Technische Hochschule
Brandenburg
University of
Applied Sciences
Fachbereich
Wirtschaft

Master's thesis
Price prediction of real estate using machine learning techniques in the city of Berlin

Technische Hochschule Brandenburg

Lucas Gerardo Gonzalez Sonnenberg

Advisor 1: Dr. Prof. W. Pfister
Advisor 2: M. Raddatz

29th of January 2024

Contents

List of Figures

List of Tables

List of Abbreviations

AB	Adaptative Boosting
CART	Classification and Regression Trees
CV	Cross Validation
DT	Decision Tree
LR	Linear Regression
MAD	Mean Absolute Deviation
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MSE	Mean Squared Error
PDP	Partial Dependence Plot
PFI	Permutation Feature Importance
R^2	Coefficient of Determination
RF	Random Forest
RMSE	Root Mean Squared Error
VIF	Variance Inflation Factor

Abstract

House pricing with Machine Learning is a new technique to estimate prices in a more accurate and swift way. A data frame based in the city of Berlin is constructed using web scraping. The data set is conformed with 14.869 house examples. There are thirteen different features, which intend to explain the target. Four Machine Learning models are implemented to research in this field; these are Linear Regression, Decision Tree, Random Forest, and Adaptative Boosting. Three techniques are used to weight the importance of each feature. There is no evidence of a general rule for the relative importance of each predictor, since the algorithms choose individually the preferences to prioritize each parameter for the predictive job. It is shown that Linear Regression models tend to estimate worse than other the other Machine Learning compared algorithms. Algorithms, those use bagging tend to overfit the dataframe, and those use boosting do not.

Chapter 1

Introduction

The main objective of this thesis is to estimate the transaction price of properties for the city of Berlin. Machine Learning (ML) was used for this purpose. This task involves finding a reasonable transaction price for each property.

Predicting the price of a house is a complex task, as each estimate can differ significantly from one stakeholder to another. This thesis focuses on home buyers looking for an affordable property in Berlin to maximize their living standards. For many people, purchasing a home represents the largest and most expensive transaction in their lives (Aljohani, 2021). This thesis will concentrate on this to help buyers make an intelligent purchase, supported by ML, a subset of artificial intelligence. This research seeks to increase the rationality, objectivity, simplicity, and ease of the home-buying transaction process. By supporting the observations provided by the dataset, ML will help make inferences from the patterns hidden in the collected observations (Alzubi, Nayyar and Kumar, 2018).

When home buyers make purchasing decisions, factors like the district, whether the house has a garden, or whether the property is already rented could be significant for their life standards. However, at the same time, it is hard to estimate how much value should be attributed to these factors. How much importance should be given to an extra bathroom in the house or a garage? Is it true that the year of house construction is as important as it seems? Is it true that rented houses in Berlin have lower selling prices?

House price forecasting has become necessary because the real estate market changes yearly (Thamarai and Malarvizhi, 2020). The German property market is familiar with this trend. From 2010 to 2022, the value of the property market rose steadily. However, there were signs of a potential speculative bubble bursting between 2021 and 2023, as the value of rents failed to keep pace with the rising property sales prices Kholodilin and Michelsen (2020). This prediction materialized when, in the last four months of 2022, house prices fell by 3.4%, followed by a 6.8% decline in the first four months of 2023. Among the hardest hit by these price drops were the seven largest German cities: Berlin, Hamburg, Munich, Cologne, Frankfurt am Main, Stuttgart, and Düsseldorf (Bundesamt, 2023). This example underscores the fact that property values experience fluctuations over time, emphasizing the need for real estate market price estimation.

ML is a tool for analyzing and interpreting large amounts of data to predict outcomes. It can analyze historical sales data and other relevant factors such as demographics, location, size, and amenities to predict the value of a property. In this way,

ML can simplify property exchange and help in the decision-making process for buyers and sellers (Choy and Ho, 2023). However, it is essential to clarify that in this master's thesis, the focus is only on maximizing the quality of life of home-buyers.

