on sentiment (Pang & Lee, 2008; Pak & Paroubek, 2010). A broad approach assigns a positive or negative class to a text, depending on its content. A more specific approach can include different gradations of negative and positive or it can incorporate a range of different emotions.

Before a text mining technique can make any predictions it needs to train its model. A corpus of a large number of texts is provided which already has the classes assigned. From this corpus, the leaning algorithm can induce rules, as seen in Table 4, based on reoccurring patterns in the text.

Generally, a corpus is used to test the performance of the algorithm. This test corpus also contains pre-assigned classes but they are only used to match them to the classes assigned by the algorithm. When the classifying algorithm shows sufficient performance it can be used to classify texts without any pre-assigned classes. This performance cannot be measured and it is therefor highly important that the training- and test corpus is sufficiently representative.

2.4 Regression

Predicting texts to a certain class can be highly useful and it can separate large collections of texts into relevant groups. However, in some situations it can be more suitable to predict a specific value and not a global range. Research done by Ghani and Simmons (2004), in which a system is proposed that can forecast end-prices of online auctions using text mining, is an example of such an approach. A specific value could be predicted: the end-price of an online auction. This type of prediction can be done using regression. Although Ghani and Simmons found that their regression results were not sufficient, other dataset may show more potential for specific value prediction.

Classifiers that solve regression problems analyze the relationship between the attribute- (independent) variables and the numerical target (dependent) variables (van Wijk, 2008). This can be done using a variety of techniques that predict a continuous numerical value. Regression is suitable in solving complex prediction problems and can handle both linear- and nonlinear relationships (Fayyad, Piatetsky-Shapiro & Smyth, 1996; van Wijk, 2008).

In general, regression tasks have a wider range of possible outcomes compared to classification using broad classes. Compared with classification, predictions for regression are "more sensitive to inaccurate probability estimates" (Frank, Trigg, Holmes & Witten, 1999).

This means that predictions made on certain statistical assumptions have a higher chance of failure due to the wider range of potential predictions. However, when predicting specific values is needed, regression can lead to a more useful and distinct system.

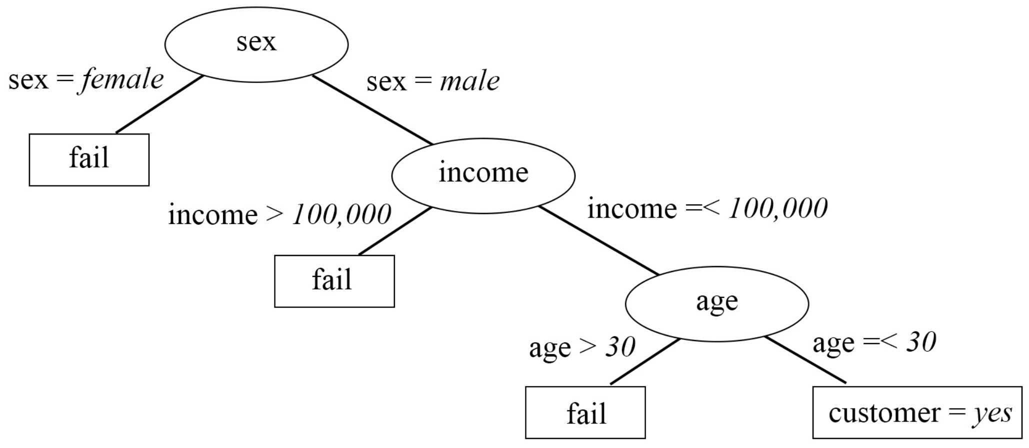
2.5 Machine Learning Algorithms

Machine learning techniques can be divided into supervised and unsupervised (Witten, Frank & Hall, 2011). Since our experiment takes the provided target variables into account in making predictions, thus is supervised, we will focus this section on supervised machine learning techniques. Witten, Frank and Hall identify the following supervised machine learning techniques: Decision Trees, Linear- and nonlinear regression and classification, Instance-based methods and Probabilistic Dependency Models.

2.5.1 Decision Trees

Decision trees are constructed in a recursive fashion (Witten, Frank & Hall, 2011). A *branch* is constructed for each value for an attribute. This branch leads to a decision node, which can also be divided into different branches. This step is repeated until the target variable is reached. A branch' end is called a *leaf* and expresses a value for the target variable. For example, a decision tree for the data in Table 1 (section 2.1) can lead to a branch as seen in Figure 1.

Figure 1. An example of a decision tree



Other than dividing the attributes according to a simple yes-or-no choice, decision trees can also their own construct rules. This is called a *covering* approach because the rule covers some of the attribute instances (Witten, Frank & Hall, 2011). The rules for 'income' and 'age' seen in Figure 1 provide an example.

A decision tree can be *pruned* in order to create faster estimations and this generally improves performance (Kwok & Carter, 2006). In short, pruning removes branches in the decision tree that do not provide vital information for the decision making task. This reduces complexity of the classifier, which leads to faster estimations.

Unlike the single Decision Tree classifier (DT), classifiers as *Gradient Boosting Regressor* (GBR) and *Random Forest Regressor* (RFR) operate using multiple decision trees. GBR builds a model in a stage-wise fashion based on a *loss function* (Friedman, 2002). At each stage a regression tree is fitted in order to assemble the whole model based on 'loss', or 'cost', associated with each decision. RFR creates multiple trees without using a stage-wise approach. Leaves represent the dependent variable and branches for independent variables are created during training. This creates a measure of importance and internal structure of the data and prediction is based on the combined output of all decision trees (Liaw & Wiener, 2002).

2.5.2 Linear- and nonlinear regression and classification

Linear and nonlinear algorithms require numerical input (Witten, Frank & Hall, 2011). Both use weights for each attribute variable in order to make predictions but the main difference between the linear- and nonlinear approach is the relational line for the expected values. A positive linear relation between an attribute variable *x* and target variable *y* indicates that *y* increases when *x* increases. A negative linear relation shows a decrease of *y* when *x* increases. A nonlinear relation between *x* and *y* can be highly complex and nonlinear algorithms can handle a variety of functions to calculate prediction values (Hocking, 2013). Among the different functions, logarithmic, exponential and Lorenz curves can be included. In short, successive approximations are used in order to fit the nonlinear model to the data.

When all input is numerical, both methods can predict exact values or a particular class. A variety of learning algorithms are based on linear- and nonlinear functions (Witten, Frank & Hall, 2011). For example, Linear Support Vector Machines (SVC for classification and SVR for regression), Kernel Ridge (Ridge) and Kernel Perceptron can be used for linearly separable problems. Ridge creates a regression model by optimizing its decision functions. This controls the complexity of the model and prevents overfitting (Malthouse, 1999).

Nonlinear learning algorithms include Multilayer Perceptron, SGD, and Nonlinear Support Vector Machine among others. SGD stands for Stochastic Gradient Descent and contains different classes of methods. This regressor updates its model with a decreasing robustness, or learning rate, in order to fine-tune its functions. Although SGD is fairly simple, it can be highly effective (Zhang, 2004). In principle, the loss minimization based SGD can be superior to other methods as the perceptron and SVM.

2.5.3 Instance-based methods

Instead of creating rules, instance-based learning methods (IBL) work directly from the examples themselves (Witten, Frank & Hall, 2011). These methods use the instances given in the training set to represent what is learned, rather than a set of rules or a decision tree.

IBL uses similarity functions on the generalized instances to yield graded matches in the data (Aha, Kibler & Albert, 1991). This means that instances are stored in memory and are separated by the IBL algorithm. All instances are represented as an attribute-value pair on an *n*-dimensional *instance space*. This requires a large amount of storage compared to non-instance-based methods and is often slow (Witten, Frank & Hall, 2011). The distance of separation between the given examples can vary and different curved boundaries are used for this task (e.g. linear or nonlinear). An example of this partitioning by boundaries (black rectangular) can be seen in Figure 2.

Figure 2. Partitioning of an instance space

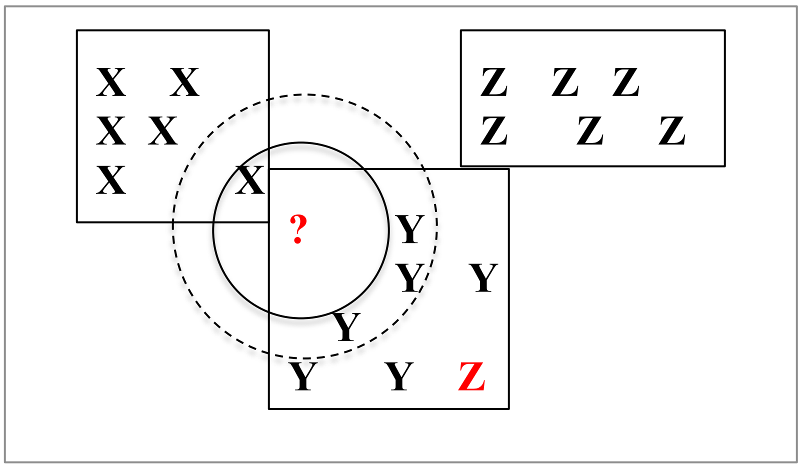


Figure 2 illustrates a two-dimensional instance space for the attribute-value pairs in classes *x, y* and *z*. Simple rectangular boundaries between the three classes are constructed and we see that one instance of *z* (marked in red) falls within the class for *y*. This means that these rectangular boundaries will falsely classify at least one *z* as *y*. More complex functions have the ability to exclude this instance with the creation of different boundaries.

An effective approach for IBL is k-Nearest Neighbors (kNN). kNN is used for classification and regression and is a non-parametric method (Altman, 1992). This means that kNN makes the assumption that the data does not contain any characteristic structure or distribution. An approximation of output is made on the basis of a majority of its neighbors, where *k* is the number of neighbors used for determination. The circles around the question mark in Figure 2 illustrate the principle of kNN: if $k = 1$, illustrated by the solid circle, the unknown instance will be classified as *x* since an instance of *x* is the nearest neighbor. However, if $k = 4$, illustrated by the dashed circle, we see that three of four nearest neighbors are instances of *y*. This will lead to a classification of *y* for the unknown instance. A larger *k* reduces the effect of noise in the data, but creates less distinction between different classes.

2.5.4 Probabilistic Dependency Models

Probabilistic Dependency Models (PDM), also known as statistical modeling, uses statistical dependencies between attribute variables (Witten, Frank & Hall, 2011). These statistical dependencies assign importance to the attribute variables on the basis of probability. One of the most used probability theory is Bayes' theorem. This theorem is based on Bayes' rule of conditional probability: the probability *P* of finding *A* considering evidence *B*:

$$P[A \mid B] = (P[B \mid A] * P[A]) \;/\; P[B]$$

The machine learning method based on Bayes' rule is called Naïve Bayes (NB). NB assumes independence 'naïvely' which lead to the possibility of multiplying probabilities. This approach is very fast and highly effective in classification and has the advantage that it is not vulnerable for missing values in the data since missing values do not affect probability.

NB is a popular approach for document classification and can be powerful when using *bag-of-words* models (Witten, Frank & Hall, 2011). For these models (see section 2.6.1), a modified NB form can be applied called *multinominal* Naïve Bayes. The multinomial Naïve Bayes classifier implements the Naïve Bayes algorithm for multinomial distributed data. This typically works for word vector counts but tf-idf vectors have been reported to work as well (Kibriya, Frank, Pfahringer & Holmes, 2005).

Generally, NB performs well on classification tasks and assigns classes on the basis of a maximum probability. NB has been reported to be more suitable for classification than regression (Frank, Trigg, Holmes & Witten, 1999).

2.6 Features

2.6.1 Word Representation

The ‘standard’ approach towards classifying text has been a simple word-based representation as a vector in a high dimensional space. This single word-based approach is also known as *bag-of-words (BOW)* where each dimension in the vector space corresponds to a word (Bekkerman & Allan, 2003). This relatively simple *BOW* remains highly effective despite the emerging of numerous more sophisticated techniques for document classification. Even though this approach generally produces the best results, the main flaw remains that this representation destroys any semantic relation between words. This means that the meaning of consecutive words is lost in the process. For example: the short phrase “not beautiful” is represented in the *BOW* as “not” and “beautiful”, thus one can suggest the document has a positive sentiment if the word “beautiful” is associated with the document. This suggestion is of course false when the whole short sentence is taken into consideration.

Researcher Lewis proposed a solution to the before mentioned problem as early as 1992. His idea of enriching the *BOW* representation involved the use of pairs of consequent words called *bigrams* instead of using single word *unigrams*. This *Bag-of-Bigrams* has shown to be a significant improvement in some cases of text categorization (Mladenic & Grobelnik, 1998) but other research also showed only marginal, or even a decrease in improvement. Researchers commonly refer to n-grams where *n* stands for the number of consequent words used for classification. Although there is no uniform picture on the improvements that bigrams, or n-grams, can offer text classification, there has been a major increase in computational power and algorithmic innovations in the past years (Bekkerman & Allan, 2003). This has lead to a new generation of text classification techniques that use both bigrams and unigrams and the use of multiple classifiers, also known as *two-staged classifiers* (Tan, Wang & Lee, 2002). Thus, a combination of the two can positively contribute to a classifiers performance.

2.6.2 Stemming

Words have many morphological variants (e.g. have, has, had). This can lead to recognition problems in term-matching algorithms when searching for variations of one lemma (Hull, 1995). Information Retrieval applications can therefore benefit from a reduction of morphological variants of the same semantic interpretation. The algorithmic transformation of term to root (*e.g.* 'beautiful' and 'beautifully' both become 'beautif') is called *stemming.*

A widely used algorithm based on suffix removal is the Porter stemmer (1980). This iterative algorithm removes the suffix using a few context-sensitive recoding rules. This leads to a drawback of this approach: the Porter stemmer ignores word meaning and this can lead to certain stemming errors in related- and nonrelated word pairs (Hull, 1995). In addition, stems are often not real words, which make them harder to interpret for any other purpose than information retrieval.

Even though stemming does have its drawbacks, it can greatly improve the performance of an information retrieval task when it is compared to an algorithm that does not use a stemming algorithm. A BOW model that includes stemming and term weighting has shown to be beneficial in a variety of studies (Hu, Downie & Ehmann, 2009; Mittermayer, 2004).

2.6.3 Term Weighting

In order to perform calculations on the document attribute variables, it is required to represent the frequency of each n-gram term in the document as a numeric value (Manning, Raghavan & Schutze, 2008; Jing, Hung & Shi, 2002). These corresponding frequencies are contained in the document vectors and can be seen as a dictionary within the document. The transformation of terms into features allows learning algorithms to be executed on the data in order to perform classification tasks. The Term Frequency (tf) is defined as follows:

$$\mathrm{tf}(t,d)$$

In this formula tf is the number of times that term t occurs in document d.

