TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Aleksander Belov 155549

TALLINN HOUSE PRICE PREDICTION MODEL

Bachelor's Thesis

Supervisor: Margarita Spitšakova PhD

Tallinn 2019

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Aleksander Belov 155549

TALLINNA KINNISVARA HINDADE ENNUSTAMISE MODEL

Bakalaureusetöö

Juhendaja: Margarita Spitšakova PhD

Tallinn 2019

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Aleksander Belov

07.01.2019

Abstract

Real estate market is very huge and diverse. These features complicate the process of distinguishing between good and bad offer. Apartments can be overpriced, and offer is not profitable for the buyer.

For a person with small experience in this field it can be difficult to make the right choice. He lacks the source of simple and understandable information about real estate market. When buying a house people often choose not the most optimal variant among the houses with the parameters provided by them.

Author offers model which gives additional information for making a decision during the choice of the house. Solution includes statistical method of exploring the problem. Data is collected from web resources that provide real estate objects offerings. The data is processed into suitable format for the model. Objects include house parameters and price. Collected data is used as input for training mathematical model. Desired or arbitrary parameters are used as input in training model. The average price of the house is predicted in the model and comes as output.

The result of the work will be simple web application that uses trained model. User inputs desired parameters in the web form and gets normal distribution graphic visualization with the average price. Horizontal axis includes price, vertical includes the number of houses with the given parameters.

This thesis is written in English and is 50 pages long, including 9 chapters, 26 figures and 6 tables.

Annotatsioon

Kinnisvaraturg on väga suur ja mitmekesine. Need omadused muudavad hea ja halva pakkumise eristamise protsessi keeruliseks . Korterid võivad olla ülehinnatud ja pakkumine ostjale mitte kasumit toov.

Valdkonnapõhiselt väikese kogemusega inimesel võib olla raske teha õigeid valikuid. Tal puudub lihtne ja arusaadav teave kinnisvaraturu kohta. Maja ostmisel valivad inimesed üsna tihti pakutud variantide seast mitte kõige optimaalsema näitajatega variandi.

Autor pakub välja mudeli, mis annab lisainformatsiooni maja valimise ajal otsuste tegemiseks. Lahendus sisaldab statistilist meetodit probleemi uurimiseks. Andmeid kogutakse veebiressurssidest, mis sisaldavad kinnisvaraobjektide pakkumisi. Info töödeldakse mudeli jaoks sobivas vormingus. Objektide sisaldavad maja parameetreid ja hinda. Kogutud andmeid kasutatakse sisendina matemaatilise mudeli väljatöötamiseks. Koolitusmudeli sisendina kasutatakse soovitud või suvalisi parameetreid. Mudeliga prognoositakse maja keskmine hinda ja see saadakse väljundina.

Töö tulemuseks on lihtne veebirakendus, mis kasutab väljatöötatud mudelit.

Kasutaja sisestab soovitud parameetrid veebivormi ja saab mugavalt loetavate graafilise visuaaliga jaotuse, koos keskmise hinnaga. Horisontaaltelg sisaldab hinda, vertikaalne sisaldab antud parameetriga majade arvu.

See töö on kirjutatud inglise keeles ja on 50 lehekülge pikk, sisaldades 9 peatükki, 26 arvandmeid ja 6 tabelit.

List of abbreviations and terms

CSV	Comma-Separated Values, file format
MSE	Mean squared error
RSS	Residual sum of squares
RMSE	Root mean square error

Table of contents

List of figures

List of tables

Introduction

Predicting housing prices has always been a challenge for many machine learning engineers. It has been hosted as part of the Kaggle competitions [1]. Several researchers have tried to come with a model to predict housing prices with high accuracy and least error. These models are created using various features such as square feet of the house, number of bedrooms, ambiance etc. Some of the researchers have used techniques like clustering [2] for grouping same houses together and then estimating the price. Another way is giving certain weight to the features in the model, which determines how important is that feature towards model prediction. This step is part feature engineering.

Usually most companies which do real estate business have probably a lot of different features to choose from however one of the drawbacks of having a large number of features involved is the heavy computations involved in making the regression model and computing the gradient descent solution. One of the techniques is used in companies is regression [3]. Later we will be introduced to the algorithm called as gradient descent algorithm which drastically reduces the computation workload and limits the number the features while selecting the only important ones and also some of the state-of-the-art model.

With the availability of computing power and expressive data analysis software, exploratory data analysis has evolved well beyond its original scope. Key drivers of this discipline have been the rapid development of new technology, access to bigger data, and the greater use of quantitative analysis in a variety of disciplines [4].

Problem overview

Real estate market is very huge and diverse. These features complicate the process of distinguishing between good and bad offer. Houses can be overpriced, and offer is not profitable for the buyer.

For a person with small experience in this field it can be difficult to make the right choice. He lacks the source of simple and understandable information about real estate market. When buying a house people often choose not the most optimal variant among the houses with the parameters provided by them.

One of the ways to assist person in decisions is using statistical models that can help answering the questions. Perhaps the most common goal in statistics is to answer the question: Is the variable X associated with a variable Y, and, if so, what is the relationship, and can we use it to predict Y [4]?

To answer on this question here comes data analysis and linear regression model training. Data analysis helps to measure the correlations between different parameters and to prepare data for effective modeling. Simple linear regression estimates exactly how much Y will change when X changes by a certain amount. With the correlation coefficient, the variables X and Y are interchangeable. With regression, we are trying to predict the Y variable from X using a linear relationship [4].

Objectives

The main goal of this thesis is to research, develop and deploy linear regression model that predicts estimated price for the apartments located in Tallinn, based on certain parameters. The goal is to use the latest state of the art models and technologies that are considered successful in solving regression problem. This entails viable usage of machine learning libraries and implementation of stacking technics along with representation as well as provision of baselines for subsequent analysis. In addition, it includes comparison of the models and deployment of the most effective model using standalone Python web application that allows to configure and run reproducible experiments using said models and allow for the outcome to be analyzed using external tools.

One of the similar works was already done. In that work author uses old dataset of 2006 year collected from housing portal. Tallinn apartments are processed, and some statistical meaning is extracted from them. The histogram plot is used to show his results [5].

Outline

The thesis is organized in the following way. Chapter 2 contains the problem statement. Chapters 3 introduces the basic theoretical concepts behind statistical data analysis and base models. It also provides a brief overview of existing solutions and applied techniques. Chapter 4 elaborates on the choice of tools, general strategy and validation methods used in the thesis. Chapters 6, 7, 8 are dedicated to the detailed explanation of the proposed solution. Chapter 9 documents the conducted experiments and focuses on outcome analysis and discussion of possible interpretations before leading to conclusions.

Problem statement

Every linear model requires training using data, during which model weights are defined. Weights will have influence on the predictions. The main problem is to find that data. Next step is data verification for its quality and variability through feature engineering and processing. Another obstacle is model comparison and finding the most effective one. The final is model deployment.

Data collection is the first step of our research. There are no publicly accessible resources that provide APIs for collecting data that contains all necessary features for house price prediction model in Tallinn. The data collected comes in inappropriate format for regression model. The result of data scraping contains a lot of missed data where some features are not correctly defined or has null values. Our target is to clean the data from missed row and convert feature columns to certain format that is readable to our linear regression model. Many of the variables has null values that needed to be dealt with.
Next step is data preprocessing. It includes raw data analysis and visualization. During this step we decide which variables correlate with the house sale price and improve our prediction model. If needed we scale some columns, remove outliers, create dummy variables for the categorical variables and split data into training set and a test set. Most of the variables that deal with the actual physical space of the apartment are positively skewed—which makes sense, as people tend to live in smaller homes/apartments apart from the extremely wealthy. Sale price also has a similar positively skewed distribution. One of the hypothesizes for this reason is that the variables dealing with the actual dimensions of the apartment have a large impact on sale price. Some features may not have a strong relationship with sale price, such as the year of the build, for example. However, a few variables, like overall quality are highly correlated with sale price.

One of the important steps is different model fitting, training, evaluation and comparison. Linear regression models are based on one mathematical formula but differ in error estimation and the way its algorithm optimize it weights that decrease error. We need to find these optimal parameters for each model and make it perform effectively.

In the production environment we need the most effective model, which we choose by comparison. After choosing the candidate we deploy the model on the server and implement the way it communicates with the potential client.

1 Theoretical background

1.1 Data analysis

Every data analysis starts from basic methods of exploration. Depending on the particular characteristics of a distribution, we can summarize it using the mean, standard deviation, finding maximum and minimum points, analyzing the shape of the distribution, finding the gold ration between bias and variance.

1.1.1 Skewed distribution

A distribution is negatively skewed, or skewed to the left, if the scores fall toward the higher side of the scale and there are very few low scores. In positively skewed distributions, the mean is usually greater than the median, which is always greater than the mode [6].

This distribution is clearly right skewed. Generally, the location of the mode, median and mean is predictable for a right-skewed distribution:

- The mean takes into account each value in the distribution, and it will be affected by the outliers in the right tail. This will generally pull the mean to the right of the median [6].

In the right-skewed distribution, the mean will usually be to the right of the median, and the median will be to the right of the mode [6]. This holds true for the distribution of the price variable:

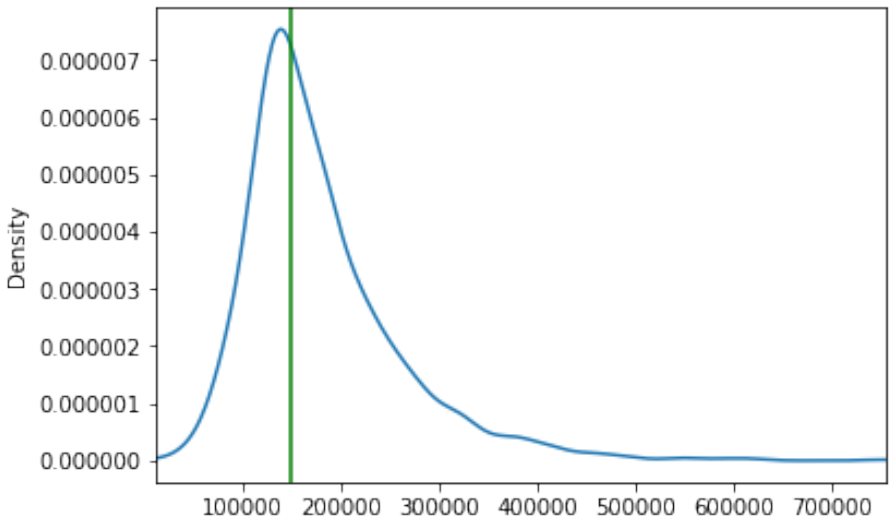

Figure 1: Right-skewed distribution

The normal distribution is the easiest distribution to work with in order to gain an understanding about statistics. Real life distributions are usually skewed. Many statistical techniques don't work if there are too much skewness. As a result, mathematical techniques including logarithms and quantile regression are used to normalize distribution [7].

1.1.2 Bias and the variance

In statistics, there are two critical characteristics of estimators to be considered: the bias and the variance.

Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.

$$Bias(\hat{\beta}\, ols) = E(\hat{\beta}\, ols) - \beta$$

Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it has not seen before. As a result, such models perform very well on training data but has high error rates on test data [8].

In general, having high bias reduces the performance of the algorithm on training set while having high variance reduces performance on unseen data [9].

This measure of variability is sometimes called mean squared distance or mean squared deviation. However, it is more commonly known as variance.

$$mean\ squared\ distance = \frac{(x_1 - \mu)^2 + (x_2 - \mu)^2 + \ldots + (x_N - \mu)^2}{N}$$

$$= \frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}$$

Squaring the distances or taking their absolute values ensure that we get a variability value that is greater than 0 for all distributions that show some variability [10].

1.1.3 Standard deviation

The square root of variance is called standard deviation and it can be expressed like this in an algebraic definition:

$$standard\ deviation = \sqrt{mean\ squared\ distance}$$

In practice, standard deviation is perhaps the most used measure of variability. Standard deviation tells us how much the values in a distribution vary (on average) around the mean of that distribution [10].

