Универзитет „Св. Кирил и Методиј" во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Predicting apartment prices in Skopje

Elena Stefanovska[1] , Marija Velichkovska[2]

[1] *Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University in Skopje.*
[2] *Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University in Skopje.*

Abstract

Buying a new apartment, whether you are a university student or a middle-class family, is always a daunting process that often seems impulsive or risky. If the decision itself is already hard, doing it in Skopje doesn't make it any easier. In this project, we will go through an entire Data Science pipeline for apartment price prediction in Skopje, starting from the basics of gathering data, data integration, data cleaning, visualization, up until using multiple machine-learning models and hyperparameter tuning to develop valuation models for the city's apartments price. There are also a lot of discussions about interesting machine learning topics that were the biggest challenge along the way, such as handling the missing and noisy data, cleaning the inconsistent and duplicate records, trying to improve the metrics of our regression models. The statistical methods we employed in this analysis were our own value-judgment after deep exploration regarding the topic. In spite of that, we will link deeper explanations to all the methods we applied.

Key words: *regression models, predictive modeling, hyper-parameter tuning, boosting, bagging, correlation, outliers*

1 Introduction

The main aim of this project is to build a stable model capable of making as precise as possible predictions for the real price of an apartment for sale given some characteristics in the city of Skopje. In spite of this idea, the first decision we had to make was to choose the appropriate attributes that are crucial when it comes to determining the price of an apartment. The first thought that came to our mind was to explore the advertisements' content for apartments in the target city.

Anyway, we should still be aware that the advertisement price may vary depending of the deal made between seller and buyer explicitly and not remain exactly the same as the one stated. However, we can still create this model that will serve as consultative tool for how much budget would be necessary in order to buy an apartment in the city with some specific requirements the buyer would have.

There are plenty of Macedonian websites that offer a possibility to post an advertisement regarding this topic, and additionally the number of advertisements per website is pretty large. Most of the advertisements contain the same information about an apartment like the place of settlement, its surface, rooms count, floor number, whether or not there is an elevator, parking and heating in the building.

With this observation, we could conclude that these characteristics are used to describe an apartment and as such, they would be exactly the 7 attributes we could use for building our model since they contribute to the final decision of what will the apartment price be. Having this in mind, the next step was to perform web scraping in order to obtain the information from as much advertisements as we can.

1.1 Web scraping

Beautiful Soup is a Python library for getting data out of HTML, XML, and other markup languages. Beautiful Soup is helpful when the data is located on some website and it does not provide option for downloading the data itself. Beautiful Soup helps in pulling particular content from a web page, remove the HTML markup, and save the information. It is a tool for web scraping that helps with cleaning up and parsing the documents that have been pulled down from the web.

The attributes that we were concentrated on while taking the information for the apartments were number of rooms, parking and elevator available, surface, heating options, floor, price, municipality, the type of the advertisement and the date. The type of advertisement was used in order to filter the ads that were related with the apartments for renting which were later

on extracted from the data set since they were not necessary for this purpose. Also the date was just used to filter out the ads that were older than one year. This filtering was done in order to have newer information about the apartments.

1.2 Saving the data and generating profile reports

After we have successfully performed the web scraping of two websites, we saved the data structured in a table, to two different csv files named sk_stanovi.csv and sk_stanovi2.csv accordingly. Before starting with our detailed exploratory data analysis, we created a profile report for each of the newly created dataframes using the python library pandas_profiling, named profiling1.html and profiling2.html respectfully [9].

Pandas profiling is an open source Python module with which we can quickly do an exploratory data analysis with just a few lines of code. It also generates interactive reports in web format that can be presented to any person, even if they don't know programming. In short, what pandas profiling does is save us all the work of visualizing and understanding the distribution of each variable. It generates a report with all the information easily available. One of the strong points of the generated report are the warnings that appear at the beginning. It tells us the variables that contain NaN values, duplicate values, variables with many zeros, categorical variables with high cardinality, etc. These are the primary observations we will use for our data preprocessing steps. Furthermore, this report contains different correlation matrices that may help us determine the degree of correlation of our attributes and sufficiently plan our actions regarding that state.

Below we have two figures that display the warnings we received regarding our data sets. Regarding the first dataset, we can observe we have problem with a lot of missing values in all attributes, so we must be aware that in this case we need to carefully examine these data values.

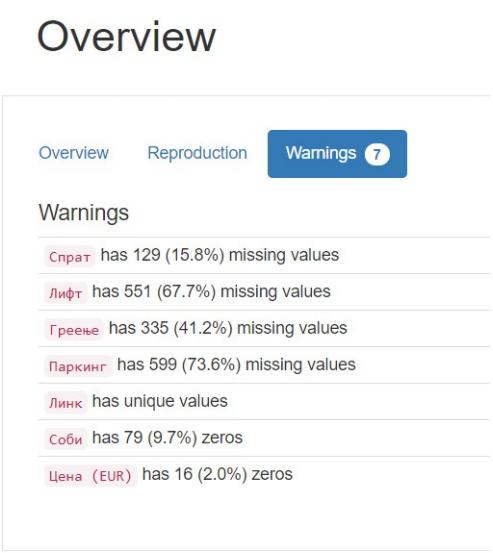

Figure 1: Warnings in first profile report

As we can see in the second figure, we have obtained more advertisement data, moreover we have significantly less missing values in this case, so the assumption is we will have to put in some more effort in cleaning the first rather than the second data set.

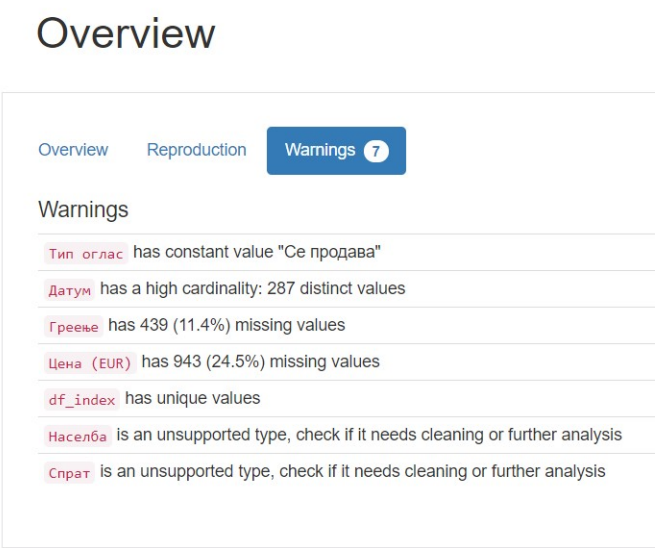

Figure 2: Warnings in second profile report

2 Data preprocessing

In this section, we will separately describe the data cleaning and preprocessing steps we applied on each data set separately, such that we will start from the first and at the same time more problematic one.

Since we were mostly curious about the distribution of the price attribute values, we first created a box-plot of these values that is displayed in the figure below.

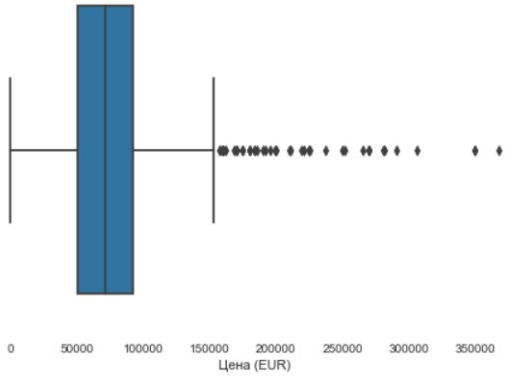

Figure 3: Box-plot of Price attribute

From this plot we can observe that most of the values lie in the range from 0 - 150 000 Euros for apartment, whereas very small portion from all 814 observations lies in the range 150 000 - 350 000 Euros. This may indicate that we may need to drop some of these values since they might be some outlier points that would introduce an undesirable bias or noise in our model further on, which is not what we want. However, we firstly proceeded to the procedure of obtaining clean data and afterwards ignoring some of the problematic data points.

Cleaning this data was one of the biggest challenges in our project, since we observed that there are a lot of invalid advertisements with a lot of missing information that the apartment owners did not enter.