The next step is to calculate the *inverse document frequency* (idf). Idf calculates how common or rare a specific term is across all documents (Manning, Raghavan & Schutze, 2008; Salton & Yang, 1973; Jing, Hung & Shi, 2002). For example: the common term "the" is presumed to occur more frequently than a less common term. This does not mean it is more important, so idf calculates this by looking at the frequency of the word across all documents as follows:

$$\text{idf}(t) = \log\left(| D | \; / \; (\text{df}(t)) \right)$$

The formula divides the total number of documents $| D |$ with df(t), this is the total frequency of documents *DF* containing the term *t*. The logarithm of that quotient is than taken to calculate the idf. The logarithm normalizes high counts.

Both tf and idf are used to calculate the final weight of a term by taking the product of the two values (Salton & Yang, 1973):

$$\text{tf-idf}(t) = \text{tf}(t,d) * \text{idf}(t)$$

This tf-idf score helps probabilistic models to find relevant features across the whole dataset. There are various mathematical derivations of the tf-idf score, each modeled around its specific domain.

2.7 Evaluation

Machine learning algorithms need to be evaluated on performance (Witten, Frank & Hall, 2011). Evaluation is the key to determining how well our methods work and which method shows a better performance compared to others. We need to evaluate systematically and appropriately in order to construct valid conclusions from our results. To do this, we will use accepted and proven forms of performance evaluation seen in similar machine learning research.

2.7.1 Training and Test Sets

Our algorithms will be trained on our data, but in order to generalize these results we need to use a separate test set (Witten, Frank & Hall, 2011). Therefor, a training set is made from part of the dataset and a test set is made from the remaining data. Apart from these two sets, we also create a development set for the selection of machine learning algorithms for our experiment. More information on the development set can be found in section 3.3.1.

We test our classifier with a 20-80 ratio as seen in other machine learning research (Badrul, Sarwar, Karypis, Konstan & Riedl, 2000; Brindle et al, 2002). This means that of our data, 80% is used to train our classifier while 20% is used for testing its performance. Four additional runs will be done on the best performing configuration for classification (see section 3.3.3). This should allow us to predict the quality of our classifier sufficiently. This approach will insure that our test set is not part of our training set. This is to avoid overfitting the data and create overoptimistic and therefor misleading performance results.

In addition, it is advisable to test the trained algorithm on different training set proportions (Peterson, 2008). This is due to the effect that training set size can have on performance. As Peterson mentions in his research of 2008, algorithms like Neural Networks perform better on larger training sets. This effect was not seen using a linear method. In addition, research by Zhang, Jin, Yang and Hauptmann (2003) indicate that Support Vector Machines are also more efficient when the amount of training samples is large.

2.7.2 Evaluation of Classification

Appropriate metrics need to be selected in order to measure a systems performance. For classification tasks, the F^1 accuracy measure is a common performance metric (Hu, Downie & Ehmann, 2009; Rajman & Basancon, 1997; Jing, Huang & Shi, 2002). This metric is used in a variety of studies concerning classification and seems to be a good indication of performance (Beitzel, 2006).

The F^1 measure can be seen as the harmonic mean of *precision* and *recall* (Ash, 2013). Recall is a measurement of the proportion of correctly classified instances, where precision shows a measurement of the amount correctly classified instances out of all instances of that class (Witten, Frank & Hall, 2011). Precision and recall are calculated as follows:

$\text{Precision}_{\text{class}} = TP / (TP + FP)$

$\text{Recall}_{\text{class}} = TP / (TP + FN)$

Where:

TP: number of correctly classified instances to class

FP: number of falsely classified instances, as belonging to class

FN: number of instances belonging to class, not correctly classified

This means that, if all instances are classified to a certain class, recall will be 100% for that class. However, precision will be low. In a situation where only one instance is classified to a class and this is correct, precision will be 100% disregarding the not correctly classified instances belonging to this class. However, recall will be low in this situation.

The precision (*P*) and the recall (*R*) measurements are then used to calculate our F^1 measure. F^1 is calculated using the following formula:

$F^1_{\text{class}} = 2PR / (P + R)$

We use the micro averaged F^1 score for our experiment due to its equal weighting approach for each instance prediction (Ash, 2013; Beitzel, 2006). This means that no normalization is done regarding class proportion. Our interest goes out to the quality of our prediction and not the effect of each class' magnitude.

The F^1 score can range between 0 and 1, where 0 is the worst score and 1 the best score possible.

2.7.3 Evaluation of Regression

We have established that algorithms using regression are inherently different than ones using classification. Classification tasks predict to a certain class and this prediction can only be correct or incorrect. This is different to regression tasks where numeric values are predicted. Values predicted by the algorithm can closely approach the actual values, but less accurate predictions can also occur. Suitable evaluation metrics need to be selected to assess the technique's performance.

Similar studies, where price is predicted using machine learning algorithms for regression tasks, show two principal evaluation metrics: Root Mean Squared Error (RMSE) and R^2; the coefficient of determination (Peterson, 2008; Nguyen & Cripps, 2001; Dubin, 1998).

In short, the RMSE is calculated by taking the sum of all our predicted values (p_1), subtracted by the actual values (a_1) squared. This is then divided by the number of predicted cases (*n*). In order to provide the same dimension as the predicted value, the square root is taken (Willmott & Matsuura, 2005). The formula for RMSE is as follows:

$$\text{RMSE} = \sqrt{}\,(\,(\,\Sigma\,(\,(\,p_1 - a_1\,)^2\,)\,)\,/\,n\,)$$

RMSE is the most commonly used performance score for numeric prediction. It shows a weighted measurement of the difference between the value predicted and the actual value (Wijk, 2008). As mentioned by Willmot and Matsuura, larger errors have a stronger influence on RMSE than smaller errors because the predictor has been squared before the scores are averaged. The goal is to minimize RMSE in order to make predictions as close to the actual value. RMSE scores from our experiment can be compared side by side to examine the difference in error magnitude.

The use of the coefficient of determination (R^2) is well established for regression analysis (Nagelkerke, 2008). This is a useful measure of the success of predicting the target variable from the provided attribute variables. The coefficient of determination (R^2) is calculated using the following formula (Nagelkerke, 2008; Wijk, 2008):

$$R^2 = 1 - (\,SS_{residual} / SS_{total}\,)$$
$$SS_{residual} = \Sigma\,(\,a_1 - \hat{}a\,)^2$$
$$SS_{total} = \Sigma\,(\,a_1 - p_1\,)^2$$

This formula takes the Sum of Squares of the predicted values ($SS_{residual}$) and divides it by the Sum of Squares of all (SS_{total}). This value is then substracted from one.

$SS_{residual}$ is calculated by taking the sum of all squared differences between the prediction (p_1) from the actual value (a_1). SS_{total} is the sum of all squared differences between the actual value (a_1) from the overall mean of these values (^a).

When $R^2 = 0$, no is relationship found between the dependent and independent values, thus the system cannot make any systematic predictions. A perfect predictive system which predicts 100% of the cases correctly, would lead to $R^2 = 1$. A negative R^2 score indicates that the mean of the data provides a better fit in predicting the outcomes in comparison to our predictive system (Wooldridge, 2012).

2.8 Previous work on price prediction

Studies regarding the prediction of real estate selling price can be widely found (Knight, 2002; Dubin, 1998; Peterson, 2008; Nguyen & Cripps, 2001). However, to the best of our knowledge, no studies regarding real estate price using text mining are published. This means that comparison of our results have to be made with similar studies using different attribute variables, also known as dependent variables. These studies commonly use general real estate characteristics to make predictions. This method is generally referred as hedonic price prediction (Peterson, 2008). Characteristics such as the *number of rooms, number of bathrooms, basement-* and *fireplace presence* are used as attribute variables. In this section, we give a summarization of different studies done on price prediction using machine learning techniques. Since all studies found focus on selling price prediction, we will adapt these as comparison to our results on asking price prediction. The studies mentioned form the basis for our before mentioned hypotheses and provide comparison for results found in our experiment.

A study done by Dubin in 1998 shows a method of predicting real estate price based on its characteristics. This method estimates the price values using maximum likelihood regression and had the purpose to show that this approach can be used in predicting real estate prices. General real estate characteristics were used for attribute variables. In short, his method could make predictions with an R^2 of .731 (n = 1,000, RMSE = 81,117). Dubin notes that the model makes large prediction errors and was able to improve its model by adding geographical information. This new model used nearby real estate information as a trend to improve estimations to $R^2 = .745$ and RMSE = 76,278. He notes that this offered a significant improvement for RMSE but advised usage of this method in conjunction with standard techniques.

With digital technology rapidly improving, Nhuyen and Cripps published their paper on prediction of real estate prices in 2001. In their study, the researchers used Multiple Regression

Analysis (MRA) for price prediction. General house characteristics, similar to Dubin (1998) were used for their predictions and three different sized training sets were used. Nhuyen and Cripps obtained price fluctuation results, which is the difference between asking- and selling price, and suggest that MRA performs best on smaller sized training sets. The researchers do not provide concluding values for their measurements. In addition, they did found that the size of their training set had a severe effect on overall performance.

A study providing useful comparison for R^2 and RMSE measurements is done by Peterson (2008). A total of 46,467 residential properties, spanning the year 1999 to 2005, were collected for this study. In this research, Artificial Neural Networks (ANN) are used as a nonlinear modeling strategy and is compared to a linear regression. Both are used to predict real estate price using general house characteristics, as seen in previous research, for attribute variables. Due to the influence of training set size on performance, Peterson uses portions of 10%, 25%, 50% and 75% of his total dataset for training. His experiment shows that ANN generates less mispricing errors than linear regression. Average absolute pricing errors ranged from $857 at 10% of the data and $2,126 at 75% of the data. This led to 0.43% and 1.06% less error respectively. Peterson also notes that RMSE scores get larger with the size of the training group. Thus, more instances mean larger errors. R^2 scores are only provided for linear regression results and show acceptable performance (from $R^2 = .75$ at $n = 10\%$, to $R^2 = .73$ at $n = 75\%$). He concludes that pricing errors in linear models can be severe and are avoidable with nonlinear models like ANN.

Other than real estate, studies on price prediction in other domains can also be found. Ghani and Simmons (2004) did experiments with both classification and regression in order to predict end-prices for online auctions. Attribute variables as sellers rating, minimum price of the auction and auction dates were used (1,300 training- and 400 testing samples). Linear regression and Regression Trees were used to predict exact values for price. In classification, the researchers created classes with a five-dollar interval (10% window of the average price). Decision Trees and Neural Networks were used for classification tasks. In their results, Ghani and Simmons only provide RMSE measurements (Linear regression = 5.9, Decision Trees = 5.4). For classification they were able to achieve 96% accuracy when price direction was predicted (more or less). Decision Trees lead to 72% accuracy and 75% accuracy was achieved with Neural Networks. Accuracy calculations are not specified in this study.

3 Methodology

This section describes the dataset used for research, the transformations to the dataset and the various analyses performed. We transform our dataset in order to make it suitable for analysis. All words are stemmed using the Porter method and unigrams as well as bigrams are created in order to achieve the best possible results (Tan, Wang & Lee, 2002).

3.1 Real Estate Dataset

The dataset used for this research was constructed by Nanne van Noord in November 2012. The dataset consists of 3,269 comma-separated value (CSV) files, each containing data on a number of properties listed for sale. The amount of separate files is due to the fact that different files were created for each run to collect data. The data is collected from Zillow[1], a real estate website that operates in the United States of America. This means that the data is in English and that it only contains properties located in the USA.

The data has 28 different attributes and every line, or instance, in the file represents a house. It varies per house how many of the 28 attributes are listed and missing values in the data are replaced by '- -' or simply left blank. Natural language text strings are provided with opening- and closing quotation marks that enable an automatic system to separate comma's inside quotations from a comma that needs to separate different values in the data. Table 5 shows an example of information for one instance.

Table 5. Feature information on a single instance in the dataset

ID,	01008-2,
$/sqft,	$117,
sold_price,	"$219,500",
sale_duration,	,
Heating: ,	--,
extended_address,	"1963,10371 Angela Ave",
Listing website: ,	,
Year built: ,	1972,

[1] http://www.zillow.com

street,	22527 113th Pl SE,
Availability: ,	,
Parking: ,	--,
On Zillow: ,	,
Lease term: ,	,
Cooling: ,	Yes,
Lot: ,	"3,200 sq ft / 0.07 acres",
MLS #: ,	,
photodir,	,
Fireplace: ,	--,
description,	"This is a mobile / manufactured home. It is located at 22527 113th Pl SE Kent, Washington. This home is in the Kent School District. The nearest schools are Park Orchard Elementary School, Meridian Middle School and Kentridge High School.",
citystate,	"Kent, WA 22527",
price,	"$249,000",
photos,	,
Property website: ,	,
Sqft: ,	1224,
Baths: ,	2,
Type: ,	Single Family,
Beds: ,	4,
History	"Date @@@ 07/10/2012 Description @@@ Sold Price @@@ $219,500 Change @@@ 0.2% $/sqft @@@ $117 Date @@@ 05/26/2011 Description @@@ Listed for sale Price @@@ $249,000

3.2 Dataset Transformation

We need to make the data properly manageable and transparent. The data needs to be combined, structured and transformed in order to analyze it. Various Python scripts were developed and used to transform the data.

3.2.1 Feature Extraction & Transformation

In order to answer our before mentioned research questions we need the target variables, thus, the values we want to predict: selling price, asking price and price fluctuation. In addition, we need to incorporate the property description for our attribute variables. These variables are used for training our classifiers.

Reading the data using Python showed that the attributes *description* and *sold_price* were reliably listed thus they could be easily extracted. However, the attribute *asking price* is not provided in a large number of cases. Using only the houses that have this information available would reduce our dataset to a couple hundred of houses. Luckily, the attribute *history* provides a solution. In this attribute the selling- and listing dates with their corresponding prices are recorded in a single chunk of data. Extracting this information offers a challenge due to the ambiguity of the text but could be accomplished by identifying the part containing '@@@' and the prior text 'Listed for sale Price'.

Data transformation resulted in a single 124MB CSV file containing 286,189 instances. The length of the descriptions, with approximately 85 words each, explains the considerable size of our file. Descriptive statistics of all attribute variables can be found in Table 6. Skewness for selling price, asking price and price fluctuation are all within the -1 and 1 range (respectively .854, .935, -.778). These are acceptable level of skewness (Bulmer, 1979).