The mean of the price variable is EUR 180,796:

The standard deviation gives us a picture about this variability around the mean sale price. So, on average, sale prices vary by roughly EUR 79,873 above and below a mean of EUR 180,796.

The standard deviation doesn't set boundaries for the values in a distribution: the prices can go above and below the mean more than EUR 79,873.

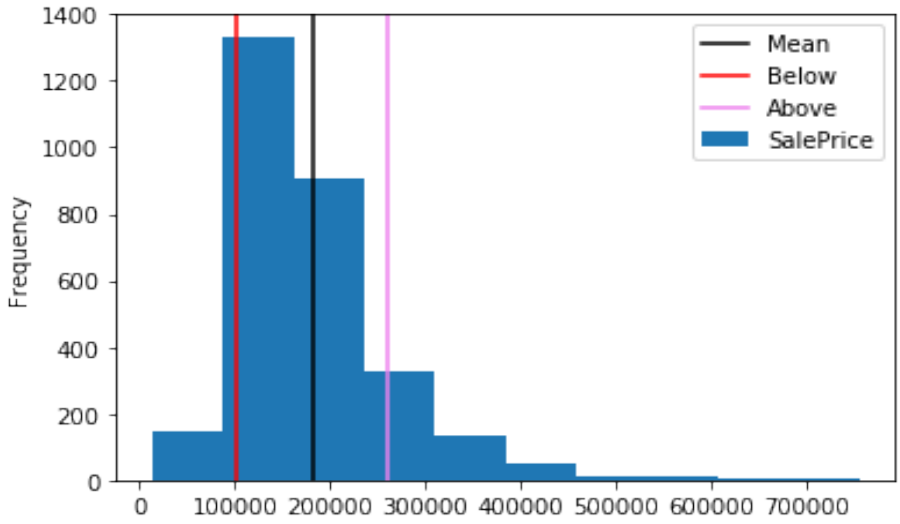

Figure 2: Histogram with the varying prices around the mean

1.2 Linear regression model

The linear regression model is the most commonly used machine learning model. Parametric machine learning approaches work by making assumptions about the relationship between the features and the target column. In linear regression, the approximate relationship between the feature columns and the target variable [4].

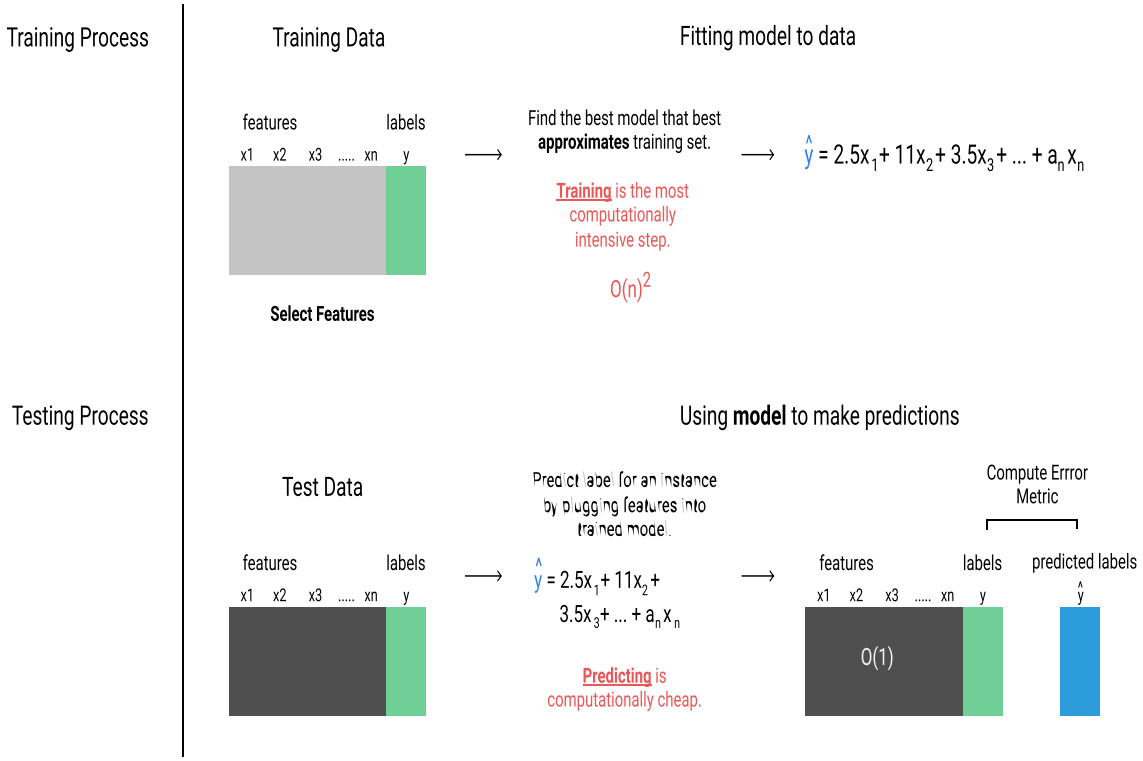

Figure 3: Overview of the machine learning process for linear regression [4]

1.2.1 Simple linear regression

We will start by understanding the univariate case of linear regression, also known as simple linear regression. The following equation is the general form of the simple linear regression model:

$$\hat{y} = a_1 x_1 + a_0$$

$\hat{y}$ represents the target column while x_1 represents the feature column we choose to use in our model [11]. a_0 and a_1 represent the parameter values that are specific to the dataset. The goal of simple linear regression is to find the optimal parameter values that best describe the relationship between the feature column and the target column. The following diagram shows different simple linear regression models depending on the data [12]:

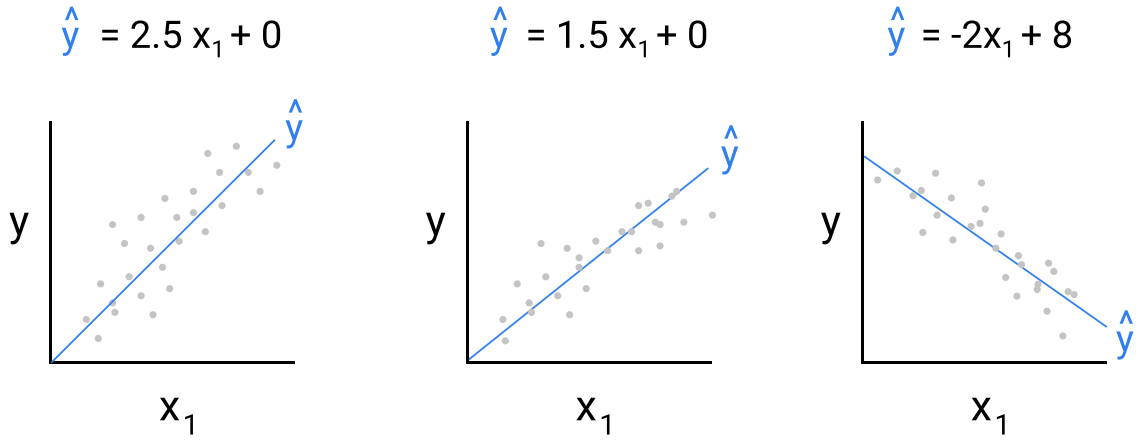

Figure 4: Different simple linear regression models depending on the data

A multiple linear regression model allows us to capture the relationship between multiple feature variables and the target column. When using multiple features, the main challenge is selecting relevant features [13]. Here's what the formula looks like:

$$y = a_1x_1 + a_2x_2 + a_3x_3 + \ldots + a_nx_n$$

1.2.2 Residual Sum of Squares

To find the optimal parameters for a linear regression model, we want to optimize the models residual sum of squares (RSS). Residual describes the difference between the predicted values for the target column $\hat{y}$ and the true values (y) [11]:

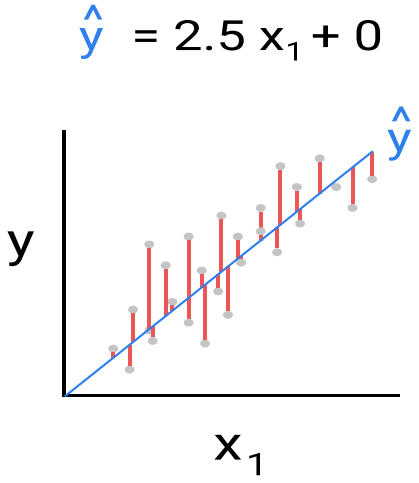

Figure 5: Difference between the predicted values for the target column and the true values [4]

We want this difference to be as small as possible. Calculating RSS involves summing the squared errors [11]:

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

1.3 Gradient descent

Each combination of unique parameter values forms a unique linear regression model, and the process of finding these optimal values is known as model fitting [14]. Model fitting aims to minimize the following function:

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

Gradient decent is the most common way for finding the optimal parameter values for a linear regression model. The gradient decent algorithm works by iteratively trying different parameter values until the model with the lowest mean squared error is found. Here's an overview of the gradient descent algorithm for a single parameter linear regression model [15]:

- select initial values for the parameter
- repeat until convergence
- calculate the error (MSE) of model that uses current parameter value: $MSE(a_1)$

- calculate the derivative of the error

$$\frac{d}{da_1} MSE(a_1)$$

- update the parameter value by subtracting the derivative times a constant α:

$$a_1 := a_1 - \alpha \frac{d}{da_1} MSE(a_1)$$

The main challenges with gradient descent include:

- choosing good initial parameter values
- choosing good learning rate

Selecting an appropriate initial parameter and learning rate will reduce the number of iterations required to converge and is part of hyperparameter optimization [14].

1.4 Base models

1.4.1 Kernel Ridge Regression

We would like to decrease the model complexity that is the number of the predictive features. Removing predictors from the model can be seen as setting their coefficients to zero. Instead of forcing them to be exactly zero, let's penalize them if they are too far from zero, thus enforcing them to be small in a continuous way. This way, we decrease model complexity while keeping all variables in the model. This, basically, is what Ridge Regression does. In Ridge Regression, the loss function is augmented in such a way that we not only minimize the sum of squared residuals but also penalize the size of parameter estimates, in order to shrink them towards zero. Ridge regression adds a factor of sum of squares of coefficients in the optimization objective. Thus, ridge regression optimizes the following [16]:

$$Objective = Loss\ function + a * (sum\ of\ square\ of\ coefficients)$$

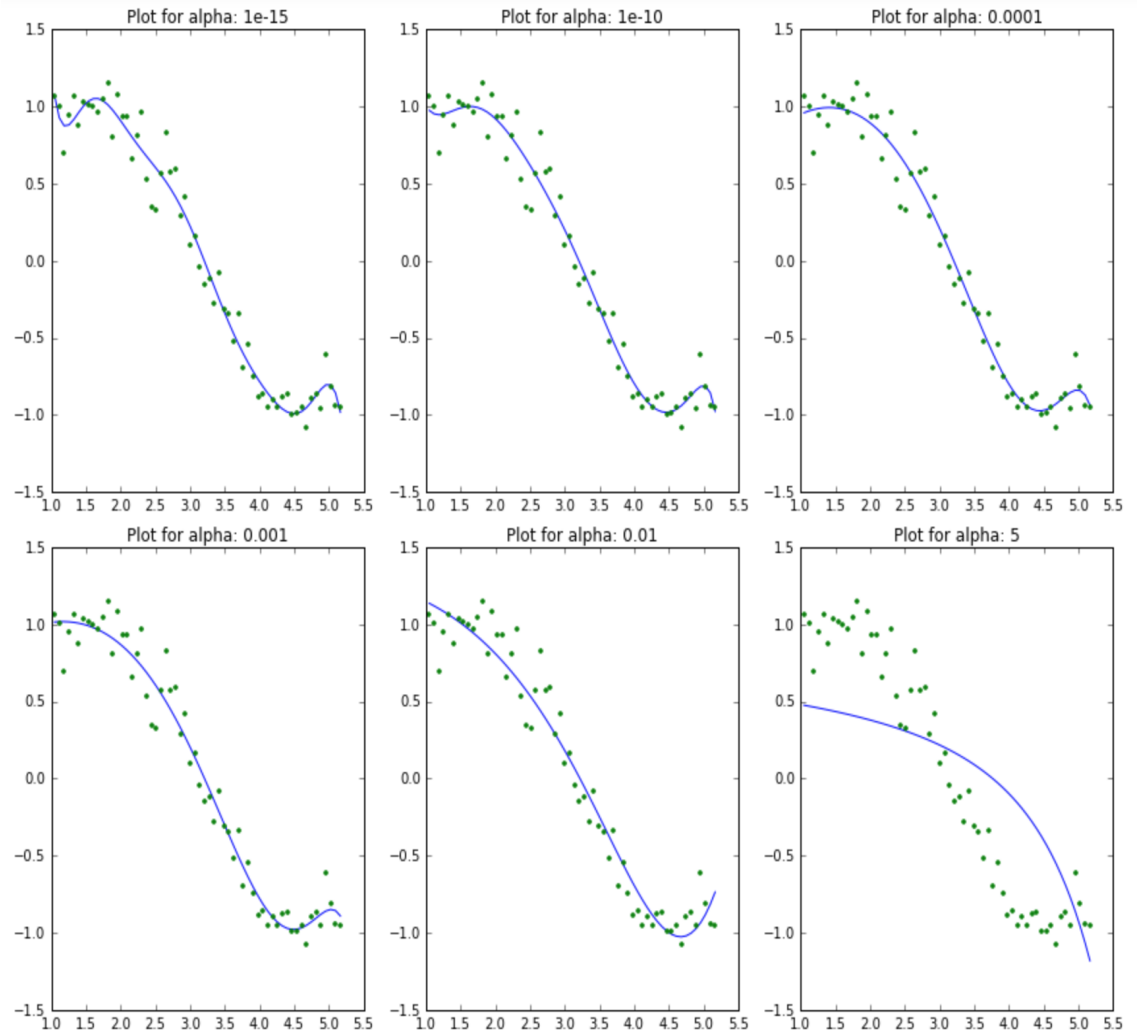

Figure 6: Model complexity changes in the Ridge regression [16]

1.4.2 LASSO Regression

Lasso, or Least Absolute Shrinkage and Selection Operator, is quite similar conceptually to Ridge regression. It also adds a penalty for non-zero coefficients, but unlike ridge regression which penalizes sum of squared coefficients, lasso penalizes the sum of their absolute values. As a result, for high values of λ, many coefficients are exactly zeroed under lasso, which is never the case in ridge regression [16].

$$Objective = Loss\ function + a * (sum\ of\ absolute\ value\ of\ coefficients)$$

Here, α (alpha) works similar to that of ridge and provides a trade-off between balancing the loss function and magnitude of coefficients. Like that of ridge, α can take various values [16].

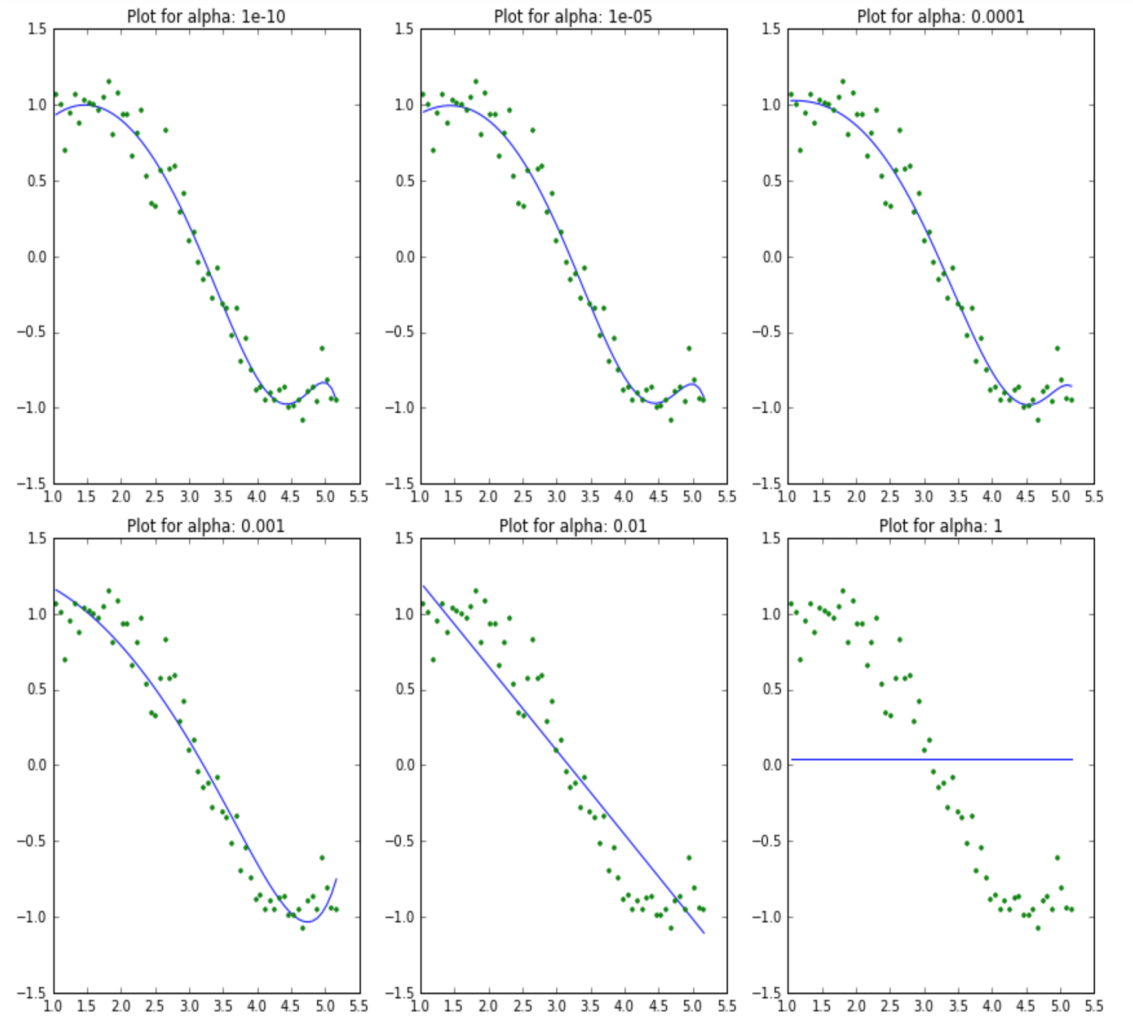

Figure 7: Model complexity is changing in the LASSO regression [16]

1.4.3 Elastic Net Regression

Elastic Net first emerged as a result of critique on lasso, whose variable selection can be too dependent on data and thus unstable. The solution is to combine the penalties of ridge regression and lasso to get the best of both worlds. Elastic Net aims at minimizing loss function where α is the mixing parameter between ridge ($\alpha = 0$) and lasso ($\alpha = 1$) [16].

1.4.4 Gradient Boosting Regression

We want our predictions, such that our loss function (MSE) is minimum. By using gradient descent and updating our predictions based on a learning rate, we can find the values where MSE is minimum. So, we are basically updating the predictions such that the sum of our residuals is close to 0 (or minimum) and predicted values are sufficiently close to actual values.

The boosting algorithm can be seen as a form of gradient descent that optimizes the squared error loss function, because in each step, it adds a sub-model that tries to mimic the negative gradient of this loss [17].

In simple words gradient boosting combines a set of weak learners and delivers improved prediction accuracy. At any instant t, the model outcomes are weighed based on the outcomes of previous instant t-1. The outcomes predicted correctly are given a lower weight and the ones miss-classified are weighted higher [18].

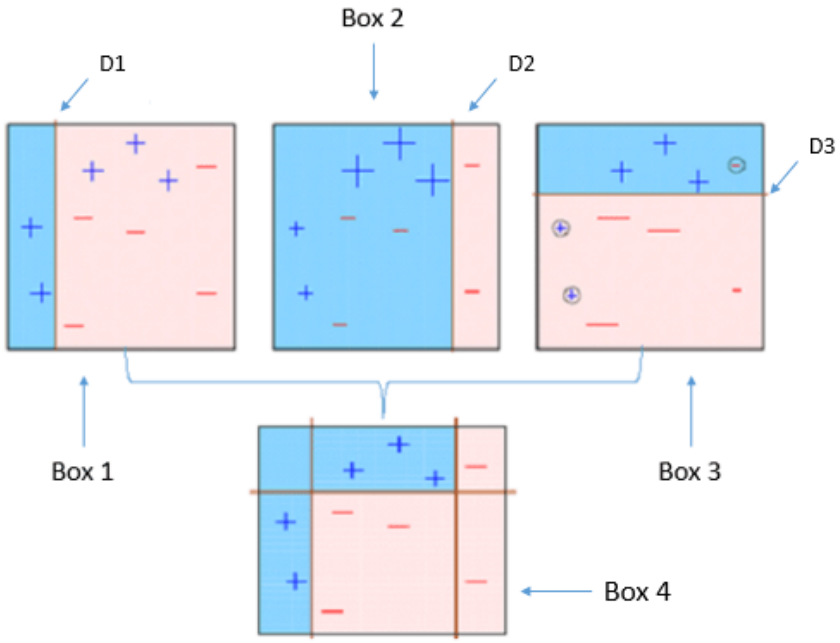

Figure 8: Gradient boosting classifies different features [18]

Box 1: It says anything to the left of D1 is + and anything to the right of D1 is -. However, this classifier misclassifies three + points.

Box 2: The second classifier gives more weight to the three + misclassified points.

Box 3: Again, the third classifier gives more weight to the three - misclassified point.

Box 4: This is a weighted combination of the weak classifiers.

That's the basic idea behind boosting algorithms is building a weak model, making conclusions about the various feature importance and parameters, and then using those conclusions to build a new, stronger model and capitalize on the misclassification error of the previous model and try to reduce it [18].

1.4.5 XGBoost model

XGBoost belongs to a family of boosting algorithms and uses the gradient boosting framework at its core. The default base learners of XGBoost are tree ensembles. XGBoost specifically, implements gradient boosting for ensemble trees with an additional custom regularization term in the objective function [18].
The tree ensemble model is a set of classification and regression trees
Trees are grown one after another and attempts to reduce the misclassification rate are made in subsequent iterations. The prediction scores of different trees then are summed up to get the final score in the objective function [19].

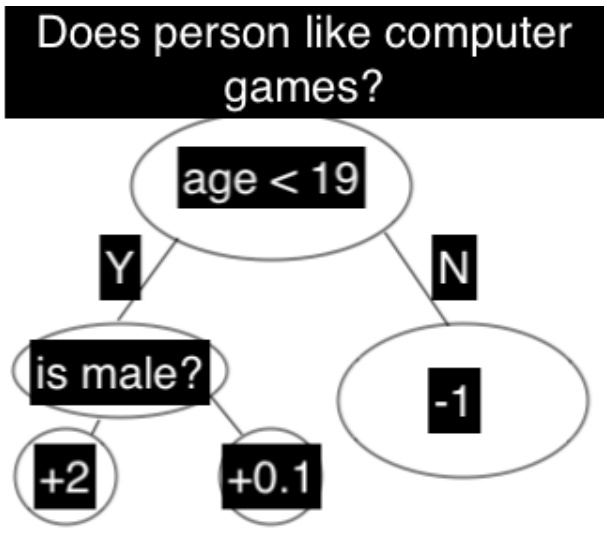

Figure 9: Ensemble tree that classifies whether someone will like computer games

The training process is very similar to previous models. We define an objective function with custom regularization and optimize it. Optimization happens using additive strategy. Learning tree structure is much harder than traditional optimization problem where you can simply take the gradient. It is intractable to

learn all the trees at once. Instead, we use an additive strategy: fix what we have learned and add one new tree at a time [19].