	Населба	Квадратура	Соби	Спрат	Лифт	Греење	Паркинг	Цена (EUR)
0	К.Вода	35	2	NaN	NaN	NaN	NaN	30000
1	Центар	60	2	NaN	NaN	NaN	NaN	47000
2	Аеродром	61	3	NaN	NaN	NaN	NaN	1
3	Хиподром	67	3	1	NaN	1.0	1.0	47000
4	Центар	93	3	7	1.0	1.0	1.0	102000
...	...	...	...	...	...	...	...	...
809	Мацари	78	3	4	NaN	NaN	1.0	51000
810	Тафталиџе	110	4	5	1.0	1.0	1.0	137000
811	Козле	209	4	3	1.0	1.0	1.0	219000
812	Козле	90	4	2	1.0	1.0	1.0	129000
813	Козле	70	3	1	1.0	1.0	1.0	92000

814 rows × 9 columns

Figure 4: First data table

In addition, we have also included one more attribute column in this case, that is the attribute that provides the link to the advertisements that we used in order to examine an advertisement separately if necessary.

2.1 Feature Engineering (Attribute Conversion)

Before exploring the most efficient ways of filling the missing data, we need to convert all categorical attributes to numeric ones, as well as find some inconsistent string values in numeric attributes if such, and replace them with the appropriate value. The first attribute has the settlement information about the apartments, that we have found it contains 35 unique values that were replaced with numbers ranging 0 - 34. Regarding the floor number attribute, we replaced the expression for basement with the integer -1. Finally for easier manipulating purposes we replaced the missing values with the value of 404.0. Another important notice is that on this website, people only enter information about, elevators, heating and parking only if one is provided in the building, so further on we replaced the missing values for this boolean attributes with 0.

2.2 Filling missing values

Python libraries basically represent missing numbers as nan which is short for "not a number". The datasets containing nan values, however, are not consistent with Sklearn estimators which assume that all values in an array are numerical and that all have and hold meaning so that mathematical operations may be applied to them, because it will affect the strategy we will follow for missing value treatment. That is why we decided to treat missing values separately for each attribute and later join them together.

A basic strategy to use datasets containing missing values is to discard entire rows and/or columns consisting of missing values. However, this comes at the cost of losing valuable data which may be important(even though incomplete). A better strategy is to fill in the missing values is to estimate them from the part of the data which is not missing. We can also figure out which cells have missing values and what percentage of values are missing in each column using our previously made profiling report.

Missing values can be imputed with a provided constant value, or using the statistics (mean, median, or most frequent) of each column in which the missing values are located.

After some research, we have found that some options like so called multivariate imputers that estimate each feature from all the other are way better strategy for imputing missing value. This is actually done by modelling each feature with missing values as a function of other features in a round-robin fashion.

For example, a regressor is fit on (X,y) for known y. Then, the regressor is used to predict the missing values of y.

The k nearest neighbors regressor (KNNRegressor) algorithm can be used for imputing missing data using some distance metric by finding the k closest neighbors to the observation with missing data and then imputing them based on the non-missing values in the neighbors. There are several possible strategies for this. One can use 1NN, where we search the most similar neighbor and then use its value to replace the missing data. Alternatively, we can use kNN, with k neighbors and calculate the mean of the neighbors, or weighted mean, where the distances to neighbors are used as weights, so the closer the neighbour is, the more weight it has

while calculating the mean.

The target in the KNNRegressor model we used from Sklearn library for estimating missing values, is predicted by local interpolation of the targets associated of the nearest neighbors in the training set [16]. The regressor we build in our case is based on kd-tree algorithm where k=3 or in some cases k=5 neighbours, that aims to find the point in the tree that is nearest to a given input point. This search can be done efficiently by using the tree properties to quickly eliminate large portions of the search space. As a distance metric we used minkowski distance metric that represents a distance/ similarity measurement between two points in the normed vector space (N dimensional real space) and is a generalization of the Euclidean distance and the Manhattan distance [14]. The figure below displays how we applied this model in our code.

```
from sklearn.neighbors import KNeighborsRegressor

d_s1 = data_predictions
cena_stan = d_s1["Цена (EUR)"]
d_s1 = data_predictions.drop("Цена (EUR)", axis=1)
cena_stan = pd.to_numeric(cena_stan)
print(cena_stan)

neigh = KNeighborsRegressor(n_neighbors=5, algorithm ="kd_tree")
neigh.fit(d_s1, cena_stan)
#KNeighborsRegressor(...)
#print(neigh.predict([[1.5]]))
```

```
0       30000
1       47000
3       47000
4      102000
5      130000
        ...
809     51000
810    137000
811    219000
812    129000
813     92000
Name: Цена (EUR), Length: 748, dtype: int64

KNeighborsRegressor(algorithm='kd_tree', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                    weights='uniform')
```

Figure 5: Model used for predicting missing values

This procedure was applied for predicting the missing values of the Price and the Floor Number attribute. Below we have figure of the final distribution of the floor number attribute. We notice that very few of our records contain apartments for sale with floor number in the range 9-15.

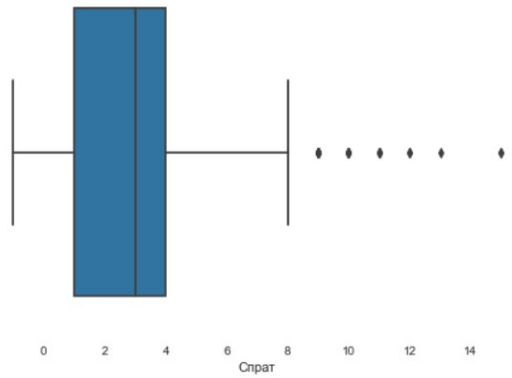

Figure 6: Distribution of attribute Floor Number

After filling as much values as our data permits, we ended up with very few data points that have missing values for multiple attributes at the same time and are displayed below. We have decided to drop these advertisements due to the poor data they contain since we cannot make predictions for such a large quantity of missing values where the information they contain itself is pretty useless.

	Населба	Квадратура	Соби	Спрат	Лифт	Греење	Паркинг	Цена (EUR)
151	NaN	57	3	3	NaN	NaN	NaN	55000
180	NaN	0	0	NaN	NaN	NaN	NaN	40000
530	NaN	60	2	NaN	NaN	NaN	NaN	15000

Figure 7: Dropped data entries

2.3 Scatter plots and final profile report

Before running the final profile report for this data set, we wanted to examine the scatter plots of the apartment surface and floor number with the price as the target attribute. From the first scatter plot, we cannot conclude that there exists some relationship between the floor number of an apartment and its price meaning we should expect that this attribute will not contribute much to the final prediction and it will be assigned a small weight coefficient.

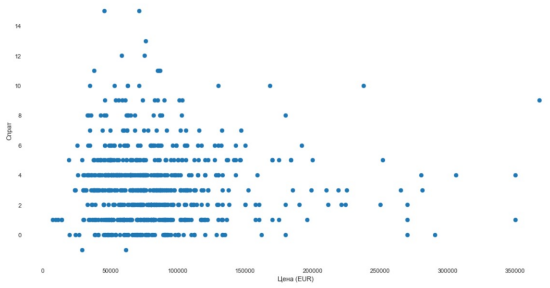

Figure 8: Price vs Floor Number

However, when we plot the scatter plot of the price and the apartment surface, we can observe higher linear correlation between these two features, which is as expected [2]. After this observation we can say that certainly the apartment surface has the greatest impact in determining the final price and as such will be assigned the largest weight when building the regression model.

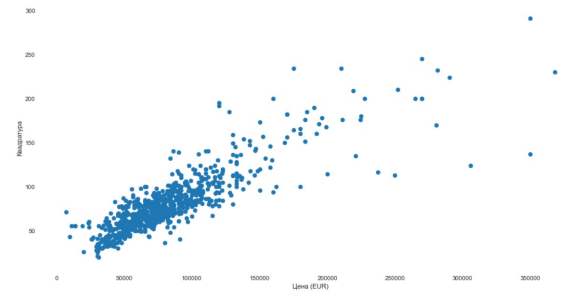

Figure 9: Price vs Surface

Finally, after all careful examinations, we have obtained the final clean data set, displayed on the figure below.

	Населба	Квадратура	Соби	Спрат	Лифт	Греење	Паркинг	Цена (EUR)
0	5.0	35.0	2.0	3	-1.0	-1.0	-1.0	30000.0
1	8.0	60.0	2.0	4	-1.0	-1.0	-1.0	47000.0
3	26.0	67.0	3.0	1	-1.0	1.0	1.0	47000.0
4	8.0	93.0	3.0	7	1.0	1.0	1.0	102000.0
5	4.0	136.0	4.0	1	-1.0	1.0	1.0	130000.0
...	...	...	...	...	...	...	...	...
611	33.0	76.0	3.0	1	1.0	1.0	-1.0	77640.0
726	6.0	74.0	2.0	3	-1.0	1.0	-1.0	81260.0
754	8.0	100.0	4.0	3	1.0	1.0	-1.0	105606.0
776	8.0	118.0	5.0	3	-1.0	-1.0	-1.0	122680.0
803	8.0	94.0	3.0	1	-1.0	1.0	-1.0	118200.0

796 rows × 8 columns

Figure 10: Final version of first dataset

This data is saved in csv file named sk_stanovi_final.csv, and the final profile report for this data set is named profiling11.html.