Several sources confirm that ML can effectively predict property prices. Sanyal, Biswas, Das, Chakraborty and Purkayastha (2022) achieved an 88,79% success rate using Lasso Regression, Mora-Garcia, Cespedes-Lopez and Perez-Sanchez (2022) achieved a 91% success rate using Extra Trees, Choy and Ho (2023) achieved a 91.92% success rate using Random Forest (RF), Crosby, Davis, Damoulas and Jarvis (2016) achieved 96.6% accuracy using the Gaussian Process Regression model, Rahman, Maimun, Razali and Ismail (2019) achieved 99% accuracy using an Artificial Neural Network model and Mora-Garcia et al. (2022) achieved 99.95% accuracy.

In this thesis, the scope will be solely on four supervised ML algorithms: Linear Regression (LR), Decision Tree (DT), RF, and Adaptative Boosting (AB). These algorithms were chosen to be compared with the results of Ragapriya, Kumar, Parthiban, Divya, Jayalakshmi and Raman (2023), who employed LR, Ridge Regression, Lasso Regression, DT, RF, Gradient Boosting Trees. They found in their study that LR achieved a better score than DT, AB, Gradient Boosting Trees, and Lasso Regression. Furthermore, the results of Ragapriya et al. (2023) contradict the main idea of Mora-Garcia et al. (2022) saying that LR outperforms the other algorithms. Therefore, in this thesis the aim is to shed light on this contradiction.

The city of Berlin was selected for this thesis because of its global significance Eckardt (2005) and because no previous studies have predicted property sales prices in this city using the ML algorithms of LR, DT, RF, and AB. Berlin's role as both a prosperous economic and political hub, coupled with its central position in Europe and increasing desirability for people to live there Berlin-Brandenburg (2023), further motivates our research.

This thesis aims to predict accurate property prices in Berlin. More than 14,000 observations from the Berlin property market were collected using web scraping. With the gathered observations, a dataset of real estate data is created. This enables the training of different ML models with various statistical methods capable of predicting the sales prices of houses. The estimated output is then compared against the actual price to determine how well a model works. The choice of a suitable model must be based on the performance of a modeling technique, the advantages and disadvantages of the model, and the objective of the case study (Mohd, Jamil, Johari, Abdullah and Masrom, 2020).

An open-source optimizer named 'Optuna' will be used to train the algorithms. It has the particularity to automatize the optimization process of the hyperparameters. For this, the user must enter a 'space search' margin and a k number of tries to train and evaluate the model. As the model is trained, the hyperparameters will be adjusted, minimizing the expected error and increasing the prediction quality (Akiba, Sano, Yanase, Ohta and Koyama, 2019).

The ambitious aim of this research work is to optimize a machine-learning model that can reasonably predict the selling price of a property. The ultimate goal of this research would be to achieve a sufficient degree of certainty in the prediction of real estate to allow home buyers to trust the sale value estimated by the model. In this case, the buyer should compare the property's actual value with the value estimated by the model. If the house is below this price, it is a signal to buy. If it is the other way around, it is a signal not to buy or to negotiate the property's value below the predicted value.

This thesis can serve as an inspiration for an application that can be highly valued.