1.4.6 Stacking model

Stacking is a model technique used to combine information from multiple predictive models to generate a new model. Often times the stacked model will outperform each of the individual models due its smoothing nature and ability to highlight each base model where it performs best and discredit each base model where it performs poorly. For this reason, stacking is most effective when the base models are significantly different. The algorithm can be visualized as follows [20]:

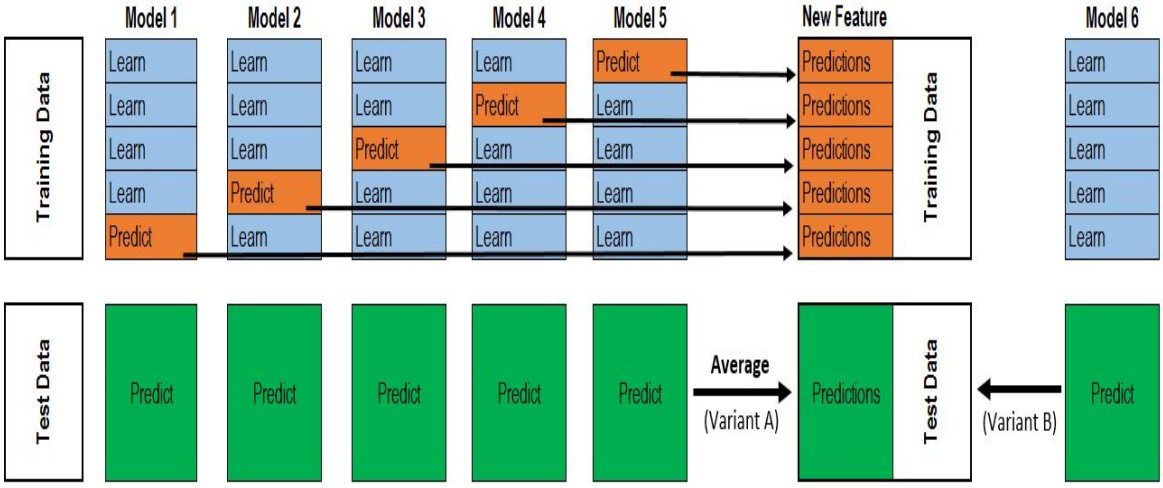

Figure 10: Visualization of the stacking model algorithm [20]

2 Methodology

Various methods and technologies can be utilized to tackle the house price prediction model. The ones used in this thesis are chosen with accessibility and efficiency in mind.

2.1 Tools

Most of the thesis problems is solved by using Python because it has very active community of the developers that implement open source libraries for server, data collection and scientific solutions. Frontend is also represented by free to use framework written in JavaScript.

One of the popular programming languages for machine learning is Python. One of the reasons is its simplicity and readability. Python boast a good number of libraries and frameworks for machine learning, compiled languages are much more suitable due to their efficiency and speed. In this thesis all the data preprocessing, visualization, analysis and cleaning is done by using next Python libraries: Pandas, NumPy, Matplotlib. Most of the prediction models are provided by Skearn library [21].

Data collection is implemented using Python library Scrapy. The library suits very well for single and not autonomous scraping of webpages. The code implemented is able to walk through pages in the City24 web site and collect raw data based on the defined criteria [22].

Open source and widely spread server framework is Flask [23]. It is used on the server side where the model is deployed. One of its features is modularity and structure. Structure makes it light-weighted framework and allows usage of different modules on demand.

Frontend is implemented using modern framework called VueJs [24]. It is simple, light-weight and provides single page application solution, which is very popular nowadays. It communicates with the server using Ajax requests.

2.2 Strategy

We need data on which we can train our models. The way we choose the web resource from which we collect data is based on its popularity and amounts of data accessible in it.

After choosing the resource, we will tend to collect as much data as we can because linear regression models perform better if they are trained on huge amounts of unbiased data. Tallinn city has a lot of modern and new build apartments. There are outliers like luxury apartments in the Old Town. We will try to escape these exceptions and remove them from our distribution as people tend to live in smaller homes/apartments apart from the extremely wealthy [4].

The way predicting models are chosen is based on the analysis of the machine learning competitions, where the most popular models are used because they have high winning score and low error rate. The effectiveness of these models is also confirmed by the last researches in this area.

Hyperparameter optimization of the models is tending to be called an art and does not have any optimal formula for finding the most effective parameters. It requires a lot of experience in machine learning in production environment and since the author of this work is lacking the strong experience in this area, we will adopt the experience from the articles and successful works of machine learning competitions and try to use it in our models. We will improve model optimization parameters through iterative approach [4].

3 Data collection

The beginning of data collection starts from defining the resource. During the research, no accessible web portal providing API for data collecting was found. The decision was to implement web spider that collects data. One of the best candidates is City24 [23] portal. It contains no captcha checking and data is publicly available.

3.1 Web scraper

Every web resource has its own HTML markup and sitemap.xml file that contains information about website structure for web spiders. Our task is to define it correctly in the programming code. In our case we use XPath syntax that parses HTML structure and assists in extracting the data from HTML elements. City24 web page structure consists of the list page where different house offers are located with the links to full description. Web spider finds for the link to the next page with the list of offers. Traverses the offers and collects the data from the links and goes to the next page. This whole system is working according to simple algorithm [22].

```
def parse(self, response):
       link_list =
       response.xpath('/a[@class="addressLink"]/@href').extract()
       for house_page in link_list:
              if house_page is not None:
                     yield scrapy.Request(house_page,
              callback=self.parse_property)

       next_page =
       response.xpath('///a[@class="next"]/@href').extract_first()
              if next_page is not None:
              self.log("Next page link {}".format(next_page))
              yield response.follow(next_page, self.parse)
```

Figure 11: Simple page scraping algorithm

4 Data preprocessing

4.1 Raw data analysis

During the initial data lookup there were unnecessary feature variables such as: descriptions, links, broker company names and etc. They were removed immediately. Columns that have a lot of null values can have some impact on our prediction model, but there are no ways to fill these columns with some average data because overall amount of null values is huge. The raw data consists of 10340 rows and 15 columns of features.

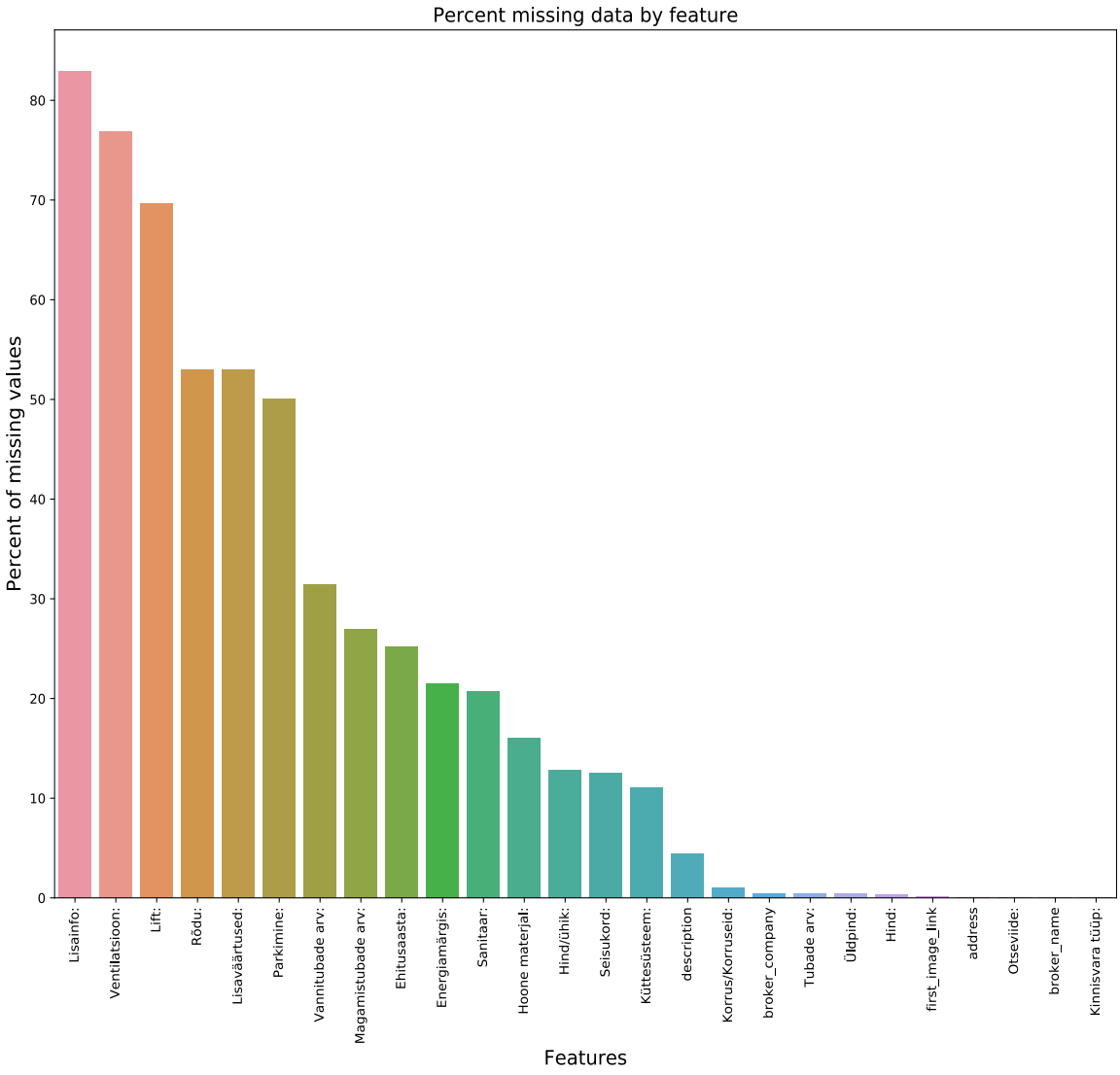

Figure 12: Percentage of missing data

Estonian	English
Lisainfo	Additional information
Ventilatsioon	Ventilation
Rõdu	Balcony
Ehitusaasta	Year of construction
Enrgiamärgis	Energy label
Hind/Ühik	Price per square meter
Seisukord	Condition
Küttesüsteem	Heating system
Üldpind	Total area

Table 1: Feature variable translation from Estonia to English

4.2 Missing values

Some data objects were lack of address information. This fact is important because address defines house itself. Without location we cannot determine if house is located in Tallinn and classify it. If there are no address, data object cannot be used in production environment. Next step was to remove such objects.

In our data distribution there are three important features that influence the model. They are: “hind”, “üldpind”, “tubade arv” and “hind/ühik”. Column “hind” is required for training because it is our target. The decision was to extract price if it is null by multiplying “üldpind” and “hind/ühik” columns.

Build year is another feature that can tell something about the price. Houses with the same address were given the same build year if they were missing it.