From this report, we can see that according to the basic statistics for the final data set given on the figure below, we do not have any missing nor duplicate values or rows in our data.

Dataset statistics

Number of variables	9
Number of observations	796
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	56.1 KiB
Average record size in memory	72.2 B

Figure 11: Data set statistics

In addition, we reviewed the distributions and also the minimal and maximal values of all attributes, to check for existence of some possible inconsistencies. Regarding the surface attribute values, the range is 20 - 291 squared meters, which are two pretty extreme values. However, after examining these kinds of advertisements, we could see that they are actually valid and there are such apartments for sale so we decided not to remove these points for now. We can also notice from the histogram of this distribution that the values for surface larger than 100 are very rare.

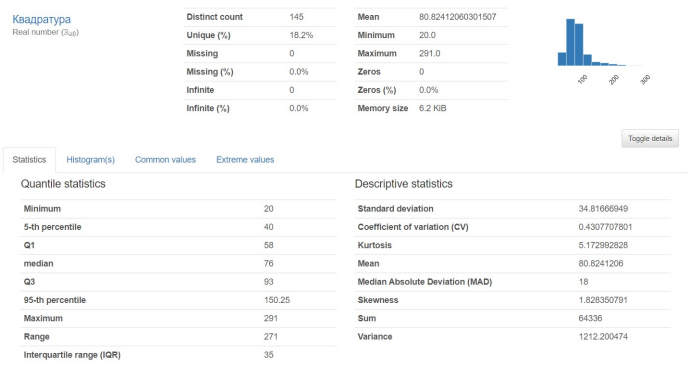

Figure 12: Attribute Surface statistics

Regarding the target attribute, that is the Price, we observe values in a large range i.e 7000 - 368000 Euros which once again are valid prices for apartments with suitable surface. However, values larger than say 150 000 Euros are very rare, as we can see from the attribute statistics on the figure below.

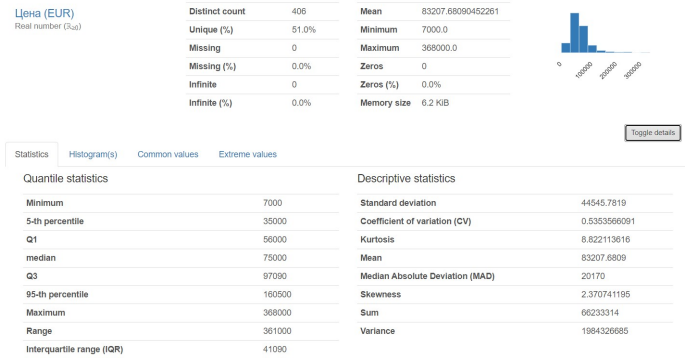

Figure 13: Attribute Price statistics

Before proceeding towards analysis of the second data set, we would like to add some comments on the Pearson's correlation Matrix values [6] for all attributes, given on the figure below.

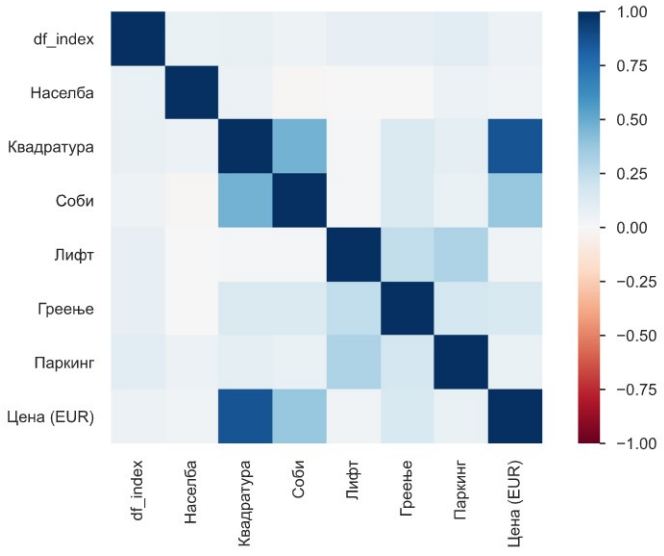

Figure 14: Attribute Price statistics

We can observe that there only exists some positive correlation between the attributes. The most obvious and highest correlation as we previously concluded is between the attributes Surface and Price that is approximately around 0.8-0.9. This would be interpreted as the larger the surface the larger the price of the apartment and vice versa. There are also some features whose linear correlation is around 0.6, that is Rooms number and Price, Rooms number and Surface which are also expected observations about this topic. In addition there exist some correlation of around 0.3 between the elevator and the parking existence in the building, meaning that their mutual existence or non-existence may be often in some cases.

2.4 Cleaning the second data set

Regarding the second data set, we have obtained much more data than in the first case since most of the advertisements from this website appeared to be in the same format. However, this website has the apartments for rent and apartments for sale in one place, so we had to add additional attribute in the table so that further on we can eliminate advertisements for rent. Also as with the first data set, we extracted the data that is not older than 1 year. Furthermore, we made the same cleanup for the Settlement attribute values, meaning we converted it to numerical attribute. We also replaced some string values for the floor number in suitable numerical ones.

During the examination of the advertisements from this website, we noticed that some of them which are in the sale category are actually for rent. This situation was a problem since we had to figure out way of how to distinguish these cases from the apartments that are really for sale. The only attribute that can make this distinction valid was the price. In spite of this observation we concluded that the rent prices are maximally 500 Euros, so we dropped these data points. Furthermore we eliminated some invalid values for the surface, price and floor number which can also be considered as missing.

During these steps of data cleaning, we used the profiling reports named profiling2.html and profiling2_full.html that describe the data set before and after cleaning the invalid entries. The first figure below displays the basic statistics before preprocessing, where we can see that we have 1462 missing cells or 3.2% overall.

Dataset statistics

Number of variables	12
Number of observations	3843
Missing cells	1462
Missing cells (%)	3.2%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	360.4 KiB
Average record size in memory	96.0 B

Figure 15: Second Data Set Statistics

After applying the specified steps for data cleaning, we obtained the following situation displayed below. The second data set is finally clean with no missing entries.

Dataset statistics

Number of variables	9
Number of observations	2508
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	176.5 KiB
Average record size in memory	72.1 B

Figure 16: Second Data Set Statistics After Cleaning

Regarding the price attribute values range, this time the values fall into the interval 2 200-521 000 Euros. Once again, these values are checked and confirmed to be valid. The median of this feature is 65 000 Euros which is a the same time the most frequent price value.

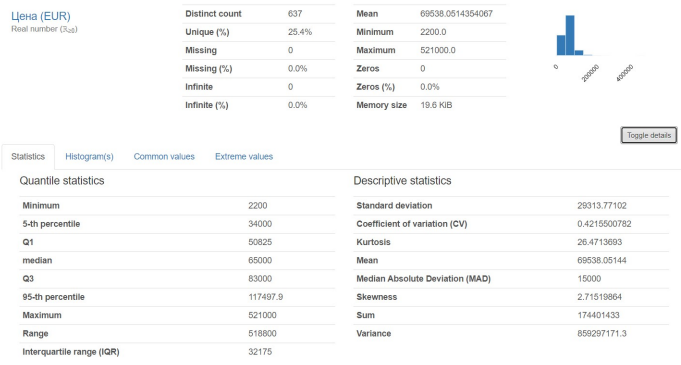

Figure 17: Second data set Price Statistics after cleaning

After examining this data's correlation matrix, this time we can only detect higher correlation between the target attribute and apartment surface. We also had this situation with the previous data set, but this time this relationship is the only one that stands out. The matrix is displayed on the figure below.