Chapter 2

Literature Review

After a review of the literature on house price prediction, it's evident that there remains a lack of consensus regarding the most suitable ML algorithms. However, there is some agreement that ML algorithms tend to outperform traditional linear models in performance (Mora-Garcia et al., 2022). An example of this is Soltani, Heydari, Aghaei and Pettit (2022), who compared Multiple Linear Regression (65%), DT (79.7%), Gradient-Boosted Tree (89.6%) and RF (87.5%) using a dataset of 352.024 house price observations with 46 features. Their result reflects that the LR has the lowest performance compared to the algorithms above. Gokalani, Das, Ramnani, Kumar and Shah (2022) consider LR (76.93%), DT (98.01%) and RF (98.08%) side by side on a dataset of 1,000 observations. The RF algorithm performed better than the others, and the LR accomplished the lowest result. Another example of this consensus is the article from Manasa, Gupta and Narahari (2020), who compared LR (-212% for the test set[1], and 41.8% for the training set), Ridge Regression (43.58%), Lasso Regression (44.30%) and the Extreme Gradient Boosting (75.84%) using a dataset of 12.680 observations in the training set and test set. In this case, the Extreme Gradient Boosting algorithm achieved the highest performance over the other linear models.

We find an exception to this consensus in the investigation of Ragapriya et al. (2023), who investigate the following models: Modified Extreme Gradient Boosting, LR, Ridge Regression, Lasso Regression, DT, RF, AB and Gradient Boosting Trees. Their results display that LR (78%) performed better than some ML algorithms, such as AB (69%), Gradient Boosting Trees (60%), DT (74%) and Lasso Regression (76%). Overall, the best performance model was Modified Extreme Boosting (82.9%).

The choice of the right features is of great importance as they serve as the exclusive source of information for any learning algorithm applied to the relevant data (Piramuthu, 2004). However, it is worth noting that there is no consensus in the literature as to which set of predictors is relevant for estimating house sales prices. According to Choy and Ho (2023), several factors can influence house prices, including demographic shifts, real interest rates, speculation, tax incentives, construction costs, green space, and government regulations and policies. Yağmur, Kayakuş and Terzioğlu (2022) constructed their dataset using the house characteristics of location, floor area, number of rooms, age of dwelling, floor and social amenities, and presence in complex buildings. This lack of consensus is also evident in the paper by Aljohani (2021), who acknowledge the uncertainty in selecting the appropriate features to reduce the error in the performance measure. In addition, they face the challenge of deciding which features

[1] Manasa et al. (2020) explicit that their result on the test set needs further investigation.

to include in training, as including too many features can potentially lead to overfitting and subsequently increased errors on the test set. Özöğür Akyüz, Eygi Erdogan, Yıldız and Karadayı Ataş (2022) used a heat map technique to assess the correlations between the predictors and the dependent variable. In their study, the predictors with higher correlation were identified as house size, number of rooms, and number of bathrooms. They keep the predictors identified by the heat map in their machine-learning models.

In the paper by Gokalani et al. (2022), the authors adopted a feature selection strategy based on a correlation matrix. This matrix provides insights into the relationships among various parameters, with *-1* signifying the strongest negative correlation, *1* indicating the highest positive correlation, and *0* denoting no correlation at all. Notably, the highest correlation they observed was 86%, specifically between the number of bedrooms and bathrooms. For the authors, it is essential to understand how the predictor variables and the target variable correlate. By employing this approach, they were able to address fundamental questions such as whether a sizable correlation exists, and if so whether the correlation is positive or negative.

The researchers Mora-Garcia et al. (2022) note that almost all ML algorithms can calculate a measure of the relative importance of features and identify those that are more relevant for predicting the dependent variable. However, the metrics employed in these algorithms lack comparability across different statistical methods due to variations in the strategies they used during their calculations. Mora-Garcia et al. (2022) give an example of this with LR, where the importance of the features is determined by the standardized regression coefficients obtained after standardizing the original features introduced into the model. Therefore, they used Permutation Feature Importance (PFI) and Partial Dependence Plot (PDP). PFI presents three advantages. Firstly, the strategy is agnostic. This means that any algorithm can calculate the importance of a feature. Secondly, the metric obtained is easy to interpret and allows comparison of results between different algorithms. Thirdly, this strategy is computationally lighter than other strategies since it does not require more models with different combinations of features to be trained (Molnar, 2020).