Table 6. Descriptive statistics

Attribute variables	N	Mean	Standard deviation
Description	286,189		
Selling price	268,126	186,611	124,006
Asking price	260,629	160,014	104,995
Price fluctuation	253,215	-22,682	100,611

We see that our standard deviation is fairly high, which means that our scores are widely spread (Chen, 2001; Lalys, Riffaud, Morandi & Jannin, 2010). This is not surprising since real estate prices vary greatly from one another.

3.2.2 Machine Learning Tools

There are a variety of options available for dataset analysis. We did trial experiments with the two Python modules Pattern[2] and SciKit-Learn[3] and the WEKA toolkit version 3.6.10 (Hall, Frank, Holmes, Pfahringer, Reutemann & Witten, 2009) in order to find a tool that was the best fit for our needs.

The SciKit-Learn module in Python provides the option to perform classification tasks as well as regression tasks. This module requires an n-dimensional array dataset, can handle multi-values attribute variables and is highly customizable. Due to the considerable usage among other data scientists, a substantial body of knowledge is also available.

The choice was made to use the SciKit-Learn module for dataset analysis. This tool seems to fit all our needs and provides the option to run our experiments on a remote Linux server. A remote server provided by the Tilburg University is used for these computations. Considering the high level of complexity, we were successful in creating an effective system to use for our experiments.

3.2.3 NumpPy Dataset Creation

The SciKit-Learn module uses n-dimensional array datasets for data analysis. A powerful Python package for scientific computing that can create these datasets is NumPy.

[2] http://www.clips.ua.ac.be/pattern
[3] http://www.scikit-learn.org

NumPy is a Python extension that can modify data into large, multi-dimensional arrays and matrices on which high-level mathematical functions can operate.

During the creation of our dataset we first transform our *description* attribute from a single string into a list of stemmed unigrams and bigrams. We perform this step using the SciKit-Learn CountVectorizer tool that uses a given *n-gram range (min_n, max_n)* to create our n-grams. As before mentioned, we will use bigrams as well as unigrams for the creation of our dataset. Also, we build our own tokenizer that stems the individual words using the NLTK Porter stemmer in order to increase performance (Hu, Downie & Ehmann, 2009; Mittermayer, 2004). This tokenizer is used within the CountVectorizer tool. The n-gram counts are then transformed into tf-idf scores using the SciKit-Learn tool TfidfTransformer. Our dataset is constructed in the following steps:

1. Read the csv file
2. From every row extract *Description* to a corpus list
3. From every row extract target attribute (e.g. *selling price*) to a list
4. Create stemmed n-grams from words
5. Vectorize n-gram counts
6. Keep a maximum of top n-attributes according to term frequency
7. Transform n-gram vectors into tfi-idf scores
8. Create a vocabulary of n-grams used for analysis
9. Create a dataset with NumPy containing the data: n-gram tf-idf scores, and the numerical targets variables

These steps create several 2-dimensional array datasets that can be used for analysis using SciKit-Learn. For example, a 2-dimensional array can be ({1,2,3,4},{4,5,6,7}), where the first list of numbers corresponds with the second list. In this example a difference of three is seen across each position in both lists. A different dataset is created for every target variable and every dataset can be saved for future analysis.

3.2.4 Creating Classes

Our features now include specific values for price, which can be used for regression tasks. However, if we want to perform classification tasks, we need to divide our values for prices into classes.

Since our attributes selling price and asking price are continuous numerical values, they cannot be divided into a natural set of classes. We therefor have to find a price interval that can be used to create our classes. In order to avoid classification on a large amount of classes we will divide our real estate prices into three classes: *low prices* (LOW: *price* < 150,000)*, medium prices* (MEDIUM: 150,000 < *price* < 300,000) *and high prices* (HIGH: *price* > 300,000). The class distribution can be seen in Table 7. We did not divide our instances equally per class due to the skewness in our distribution. This would mean that we would have two relatively small ranged classes for low prices and medium prices and one large ranged class for high prices. Although our classes are not uniform in size, we have deliberately chosen to create three classes that convey meaning in its price range.

Price fluctuation forms an exception, since this attribute can be divided into the three natural classes: *price decrease*, *equal price* and *price increase.* The class distribution can be seen in Table 7. Using these three classes will be more informative than an interval distribution because it predicts a specific direction in price shift.

Each instance in our dataset now encloses seven attributes after the creation of our classes: *description, selling price, asking price, price fluctuation, selling price class, asking price class* and *price fluctuation class*. Baseline values for classification will be taken from the largest class for each pricing indicator since a default prediction to that class will be correct for the highest percentage of cases.

Table 7. Descriptive statistics classes

Pricing indicator	Number of instances and percentage of total		
	LOW	MEDIUM	HIGH
Selling price	12,7217	91,250	49,659
268126	(48%)	(33%)	(19%)
Asking price	146,200	83,319	31,110
260629	(56%)	(32%)	(12%)
	DECREASE	EQUAL	INCREASE
Price fluctuation	132,766	3,529	116,920
253215	(52%)	(2%)	(46%)

3.3 Analysis

The analysis we select is highly related with the type of prediction we want to make. As mentioned before, machine learning techniques that use regression predicts a value, learning from the relation between the input variables and the target value. Machine learning techniques that use classification predict a class, learning from the relationship between the attribute variables and the target variables. In our experiment we are interested in the possibility to predict pricing indicators based on description. Our experiment will show how well both methods perform on our dataset.

3.3.1 Classifier Selection

To select our classifiers for regression and classification tasks, we created a development set. This set includes 100 attributes (n-grams), was trained on 1,000 instances and was tested with 250 instances using a variety of classifiers. This development set gave us the opportunity to compare a range of classifiers in a manageable amount of time, relative to using 80% of the total dataset. This development set was created from data outside our experiment in order to prevent overfitting. Overfitting can occur when a classifier is trained and tested on overlapping data (Witten, Frank & Hall, 2011). This gives a high chance of finding similar results for training and testing and can lead to invalid conclusions.

Experiments on our development set roughly showed which classifiers outperformed others. We then could conduct our final analysis with the classifiers most suitable for our dataset. It is wise to use multiple approaches to classify our dataset in search for best performance (Witten, Frank & Hall, 2011). More information on the machine learning algorithms presented below can be found in section 2.5.

Based on the testing using our development set, our classifiers were chosen. These performed best out of all classifiers provided by the SciKit-Learn module (see section 3.2.2). We have chosen the following five classifiers for our classification task: *Support Vector Classification* (SVC), *Linear SVC* (LinSVC), *K-Nearest Neighbor Classifier* (kNN), *Multinominal Naïve Bayes* (MNB) and *Decision Tree Classifier* (DT). For regression, the following five classifiers were chosen for our experiment: *Gradient Boosting Regressor* (GBR), *Suppport Vector Regressor* (SVR), *Ridge*, *Random Forest Regressor* (RFR) and *SGD Regressor* (SGD).

3.3.2 Number of n-grams

For our analysis we vectorize our corpus into stemmed n-grams as mentioned before and we create six datasets with a varying amount of top n-attributes: 10, 50, 100, 150, 200 and 250. We use these six datasets to find which amount of n-grams shows the best performance. Dimensionality reduction will be beneficial for our study for two reasons. We limit the amount of n-grams to a maximum of 250 in order to restrict the time of computing and because our trial run experiments indicated that performance increase ceased around 150 n-grams or more.

3.3.3 Training & Testing

We train our classifier with a varying amount of instances: 1,000, 10,000, 50,000 and 100,000. We use these increments so we can learn what our model does with a relatively small- and large amount of instances. Our task would be less manageable if we took standard increments of 1000 instances since this would provide a hundred different datasets. Large increments would not provide results for smaller datasets.

This allows us to see learning curves for our classifiers regarding its performance using various amounts of data. In short, the effect of sample size on performance. If an increase of instances has an effect on the performance we should be able to identify its effect using the determined distribution. In order to save enough instances for testing and to keep the computing times manageable, we use 100,000 as the maximum amount of instances, which already takes up to a full two days of computing when using 250 n-grams. This is also the main reason we did not use multiple runs since this would lead to almost a full month of computing. However, we do use four separate runs for classification on our best performing configurations in order to calculate significance from baseline. This is done for selling price, asking price and price fluctuation.

We test our classifier with a 20-80 ratio as seen in other machine learning research (Badrul, Sarwar, Karypis, Konstan &Riedl, 2000; Brindle et al, 2002). A training set of 10,000 instances would therefore have a test set of 2500 instances, so a total number of 12,500 instances are used for this task.

To summarize, our analysis is conducted according to the following steps:

1. Create or load NumPy datasets
2. Split dataset into training set and test set (e.g. 80% training, 20% testing)
3. Fit the training data into an estimator using the SciKit-Learn module. Fitting is done using each regression algorithm
4. Calculate performance indicators using the SciKit-Learn module

3.4 Performance Indicators

To assess the quality of our predictive system, we will use the F^1 metric to evaluate classification performance and R^2 and Root Mean Squared Error (RMSE) for regression. All metrics are calculated by the SciKit-Learn module.

4 Results

In this section we will present the results of our experiment. In our experiment we used classification and regression methods, separately involving all pricing indicators: selling price, asking price and price fluctuation. For classification, we will reveal our best performing configurations for F^1. For regression, we present our best performing configurations for R^2 and the corresponding RMSE. Detailed results for each pricing indicator are also presented.

All graphs in this section maintain similar horizontal axis for comparison reasons. Horizontal axis starts at 1,000 instances for training with 10 n-grams (*1Kx10*) and ends at 100,000 instances for training with 250 n-grams (*100Kx250*). As before mentioned in section 3.3, four amounts of training instances were used, each divided into six amounts of n-grams used for analysis. This adds up to 24 measurement points on the horizontal axis. Least n-gram configurations are chosen when equal values show for best performance.

4.1 Classification Results

Table 8 shows the best performing configurations for classification. Firstly, MNB shows the best performance on for the prediction of selling price (F^1 = .652) while MNB, SVC and LinSVC show equal best performance scores predicting asking price (F^1 = .682). For the prediction of price fluctuation, the LinSVC classifier achieves best results (F^1 = .617).

Secondly, we can see that our system achieves results higher than our baseline values for all our pricing indicators. Our system scores selling price 38% higher than our baseline value, 27% higher than our asking price baseline and 18% higher than our price fluctuation baseline. All improvements are significant compared with our baselines ($p < .001$). Significance was calculated using the SPSS paired t-test on four separate runs using the best performing configurations.

Best performance is achieved at different configurations across all pricing indicators. Tables for all F^1 results can be found in appendices A to C.

Table 8. Best Performing Classification Configurations

Pricing Indicator	Classifier	$N_{classes}$	Variables	F^1	Baseline	Difference
Selling Price	MNB	3	250 n-grams, 1,000 instances	.652**	.474 (LOW)	.178 (38%)
Asking Price	MNB & SVC & LinSVC	3	10 n-grams, 100,000 instances	.682**	.561 (LOW)	.121 (27%)
Price Fluctuation	LinSVC	3	50 n-grams, 10,000 instances	.617**	.524 (DECR.)	.093 (18%)

*Note: * = significant with p < .05 ** = significant with p < .001*

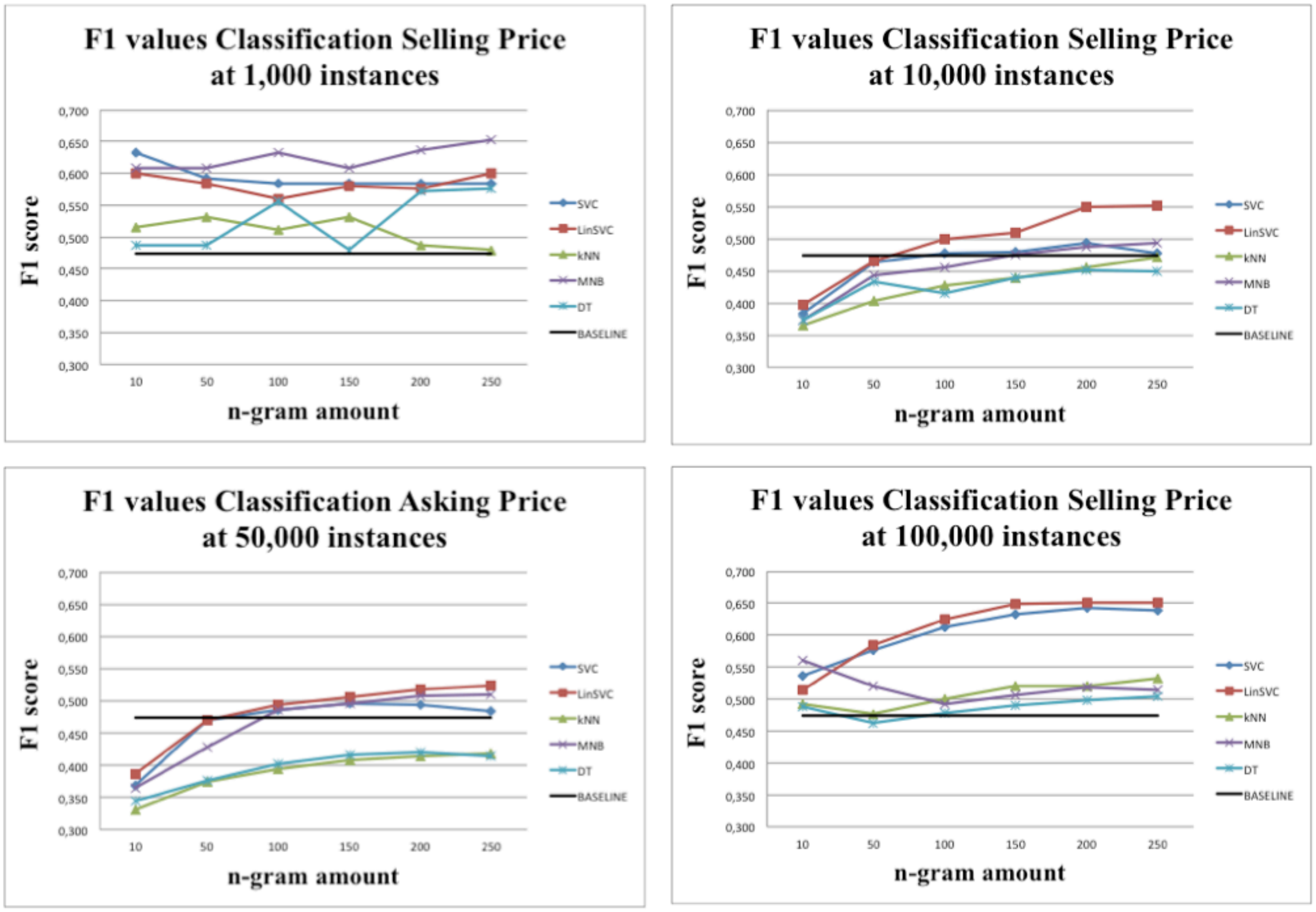
4.1.1 Selling Price Classification

Figure 3 shows all classification results for selling price prediction. The highest F^1 score is achieved by MNB using 250 n-grams at 1,000 training instances (F^1 = .652), which outperforms our default class (LOW) baseline of .474 with 38% ($p < .001$).