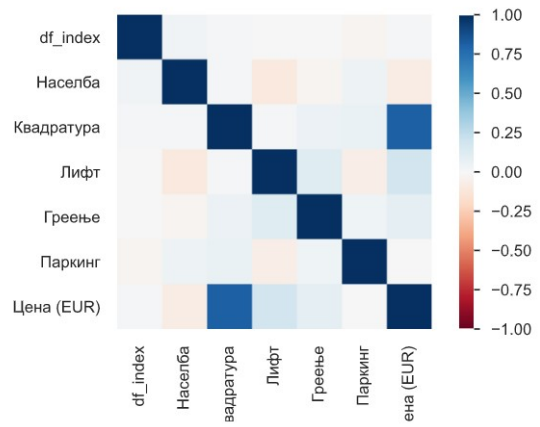

Figure 18: Correlation Matrix for second Data Set

2.5 Data Integration

The final step after cleaning the two data sets separately, is to merge these two data frames into one, which is an easy procedure since both were initially created with the same attribute order. This is done using the pandas function concatenate as displayed on the figure.

```
import pandas as pd

data_f1 = pd.read_csv(r'C:\Users\User\Desktop\faks\VI semestar\PR\sk_stanovi_final.csv')
data_f2 = pd.read_csv(r'C:\Users\User\Desktop\faks\VI semestar\PR\sk_stanovi_final2.csv')

frames = [data_f1, data_f2]

stanovi_data = pd.concat(frames)
stanovi_data
```

	Населба	Квадратура	Соби	Спрат	Лифт	Греење	Паркинг	Цена (EUR)
0	5.0	35.0	2	3.0	-1.0	-1.0	-1.0	30000.0
1	8.0	60.0	2	4.0	-1.0	-1.0	-1.0	47000.0
2	26.0	67.0	3	1.0	-1.0	1.0	1.0	47000.0
3	8.0	93.0	3	7.0	1.0	1.0	1.0	102000.0
4	4.0	136.0	4	1.0	-1.0	1.0	1.0	130000.0
...	...	...	...	...	...	...	...	...
2503	8.0	40.0	2	0.0	0.0	1.0	1.0	25000.0
2504	6.0	65.0	2	5.0	1.0	1.0	1.0	58500.0
2505	2.0	73.0	2	2.0	1.0	1.0	0.0	65000.0
2506	5.0	49.0	1	3.0	0.0	1.0	0.0	43000.0
2507	8.0	60.0	3	1.0	1.0	1.0	1.0	70000.0

3304 rows × 8 columns

Figure 19: Integrating the data

After merging the data sets, we noticed a problem with the Rooms attribute. Namely, in some advertisements, floating point values are written using ',' instead of '.', so we made this change in order to fix the inconsistency.

Finally, we generated the final profiling_stanovi_data.html report and saved the data to file named final_sk_stanovi.csv. On the figure below we can observe the between attribute correlation of this merged data. It is interesting how the correlation between some attributes got higher values than before. However, these values are around 0.5-0.6 which is are not that significant.

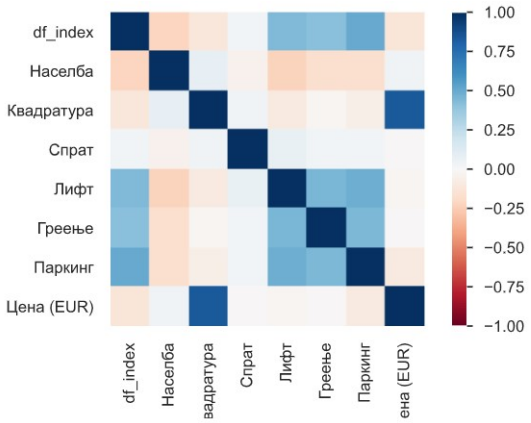

Figure 20: Correlation matrix after data integration

2.6 Anomaly Detection and Removing Outliers

Now, that we have all the data in one place, we thought that it would be a good idea to run an algorithm capable of detecting the outlier data points [4]. This method has proven to be effective and the right thing to do since we obtained better model performance in our next steps of the project. Outlier detection, also named as anomaly detection, is one of the hot issues in the field of data mining [7]. For the purpose of handling this challenge, we used the popular Isolation Forest, or iForest for short that is a tree-based anomaly detection algorithm [5].

It is based on modeling the normal data in such a way as to isolate anomalies that are both few in number and different in the feature space [12]. This proposed method takes advantage of two anomalies' quantitative properties: i) they are the minority consisting of fewer instances and ii) they have attribute-values that are very different from those of normal instances.

The scikit-learn library provides an implementation of Isolation Forest in the IsolationForest class.

Perhaps the most important hyperparameter in the model is the "contamination" argument, which is used to help estimate the number of outliers in the dataset. This is a value between 0.0 and 0.5 and by default is set to 0.1.

After running this algorithm on our 3012 data points it yielded a result of 865 outliers present. After removing these entries, we had significant decrease of the error and improvement in the regression metrics of our models even though we ended up with 2147 data items.

With this step we end our adventures regarding the data cleaning process and of course for one last time we run a profile report named profiling_noOutliers.html, where we have the final statistics of our data set. According to this report, the final attribute value ranges for the most significant attributes are the following:

- Surface: 20-170 meters sq.
- Room number: 0-5

- Price: 4086-190000 Euros

On the figure below we can also see the final state of the popular correlation matrix. We have this interesting situation at the end, that is besides the already familiar correlation between Price and Surface, this time there is also high correlation between the attributes Rooms and Surface, as well as Rooms and Price. This situation is the closest to the expected correlation scenario for our attributes and represents one of the pros of our data.

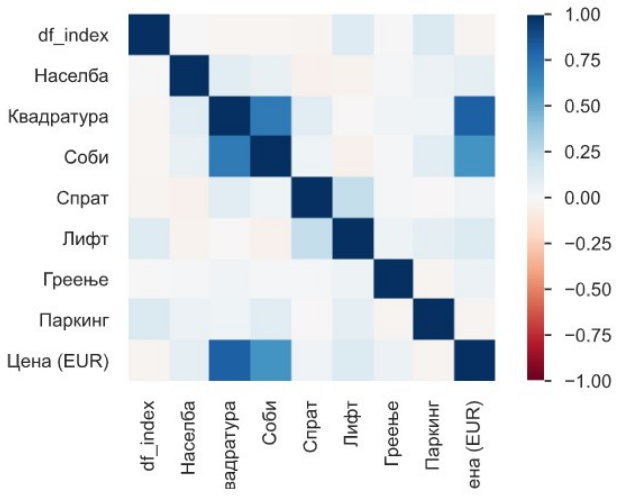

Figure 21: Final Correlation Matrix

Another good notice is that the distribution of the Price and Surface attribute values is nearly normal and that can be seen on the figure below. This can also be interpreted as another confirmation for their mutual correlation.

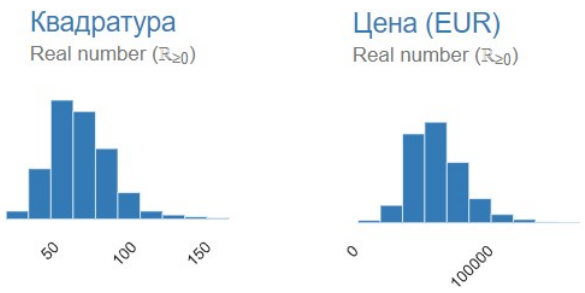

Figure 22: Distribution of Surface and Price

2.7 Feature Interactions and Importance

In order to examine all feature correlations and interactions at once, besides the correlation matrix we can also use the seaborn library and create so called pair plots for all features. The resulting plot is displayed on the figure below.

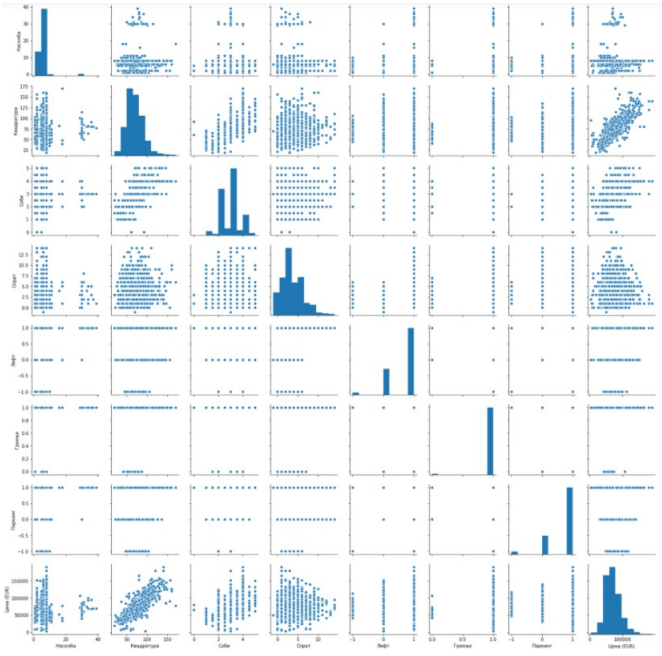

Figure 23: Feature pair plots

This plot once again confirms the higher linear correlation between the surface and price attributes. In addition we can see that the data values of these two attributes and also almost in the floor number attribute are normally distributed.

Since we saw few times earlier that we have significant linear relationship between above two attributes, we decided to make two more plots regarding this subject.