Another interesting observation is the city, which the authors chose for their house price prediction studies. To carry out their studies, the authors strategically opted for cities characterized by vibrant real estate markets. Choy and Ho (2023) decided to investigate four private housing estates: these are Grand Promenade, Kornhill Garden, Les Saisons, and Taikoo Shing in the Quarry Bay, city of Hong Kong. Sanyal et al. (2022) used the Boston housing dataset provided by Kaggle of the city of Boston in the USA, and Mora-Garcia et al. (2022) selected the city of Alicante, where 18 % of the population of Spain lives.

There are two previous studies based on the city of Berlin. First, Iliev and Anand (2023) [2] construct a dataset of 12.743 observations using web scraping. They research three different types of Neural Networks: simple Neural Networks, Recurrent Neural Networks, and Convolution Neural Networks. Second, Baur, Rosenfelder and Lutz (2023), concentrate on forecasting rental apartment prices in Berlin using ML. They don't delve into the broader real estate market of the city. Even though they employed ML to estimate sales prices, they applied it exclusively to Los Angeles in the United States, not Berlin.

[2] This paper was found in Google Scholar on 23.01.2024, one week before the deadline of this thesis. It was not used for the inspiration of this research. The publication dates back to 2023/3/2.

Chapter 3

Research Questions & Hypothesis

ML is a tool developed and used in various contexts and businesses to optimize solutions. For almost every person, buying a house is the most important transaction in their lives, so it is desirable to allow buyers to use new technologies to lead to a better transaction price and thereby to a superior quality of life. For this thesis, the ML algorithms of LR, DT, RF, and AB and the feature selection methods of heat map, PFI, and PDP are used as the key elements to answer the following research questions. In this thesis, it is expected that the use of ML will help home-buyers with the tedious task of buying a house. Getting goods of the same or even better quality for a cheaper price is an optimization task where technologies can take a primordial role (Bechwati and Morrin, 2003).

Research Questions

1. Which house characteristics are essential for predicting house prices? Which conclusions can be drawn using the tools of Heat map, PFI, and PDP?
2. Mora-Garcia et al. (2022) say that there is an agreement that ML algorithms tend to outperform traditional linear models in terms of performance. Still, Ragapriya et al. (2023) demonstrate that LR achieved a better result than AB, Gradient Boosting Tree, DT, and Lasso Regression. What results are obtained, when making the same comparison using the algorithms of DT, RF, and AB in the Berlin dataset? Will LR outperform DT, RF, and AB?
3. Which of the models above best predicts the real estate market prices in Berlin? Are any of these models reliable enough (a smaller difference than 30.000 Euros between actual and predicted prices) to be trusted when buying a property?

Hypothesis

1. Area, rooms, number of bathrooms, construction year, district, and rented mainly determine the house prices (Özöğür Akyüz et al., 2022).
2. It is expected that LR algorithm estimates property prices worse than DT, RF, and AB algorithms (Mora-Garcia et al., 2022).
3. RF predicts the most accurate property prices Ragapriya et al. (2023), Gokalani et al. (2022), and Choy and Ho (2023). It is desirable, but not expected, that the

model can be sufficiently optimized to predict property values with reasonable confidence (a smaller difference than 30.000 Euros between actual and predicted prices).

Chapter 4

Methods

1. **Data collection**: The dataset is gathered using web scraping techniques from the following property sites:

   https://www.immowelt.de/
   https://www.wohnungsboerse.net/

   Nine crawls were performed from 01/07/23 to 06/09/23 to construct the Berlin Housing data frame. For this purpose, Python and the 'Scrapy' framework were implemented (Kouzis-Loukas, 2016). No observation was gathered using any technique other than web scraping.

2. **Data cleaning & Data analysis:** For this step, the Python libraries Pandas pandas development team (2020) and Numpy Harris, Millman, van der Walt, Gommers, Virtanen, Cournapeau, Wieser, Taylor, Berg, Smith, Kern, Picus, Hoyer, van Kerkwijk, Brett, Haldane, del Río, Wiebe, Peterson, Gérard-Marchant, Sheppard, Reddy, Weckesser, Abbasi, Gohlke and Oliphant (2020) were used.