We see that best LinSVC performance is achieved using 150 n-grams at 100,000 instances (F^1 = .650). Although F^1 results for SVC (F^1 = .642) are near the results obtained using LinSVC, it cannot outperform its linear variant.

kNN and DT are the two lowest performing classifiers with highest F^1 scores of .532 and .568 respectively.

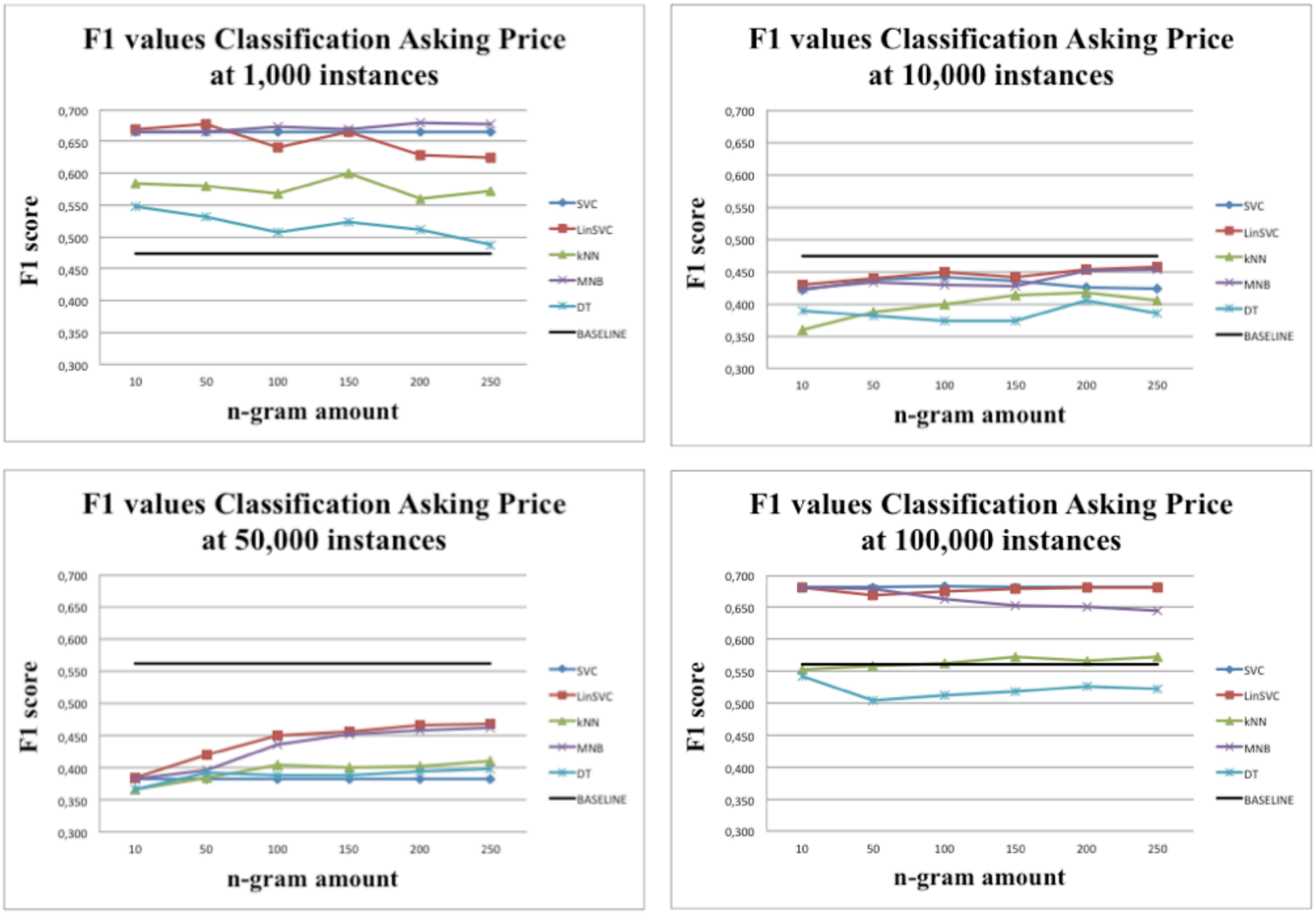
Figure 3. Selling Price F^1 Charts at each instance amount



4.1.2 Asking Price Classification

All classification results for the prediction of asking price are shown in Figure 4. SVC, LinSVC and MNB outperform kNN and DT using 10 n-grams at 100,000 training instances (F^1 = .682). Achieved results surpass our default class (LOW) baseline of .561 with 27% ($p < .001$). Our findings show that SVC, LinSVC and MNB all show higher performance than our baseline when using 1,000 or 100,000 instances. All classifiers perform under our default baseline when using 10,000 or 50,000 instances.

Figure 4. Asking Price F^1 Charts at each instance amount



We can see in Figure 4 that DT (Decision Tree classifier) cannot outperform our baseline at any configuration (F^1 = .548). Finally, increasing the amount of n-grams does not seem to have a general effect on F^1 scores.

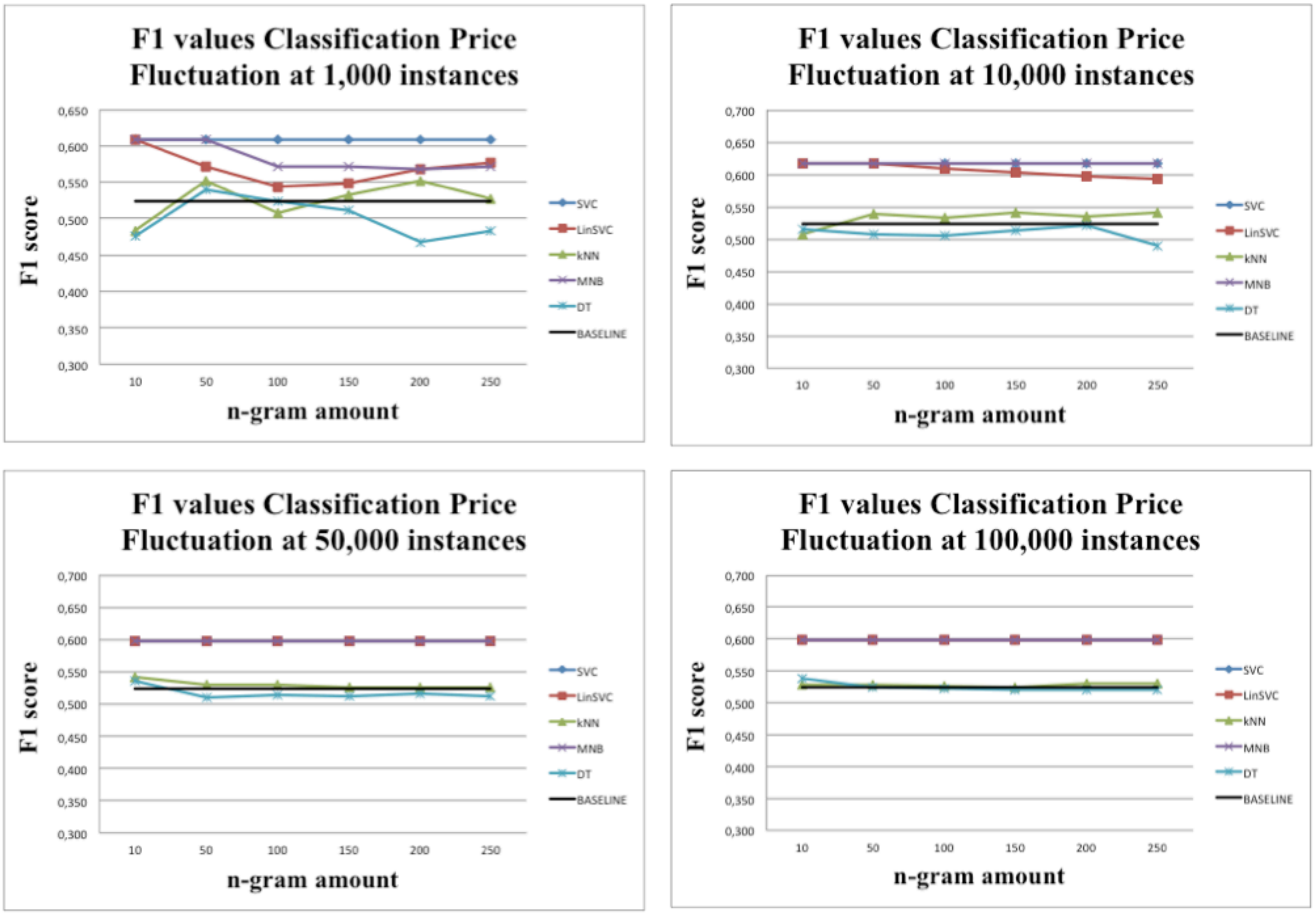
4.1.3 Price Fluctuation Classification

Figure 5 shows all classification results for the prediction of price fluctuation. Our default class (DECREASE) gives us a baseline of .524. We can clearly see that SVC, LinSVC and MNB all achieve F^1 scores over baseline for all configurations.

Best performing classifier is LinSVC with an F^1 score of .617 (50 n-grams, 10,000 instances), which is 18% higher ($p < .001$) than our set baseline.

kNN and DT both consistently show lower F^1 scores with highest F^1 achieved of .552 and .540 respectively. Increasing the amount of n-grams does not seem to have any effect on performance for any classifier. It must be noted that we see a all graphs appear to be highly flat in nature and show little variation in results.

Figure 5. Price Fluctuation F^1 Charts at each instance amount



4.2 Regression Results

Best performing configurations for regression can be seen in Table 9. Although R^2 scores vary greatly for our three pricing indicators, we can see that the Gradient Boost Regressor performs best among all classifiers.

For the prediction of selling price, best performance is achieved using 200 n-grams at 10,000 instances ($R^2 = .303$). Asking price prediction achieves a highest R^2 score of .124 at the same configuration as selling price prediction, while price fluctuation achieves its best performance using 250 n-grams at 50,000 instances ($R^2 = .065$).

RMSE shows us the weighted average difference between the prediction and true value. On average, selling price is predicted with a RMSE of 139,177 and the predicted asking price has a RMSE of 158,106. Price fluctuation was predicted with an average RMSE of 122,823.

Table 9. Best Performing Regression Configurations

Pricing Indicator	**Classifier**	**Variables**	**R^2**	**RMSE**
Selling Price	SGD	200 n-grams, 10,000 instances	.303	139,177
Asking Price	SGD	250 n-grams, 10,000 instances	.124	158,106
Price Fluctuation	SGD	250 n-grams, 50,000 instances	.065	122,823

SVR shows worst performance for regression tasks. Across all pricing indicators MNB shows R^2 scores close to zero or lower. Tables for all R^2 and RMSE results can be found in appendices D to I.

4.2.1 Selling Price Regression

Figure 6 presents the R^2 scores for selling price prediction at each different configuration. Figure 7 presents the RMSE values for selling price prediction.

The SGD classifier shows the best overall performance when trained with 200 features and 10,000 instances ($R^2 = .303$; RMSE = 139,177). Figure 6 shows R^2 scores for Ridge comparable to SGD when using 10,000 instances or more. Best performance by Ridge is achieved using 250 n-grams at 10,000 instances ($R^2 = .267$; RMSE = 138,737).

In addition, increasing the amount of features seems to results in a higher R^2 score. This is relative to the amount of instances used for training. More features must be used for larger sets of training data to obtain the same R^2 score.

The classifier that performed worst on our dataset is SVR. This classifier does not manage to produce positive R^2 scores.

Finally, we see in Figure 7 that increasing the amount of instances leads to a larger RMSE. This effect can be seen until 100,000 instances are used for training. RMSE decreases drastically when using 100,000 instances compared to 50,000 instances.