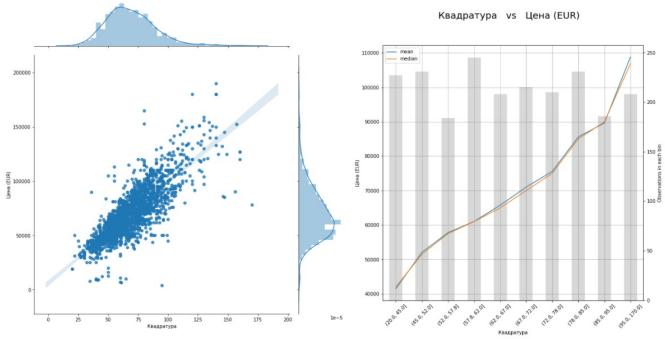

Figure 24: Surface vs Price

The first plot displays the same observations as in the overall pair plots. In the second plot, we can observe the distribution of values within each bin from the surface attribute. Surface is predictive, there is a clear pattern: on average, the larger the apartments the higher the price, even though there are some outliers.

All the above analysis we made regarding the attributes interactions were mainly conclusions based on the attributes visualizations. This time we want to try a different approach to asses the feature importance. Just like before, we can test the correlation between 2 variables. Since they are both numerical, we can use the Pearson's Correlation Coefficient: assuming that

two variables are independent (null hypothesis), it tests whether two samples have a linear relationship. If the p-value is small enough (<0.05), the null hypothesis can be rejected and we can say that the two variables are probably dependent.

This kind of analysis should be carried on for each variable in the dataset to decide what should be kept as a potential feature and what can be dropped because not predictive. The obtained p-values are displayed on the figure below.

```
import matplotlib.pyplot as plt
import seaborn as sns
import scipy

attributes = ["Населба", "Квадратура", "Соби", "Спрат", "Лифт", "Греење", "Паркинг"]
dtf = stanovi_data

for a in attributes:
    x, y = a, "Цена (EUR)"

    dtf_noNan = dtf[dtf[x].notnull()]
    coeff, p = scipy.stats.pearsonr(dtf_noNan[x], dtf_noNan[y])
    coeff, p = round(coeff, 3), round(p, 3)
    conclusion = "Significant" if p < 0.05 else "Non-Significant"
    print("Pearson Correlation:", coeff, conclusion, "(p-value: "+str(p)+") for attribute: " + a)
```

```
Pearson Correlation: 0.088 Significant (p-value: 0.0) for attribute: Населба
Pearson Correlation: 0.817 Significant (p-value: 0.0) for attribute: Квадратура
Pearson Correlation: 0.595 Significant (p-value: 0.0) for attribute: Соби
Pearson Correlation: 0.038 Non-Significant (p-value: 0.078) for attribute: Спрат
Pearson Correlation: 0.135 Significant (p-value: 0.0) for attribute: Лифт
Pearson Correlation: 0.06 Significant (p-value: 0.006) for attribute: Греење
Pearson Correlation: -0.025 Non-Significant (p-value: 0.255) for attribute: Паркинг
```

Figure 25: Features' significance

The Settlement, Surface, Rooms number, Elevator and Heating are examples of predictive features, therefore we can try keeping only them for modeling. This observation indicates that these 5 features have the greatest impact in the decision what would the final price for the apartment be. Using the Gradient Boost Regressor for example, we can create another plot describing the features' importance.

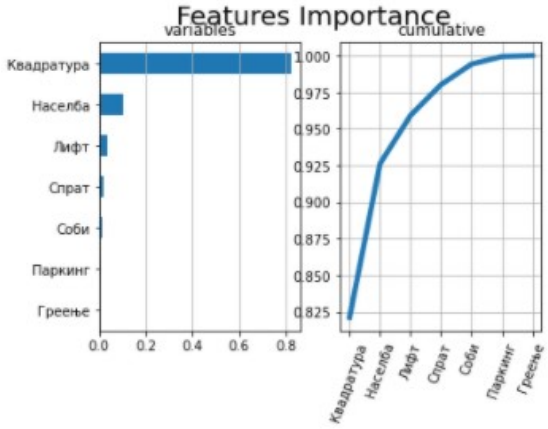

Figure 26: Features' importance

This plot confirms that Settlement, Surface, Rooms number and Elevator are the features that mostly influence the final decision. However, in contrary of the previously calculated p-values, it suggests that the Floor number attribute is much important than the heating attribute. Therefore we see that it is not quite clear what is the final impact of these two attributes. This can also be confirmed by the fact that we obtained worse model performance by removing the specified attributes. However, one thing we certainly know is that the parking existence attribute is the least important one. In that case we may try dropping this attribute. Even though we tried this, we did not obtain XGBoost model with better performance, actually it was a little bit worse than before. As a final remark, it is clear that most of the variance in the data is explained by five or at most six of all 7 attributes, but anyway excluding the remaining features yields worse model performance, so we will keep all attributes.

3 Machine Learning Models

The problem we are solving with this project actually belongs to the regression analysis domain. Regression analysis is a predictive modelling technique that analyzes the relation between the target or dependent variable and independent variable in a data set. The different types of regression analysis techniques get used when the target and independent variables show a linear or non-linear relationship between each other, and the target variable contains continuous values [10]. The regression technique gets used mainly to determine the predictor strength, forecast trend, time series, and as in our case, to predict the value of the price target variable [10].

Regression analysis is the primary technique to solve regression problems in machine learning using data modelling. It involves determining the best fit line, which is a line that passes through all the data points in such a way that distance of the line from each data point is minimized [10].

There are many types of regression analysis techniques, and the use of each method depends upon the number of factors. These factors include the type of target variable, shape of the regression line, and the number of independent variables.

In our project, we applied as much regression models as possible so that we can see how well will our data fit in different cases.

Before fitting the data into the models, we split the data set to train and test with ratio 70%-30%. Standardizing the data in most cases did not yield some specific improvement in models' performance, so sometimes we applied it and sometimes not.

As a performance metrics for our model we used the squared mean absolute error and R-squared or also called coefficient of determination. R-squared is a goodness-of-fit measure for linear regression models. This statistic indicates the percentage of the variance in the dependent variable that the independent variables explain collectively. R-squared measures the strength of the relationship between your model and the dependent variable on a convenient 0 – 100% scale. After fitting a linear regression model, it is necessary to determine how well the model fits the data. Does it do a good job of explaining changes in the dependent variable? There are a several key goodness-of-fit statistics for regression analysis.

R-squared evaluates the scatter of the data points around the fitted regression line. It is also called the coefficient of determination, or the coefficient of multi-

ple determination for multiple regression. For the same data set, higher R-squared values represent smaller differences between the observed data and the fitted values.

At first glance, R-squared seems like an easy to understand statistic that indicates how well a regression model fits a data set. However, it doesn't tell us the entire story. To get the full picture, you must consider R2 values in combination with residual plots, other statistics, and in-depth knowledge of the subject area [1].

The RMSE is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data or how close the observed data points are to the model's predicted values. Whereas R-squared is a relative measure of fit, RMSE is an absolute measure of fit. As the square root of a variance, RMSE can be interpreted as the standard deviation of the unexplained variance, and has the useful property of being in the same units as the response variable. Lower values of RMSE indicate better fit. RMSE is a good measure of how accurately the model predicts the response, and it is the most important criterion for fit if the main purpose of the model is prediction [1].

3.1 XGBoost

XGBoost stands for "Extreme Gradient Boosting". XGBoost is used for supervised learning problems, where we use the training data (with multiple features) xi to predict a target variable yi. The model choice of XGBoost are the decision tree ensembles. The tree ensemble model consists of a set of classification and regression trees (CART). Usually, a single tree is not strong enough to be used in practice. What is actually used is the ensemble model, which sums the prediction of multiple trees together. Basically random forests and boosted trees are really the same models, but the difference arises from how we train them. 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. Therefore additive training strategy is used, that is fix what we have learned, and add one new tree at a time. [11]

XGBoost is developed with both deep consideration in terms of systems optimization and principles in machine learning. The goal of this library is to push the extreme of the computation limits of machines to provide a scalable, portable and accurate library. Due to all this careful considerations, we have decided to firstly train this model to our data. [13]

For apartment prices which range from 4086 - 190 000 Euros, a RMSE of around 11507, is not such large value. Moreover, we obtained R-square value of 0.77 meaning this model successfully described around 77% of the variance in our data.

```
import xgboost as xgb
import math
from sklearn.metrics import mean_squared_error

model=xgb.XGBRegressor(random_state=1, n_estimators=5000, learning_rate=0.01)
model.fit(X_train, y_train)
print(math.sqrt(mean_squared_error(y_test, model.predict(X_test))))
print("Training set score: {:.2f}".format(model.score(X_train, y_train)))
print("Test set score: {:.2f}".format(model.score(X_test, y_test)))
```

```
11507.980327135647
Training set score: 0.96
Test set score: 0.77
```

```
from sklearn.metrics import r2_score

coefficient_of_dermination = r2_score(y_test, model.predict(X_test))
coefficient_of_dermination
```

```
0.7661505248804508
```

Figure 27: XGBoost Results

Furthermore, since we obtained pretty good results using this model, we made a plot of the true and predicted labels for the test set, displayed on the figure below.