   Once the data was locally available, the various crawled files were concatenated into two datasets, one from Immowelt and the other from Wohnungsboerse. This decision was made because each web source has a different HTML structure, which means that the data cleaning process of each web source is different from each other. Therefore, the cleaning steps have to be different.

   Firstly, the attributes provided by the websites were cleaned; these features are price, area, rooms, and location. Secondly, two attributes were created from the location feature: district (neighborhood) and quarter. This decision was made with the expectation that it would provide more predictive input to the algorithms rather than keeping only one. Thirdly, regular expressions Aho (1991) were applied to the collected strings to obtain new predictors such as year of construction, electricity class, and the dummy variables of balcony, bathroom, elevator, garage, garden, rented, and restoration. However, the electricity class was not implemented in the final dataset. Had this been done, the use of 8.031 examples that did not have an electricity class would have been discarded. Including the bathroom feature means that the property has a second bathroom, like a guest bathroom.

   The next step was to eliminate possible outliers that could lead to prediction errors. A threshold was set for both area and price. The area threshold is 7 square meters, and any house below this value is removed from our observations.

In addition, a maximum price of €3,000,000 and a minimum price of €70,000 have been set. The logic behind these thresholds is that the crawled websites may contain carports for parking, gardens for recreation, storage facilities, castles for purchase, or large groups of properties, distorting the algorithms' predictive capacity. The algorithms have been trained to predict the final price of a single household. By applying this filter, the outliers were removed from the dataset.

Lastly, redundant or duplicate observations were deleted. The process for deleting duplicated rows was as follows: All records with identical locations, prices, areas, rooms, years of construction, and balconies were deleted using Python and the Pandas library. After data cleaning, the two datasets were merged to obtain a complete dataset containing the final 14.869 observations from Immowelt and Wohnungsboerse.

3. **Feature Engineering:** After eliminating unrepresentative elements of households (refers to point 2), the data was transformed by what is known as **normalization** also known as **min-max-scaling or min-max-normaization** (Géron, 2019).

   This process compares results obtained using different scales, for example, in square meters for area and euros for household price or Boolean values for whether a property is rented. Normalization treats data as a vector in a multidimensional space. Normalizing the data means transforming the vector into a new vector whose norm (length) equals one (Abdi, Williams et al., 2010). Normalized data is expected to provide better-predicting results (Géron, 2019).

Min-max-scaling formula

$$X_{\text{normalized}} = \frac{(X - X_{\text{minimum}})}{(X_{\text{maximum}} - X_{\text{minimum}})}$$

Where X is an original value from a feature of the dataset, $X_{\text{normalized}}$ is that normalized value, X_{minimum} is the minimum value from the same feature, and X_{maximum} is the respective maximum value.

However, for this thesis, the final decision was not to use the normalized data because the prediction rate of the algorithms did not improve significantly. However, the normalized Table 11.5 is exposed.

Another essential step for this feature engineering phase is to analyze the correlation of the features using the Variance Inflation Factor (VIF) method. It was also implemented in the article of Mora-Garcia et al. (2022). It is a very informative technique about the correlation of the attributes. The recently mentioned author, and also O'brien (2007), say that if an independent variable achieves a score higher or equal to 10, it is a highly correlated predictor. Thus, it should be removed from the dataset. In this thesis the same threshold value for the VIF was applied.

The VIF test was successfully carried out without having the necessity of deleting any predictor. The highest score was 8,12 for the feature construction year. A higher VIF score can lead to a higher variance in the final scores.