Author of this work tried to save as much data as it is possible because a lot of data were filtered. Many missed values were extracted and filled in by using information from rows with the same address.

```
def extract_column_from_similar_object(address_list_with_nan, nan_column, df):
    for address in address_list_with_nan:
        column_with_notna = df[nan_column].notnull()
        has_addr = df['address'] == address
        found_similar = df[column_with_notna & has_addr]
        if not found_similar.empty:
            found = found_similar[nan_column].value_counts().index[0]
            df.loc[df['address'] == address, nan_column] = found
```

Figure 13: Algorithm of filling rows with the same address

4.3 Categorical features

Pandas [24] library is a good tool to work with the data and contains all the utilities to create dummy columns and convert categorical features. The data set we used to train the model contains a lot of categorical features. Initially they were in the string format, what is inappropriate. The way columns were converted to categorical features called dummy coding. We check if row contains the value regarding the column and depict this by creating new column with the value in the header. Column is populated with true or false values accordingly. Moreover, many columns in the raw set contained array of values. Our task was to extract unique values and create new categorical columns with the technique described above.

```
unique_heating_types = set()
def extract_unique(row, unique_set):
    for el in row.split(','):
        el = el.lstrip().rstrip()
        unique_set.add(el)
tln_df['küttesüsteem'].apply(lambda x: extract_unique(x, unique_heating_types))

for heating in unique_heating_types:
    tln_df[heating] = np.nan
    tln_df.loc[tln_df['küttesüsteem'].str.contains(heating), heating] = heating
    tln_df[heating] = pd.get_dummies(tln_df[heating])
```

Figure 14: Algorithm of unique values extraction from array of values

address	küttesüsteem
A. Reinvaldi 3	elektriküte
J.Vilmsi 29	keskküte
Vesivärava tn 50	keskküte, põrandaküte
Vesivärava tn 50	keskküte, põrandaküte
Narva mnt 57	gaasiküte

Table 2: Categorical column before dummy coding

address	elektriküte	ahjuküte	keskküte
A. Reinvaldi 3	1	0	0
J.Vilmsi 29	0	0	1
Vesivärava tn 50	0	0	1
Vesivärava tn 50	0	0	1
Narva mnt 57	0	0	0

Table 3: Categorical column after dummy coding

4.4 Numerical features

The data set contains 5 numerical features. Some of the features include units and currency, which were removed by using regex. Numeric part of the data was converted to Pandas data type int64 and float64.

One of the important feature columns is “seisukord”, which describes the condition of the house. Initial column value is string consisting of array of description words. For example: ‘uus viimistlus’, ‘valmis’, ‘san remont tehtud’, ‘renoveeritud’. The decision was to make grade metric and group these definitions to the classes 1-3. Many of the description values have the same meaning, which gave us 3 logical categories.

1	new
2	ready
3	requires fixes

Table 4: Column 'seisukord' ordinal values

Another column that can improve model prediction is ‘'korrus/korruseid'. This value gives us information about the apartments floor. Usually this fact plays role in decision making when buying a house. The first part of the value was extracted to present a new column ‘'korrus’.

5 Data analysis

On this step data set is almost ready to fit in the model. Data set contains 2160 rows and 46 feature columns of house offers in Tallinn. Data is still needing some processing. There are technics to improve data quality by decreasing bias and variance tradeoff and log transforming of the data features.

address	hind	üldpind	ehitusaasta	rõdu	seisukord	tubade_arv	Haabersti	Kadriorg	Kesklinn	Kristiine	Lasnamäe	Mustamäe	Nõmme	Pirita
Ao tn 2	649800	148.3	2018	1	1	4	0	0	1	0	0	0	0	0

Põhja-Tallinn	Vanalinn	terrass	elektriküte	tahkekütus	põrandaküte	autnoomne k	kombineeritud k	õhksoojuspum	keskküte	ahjuküte	maaküte	gaasiküte	kamin	kivimaja	plokkmaja
0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0

betoonmaja	palkmaja	puitmaja	palk-kivimaja	paneelmaja	energy_class_a	energy_class_b	energy_class_c	energy_class_d	energy_class_e	energy_class_f	energy_class_g	korrus	saun	bassein
0	0	0	0	0	0	1	0	0	0	0	0	5	1	0

Table 5: One sample of data ready for model

5.1 Outlier features

Outlier features increase the bias of the distribution, which does not improve model well. In our case these are the houses with the price bigger than normal distribution. These are the luxury segment of the property market, which are rare. By visualizing the relationship between target and some feature column, we can decide to remove them or not.

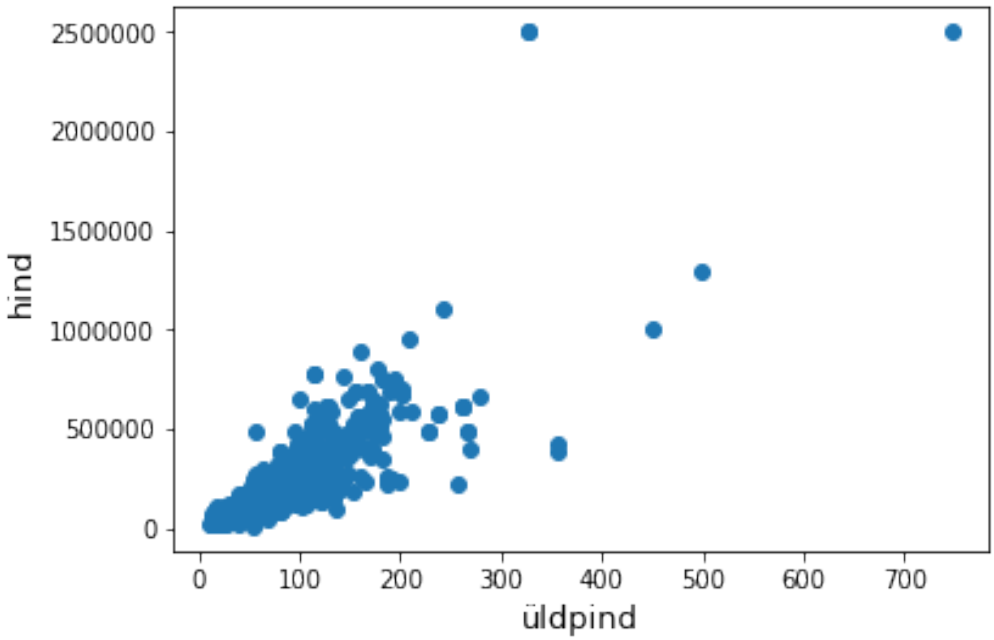

Figure 15: Plot of target and variable relationship

We can see at the bottom right two with extremely large 'üldpind' that are low in price and some at the top right with high price. These values are huge outliers because they concern to the utility area or some luxury segment. Therefore, we can safely delete the offers that are higher than 1500000 EUR and area is more than 500 square meters. We decided to delete these two as they are very huge and really bad (extremely large areas for very low prices). There are probably other outliers in the training data. However, removing all them may affect badly our models if ever there were also outliers in the test data.

5.2 Data correlation with the target column

One of the practices used in the data science is to build correlation matrix. It helps to visualize how different features influence the target variable. By using visual representation of correlation values, we can understand the strength of the features [4].

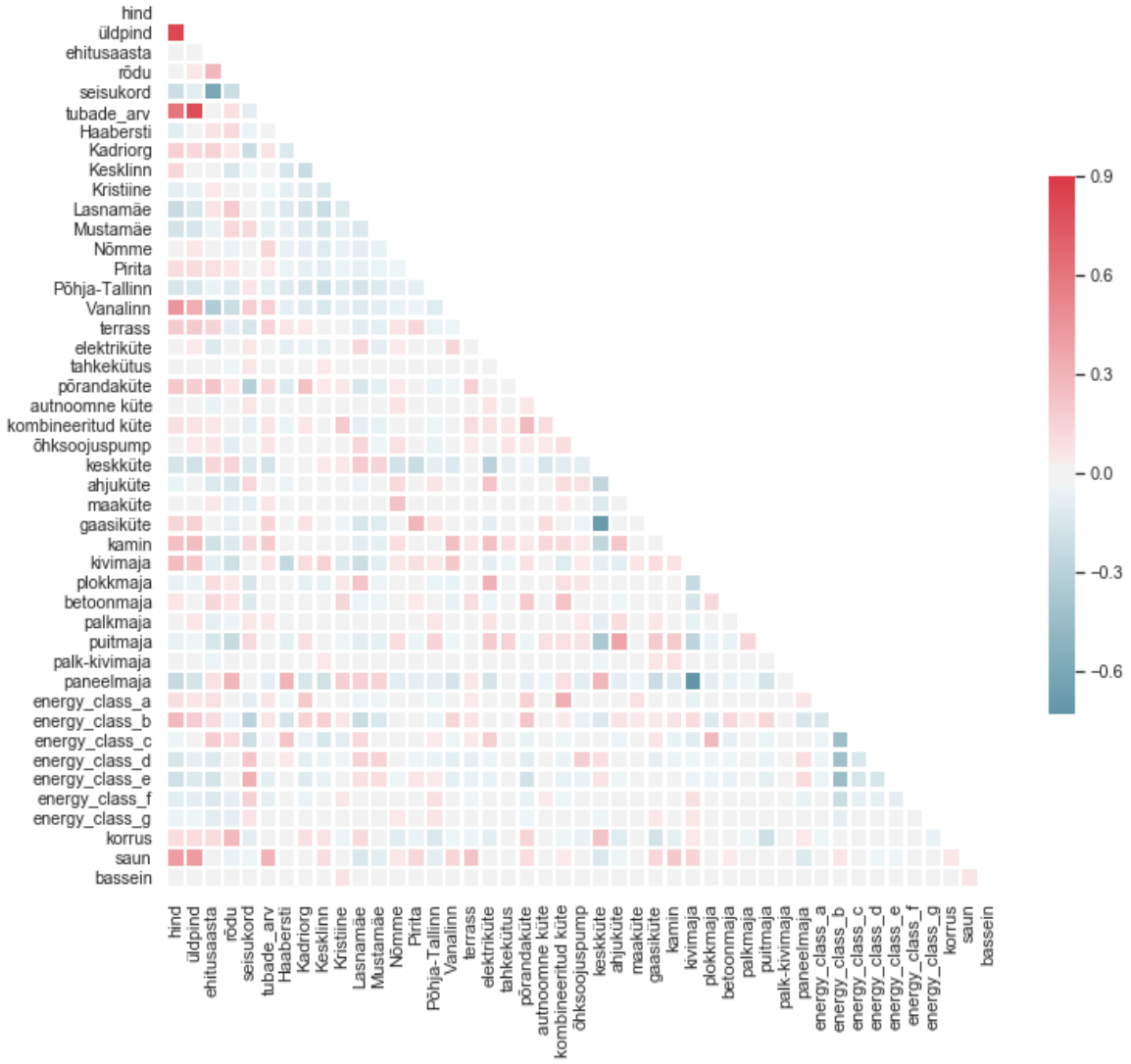

Figure 16: Correlation matrix between the features

The next thing we need to look for is for potential collinearity between some of these feature columns. Collinearity is when 2 feature columns are highly correlated and stand the risk of duplicating information. If we have 2 features that convey the same information using 2 different measures or metrics, we don't need to keep both [4].

Based on the correlation matrix heatmap, we can tell that the following pairs of columns are strongly correlated:

- ‘üldpind’ and ‘tubade_arv’

At the moment I decide to keep them both because in my opinion number of rooms can hold some useful information about architecture of the house that will have influence on the price.