Figure 6. Selling Price R^2 Charts at each instance amount

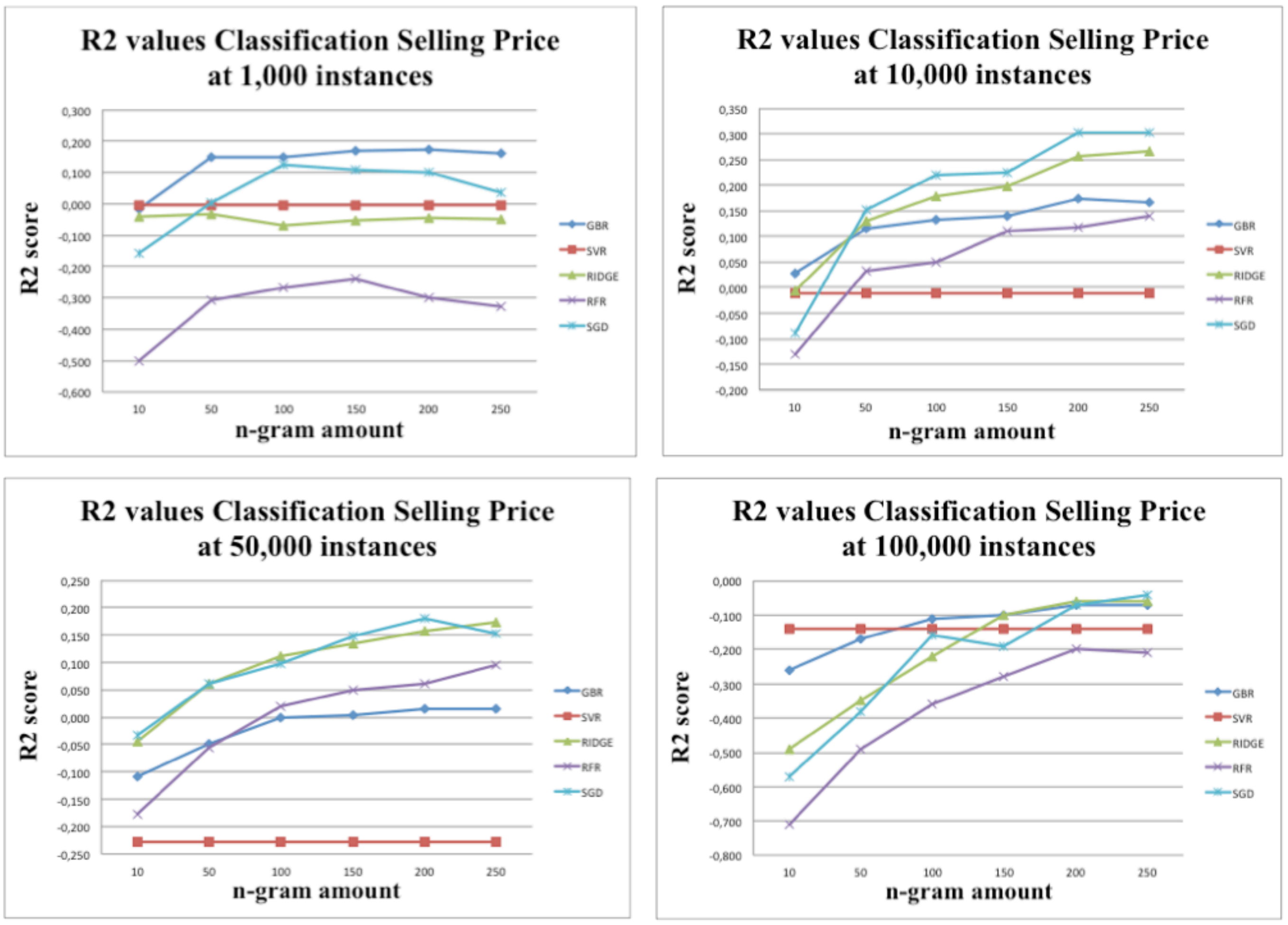
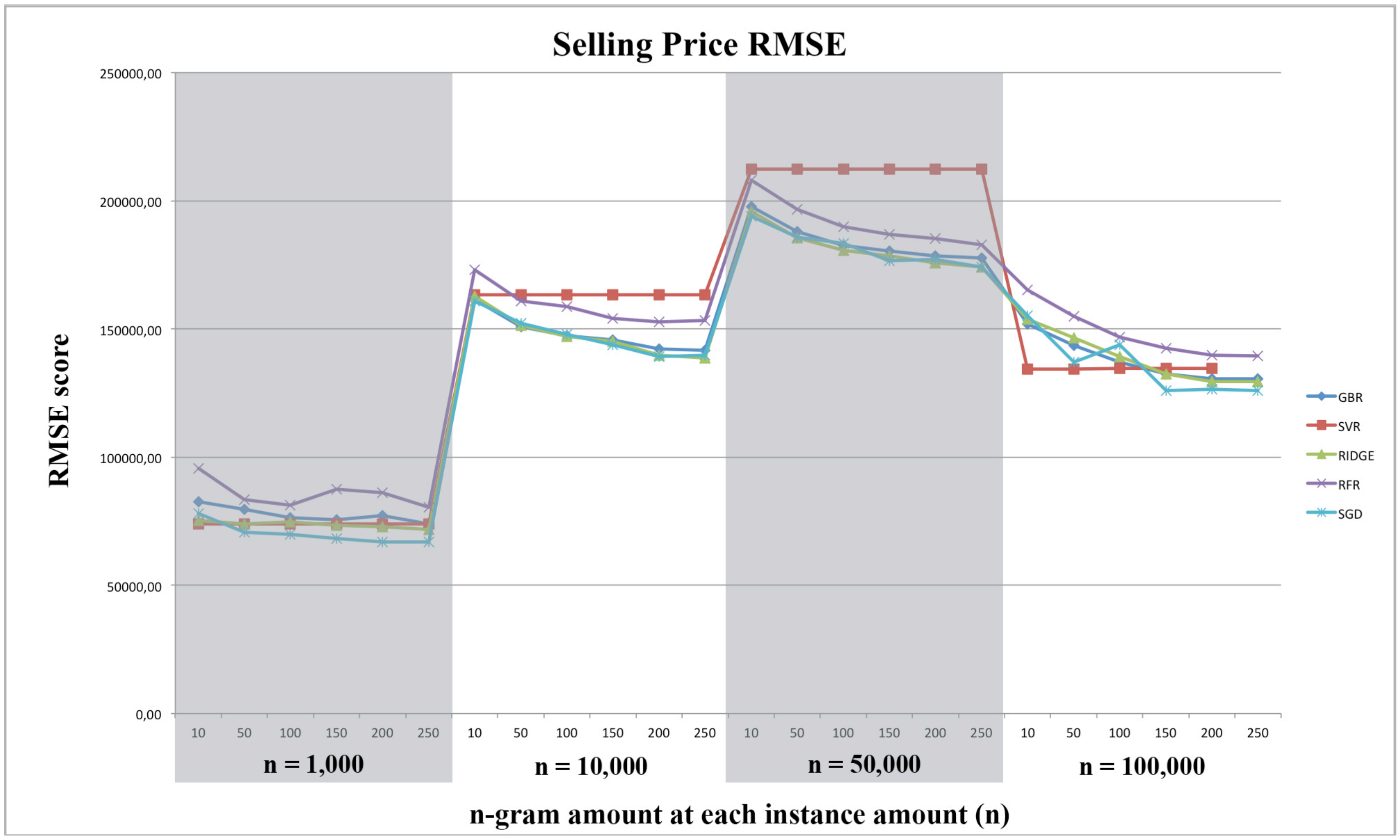


Figure 7. Selling Price RMSE Chart, n-gram amount at each instance amount



4.2.2 Asking Price Regression

Figure 8 presents the R^2 scores for asking price prediction at each different configuration while Figure 9 represents the results for RMSE.

The SGD classifier shows the best overall performance when trained with 250 features and 10,000 instances (R^2 = .124, RMSE = 158,106). This result is almost matched by the Ridge classifier at the same configuration (R^2 = .109, RMSE = 158,929). Both SVR and RFR show poor R^2 results: near zero or negative.

The effect of increasing the amount of n-grams is less apparent than R^2 scores for the prediction of selling price.

Although we can see that R^2 scores generally increase when more n-grams are used, it seems that this effect is less apparent compared to results for selling price prediction.

Finally, we see in Figure 9 that increasing the amount of instances leads to a larger RMSE until 100,000 instances are used for training. As seen for selling price RMSE scores, we notice a RMSE decrease when using 100,000 instances.

Figure 8. Asking Price R^2 Charts at each instance amount

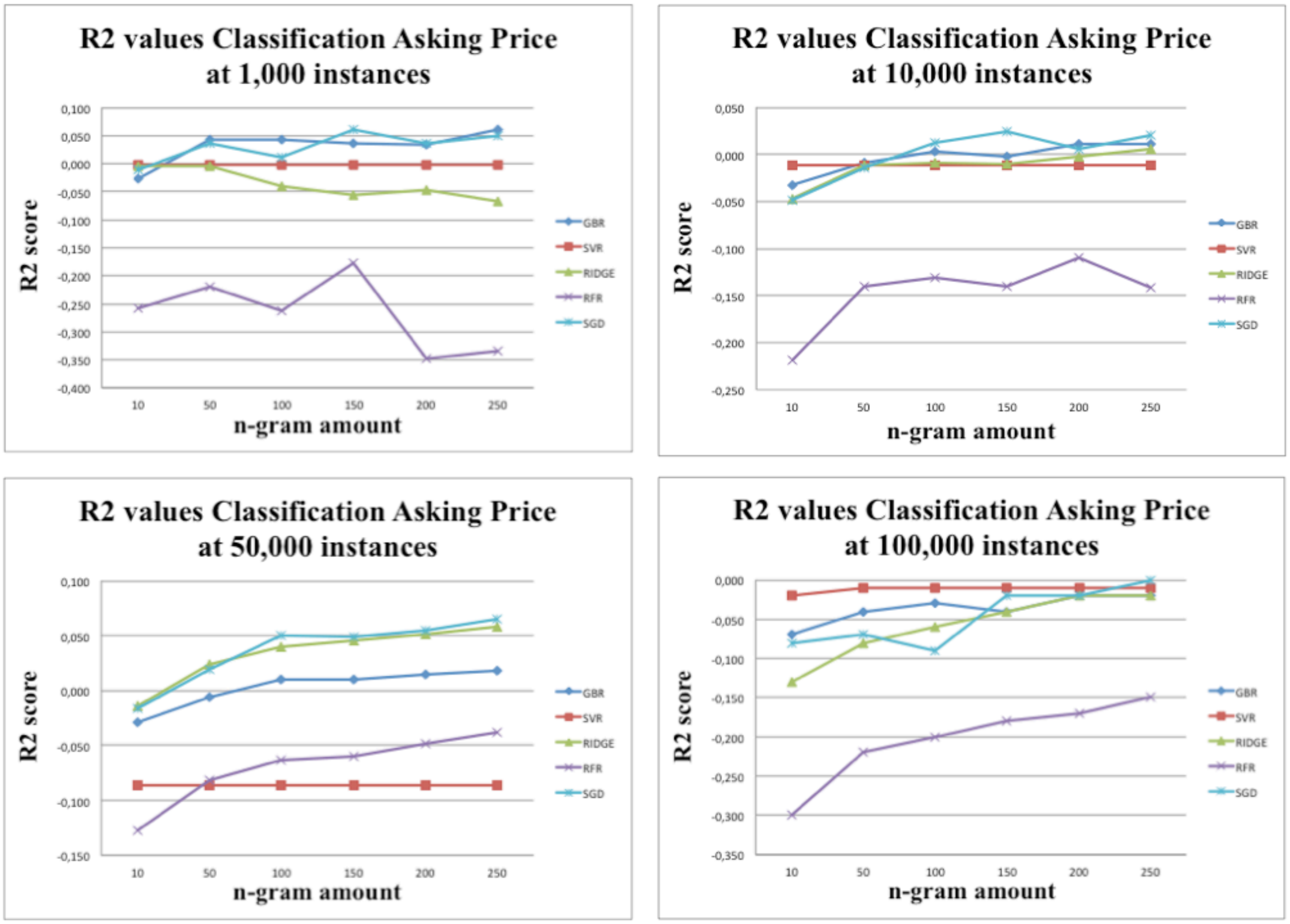
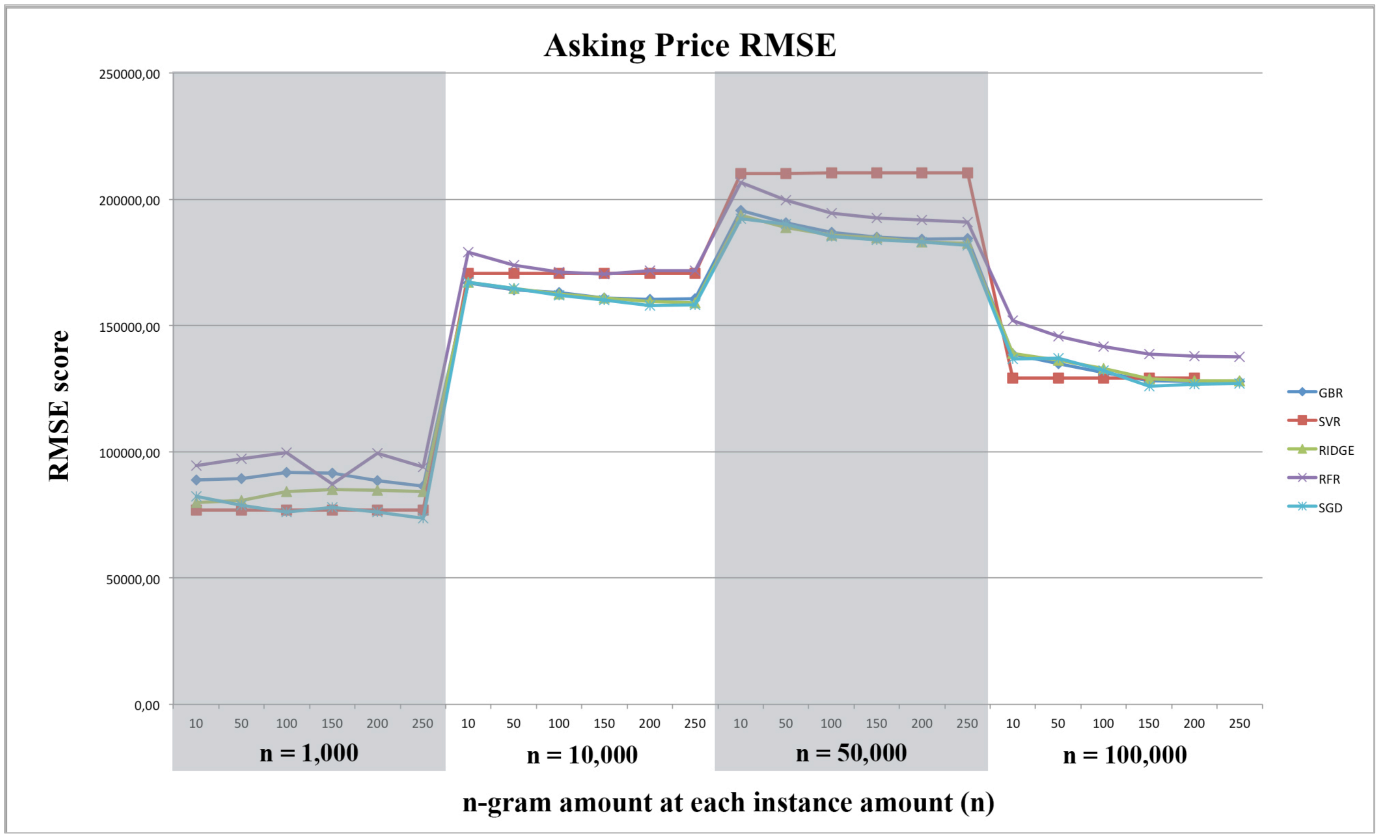


Figure 9. Asking Price RMSE Chart, n-gram amount at each instance amount



4.2.3 Price Fluctuation Regression

Figure 10 presents the R^2 scores for price fluctuation prediction at each different configuration. Figure 11 presents the RMSE values for price fluctuation prediction.

The SGD classifier shows the best overall performance when trained with 250 features and 50,000 instances (R^2 = .065, RMSE = 122,823). At the same configuration, the Ridge classifier shows its best performance with a R^2 score of .058 (RMSE = 122,987) while GBR performance peaks at 1,000 instances using 250 n-grams (R^2 = .060; RMSE = 76,352).

Increasing the amount of features does not seem to have a clear increasing effect on R^2 score. Figure 10 also shows that SVR and RFR achieve lowest performance results among all classifiers: both cannot produce positive R^2 scores.

As can be seen in Figure 11, RMSE increases from 1,000 to 10,000 instances used, but RMSE does not show an increase at 50,000 instances. Compared to 50,000 instances, we can also see a decrease of RMSE at 100,000 instances.