Position	Predictor	Score
1	*cy*	8,12
2	*ar*	6,20
3	*ro*	4,99
4	*bal*	1,86
5	*re*	1,27
6	*gan*	1,37
7	*bas*	1,19
8	*el*	1,18
9	*gar*	1,22
10	*bat*	1,04
11	*di*	4,36
12	*qu*	3,05
13	*res*	1,00

Table 4.1: Variance Inflation Factor

4. **Data exploration:**

   Real Estate Dataset over the city of Berlin
   Shape of the total dataset: 14.869, represents the 100 % of the total dataset (collected and cleaned).
   Shape of the training set: 11.151, represents the 75 % of the total dataset.
   Shape of the test set: 3.718, represents the 25 % of the total dataset.

	pr	*cy*	*ar*	*ro*	*bal*	*ren*	*gan*	*bas*	*el*	*gar*	*bat*	*di*	*qu*	*res*
mean	613.393	1.958	101	3	0	0	0	0	0	0	0	6	32	0
std	462.845	47	71	3	0	0	0	0	0	0	0	3	23	0
min	90.000	1.680	7	1	0	0	0	0	0	0	0	1	1	0
25%	299.000	1.910	60	2	0	0	0	0	0	0	0	3	13	0
50%	479.000	1.959	83	3	0	0	0	0	0	0	0	5	26	0
75%	768.500	2.001	120	4	1	0	0	0	0	0	0	8	46	0
max	3,000,000	2.029	1,250	202	1	1	1	1	1	1	1	12	100	1

Table 4.2: Descriptive Statistics dataset

It is noticed in Table 4.2 that between the mean and median (50%) values for the target, there is a difference of 134.393 Euros. All properties valued under 70.000 Euros and over 3.000.000 Euros were deleted. There is also a difference from 18 square meters in the column area.

Visualizing the dataset permitted to observe the positive relationship between area and price: In Figure 11.2, subfigure a, it is possible to see that the larger the area of the property, the higher its price.

Two assertions arise by examining the figure 11.2, subfigure b. The first one is that between 1937 and 1948, there was not a big offer for houses, probably due

Pos	Feature	Data type	Description
1	*pr*	float64	Property Price
2	*cy*	int64	Construction year
3	*ar*	float64	Area of the house
4	*ro*	float64	Number of rooms in the property
5	*bal*	int64	Whether the house has a balcony
6	*ren*	int64	Whether the house is rented
7	*gan*	int64	Whether the house has a garden
8	*bas*	int64	Whether the house has a basement
9	*el*	int64	Whether the house has an elevator
10	*gar*	int64	Whether the house has a garage
11	*bat*	int64	Whether the house has an extra bathroom
12	*di*	int64	The district to which the house belongs
13	*qu*	int64	The quarter to which the house belongs
14	*res*	int64	Whether the house was restored

Table 4.3: Overview dataset

to the effect of World War II. The second relationship of interest is that the most expensive properties are concentrated between 2022 and 2025 (under construction). On the other hand, it is also interesting to mention that in 1900 there was also an agglomeration of valuable properties (probably due the golden age).

In Figure 11.2, subfigure c, it is noticeable that some districts are more expensive or cheaper than others. The cheapest one seems to be the number 10, which is Marzahn-Hellersdorf. To understand which district or quarter number belongs to each district or quarter, see Table 11.2.

Figure 11.2, subfigure d shows a tendency: the higher the number of rooms per property, the higher the sales value.

Another notable relationship to mention is that in the Figure 11.3, the values of houses being rented are effectively cheaper than those not being rented at the time of sale. The heat map technique Figure 6.1 corroborates this observation and even quantifies it. It is within the training set with -22%.

5. **Formulation of research questions:** In chapter 3, research questions and hypotheses were exposed.

6. **Definition of KPI's:** In section 5.1 various performance measures were described including MAE (Mean Absolute Error), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE).