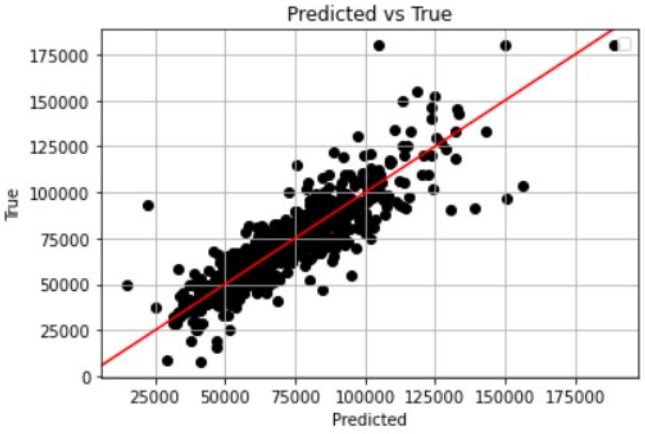

Figure 28: XGBoost - Predicted vs True values

We can see that this plot confirms the model results since most of the data of this plot is highly correlated but however there might be some larger errors at some points.

3.2 KNN and Linear Regression

We previously used KNN regression model in order to fill some missing values in our data set. This time we trained this type of model on our final data in order to examine its performance. We obtained good results, but slightly worse than using XGBoost.

K-Neighbors Regression (KNN Regression)

```
from sklearn.neighbors import KNeighborsRegressor
knn = KNeighborsRegressor(n_neighbors=4)
knn.fit(X_train, y_train)
print(math.sqrt(mean_squared_error(y_test, knn.predict(X_test))))
print("Training set score: {:.2f}".format(model.score(X_train, y_train)))
print("Test set score: {:.2f}".format(model.score(X_test, y_test)))
```

```
12460.613278168585
Training set score: 0.96
Test set score: 0.74
```

```
from sklearn.metrics import r2_score

coefficient_of_dermination = r2_score(y_test, knn.predict(X_test))
coefficient_of_dermination
```

```
0.7258318426567524
```

Figure 29: KNN Regression Model

Using the basic Linear Regression model, we obtained better values than using KNN Regression for RMSE and the coefficient of determination, whereas the training and test set score this time were little smaller.

3.3 Polynomial Regression

Polynomial Regression is a form of linear regression in which the relationship between the independent variable x and dependent variable y is modeled as an n-th degree polynomial. Polynomial regression fits a non-linear relationship between the value of x and the corresponding conditional mean of y, denoted E(y|x)

There are some relationships that a researcher will hypothesize are curvilinear. Clearly, such type of cases will include a polynomial term. If we try to fit a linear model to curved data, a scatter plot of residuals (Y axis) on the predictor (X axis) will have patches of many positive residuals in the middle. Hence in such situation it is not appropriate. Additionally, an assumption in usual multiple linear regression analysis is that all the independent variables are independent. In polynomial regression model, this assumption is not satisfied.

Implementing polynomial regression with scikit-learn is very similar to linear regression. There is only one extra step: we need to transform the array of inputs to include non-linear terms such as m. That's why the function .reshape() is used.

The results from this model regarding the RMSE and coefficient of determination are even better than the ones XGBoost yielded. The training set score is only lower in this case. After few experiments regarding the degree of the polynomial, it was established that n=2 yields best model performance.

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

# Step 2a: Provide data
x = X_train
y = y_train
x, y = np.array(x), np.array(y)

xt = X_test
yt = y_test
xt, yt = np.array(xt), np.array(yt)

# Step 2b: Transform input data
x_ = PolynomialFeatures(degree=2, include_bias=False).fit_transform(x)
x_t = PolynomialFeatures(degree=2, include_bias=False).fit_transform(xt)

# Step 3: Create a model and fit it
model = LinearRegression().fit(x_, y)

# Step 4: Get results
r_sq = model.score(x_t, yt)
intercept, coefficients = model.intercept_, model.coef_

# Step 5: Predict / Measurements
# y_pred = model.predict(x_)
rmse = math.sqrt(mean_squared_error(yt, model.predict(x_t)))

print('RMSE:', rmse)
print('coefficient of determination:', r_sq)
print("Training set score: {:.2f}".format(model.score(x_, y)))
print("Test set score: {:.2f}".format(model.score(x_t, yt)))
```

```
RMSE: 11319.27726631323
coefficient of determination: 0.7737567795487179
Training set score: 0.74
Test set score: 0.77
```

Figure 30: Polynomial Regression Results (n=2)

If we compare the True vs Predicted value plots of XGBoost and Polynomial Regression models, we can observe smaller spread in the plot of Polynomial Regression which is actually the expected scenario.

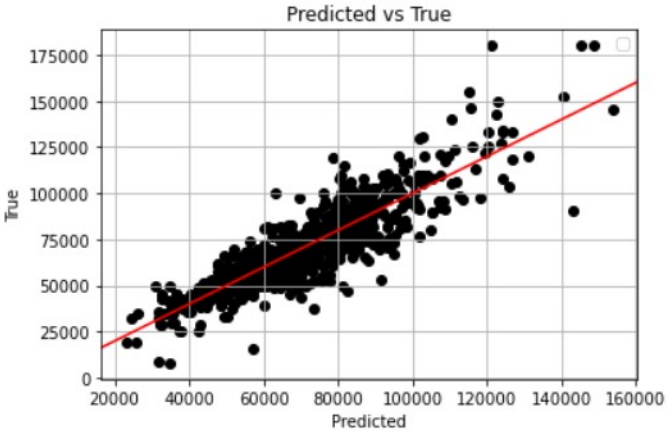

Figure 31: Polynomial Regression - Predicted vs True values

In addition, we found that the maximum error that XGBoost can make is for 17000 larger than the Polynomial Regressor's maximal error.

3.4 Logistic and Support Vector Regression

In general logistic regression model is used for classification since in its basic form uses a logistic function to model a binary dependent variable. However, outputs with more than two values are modeled by an extension called multinomial logistic regression [8].

A Support Vector Machine (SVM) performs classification by finding the hyper plane that maximizes the distance margin between the two classes. The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences.

Even though having all these adjustments in order to perform regression successfully, these two models performed poorly on our data having large RMSE and small values for the coefficient of determination.

3.5 Random Forest Regressor

Random forest is a bagging technique and not a boosting technique. The trees in random forests are run in parallel. There is no interaction between these trees while building the trees. It operates by constructing a multitude of decision trees at training time and outputting the mean prediction of the individual trees in case of regression. A random forest is a meta-estimator (i.e. it combines the result of multiple predictions) which aggregates many decision trees, with some helpful modifications and uses averaging to improve the predictive accuracy and control over-fitting. The subsample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the whole data set is used to build each tree [3]. This model performed very successfully on our data set.

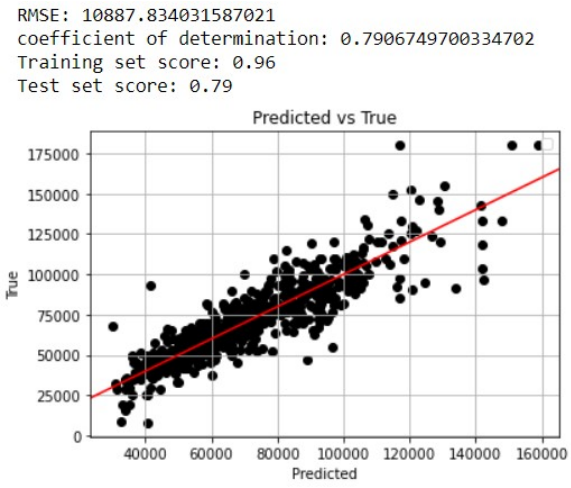

Figure 32: Random Forest Regression

In addition, we can see that besides the good RMSE and R-squared values, the True vs Predicted plot has smaller spread and errors than in the case of XGBoost or Polynomial Regression.

3.6 Ridge, Lasso and Bayesian Ridge Regression

In this section we will try to optimize our code with the help of L1 (Lasso) and L2 (Ridge) regularization.

Ridge regression is an extension for linear regression. It's basically a regularized linear regression model. The parameter is a scalar that should be learned as well, using a method called cross validation as in our case.A super important fact we need to notice about ridge regression is that it enforces the coefficients to be lower, but it does not enforce them to be zero. That is, it will not get rid of irrelevant features but rather minimize their impact on the trained model [15].

The only difference in Lasso (Least Absolute Shrinkage Selector Operator) from Ridge regression is that the regularization term is in absolute value. Lasso method overcomes the disadvantage of Ridge regression by not only punishing high values of the coefficients but actually setting them to zero if they are not relevant [15].

Bayesian linear regression is an approach to linear regression in which the statistical analysis is undertaken within the context of Bayesian inference. When the regression model has errors that have a normal distribution, and if a particular form of prior distribution is assumed, explicit results are available for the posterior probability distributions of the model's parameters. In our case, errors from this model do not quite have normal distribution, so we do not expect excellent results [17].

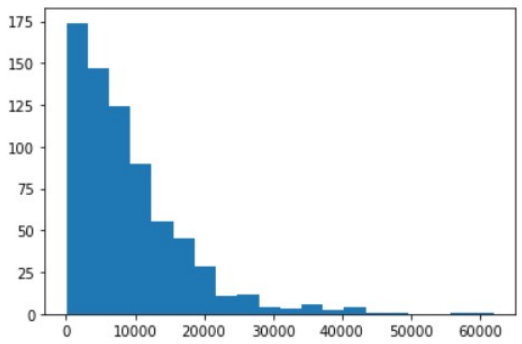

Figure 33: Errors' distribution in Bayesian Ridge Regression

Both Ridge and Lasso regression performed nearly the same when trained on our data, as well as Bayesian Ridge. RMSE value is around 12300 and R-squared around 0.73. However, better results were obtained using the RF Regressor. Actually RF Regressor performs better on its own rather than intorducing some regularization in a linear Regression model. In addition, since we also got good results using Polynomial Regression, we wanted to see if introducing regularization to this model would make some improvement but unfortunately it was not the case since nothing changed.