Figure 10. Price Fluctuation R^2 Charts at each instance amount

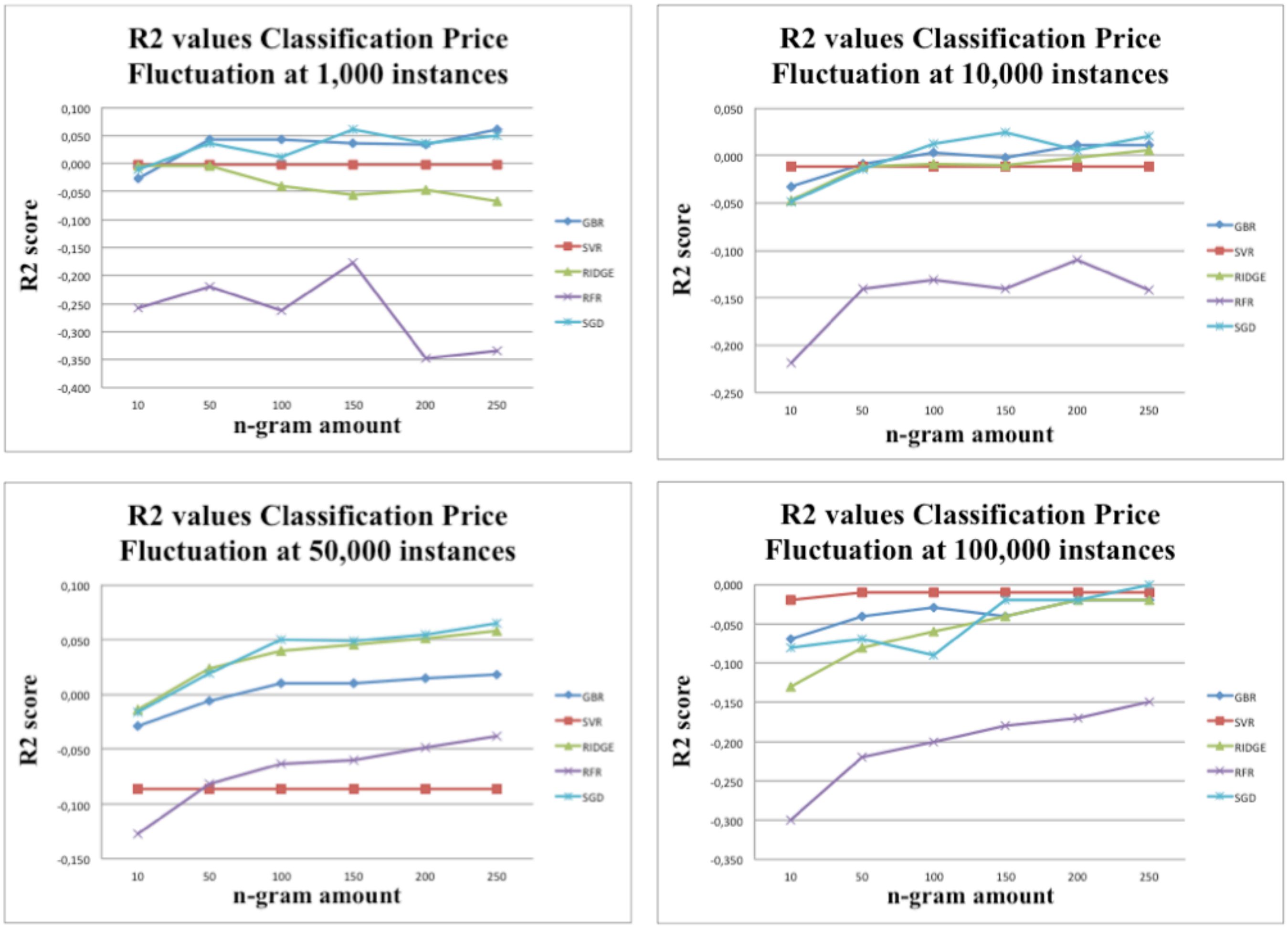
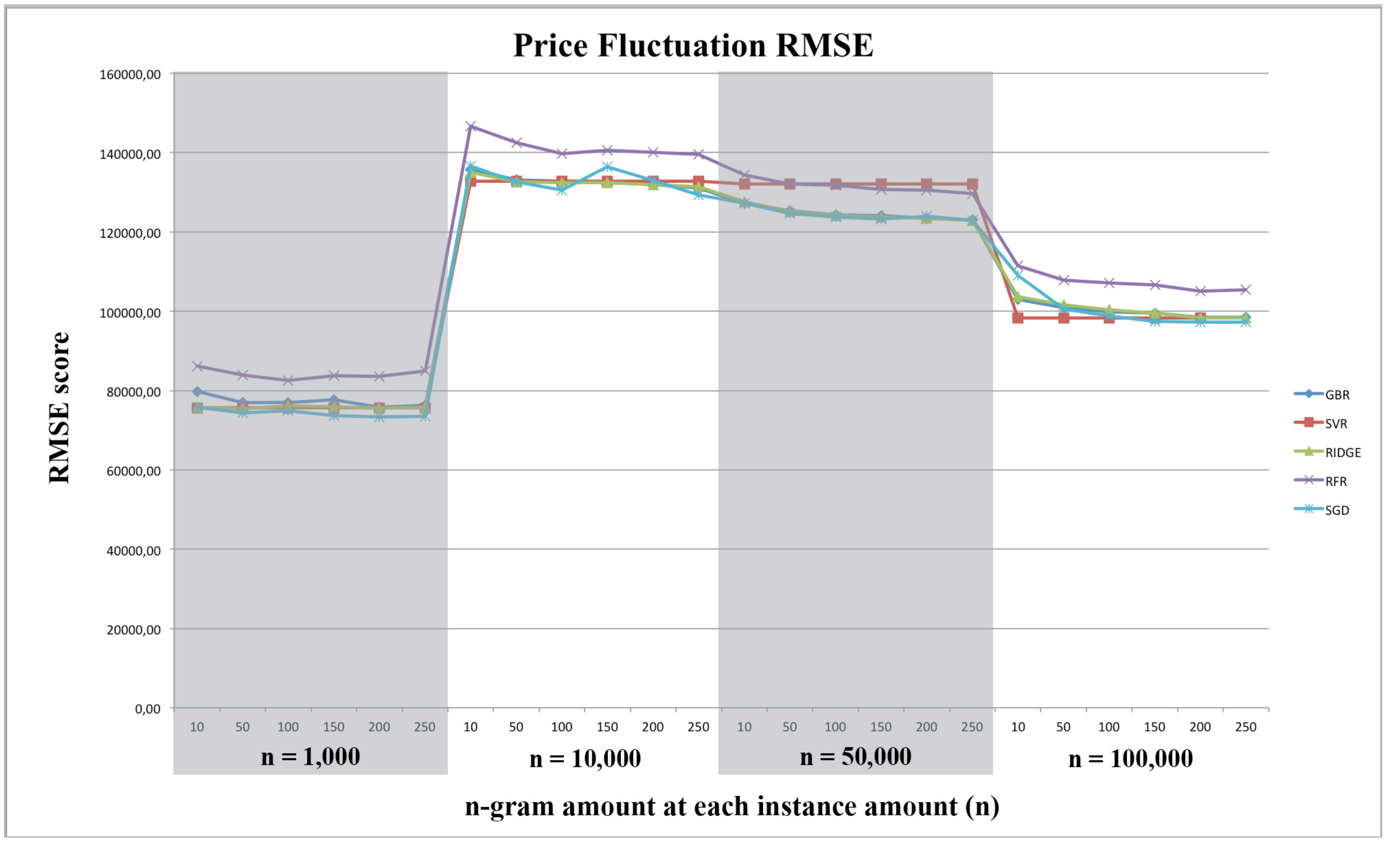


Figure 11. Price Fluctuation RMSE Chart, n-gram amount at each instance amount



4.3 Learning Curves

Increasing the amount of n-grams seems to result in a higher R^2 and F^1 scores, specifically noticeable for results on selling price prediction. This increase is relative to the amount of features (n-grams) used for training. For example, an approximate $R^2 = .15$ is achieved by SGD for the prediction of selling price using 150 features at 10,000 instances. This performance is not matched until 250 features are used at 50,000 instances.

Curve Estimation Analysis in combination with ANOVA is done using the two best performing algorithms for the prediction of selling price: MNB and LinSVC for classification tasks and SGD and Ridge for regression tasks. This analysis will determine if the increase of our performance indicators remains constant (linear curve) or reduces (logarithmic curve).

Shown in Table 10, we see that there is a significant logarithmic curve in our F^1 increase when using 10,000 or 50,000 training instances for MNB. For LinSVC we see a significant logarithmic curve in our F[1] increase when using 10,000 training instances or more. Table 10 also shows a less significant linear curve when a logarithmic curve ($p < .001$) is found.

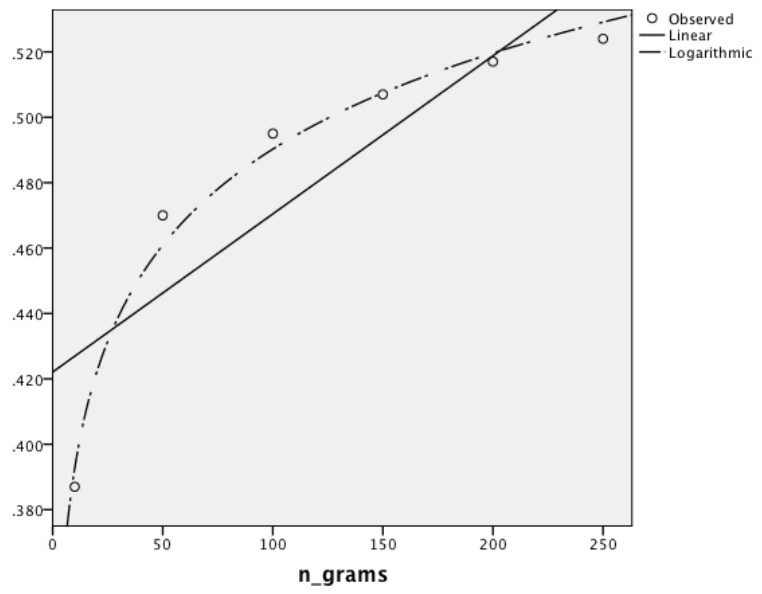
For illustration purposes, Figure 12 shows the fit for the linear- and logarithmic curve at 50,000 training instances for LinSVC.

Table 10. Curve Estimation & ANOVA, F[1] Selling Price

	MNB		LinSVC	
Instances	*Linear*	*Logarithmic*	*Linear*	*Logarithmic*
1,000	.643	.455	.000	.109
10,000	.800 *	.989 **	.891 *	.975 **
50,000	.801 *	.980 **	.747 *	.987 **
100,000	.233	.598	.752 *	.978 **

*Note: * = significant with p < .05 ** = significant with p < .001*

Figure 12. Curve Estimation for F[1], Selling Price LinSVC at 50,000 instances



In the results of our regression tasks shown in Table 11, we can see a significant logarithmic curve across all amounts of instances for SGD results. For Ridge, a significant logarithmic curve can only be seen when using 10,000 instances or more.

Like curve estimation results for classification tasks, a less significant linear curve can also be found in our results when a logarithmic curve (p < .001) is shown.

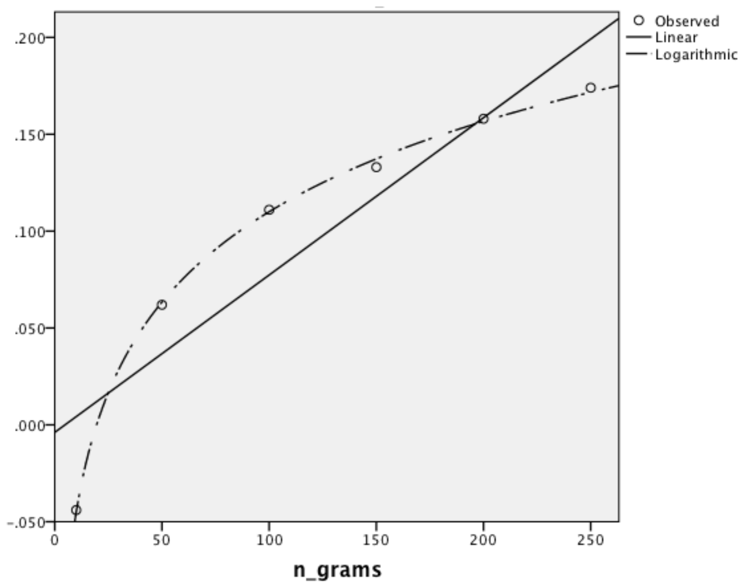
Figure 13 shows the fit for the linear- and logarithmic curve at 50,000 training instances for Ridge.

Table 11. Curve Estimation & ANOVA, R^2 Selling Price

	SGD		Ridge	
Instances	*Linear*	*Logarithmic*	*Linear*	*Logarithmic*
1,000	.358	.732 *	.057	.112
10,000	.754 *	.976 **	.859 *	.986 **
50,000	.811 *	.958 **	.846 *	.999 **
100,000	.856 *	.956 **	.894 *	.952 **

*Note: * = significant with p < .05 ** = significant with p < .001*

Figure 13. Curve Estimation for R^2, Selling Price Ridge at 50,000 instances



In addition, to test if the other pricing indicators on the effect of n-gram amount and if instance amount has influence on our performance, we conduct a Curve Estimation Analysis as well as a Correlation Analysis in SPSS. These analyses are done for all six best performing classifiers for classification and regression that are shown in Table 8 and Table 9. Groups are made for every n-gram amount in order to exclude the effect of n-grams on our results.

We can conclude that increasing the amount of n-grams mainly has a linear effect on asking price performance and no general effect on price fluctuation performance. Secondly, the amount of instances used for prediction has no significant fit on curve estimation or any correlating effect on performance. The outcomes of our Curve Estimation Analysis, ANOVA and Correlation Analysis with corresponding values for significance can be found in Appendix J.

5 Discussion and Conclusion

In our experiment, we succeeded to predict all pricing indicators above baseline using classification. Best results are all achieved using different configurations and classifiers. Our experiment on predicting pricing indicators using regression showed different results across our pricing indicators. The SGD classifier performs best overall with decent results for selling price prediction. As Zang (2004) mentioned: SGD is fairly robust but can be highly effective due to its loss minimization principle. Albeit we have done our utmost to use classifiers best suited for our approach, we must note that there can still be other classifiers and configurations that can achieve higher performance. We recommend that future research should take this into account.