5.3 Data transformation

According to [25] data normality should be tested. When we talk about normality what we mean is that the data should look like a normal distribution. This is important because several statistic tests rely on this. We are going to check univariate normality for 'hind' column, which is a limited approach. However, if we solve normality, we avoid a lot of other problems so that's the main reason why we are doing this analysis.

mean	165929.155916
standard deviation	117624.574339
min	10890.000000
25%	86950.000000
50%	139000.000000
75%	205000.000000
max	1000000.000000

Table 6: Column 'hind' main characteristics

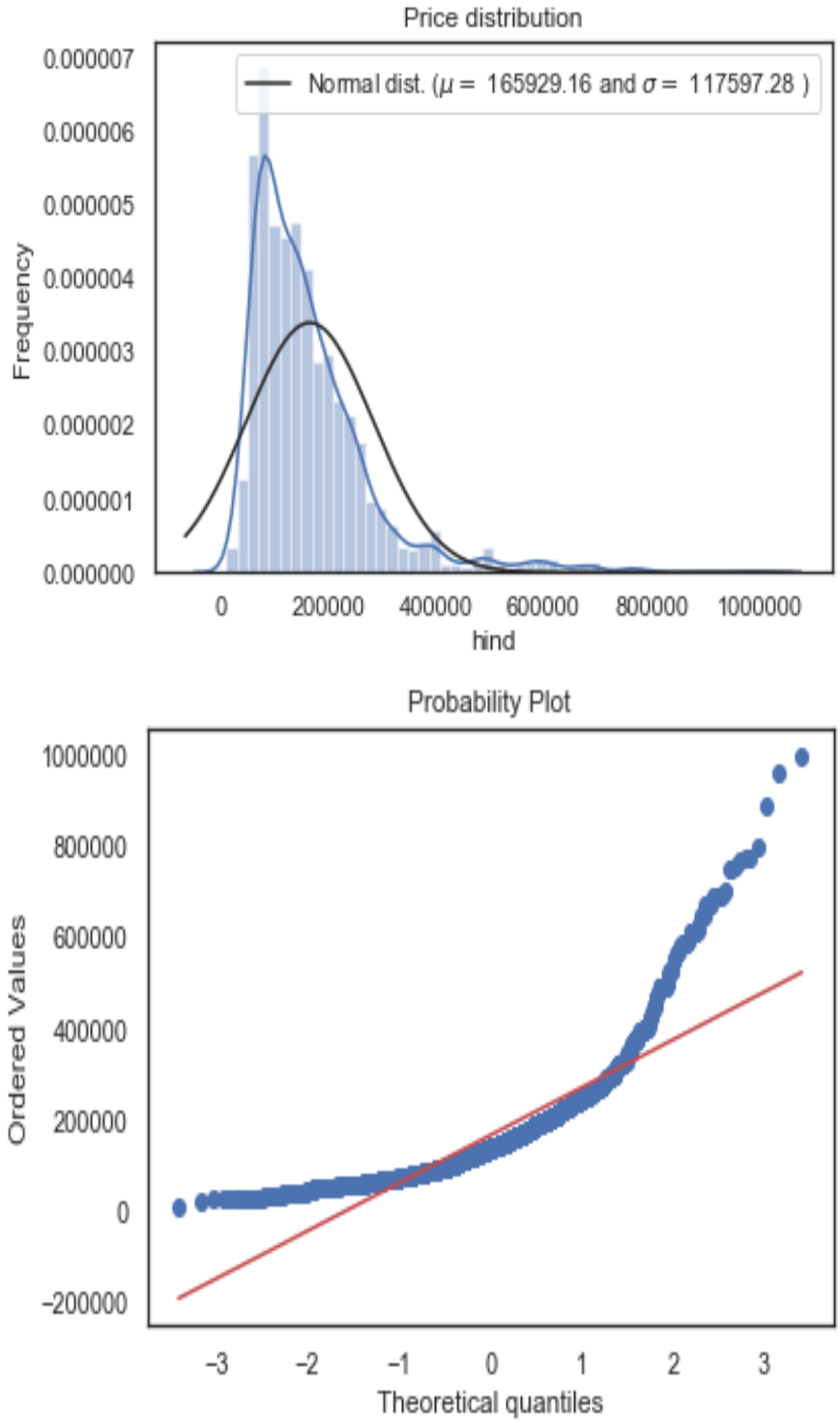

Figure 17: Skewed data distribution

'hind' column is not normal. It shows peaks, positive skewness and does not follow the diagonal line. The target variable is right skewed. As linear models prefer normally distributed data, we need to log transform this variable and make it more normally distributed [26].

Normal distribution of the data shows how data is deviated from mean. Normal distribution is symmetric, and mean is located in the center. In our case it is not normally distributed. Log transformation helps to solve this problem. We transform each value of the house price using logarithmic function in base 10. This scales the values to be in the lower numeric bounds. It is hard to discern a pattern in the upper panel whereas the strong relationship is shown clearly in the lower panel.

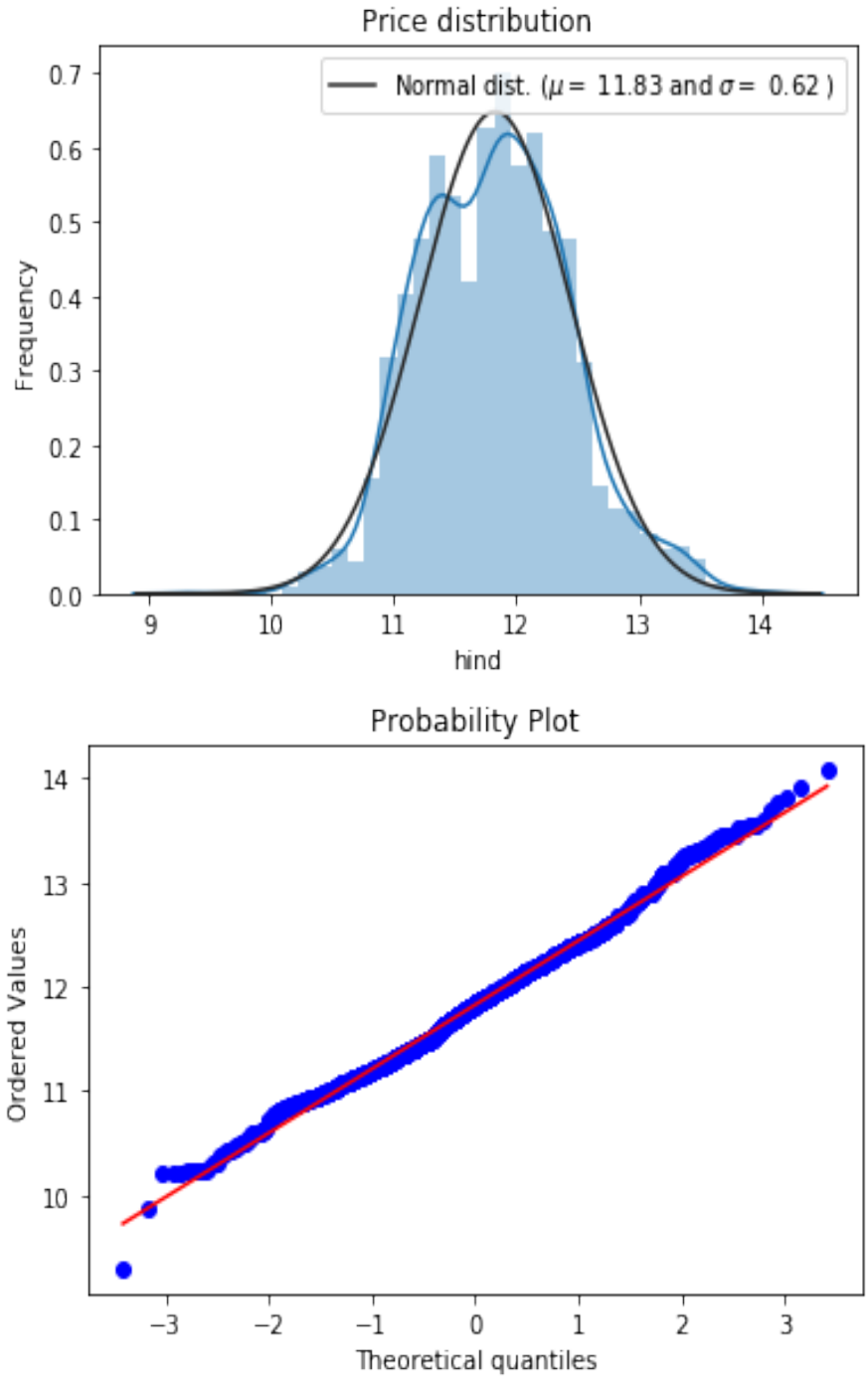

Figure 18: Normally distributed data

The skew seems now corrected and the data appears more normally distributed.

6 Prediction model

Every model fitting starts from splitting the data into training and data set. The simplest way is to split it to 50/50, 80/20 proportions. Most of the effective ways to split the data is K-Fold validation. Shortly speaking the data is split in N parts and each part is used as train and test data in the model. On every test fold error metric is computed. Repeating all of the above steps N-1 times, until each partition has been used as the test set for an iteration. The mean of the N error values is calculated [27].

The 13 is the chosen number for K-Fold validation in our models. We found it by trying different fold numbers in simple linear model and choosing N that is the average lying between the lowest and the highest error rate.

Folds number	Average RMSE
9 folds	0.22726638796045937
10 folds	0.21541617124257914
11 folds	0.21705850123209913
13 folds	0.21016656462538316
15 folds	0.20492095143391523
17 folds	0.21078822185854995
19 folds	0.2086677067383715

Table 7: K-Fold numbers and their average RMSE

6.1 Linear regression models fitting

The way models' parameters were optimized to increase performance are described in the next steps [28]:

- first fit classifier with default parameters to get a baseline idea of the performance
- train model using different parameters
- calculate RMSE
- choose parameter that show the lowest RMSE

6.1.1 Base models

```
lr = LinearRegression()
lasso = make_pipeline(RobustScaler(), Lasso(alpha =0.0005, random_state=1))
ENet = make_pipeline(RobustScaler(), ElasticNet(alpha=0.0005, l1_ratio=.9,
random_state=3))
KRR = KernelRidge(alpha=0.6)
```

Figure 19: Base model and their parameters initialization [29]

Next parameters can be configurable to optimize the performance of the model.

- LASSO and Elastic Net models may be very sensitive to outliers (extreme values).We need to make it more robust on them. For that we use the skikit-learn Robustscaler() method on pipeline, which scales data increasing data accuracy [29].
- Alpha parameter is the optimization parameter that is used in the loss function, which defines coefficient on which error weight will be multiplied [29].
- Random state is the seed of the pseudo random number generator that selects a random feature to update [29].
- L1_ratio parameter is mixing parameter that defines penalty value given to the parameter estimates [29].

6.1.2 Boosting models

There are used 2 boosting models described in the theoretical background section. They are: Gradient Boosting Regression, XGBoost model.

- Learning rate shrinks the contribution of each tree by learning_rate.
- n_estimators represents the number of trees in the forest. Usually the higher the number of trees the better to learn the data. However, adding a lot of trees can slow down the training process considerably, therefore we do a parameter search to find the sweet spot.
- max_depth. This indicates how deep the built tree can be. The deeper the tree, the more splits it has, and it captures more information the data. We fit a decision tree with depths ranging from 1 to 10.
- min_samples_split represents the minimum number of samples required to split an internal node. This can vary between considering at least one sample at each node to considering all of the samples at each node. When we increase this parameter, the tree becomes more constrained as it has to consider more samples at each node. Here we will vary the parameter from 0% to 50% of the samples
- min_samples_leaf is the minimum number of samples required to be at a leaf node. This similar to min_samples_splits, however, this describe the minimum number of samples of samples at the leaves.
- max_features represent the number of features to consider when looking for the best split. [31]