```
Ridge: 0.733383 (9093.451531)
RMSE: 12293.118757198294
coefficient of determination: 0.7331529954086577
Training set score: 0.69
Test set score: 0.73

Lasso: 0.733400 (9093.958712)
RMSE: 12292.727138465294
coefficient of determination: 0.7331699968910896
Training set score: 0.69
Test set score: 0.73

RMSE: 12300.187569100815
coefficient of determination: 0.732846021458625
Training set score: 0.69
Test set score: 0.73
```

Figure 34: Ridge, Lasso and Bayesian Ridge Regression

```
ElasticNet: 0.718390 (9321.391554)
RMSE: 12639.284439324425
coefficient of determination: 0.7179129475670221
Training set score: 0.67
Test set score: 0.72

CART: 0.675867 (9501.817819)
RMSE: 13549.63855395842
coefficient of determination: 0.6758144907890169
Training set score: 0.99
Test set score: 0.68

ExtraTreesRegressor: 0.760952 (8253.836836)
RMSE: 11639.146058827042
coefficient of determination: 0.7607894034144352
Training set score: 0.99
Test set score: 0.76

RMSE: 13748.431665540178
coefficient of determination: 0.6662321534467066
Training set score: 0.99
Test set score: 0.67
```

Figure 35: Models' Performance

3.7 ElasticNet, CART, DT and Extra-Trees Regressors

Elastic Net is a combination of Lasso and Ridge Regression and it also includes the ordinary least squares regression. This model is a generalization of these three algorithms that actually combines them, by folding all three terms into the cost function: the OLS cost, the ridge L2 penalty, and the lasso L1 penalty. Applying our data to this structure yields poor results anyways. It is displayed on the figure below.

CART, ExtraTrees and Decision Tree Regression implementations have good results, but not that significant as some previous models. Their performance is also displayed on the figure below.

There is an interesting discussion regarding the two ensemble methods Random Forest Regressor and Extra Trees Regressor. The two ensembles have a lot in common. The main two differences in these algorithms are the following:

Random forest uses bootstrap replicas, that is to say, it subsamples the input data with replacement, whereas Extra Trees use the whole original sample.

Another difference is the selection of cut points in order to split nodes. Random Forest chooses the optimum split while Extra Trees chooses it randomly. However, once the split points are selected, the two algorithms choose the best one between all the subset of features.

These differences motivate the reduction of both bias and variance. In our case, less randomization as in the case of RF Regressor performed much better.

3.8 AdaBoost and Bagging Regressors

An AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. As such, it helps in combining multiple "weak classifiers" into a single "strong classifier".

A Bagging regressor is an ensemble meta-estimator that fits base regressors each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction.

Bagging regressor has performed a lot better than AdaBoost Regressor, as we can see in figure 32. Its results are nearly the same as XGBoost since both of these algorithms are very similar to each other. In general, we saw that whenever bagging is included in a model it seems to yield very successful results.

```
RMSE: 13622.341884484118
coefficient of determination: 0.6723261911951545
Training set score: 0.68
Test set score: 0.67

Bag_Re: 0.771561 (8098.696338)
RMSE: 11374.08652568081
coefficient of determination: 0.7715604829618345
Training set score: 0.94
Test set score: 0.77
```

Figure 36: AdaBoost and Bagging Regressors

4 Repeated K-fold CV and Hyper-parameter tuning

Cross Validation is a technique which involves reserving a particular sample of a data set on which you do

not train the model. Afterwards the model is trained using the remaining part of the dataset. The reserved sample is left out for the test (validation) set. This validation technique is testing the effectiveness of the model's performance. If the model delivers a positive result on validation data, we go ahead with the current model. Later, we test your model on this sample before finalizing it.

For each model we trained by now, we applied cross-validation with ratio 70%-30% for train and test, but this time we choose the three best models and apply so called repeated k-fold cross validation. After this process, we'll get k*repetitions different model estimation errors (e1, e2 . . . ., ek*repetitions). The aim of this step is to choose different partitions for training and testing these models and examine the bias and variance of the errors we obtained. To return the model's bias, we take the average of all the errors. Lower the average value, better the model.

Similarly for calculating the model variance, we take standard deviation of all the errors. A low value of standard deviation suggests our model does not vary a lot with different subsets of training data. Since this condition is satisfied, we may say that our models have made successful performance.

Below we have the standard deviation of the errors, their mean and the minimal error that each model made after the repeated k-fold cross validation.

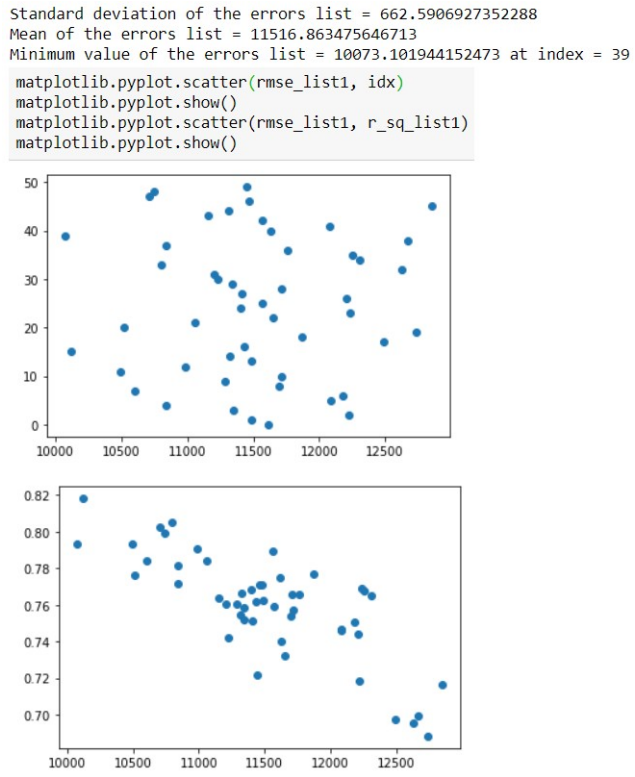

Figure 37: XGBoost Results

XGBoost's error has nearly the same standard deviation value as the best model, that is RF Regressor. The mean is for 100 lower for the RF Regressor, and the error in this model is a little bit lower than using XGBoost.

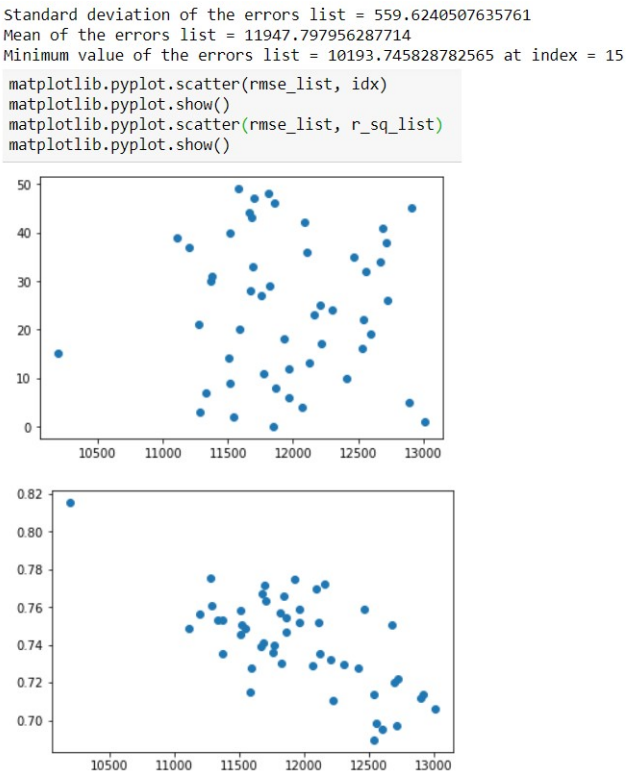

Figure 38: Polynomial Regressor Results

The polynomial regressor's error has the smallest standard deviation , but not the best overall model performance. The values for the mean are 400-500 larger than the other two models and the minimal error is larger for around 100.