5.1 RQ1: Predicting Selling Price

For the prediction of selling price using classification we found that Multiple Naïve Bayes (MNB) achieves best performance from all our researched classifiers. This is not surprising since Kibriya, Frank, Pfahringer and Holmes (2005) have reported that MNB in combination with tf-idf vectors can perform well. What is surprising is our baseline improvement. Our experiment resulted in predictions with an F^1 score significantly higher (38%) than our baseline. This is done using a relatively large amount of n-grams for the amount of instances (250 n-grams, 1,000 instances). Although MNB shows the best fit on our dataset, we can also see that this algorithm is less suited for larger sets of instances. We believe that this is largely due to the assignment of importance to the attribute variables on the basis of probability: large datasets will be harder to estimate. Results for Support Vector Classification (SVC) and Linear Support Vector Classification (LinSVC) do not surpass best performance by MNB, but we see that these classifiers perform more than sufficient on smaller as well as larger sets of instances. This makes these classifiers usable in a wider range of instance amounts than MNB. Due to computing power limitations it is important that algorithms are able to make decent predictions on large sets of instances while using a relatively small amount of n-grams.

Our numerical prediction for selling price using regression shows a performance of $R^2 = .303$ (RMSE = 139,177). This means that our best model explained 30% of the total variance in the data (Woolridge, 2012). On average, the RMSE of our estimator is 139,177, which is relatively high. We suspect that the high range of real estate selling prices leads to a number of large prediction errors. As mentioned before, large errors have a relatively high influence on

RMSE (Willmot & Matsuura, 2005). Comparing the different RMSE results, we cannot see much variation between the different classifiers but we do see a distinct drop when using 100,000 instances. We suspect that this drop is due to the decrease of large errors relative to the amount of smaller errors. This indicates that we found a point where our model reduces the strong influence of large errors in selling price prediction.

Compared with the R^2 of .75 (n = 34,850) and pricing error of RMSE = 2,126 obtained by Peterson (2008), our prediction model achieves a poorer fit on the data. Peterson's results are based on a hedonic pricing model, which is a widely used and less complex model to predict selling price in real estate. Dubin's results from 1998 are also based on a hedonic pricing model, which shows a R^2 of .73 (n = 1,000) and RMSE of 81,117. Compared to our results we can clearly that the hedonic pricing model more than doubles our obtained R^2 scores. Our results cannot match the hedonic pricing model but given the high level of complexity and difficulty of our text mining approach we recognize this as an insufficient, but nonetheless decent result.

5.2 RQ2: Predicting Asking Price

For the prediction of asking price classes we found that three classifiers show highest results on performance: MNB, LinSVC and SVC. These achieved results lead to a 22% improvement for F^1, significantly higher than baseline. This is achieved while using a small amount of n-grams (10) to predict a large set of instances (100,000). F^1 results for all classifiers fall below baseline when making predictions with 10,000 and 50,000 instances. We cannot find a specific reason for these outcomes. Our outcomes show that our classification model can provide reasonable improvement over a default baseline model while using a small amount of n-grams. Our drop in performance aside, results also show that our model can handle small, as well as large sets of instances.

Due to the lack of comparable research, we will use the before mentioned results for selling price prediction, using a hedonic pricing model, as the baseline for our regression model. We can clearly see that the results obtained by Peterson (2008) and Dubin (1998) are superior to our performance results. With a highest R^2 score of .124 achieved by SGD and a relatively high RMSE of 158,106 we cannot provide a challenging alternative to the standard hedonic approach. As seen in the results for numerical selling price prediction, we also see a RMSE decrease when using 100,000 instances. This indicates a similar decrease in influence of large prediction errors (Willmot & Matsuura, 2005).

5.3 RQ3: Predicting Price Fluctuation

LinSVC achieves highest classification F^1 for price fluctuation prediction. This performance is significantly higher (18%) than our baseline. We can also see that SVC and MNB show performance that almost equals LinSVC. Even though LinSVC just outperforms SVC with an F^1 score of .617 compared to .616 respectively, we can also see that SVC achieves more constant results when using 1,000 and 10,000 instances. This provides us with a notion that a linear model using LinSVC is less useful when using a small amount of training instances. This is why we would recommend SVC over LinSVC due to its wider effectiveness.

Insufficient results are achieved for numerical price fluctuation predictions. Best performing SGD configuration provide a R^2 of .065 and RMSE of 122,823. Despite our best efforts, we cannot find comparable results in other research. This leads us to assume that our best regression model to predict price fluctuations is still a poor fit on our price fluctuation data, but comparable results can prove otherwise. Similar to the results for selling- and asking price predictions, we see a RMSE decrease at 100,000 instances. This suggests a relative decrease in large prediction errors (Willmot & Matsuura, 2005).

5.4 RQ4: Influence of training set size

Although significant effects have been found between training set size and performance (Zhang, Jin, Yang & Hauptmann, 2003), our results do not show such a significant correlations. We can therefor dismiss our third hypothesis. A reason for this can be that our model performs relatively well on small sets of data due to the lack of complexity but also shows good results on large datasets due to the higher choice in informative n-grams for the model.

In the RMSE results for all three pricing indicators we found a drop at 100,000 instances. This is strong evidence that our models are able to reduce large errors when making predictions (Willmot & Matsuura, 2005). Even though smaller prediction errors are favorable over large prediction errors, we can also see that accompanying R^2 results show a drop as well. This can suggest that our model makes less specific predictions closer to the price mean but further research needs to be done to interpret this conclusively.

5.5 RQ5: Influence of attribute variable amount

Curve estimation analysis shows a significant logarithmic effect for the amount of attribute variables used, in our experiment n-grams, on performance. Significant effects were found for selling price classification and regression. This indicates that using more n-grams will lead to better performance, but this effect reduces exponentially when more are added. Significant linear effects were found for asking price performance and irregular results for price fluctuation due to minimal variation in performance. We will therefor retain our second hypothesis but emphasize the reduction in effect for selling price predictions. In addition, we have found that adding more attribute variables lead to a severe increase in demand for computational power and time. Algorithmic and technologic improvements have to be made in order to create competitive results for large datasets.

Comparing our results for classification and regression, we see that performance on classification does provide considerable improvement over our set baseline. This also means that we can retain our first hypothesis. Classification shows excellent overall performance and we can see that SGD is the most suited classifier for our regression task. For classification tasks, MNB, SVC and LinSVC achieve excellent results with a significant maximum baseline improvement of 38%. This suggests that our classification models can be of unique value in predicting real estate prices. SVC seems to be the most useful classifier in predicting our pricing indicators across a wide range of instances. This model also shows a respectable fit when using a small amount of n-grams, which indicates potential as an addition to the standard hedonic approach.

Our results suggest that a numerical predictive system for real estate price using text mining techniques cannot provide an alternative, but can be a unique addition to the hedonic approach based on general real estate characteristics (Peterson, 2008; Dubin, 1998). Our predictive regression model needs a relatively high amount of n-grams to make proper predictions. It appears that n-grams contain less predictive value than the widely used real estate characteristics. This means that our model needs to become exceedingly complex to achieve high performance, which poses extreme demands on computing power for larger sets of data. Another reason for the large quantity of n-grams needed may lay in the selection of n-grams. Our model vectorizes our descriptions and subsequently selecting the provided number of n-grams. This selection is solely based on term frequency. Tf-idf values are calculated and used for classification. We recommend exploring the possibility to incorporate tf-idf in the vectorizer's selection process in order to provide the model with more predictive n-grams.

Predictive systems for real estate prices using text mining techniques can be used in a variety of unique applications. For example, house owners assessing the price of their home by submitting a description online or via voice recognition software. This can provide an easy and accessible alternative to find the appropriate house price. Since this is an extremely difficult task we recommend that further research should be done on hybrid forms of price prediction models. It would be interesting to see if a combination of classification and regression leads to better performance or if a unique combination of text mining and hedonic pricing models could improve today's prediction of real estate prices. The unique advantage of including variables outside of the hedonic pricing models using text mining can lead to a more representational and tailored real estate price prediction for agencies, house owners and buyers.

References

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine learning*, 6(1), 1-2

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician 46 (3), 175–185

Ash, V. (2013). An evaluation of statistical approaches to text categorization. Information retrieval, 1(1-2), 5-7

Beitzel, S. M. (2006). On understanding and classifying web queries (Doctoral dissertation, Illinois Institute of Technology), 51-53

Bekkerman, R., & Allan, J. (2004). Using bigrams in text categorization. Department of Computer Science, University of Massachusetts, Amherst, 1003, 1-2

Brindle, J. T., Nicholson, J. K., Schofield, P. M., Grainger, D. J., Holmes, E. (2002). Rapid and noninvasive diagnosis of the presence and severity of coronary heart disease using 1H-NMR-based metabonomics, 1441

Bulmer, M. G. (2012). Principles of statistics. Courier Dover Publications, 62-65

Chen, Y. J. (2001). Transactional Distance in World Wide Web Learning Environments, Innovations in Education and Teaching International, 38:4, 333

Dubin, R. A. (1998). Predicting House Prices Using Multiple Listings Data. *Journal of Real Estate Finance and Economics,* Volume 17:1, 35-59

Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. (1996). Knowledge Discovery and Data Mining: Towards a Unifying Framework, 39, 44-46

Frank, E., Trigg, L., Holmes, G., Witten, I. H. (1999). Naïve Bayes for Regression, 1-20

Friedman, J. H. (2002). Stochastic gradient boosting. Computational Statistics & Data Analysis, 38(4), 1-3

Ghani, R., & Simmons, H. (2004). Predicting the end-price of online auctions. In *Proceedings of the International Workshop on Data Mining and Adaptive Modelling Methods for Economics and Management*, 1-11

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H. (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1, 1-18

Haurin, D. (1988). The duration of marketing time of residential housing, Real Estate Economics. *Real Estate Economics*, 16(4), 403

Hocking, R. R. (2013). Methods and Applications of Linear Models: Regression and the Analysis of Variance, 3rd Edition, 159, 262, 773

Hu, M., & Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 1

Hu, X., Downie, J. D., Ehmann, A. F. (2009). Lyric Text Mining in Music Mood Classification. International Society for Music Information Retrieval, 411-416

Hull, D. A. (1995). Stemming Algorithms – A Case Study for Detailed Evaluation, 1-4

Ifrim, G., Bakir, G., & Weikum, G. (2008, August). Fast logistic regression for text categorization with variable-length n-grams. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1-2

Jing, L. P., Huang, H. K., & Shi, H. B. (2002). Improved feature selection approach TFIDF in text mining. In *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, Volume 2, 944-945

Kibriya, A. M., Frank, E., Pfahringer, B., & Holmes, G. (2005). Multinomial naive bayes for text categorization revisited. In *AI 2004: Advances in Artificial Intelligence*, 1-2

Kwok, S. W., & Carter, C. (2013). Multiple decision trees. *arXiv preprint arXiv:1304.2363*, 213-215

Lalys, F., Riffaud, L., Morandi, X., Jannin, P. (2010). Surgical phases detection from microscope videos by combining SVM and HMM, In *Medical Computer Vision. Recognition Techniques and Applications in Medical Imaging,* 54-62

Lewis, D. D. (1992). An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, 37-50

Liaw, A., & Wiener, M. (2002). Classification and Regression by randomForest. *R news,* 2(3), 18

Malthouse, E. C. (1999). Ridge regression and direct marketing scoring models. *Journal of Interactive Marketing*, 13(4), 10-14

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval, Volume 1, 100

Mittermayer, M. A. (2004). Forecasting intraday stock price trends with text mining techniques. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*, 1-10

Mladenic, D., Grobelnik, M. (1998). Word sequences as features in text-learning, 1

Nagelkerke, N. J. D. (2008). A note on a general definition of the coefficient of determination. Biomerrika, 691-692

Pak, A., & Paroubek, P. (2010). Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In LREC

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. Foundations and trends in information retrieval, 2(1-2), 1-15

Porter, M. F. (1980). An algorithm for suffix stripping. *Program: electronic library and information systems,* 14(3), 313-316

Rajman, M., & Besançon, R. (1998). Text mining: natural language techniques and text mining applications. In *Data Mining and Reverse Engineering*, 1-15

Salton, G., & Yang, C. S. (1973). On the specification of term values in automatic indexing. Journal of documentation, 29(4), 351-372

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000, October). Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on Electronic commerce*, 8

Tan, A. H. (1999, April). Text mining: The state of the art and the challenges. In *Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases,* Volume 8, 1-2

Tan, C. M., Wang, Y. F., Lee, C. D. (2002). The use of bigrams to enhance text categorization, 1-14, 26

Wijk, C. V. (2000). Toetsende statistiek: basistechnieken. Een praktijkgerichte inleiding voor onderzoekers van taal, gedrag en communicatie, 191-201

Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research,* 30(1), 79-82

Witten, I. H., Frank, E., Hall, M. A. (2011). Data Mining: Practical Machine Learning Tools and Techniques, 3-8, 61-83, 79-99, 147-150, 328

Wooldridge, J. M. (2012). Introductory. Econometrics. A Modern Approach. *Cengage Learning,* 471-472

Ye, N. (2013). Data mining: Theories, Algorithms, and Examples. *CRC Press*, 28-29

Yu, L., Wang, S., & Lai, K. K. (2005). A rough-set-refined text mining approach for crude oil market tendency forecasting. *International Journal of Knowledge and Systems Sciences*, 2(1), 33-46

Zhang, T. (2004, July). Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, 1

Zhang, J., Jin, R., Yang, Y., & Hauptmann, A. G. (2003). Modified logistic regression: An approximation to svm and its applications in large-scale text categorization. In *ICML*, 888-895

Appendices

Appendix A

Selling Price Classes F^1 score

Instances	Features	SVC	LinSVC	kNN	MNB	DT
1,000	10	.632	.600	.516	.608	.488
1,000	50	.592	.584	.532	.608	.488
1,000	100	.584	.560	.512	.632	.556
1,000	150	.584	.580	.532	.608	.480
1,000	200	.584	.576	.488	.636	.572
1,000	250	.584	.600	.480	.652	.576
10,000	10	.384	.398	.364	.373	.372
10,000	50	.464	.465	.403	.444	.433
10,000	100	.478	.500	.426	.456	.416
10,000	150	.480	.510	.439	.476	.439
10,000	200	.493	.550	.456	.487	.452
10,000	250	.476	.552	.471	.494	.448
50,000	10	.369	.387	.331	.365	.344
50,000	50	.470	.470	.374	.428	.377
50,000	100	.486	.495	.394	.485	.401
50,000	150	.497	.507	.408	.496	.417
50,000	200	.495	.517	.413	.507	.420
50,000	250	.484	.524	.418	.511	.415
100,000	10	.536	.514	.492	.560	.488
100,000	50	.576	.585	.477	.520	.463
100,000	100	.613	.626	.501	.491	.479
100,000	150	.633	.650	.521	.507	.491
100,000	200	.642	.650	.521	.517	.498
100,000	250	.640	.650	.532	.515	.504