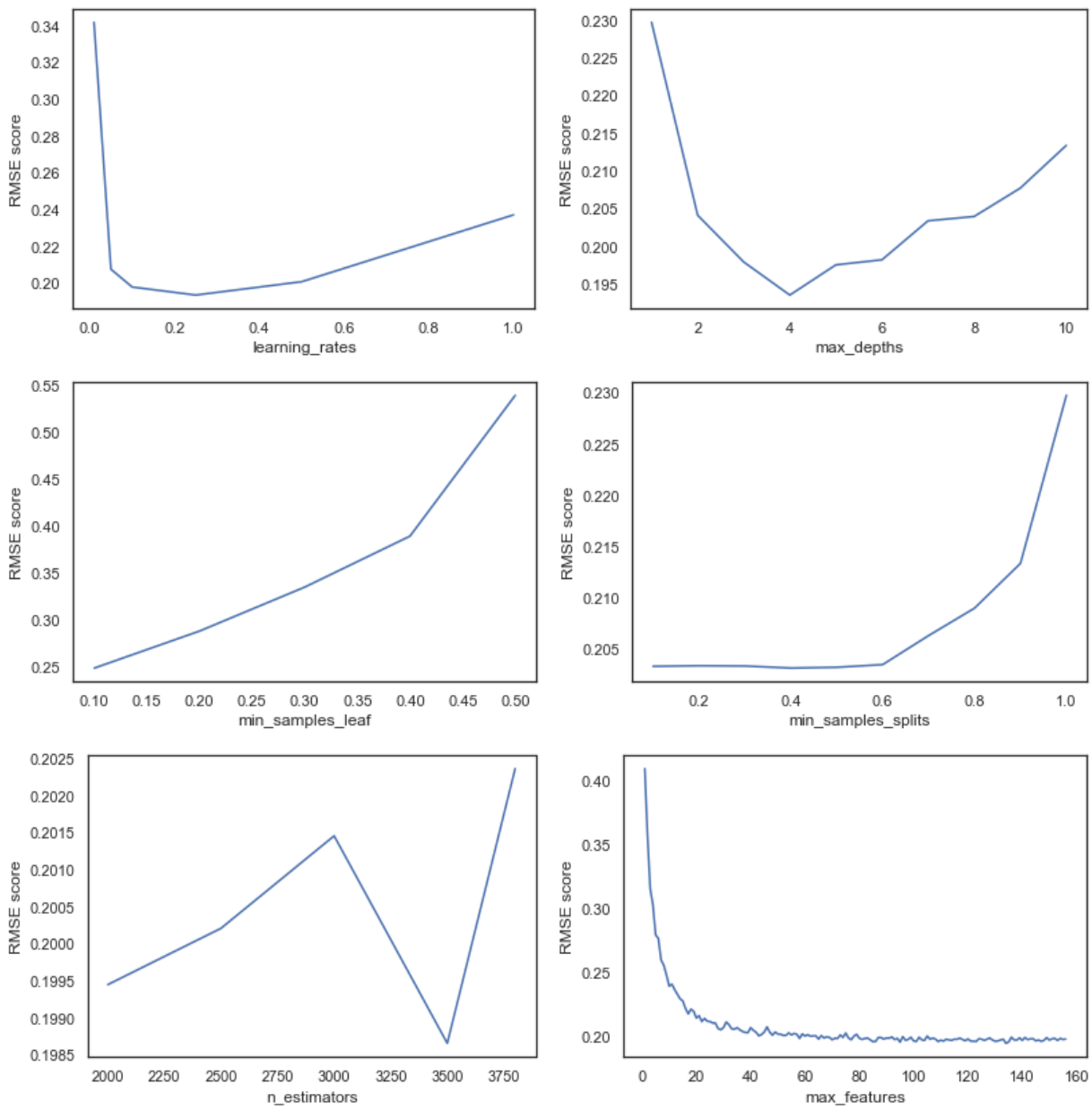

Figure 20: RMSE value changing during parameters optimization

The parameters that give the result of the prediction with the lowest RMSE value were chosen.

6.1.3 Stack and ensemble models

The reason stacking, and ensemble technique is used in this work is high scores on the machine learning competitions. For the prediction part, we average the predictions of all base models on the test data and used them as meta-features on which, the final prediction is done with the meta-model. Model provides us with a good result.

```
stacked_averaged_models = StackingAveragedModels(base_models = (ENet, GBoost, KRR),
                                                 meta_model = lasso)
```

Figure 21: Stacked model initialization

Models ensemble includes adding additional model and add up prediction results in proportions 70/30. In our case we take 70% of prediction result from stacked averaged model as it performs better. The second model is XGBoost as one of the most effective models. We take from it 30%.

```
stacked_averaged_models.fit(X_train, y_train)
stacked_test_pred = stacked_averaged_models.predict(X_test)

model_xgb.fit(X_train, y_train)
xgb_test_pred = model_xgb.predict(X_test)

ensemble = stacked_test_pred * 0.70 + xgb_test_pred * 0.3
```

Figure 22: Implementation of model ensemble

6.2 Model comparison

The performance of the models is measured by RMSE error metric. Models with the lower values predict with less errors.

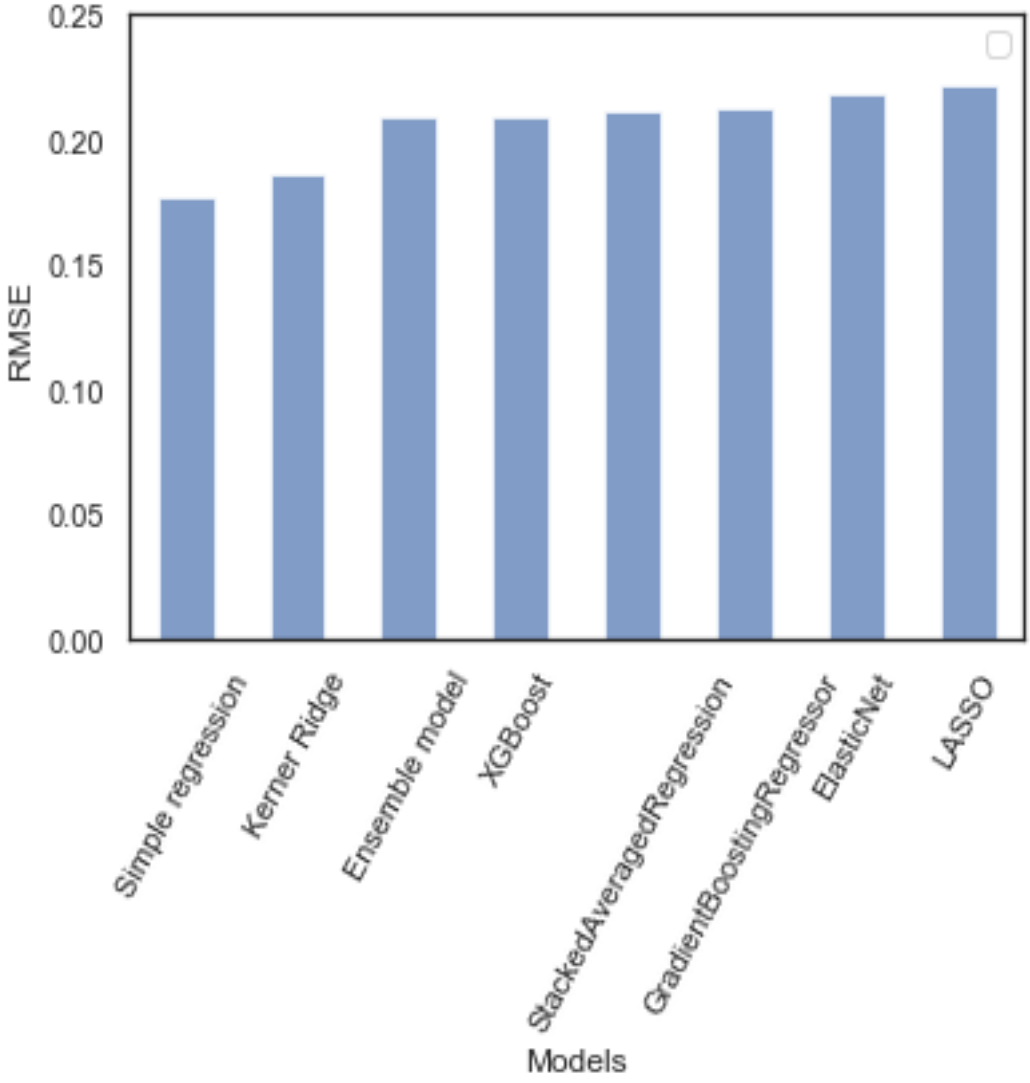

Figure 23: Model performance comparison

- The most effective base models in our case are Simple Regression and Kernel Ridge.
- Lasso and ElasticNet could not find their usage in this dataset and show worst performance. The reason lies in the dataset. It has small amount of data rows and a lot of outliers such as luxury segment houses. These models perform badly in such case.
- GradientBoostingRegressor model performs badly because it is vulnerable to small dataset. Usually it shows high performance with the dataset over 10000 rows.

- Ensemble model found itself on the third position. Although RMSE is a proper way of model estimation, but it cannot say everything about model performance. Ensemble model unites stacked base model averages and second model. In production it is considered the best model.

7 Deploying model

The base structure of the application is simple. It consists of server and view layer. Server is responsible for processing JSON requests, checking the validity of the request body and returning prediction result. On the view layer there is form with 12 parameters. On submit button form is being parsed in JSON format and sent using asynchronous HTTP request.

7.1 Backend

Backend is implemented using Flask framework. It includes annotation methods for defining request routes, libraries that assist in parsing JSON requests and utility functions to validate parsed data.

Trained models are exported as file in flexible Python ".pkl" format. This file is a serialization of the model, which holds trained weights and meta information. During server initialization model is loaded from predefined directory into the memory.

On client request server checks whether it is in the correct JSON format.

```
def get_valid_json_schema():
    valid_schema = {
                'type': 'object',
                'properties': {
                    'area_meters': {'type': 'number'},
                    'condition': {'type': 'number'},
                    'rooms': {'type': 'number'},
                    'build_year': {'type': 'number'},
                    'floor': {'type': 'number'},
                    'balcony': {'type': 'number'},
                    'terrass': {'type': 'number'},
                    'sauna': {'type': 'number'},
                    'pool': {'type': 'number'},
                    'house_material': {'type': 'string'},
                    'linnaosa': {'type': 'string'},
                    'energy_class': {'type': 'string'},
                    'heating': {'type': 'array'}
                }
    }
    return valid_schema
```

Figure 24: Implementation of JSON checking

Next step is validation of the JSON fields. It must be in the correct format. Validated data is inserted in the loaded models using “skikit-learn” library fit() method. The result is returned in JSON response. Response includes the predicted price of the property according to given criteria.

7.2 Frontend

Frontend part is made using VueJs framework. A single page application is independent from server and is rendered on client-side. User requests server by using home link. Server responds with bare-bones HTML document with a JavaScript file that will render the rest of the site using the browser [30].

Application represents a house price calculator, which predicts estimated price for house to the given criteria. It includes post form which consists of 12 fields. These fields represent criteria which will be fit in the linear model on the server for prediction. User inputs in the form parameters of the desired property and clicks button. An asynchronous request to the server is made with the help of Axios library. The result of the prediction is displayed in the table format. Table shows field values from the post form and prediction result.

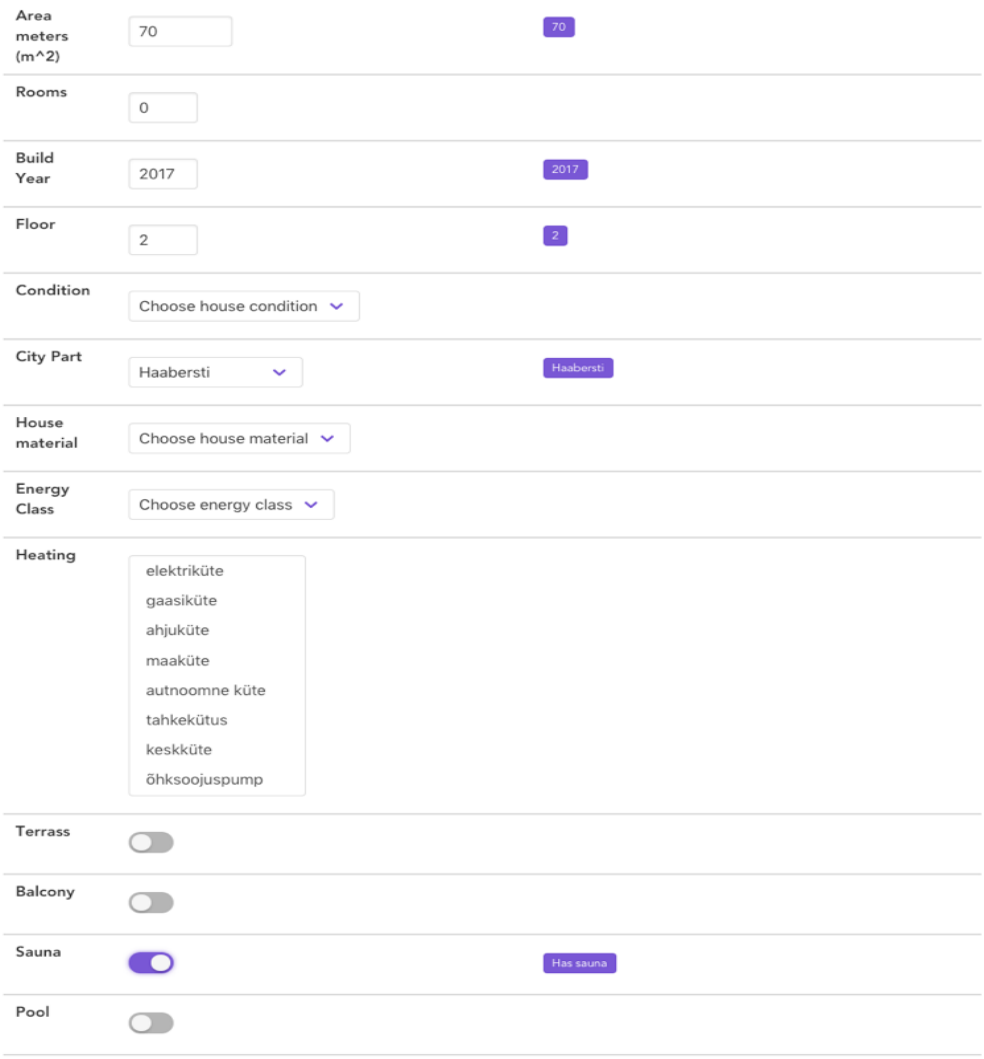

Figure 25: Post form of VueJS application

According to the given fields in the form a request to the server is made to fetch histogram plot data for normal distribution graphic visualization. Graphic is displayed in the lower part of the page. Horizontal axis includes price, vertical includes the number of houses or probability with the given parameters. The histogram shows the information about similar offers on the market, which

improves decision making.