```
Standard deviation of the errors list = 662.9318600343897
Mean of the errors list = 11431.598520174844
Minimum value of the errors list = 10019.307261274438 at index = 41
```

```
matplotlib.pyplot.scatter(rmse_list2, idx)
matplotlib.pyplot.show()
matplotlib.pyplot.scatter(rmse_list2, r_sq_list2)
matplotlib.pyplot.show()
```

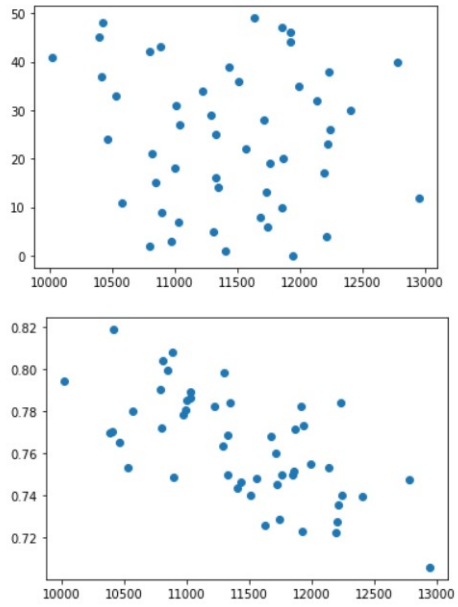

Figure 39: RF Regressor Results

Regarding the scatter plots on the above three figures, the first one displays the range of values we obtain for the errors of the given model, which are nearly the same for all three models, and smaller for the best two XGBoost and RF. The second scatter plot, describes the relationship between the different values of the coefficient of determination (y-axis) and the RMSE (x-axis), which as expected is linear. We can say that these two values are negatively correlated, meaning the higher the error, the lower the value of R-squared or in other words the poorer the model fit which is the required scenario for a successful model.

After these observations, we can conclude that our models did not perform good by coincidence, since their performance is nearly the same, with small standard deviation on different subsets of the data.

In addition, we used Grid Search in order to find the best parameters for the XGBoost model, using the train-test set split that yielded the least error in the repeated k-fold CV step. Using these parameters we trained the final XGBoost model and made one prediction. This is displayed on the figure below.

```
#Step 5: Predict / Measurements
import numpy as np
naselba = 2
kvadratura = 100
sobi = 3
sprat = 10
lift =  1
greenje = 1
parking = 1
y_pred = model.predict(np.array([[naselba, kvadratura, sobi, sprat, lift, greenje, parking], ]))
print("Предвидената цена за стан со дадените карактеристики изнесува: ", y_pred[0])
```

```
Предвидената цена за стан со дадените карактеристики изнесува:  87112.695
```

Figure 40: XGBoost Prediction

Using the best split for RF Regressor, we also trained the final model of this type. Moreover this model is officially the most successful in our project.

```
#Step 5: Predict / Measurements
import numpy as np
naselba = 2
kvadratura = 100
sobi = 3
sprat = 10
lift =  1
greenje = 1
parking = 1
y_pred = lr.predict(np.array([[naselba, kvadratura, sobi, sprat, lift, greenje, parking], ]))
print("Предвидената цена за стан со дадените карактеристики изнесува: ", y_pred[0], "EYP")
```

```
Предвидената цена за стан со дадените карактеристики изнесува:  89912.44666666666 EYP
```

Figure 41: RF Regressor Prediction

On the figure below the performances of the two best models are displayed.

XGBoost

```
10209.184595271761
Training set score: 0.95
Test set score: 0.79
 R-squared:  0.7875477449272279
```

RF Regressor

```
RMSE: 9890.834965240918
coefficient of determination: 0.7998513000879444
Training set score: 0.96
Test set score: 0.80
```

Figure 42: Best models' performance

RF Regressor has the best values for the RMSE and coefficient of determination, as well as for the train and test set score.

5 Conclusion

By performing different ML models, we aimed to get a better result or less error with max coefficient of determination and accuracy. Our purpose was to predict the price of the apartments for sale having 7 predictors and around 2200 data entries. In general, this is small amount of data since more than half of the data we collected was dropped while cleaning. Furthermore scraping may be performed on few more websites so that we obtain more entries. This process of web scraping can be automated using Airflow schedulers for example since there are also new advertisements as well as buildings in this city constantly.

Another notion is that we may introduce some type of geolocation attributes to our data set. More specifically, we can find for example the coordinates of every apartment in our data set and later compare them with a (desirable) point in the city like its center. The idea here is to get a latitude and longitude for each address in the data frame, so we can calculate the distances as precisely as possible.

In spite of all careful examinations, in the end of our project, it was concluded that RF Regressor is the best model for prediction of apartment prices.

After all we are hopeful, we have been able to shed a bit more light into how apartment prices are determined in Skopje and help a few people predict how much they should be paying for their new homes.

References

[1] Michael S Lewis-Beck and Andrew Skalaban. "The R-squared: Some straight talk". In: *Political Analysis* 2 (1990), pp. 153–171.

[2] Filip Lindskog. "Linear correlation estimation". In: *Preprint, ETH Zürich* (2000).

[3] Andy Liaw, Matthew Wiener, et al. "Classification and regression by randomForest". In: *R news* 2.3 (2002), pp. 18–22.

[4] Irad Ben-Gal. "Outlier detection". In: *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 131–146.

[5] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation forest". In: *2008 Eighth IEEE International Conference on Data Mining*. IEEE. 2008, pp. 413–422.

[6] Jacob Benesty et al. "Pearson correlation coefficient". In: *Noise reduction in speech processing*. Springer, 2009, pp. 1–4.

[7] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM computing surveys (CSUR)* 41.3 (2009), pp. 1–58.

[8] Jon Starkweather and Amanda Kay Moske. "Multinomial logistic regression". In: *Consulted page at September 10th: http://www. unt. edu/rss/class/Jon/Benchmarks/MLR_ JDS_ Aug2011. pdf* 29 (2011), pp. 2825–2830.

[9] Wes McKinney. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.", 2012.

[10] Ludwig Fahrmeir et al. "Regression models". In: *Regression*. Springer, 2013, pp. 21–72.

[11] Tianqi Chen et al. "Xgboost: extreme gradient boosting". In: *R package version 0.4-2* (2015), pp. 1–4.

[12] Wo-Ruo Chen et al. "Representative subset selection and outlier detection via isolation forest". In: *Analytical Methods* 8.39 (2016), pp. 7225–7231.

[13] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[14] Renato Cordeiro de Amorim et al. "The Minkowski central partition as a pointer to a suitable distance exponent and consensus partitioning". In: *Pattern Recognition* 67 (2017), pp. 62–72.

[15] LE Melkumova and S Ya Shatskikh. "Comparing Ridge and LASSO estimators for data analysis". In: *Procedia engineering* 201 (2017), pp. 746–755.

[16] José Ortiz-Bejar et al. "k-Nearest Neighbor Regressors Optimized by using Random Search". In: *2018 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*. IEEE. 2018, pp. 1–5.

[17] A George Assaf, Mike Tsionas, and Anastasios Tasiopoulos. "Diagnosing and correcting the effects of multicollinearity: Bayesian implications of ridge regression". In: *Tourism Management* 71 (2019), pp. 1–8.