Appendix B

Asking Price Classes F^1 score

Instances	Features	SVC	LinSVC	kNN	MNB	DT
1,000	10	.664	.668	.584	.664	.548
1,000	50	.664	.676	.580	.664	.532
1,000	100	.664	.640	.568	.672	.508
1,000	150	.664	.664	.600	.668	.524
1,000	200	.664	.628	.560	.680	.512
1,000	250	.664	.624	.572	.676	.488
10,000	10	.421	.430	.360	.422	.389
10,000	50	.437	.439	.386	.434	.381
10,000	100	.442	.449	.399	.430	.372
10,000	150	.435	.442	.412	.426	.374
10,000	200	.425	.454	.416	.452	.404
10,000	250	.424	.456	.406	.454	.384
50,000	10	.383	.385	.366	.383	.366
50,000	50	.383	.420	.384	.396	.392
50,000	100	.383	.449	.405	.436	.389
50,000	150	.383	.456	.400	.453	.388
50,000	200	.383	.467	.403	.459	.394
50,000	250	.383	.468	.409	.462	.398
100,000	10	.682	.682	.553	.682	.542
100,000	50	.682	.668	.559	.679	.503
100,000	100	.682	.675	.564	.664	.512
100,000	150	.682	.680	.573	.653	.518
100,000	200	.682	.680	.567	.650	.526
100,000	250	.682	.681	.573	.644	.522

Appendix C

Price Fluctuation Classes F^1 score

Instances	Features	SVC	LinSVC	kNN	MNB	DT
1,000	10	.608	.608	.484	.608	.476
1,000	50	.608	.572	.552	.608	.540
1,000	100	.608	.544	.508	.572	.524
1,000	150	.608	.548	.532	.572	.512
1,000	200	.608	.568	.552	.568	.468
1,000	250	.608	.576	.528	.572	.484
10,000	10	.616	.616	.508	.616	.515
10,000	50	.616	.617	.539	.616	.508
10,000	100	.616	.609	.533	.616	.506
10,000	150	.616	.604	.541	.616	.513
10,000	200	.616	.598	.535	.616	.521
10,000	250	.616	.593	.542	.616	.490
50,000	10	.597	.597	.541	.597	.536
50,000	50	.597	.597	.529	.597	.510
50,000	100	.597	.597	.530	.597	.515
50,000	150	.597	.597	.526	.597	.511
50,000	200	.597	.597	.526	.597	.515
50,000	250	.597	.597	.526	.597	.512
100,000	10	.598	.598	.528	.598	.539
100,000	50	.598	.598	.529	.598	.524
100,000	100	.598	.598	.527	.598	.522
100,000	150	.598	.598	.524	.598	.520
100,000	200	.598	.598	.530	.598	.521
100,000	250	.598	.598	.530	.598	.521

Appendix D

Selling Price R² score

Instances	Features	GBR	MNB	Ridge	RFR	SGD
1,000	10	-.016	-.004	-.043	-.500	-.157
1,000	50	.147	-.004	-.032	-.307	.005
1,000	100	.148	-.004	-.070	-.265	.126
1,000	150	.168	-.004	-.053	-.239	.109
1,000	200	.174	-.004	-.045	-.301	.102
1,000	250	.163	-.004	-.050	-.326	.034
10,000	10	.029	-.012	-.007	-.130	-.090
10,000	50	.115	-.012	.131	.032	.151
10,000	100	.133	-.012	.178	.051	.219
10,000	150	.141	-.012	.198	.111	.225
10,000	200	.173	-.012	.257	.117	.303
10,000	250	.166	-.012	.267	.140	.302
50,000	10	-.108	-.227	-.044	-.177	-.033
50,000	50	-.049	-.228	.062	-.057	.061
50,000	100	-.001	-.228	.111	.019	.097
50,000	150	.005	-.228	.133	.050	.148
50,000	200	.015	-.228	.158	.060	.181
50,000	250	.015	-.228	.174	.096	.153
100,000	10	-.260	-.140	-.490	-.710	-.570
100,000	50	-.170	-.140	-.350	-.490	-.380
100,000	100	-.110	-.140	-.220	-.360	-.160
100,000	150	-.100	-.140	-.100	-.280	-.190
100,000	200	-.070	-.140	-.060	-.200	-.070
100,000	250	-.070	-.140	-.060	-.210	-.040

Appendix E

Selling Price RMSE score

Instances	Features	GBR	MNB	Ridge	RFR	SGD
1,000	10	82692,77	73815,65	75171,31	95690,03	77942,35
1,000	50	79577,48	73815,88	73824,66	83351,53	70656,91
1,000	100	76442,56	73815,94	74694,00	81192,44	69959,45
1,000	150	75501,61	73815,97	73488,83	87459,47	68192,74
1,000	200	77087,52	73815,98	72826,60	86002,08	66967,57
1,000	250	73929,29	73815,99	71793,00	80338,88	66928,45
10,000	10	161418,89	163284,31	162867,27	173180,81	160849,69
10,000	50	150858,19	163288,08	151315,57	161025,65	152316,20
10,000	100	147326,18	163289,24	146969,45	158673,43	147756,49
10,000	150	145840,39	163289,68	145151,27	154084,12	143754,19
10,000	200	142108,62	163289,86	139732,90	152798,55	139177,52
10,000	250	141551,56	163290,01	138737,16	153243,31	139693,98
50,000	10	197797,46	212311,04	195843,38	207936,98	193931,43
50,000	50	187882,85	212386,54	185626,17	196674,29	185845,35
50,000	100	182692,20	212396,98	180702,70	189827,55	183330,45
50,000	150	180432,95	212400,78	178405,89	186866,66	176525,17
50,000	200	178580,10	212402,44	175875,16	185313,50	177103,48
50,000	250	177775,53	212403,53	174163,28	182847,63	174286,96
100,000	10	151967,15	134426,08	153835,08	165241,65	155316,50
100,000	50	143564,35	134479,01	146628,70	155040,08	137011,35
100,000	100	136972,74	134490,48	139319,01	146757,22	143696,20
100,000	150	132562,35	134495,66	132384,39	142345,00	125980,73
100,000	200	130602,80	134498,60	129571,59	139644,90	126528,09
100,000	250	130597,70	134498,60	129571,59	139530,86	125999,74

Appendix F

Asking Price R² score

Instances	Features	GBR	MNB	Ridge	RFR	SGD
1,000	10	.046	-.017	-.100	-.663	-.029
1,000	50	.069	-.017	-.142	-.843	.007
1,000	100	.010	-.017	-.280	-.618	-.028
1,000	150	.037	-.017	-.355	-.459	-.009
1,000	200	.086	-.017	-.370	-.345	.038
1,000	250	.097	-.017	-.384	-.508	.064
10,000	10	.025	-.025	.018	-.082	.019
10,000	50	.046	-.025	.046	-.069	.033
10,000	100	.047	-.025	.071	-.038	.078
10,000	150	.054	-.025	.088	-.036	.100
10,000	200	.062	-.025	.103	-.056	.118
10,000	250	.060	-.025	.109	-.015	.124
50,000	10	-.073	-.196	-.014	-.167	-.016
50,000	50	-.045	-.197	.035	-.066	.028
50,000	100	-.016	-.197	.066	-.025	.069
50,000	150	-.005	-.197	.077	-.005	.076
50,000	200	.002	-.197	.091	.021	.099
50,000	250	.001	-.197	.097	.009	.101
100,000	10	-.040	-.010	-.160	-.400	-.320
100,000	50	-.010	-.010	-.120	-.310	-.180
100,000	100	.020	-.010	-.070	-.230	-.050
100,000	150	.010	-.010	.000	-.180	.010
100,000	200	.010	-.010	.010	-.140	.020
100,000	250	.010	-.010	.010	-.140	.010

Appendix G

Asking Price RMSE score

Instances	Features	GBR	MNB	Ridge	RFR	SGD
1,000	10	88722,23	76902,25	79853,74	94598,21	82307,36
1,000	50	89387,72	76902,45	80634,11	97217,96	78791,06
1,000	100	91939,40	76902,48	84273,67	99616,41	76124,54
1,000	150	91556,93	76902,50	85173,50	87121,88	78127,20
1,000	200	88513,54	76902,51	84864,61	99410,03	76189,23
1,000	250	86285,85	76902,52	84197,56	93893,18	73608,92
10,000	10	166833,20	170729,89	167094,28	178945,99	167079,34
10,000	50	164223,30	170732,95	164602,97	173990,19	164599,59
10,000	100	163156,74	170733,63	162439,29	171103,93	162085,61
10,000	150	160799,27	170733,83	160888,18	170384,88	160157,54
10,000	200	160393,15	170733,94	159427,91	171619,17	157919,60
10,000	250	160562,86	170734,01	158929,30	171772,62	158106,22
50,000	10	195581,95	210257,37	193564,94	206646,18	192428,27
50,000	50	190800,51	210320,10	188840,18	199605,81	190233,15
50,000	100	186784,13	210335,10	185790,59	194500,45	185253,13
50,000	150	185021,90	210341,35	184614,14	192670,81	183933,21
50,000	200	184176,53	210344,32	183204,79	191689,91	183038,74
50,000	250	184341,10	210345,95	182614,75	190833,94	181816,67
100,000	10	138655,56	129271,57	138982,67	151997,54	136767,01
100,000	50	135021,29	129302,70	136362,22	145740,10	137179,86
100,000	100	131448,03	129309,37	132936,39	141739,60	132138,09
100,000	150	128154,10	129312,04	128898,17	138798,34	125864,92
100,000	200	127910,02	129313,51	128033,70	137754,17	126843,06
100,000	250	127896,22	129313,51	128033,70	137495,62	127011,03

Appendix H

Price Fluctuation R² score

Instances	Features	GBR	MNB	Ridge	RFR	SGD
1,000	10	-.027	-.002	-.004	-.257	-.011
1,000	50	.043	-.002	-.005	-.220	.036
1,000	100	.044	-.002	-.041	-.263	.012
1,000	150	.035	-.002	-.056	-.177	.061
1,000	200	.033	-.002	-.047	-.348	.037
1,000	250	.060	-.002	-.068	-.334	.050
10,000	10	-.032	-.012	-.047	-.219	-.048
10,000	50	-.009	-.012	-.012	-.140	-.015
10,000	100	.003	-.012	-.009	-.130	.012
10,000	150	-.002	-.012	-.010	-.140	.025
10,000	200	.011	-.012	-.002	-.109	.005
10,000	250	.011	-.012	.006	-.142	.021
50,000	10	-.029	-.086	-.014	-.127	-.017
50,000	50	-.006	-.086	.024	-.082	.020
50,000	100	.010	-.086	.040	-.063	.050
50,000	150	.010	-.086	.046	-.060	.049
50,000	200	.015	-.086	.052	-.048	.055
50,000	250	.018	-.086	.058	-.038	.065
100,000	10	-.070	-.020	-.130	-.300	-.080
100,000	50	-.040	-.010	-.080	-.220	-.070
100,000	100	-.030	-.010	-.060	-.200	-.090
100,000	150	-.040	-.010	-.040	-.180	-.020
100,000	200	-.020	-.010	-.020	-.170	-.020
100,000	250	-.020	-.010	-.020	-.150	.000

Appendix I

Price Fluctuation RMSE score

Instances	Features	GBR	MNB	Ridge	RFR	SGD
1,000	10	79822,55	75662,82	75723,15	86120,39	75670,50
1,000	50	76897,83	75662,84	75329,27	83896,73	74383,65
1,000	100	77033,20	75662,85	76146,35	82541,08	74806,99
1,000	150	77598,81	75662,86	75913,10	83736,39	73688,78
1,000	200	75781,96	75662,86	75521,53	83523,86	73295,38
1,000	250	76352,55	75662,87	75789,81	84946,15	73431,39
10,000	10	135686,54	132738,67	134993,83	146620,51	136650,53
10,000	50	133141,09	132738,47	132683,53	142495,30	132666,63
10,000	100	132716,84	132738,50	132464,60	139664,05	130610,82
10,000	150	132461,85	132738,53	132519,47	140601,02	136476,57
10,000	200	131977,63	132738,54	131990,01	140010,46	132994,74
10,000	250	131060,91	132738,55	131463,66	139594,91	129398,26
50,000	10	127306,96	132074,16	127638,59	134288,92	127228,18
50,000	50	125272,62	132074,92	125188,29	132050,72	124711,78
50,000	100	124233,96	132074,98	124142,77	131692,71	123793,00
50,000	150	124087,51	132074,96	123765,17	130662,39	123239,14
50,000	200	123444,12	132075,02	123410,82	130510,69	123865,58
50,000	250	123050,58	132075,04	122987,45	129722,00	122823,64
100,000	10	103042,91	98357,64	103649,00	111472,90	109000,56
100,000	50	100964,55	98351,18	101584,90	107820,75	100546,96
100,000	100	99878,32	98350,38	100325,09	107096,75	98850,16
100,000	150	99447,44	98350,31	99504,39	106535,27	97391,56
100,000	200	98391,91	98350,30	98352,49	105094,07	97180,45
100,000	250	98395,54	98350,30	98352,49	105452,89	97235,25

Appendix J

Table J.1. Curve Estimation Asking Price

	Classification		Regression	
	MNB		SGD	
Instances	*Linear*	*Logarithmic*	*Linear*	*Logarithmic*
1,000	.728 *	.603	.685 *	.455
10,000	.689 *	.508	.944 **	.881 *
50,000	.885 *	.897 **	.878 *	.976 **
100,000	.954 **	.844 *	.785 *	.965 **

*Note: * = significant with p < .05 ** = significant with p < .001*

Table J.2. Curve Estimation Price Fluctuation

	Classification		Regression	
	LinSVC		SGD	
Instances	*Linear*	*Logarithmic*	*Linear*	*Logarithmic*
1,000	.133	.474	.508	.657
10,000	.970 **	.717 *	.639	.884 *
50,000	-	-	.793 *	.976 **
100,000	-	-	.780 *	.537

*Note: * = significant with p < .05 ** = significant with p < .001*

Table J.3. Correlation Analysis, relation between instance amount and performance

Price Indicator	Classifier	n-grams					
		10	50	100	150	200	250
C1assification							
Selling Price	MNB	.110	-.184	-.424	-.393	-.423	-.482
Asking Price	MNB	.266	.260	.247	.249	.179	.171
Price Fluctuation	LinSVC	-.777	.174	.449	.502	.581	.725
Regression							
Selling Price	SGD	-.772	-.847	-.918	-.852	-.736	-.562
Asking Price	SGD	-.880	-.851	-.402	-.189	-.454	-.694
Price Fluctuation	SGD	-.711	-.778	-.703	-.798	-.501	-.498

*Note: * = significant with p < .05 ** = significant with p < .001*