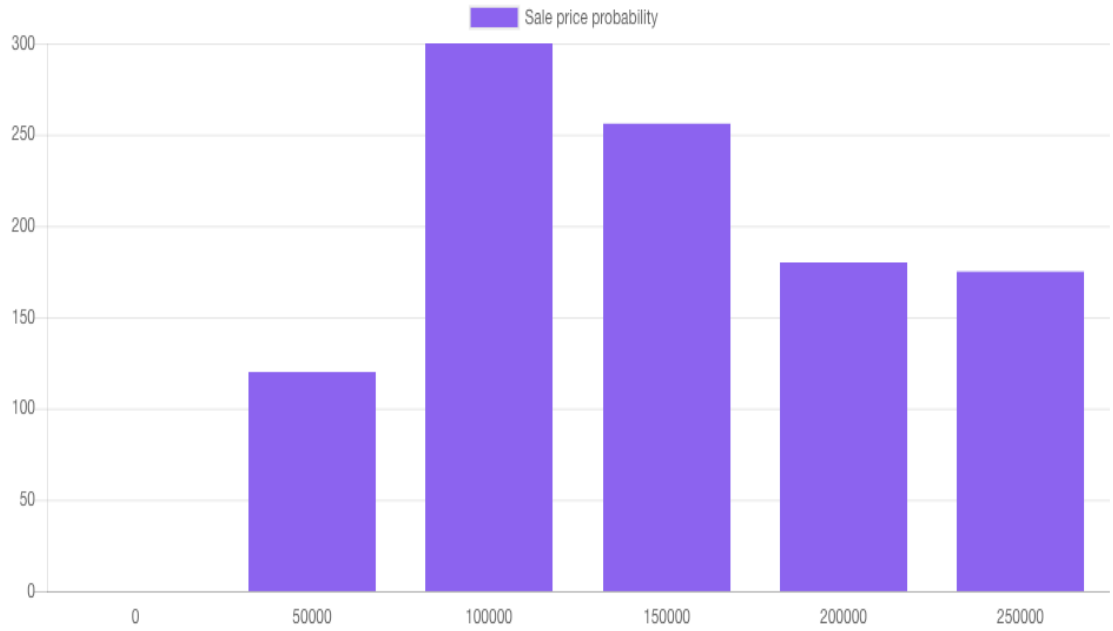

Figure 26: Web application histogram

8 Analysis

8.1 Success

The success of the models proposed in this thesis can be attributed to a number of factors. Naturally, the most important one is the proper data preprocessing. Raw dataset contained string values as features. Most of them were array of strings. Accurate feature extraction and categorical columns creation increased dataset variability. Raw dataset contained 15 columns, which were transformed into model readable columns in amount of 46.

Transformation of the data made dataset friendlier to linear models. Data columns with continuous variables were log transformed. It increased their normality and improved their distribution.

The latest growth of machine learning libraries and progress in statistical model development made it possible to archive good results in house price predictions. Model libraries used in this work were implemented for scientific purposes. They

are highly optimized and the language they are written in simplifies their usage in the real-world problems.
Significant part in every research plays visualization. Diagrams, bar plots provided by Matplotlib library contributed to quick problem identification.

8.2 Shortcomings

Linear regression models can perform differently, based on the dataset used to train them. It is very important to consider all the pros and cons of the models and use them accordingly. As far as models have tendency to overfitting, it is necessary to find the gold ratio in the bias and variance tradeoff by using more efficient data transformation technics [4].

Statistical problems prefer big distributions of data. Big datasets increase the quality of the research. The variability of the data plays very huge role in finding correct prediction results. The main difficulty is to find good resource that can provide normally distributed dataset with big number of features in it [4].

Optimization of the model parameters can decrease the number of errors significantly. This part requires a lot of experience in working with different dataset and using diverse stack of models. Learning rate, the number of leaves and other optimizations can behave differently in each model [4].

8.3 Future work

Machine learning is development is in a rapid progress nowadays. One of the ways to achieve success in this sphere is to try to solve different problems using most effective models. The latest model implementations tend to have higher scores on popular data science competitions. One of the aims is to use accumulated experience.

House price model prediction can find its usability in the business. Automation of the data collection and usage of trained models in the same time can provide fresh market price estimations. Web service with the client-friendly view and proper server architecture can be used to give its clients information about the prices of Tallinn property.

One of the perspective spheres of machine learning is neural networks. They require a lot of data to work properly. Every learning path towards neural networks lies through simple linear regression problems. By solving linear problems, it can open our eyes on the solutions regarding neural networks.

9 Summary

The aim of this work was to develop web application that produces prediction result of the house price on the request. As a part of the main objective, raw dataset was collected that was used in training the models. Dataset was processed in a suitable for the model format. The quality of the data was increased by transforming its features and decreasing the bias of it.

All the works and the narrow places of the dataset were visualized to have an idea about the context of the work. Diagrams and plots assisted in obtaining experience in data analysis.

Several models with various features and various model complexities were defined. It was realized that we need to use a mix of these models a linear model gives a high bias whereas a high model complexity-based model gives a high variance (overfit). The data overfitting was solved by using enhanced model like Ridge regression and LASSO, which tend to reduce this shortcoming.

The scores achieved by proposed models arise from a combination of a multitude of factors like parameter optimization, number of rows in the dataset, characteristics of dataset, features chosen and model behavior. These factors differ between models, so the similarity of fitness scores presents grounds for further analysis, comparison and interpretation of results.

In conclusion, the main objective of this thesis has been successfully accomplished. The end result is a working prediction model that allows for further experimentation and development.

References

[1] "https://www.kaggle.com/," Kaggle. [Online]. [Accessed 24 12 2018].

[2] Yang Li, Quan Pan ,Tao Yang and Lantian Guee, "Reasonable Price Recommendation on Airbnb Using Multi-Scale Clustering," *Proceedings of the 35th Chinese Control Conference,* pp. 27-29, 2016.

[3] M. B. a. M. A. Hasan, "Waiting to be Sold: Prediction of TimeDependent House Selling Probability," *International Conference on Data Science and Advanced Analytics,* pp. 468-477 , 2016.

[4] Andrew Bruce, Peter Bruce, "Practical Statistics for Data Scientists," O'Reilly Media, Inc., 2017.

[5] T. Pungas, "www.pungas.ee," [Online]. Available: https://pungas.ee/projektid/korterid-tallinnas-hinnastatistika/. [Accessed 21 12 2018].

[6] P. T. v. Hippel, "Mean, Median, and Skew: Correcting a Textbook Rule," *Journal Journal of Statistics Education,* vol. 13, no. 2, 2005.

[7] Dudewicz, E.J., and Mishra, S.N., "Modern Mathematical Statistics," Google Scholar, New York: Wiley, 1988.

[8] S. Singh, "Understanding the Bias-Variance Tradeoff," 2017. [Online]. Available: www.medium.com. [Accessed 13 12 2018].

[9] V. Singh, "An Intuitive explanation to Bias Variance Tradeoff," 2017. [Online]. Available: www.medium.com. [Accessed 11 12 2018].

[10] The Technical Whitepaper Series, "https://www.pbarrett.net," September 2005. [Online]. Available: https://www.pbarrett.net/techpapers/euclid.pdf. [Accessed 11 12 2018].

[11] D. Panchenko, "Simple Linear Regression," in *18.443 Statistics for Applications*, Massachusetts Institute of Technology: MIT OpenCourseWare, Fall 2006.

[12] W. a. S. T. Mendenhall, in *Statistics for Engineering and the Sciences (3rd ed.)*, New York, NY: Dellen Publishing Co, 1992.

[13] L. McIntyre, "Using Cigarette Data for An Introduction to Multiple Regression," *Journal of Statistics Education,* vol. 2, no. 1, 1994.

[14] P. Grover, "Gradient Boosting from scratch," *medium.com.*

[15] S. Ruder, "An overview of gradient descent optimization algorithms," 2016.

[16] A. Jain, "A Complete Tutorial on Ridge and Lasso Regression in Python," *Analytics Vidhya,* 2016.

[17] R. Johansson, "An intuitive explanation of gradient boosting," [Online]. Available: http://www.cse.chalmers.se/~richajo/dit865/files/gb_explainer.pdf.

[18] M. Pathak, "Using XGBoost in Python," 2018. [Online]. Available: https://www.datacamp.com/community/tutorials/xgboost-in-python. [Accessed 17 11 2018].

[19] xgboost developers, "Introduction to Boosted Trees," 2016. [Online]. Available: https://xgboost.readthedocs.io/en/latest/tutorials/model.html. [Accessed 24 11 2018].

[20] Tang, J., S. Alelyani, and H. Liu, "Data Classification: Algorithms and Applications," Data Mining and Knowledge Discovery Series, CRC Press, 2015, pp. 498-500.

[21] P. Patel, "Medium," May 2017. [Online]. Available: https://medium.com/@UdacityINDIA/why-use-python-for-machine-learning-e4b0b4457a77. [Accessed 19 12 2018].

[22] "Scrapy documentation," [Online]. Available: https://doc.scrapy.org/en/latest/intro/overview.html. [Accessed 12 10 2018].

[23] "City24," [Online]. Available: https://www.city24.ee/. [Accessed 12 12 2018].

[24] "Pandas Data Analysis Library," [Online]. Available: https://pandas.pydata.org/. [Accessed 24 12 2018].

[25] J. F. Hair, "Multivariate Data Analysis," Paperback, 2013, pp. 89-96.

[26] D. M. Lane, "Introduction to Statistics: An Interactive e-Book," 2013. [Online]. Available: http://onlinestatbook.com. [Accessed 18 12 2018].

[27] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, "An Introduction to Statistical Learning: with Applications in R," 2013, p. 181.

[28] M. B. Fraj, "In Depth: Parameter tuning for Gradient Boosting," 2017. [Online]. Available: https://medium.com/all-things-ai/in-depth-parameter-tuning-for-gradient-boosting-3363992e9bae. [Accessed 11 12 2018].

[29] "scikit-learn.org," [Online]. Available: https://scikit-learn.org/0.18/auto_examples/preprocessing/plot_robust_scaling.html. [Accessed 24 12 2018].

[30] T. Zhiyentayev, "https://hackernoon.com," [Online]. Available: https://hackernoon.com/server-side-vs-client-side-rendering-in-react-apps-443efd6f2e87. [Accessed 20 11 2018].

[31] L. S. Sterling, The Art of Agent-Oriented Modeling, London: The MIT Press, 2009.

[32] L. L. Victor Powell, "Ordinary Least Squares Regression Explained Visually," [Online]. Available: http://setosa.io/ev/ordinary-least-squares-regression/.