7. **Select the Models, Optimize the Hyper-parameters, and Test the Hypothesis:** To execute the supervised ML algorithms in the Berlin data frame, the Python library scikit-learn Pedregosa, Varoquaux, Gramfort, Michel, Thirion, Grisel, Blondel, Prettenhofer, Weiss, Dubourg, Vanderplas, Passos, Cournapeau, Brucher, Perrot and Duchesnay (2011) was used. The following supervised ML algorithms were trained for the Berlin datasets: LR, DT, RF, and AB to test the hypotheses

and estimate an accurate price (less difference than 30.000 Euros between actual and predicted values) for the real estate in the city of Berlin. To search for the optimum hyperparameters, the open-source optimization framework Optuna was used (Akiba et al., 2019).

Chapter 5

Error Metrics & Resampling Methods

5.1 Error Metrics

Error Metrics measure the accuracy of a predictive model's output. They help to determine how well or poorly the model can forecast the data (Géron, 2019). Several different error measures for property value forecasting are used in the literature. Thamarai and Malarvizhi (2020) apply MSE, RMSE, and Mean Absolute Error (MAE), and Choy and Ho (2023) used Coefficient of Determination (R^2), MSE, RMSE, and MAPE. Rahman et al. (2019) use R^2, Mean Absolute Deviation (MAD), RMSE, and MAPE. The most used performance measures appear to be R^2, MSE, RMSE, MAPE, MAD, and MAE.

Error metrics are defined as logical and mathematical constructions used to quantify how close the predicted outcome is to the actual value (Géron, 2019). Predictions made by a regression model are usually not perfectly accurate, and they should not be perfectly accurate in the training set to avoid overfitting. It occurs when a model performs well on the training data but struggles to generalize effectively on unseen data (Géron, 2019). Every prediction of each model contains errors. Aggregating these errors for each model allows for comparison across models and allows to examine their overall performance and final results (Plevris, Solorzano, Bakas and Ben Seghier, 2022). The general rule is that a model fits the test data well if the differences between observed values and predicted values are minor and unbiased (James, Witten, Hastie, Tibshirani et al., 2013).

Coefficient of Determination

$$R^2\ (y, \hat{y}) := 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Where y is the observed value and $\hat{y}$ is the estimated value, n is the number of distinct data points or observations, and where i is the number of observations, so $i = 1, 2, \ldots n$. R^2 is the proportion of the variation in the dependent variable that can be described from the independent variable. A value of 1 for R^2 says that the regression equation fitted describes the entire variability of the dependent variable values in the sample data; a value of 0 indicates that the regression equation accounts for none of the variability (Hahn, 1973). The highest possible achievable score is 1. The algorithm whose resulting score is higher indicates a better predicting performance.

Mean Squared Error

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

According to James et al. (2013), MSE is the most widely used forecasting model for predicting future events, where the error will be smaller as long as the predictions output $\hat{y}_i$ are similar to the actual values y_i. Within MSE, the result is expressed in squared values and not in the unit of measurement used to calculate the sale price of a property (in this thesis, Euros). The best possible reachable score using MSE is 0.0. A small error value indicates higher predicting performance; thus, the error rate is lower.

Root Mean Squared Error

$$\textbf{RMSE}\ (y, \hat{y}) := \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}}$$

Mathematically, it is the standard deviation of the residuals, where residuals represent the distance between the regression line and the data points. RMSE quantifies how dispersed these residuals are, revealing how tightly the observed data clusters around the predicted values (Barnston, 1992). An advantage of the RMSE over the MSE is that the result is expressed in the unit of measurement used to calculate the sale price of a property (in this thesis, Euros). The best possible RMSE score for a model output is close to 0,0. It occurs when the output value ($\hat{y}_i$) is equal to the actual value (y_i).

Mean Absolute Percentage Error

$$\textbf{MAPE}\ (y, \hat{y}) := \frac{100\%}{n} \sum_{i=1}^{n} \frac{y_i - \hat{y}_i}{y_i}$$

The advantages of MAPE are scale independence and interpretability, expressed in percentage. These mentioned advantages make it easier for the user to understand the result. On the other hand, it will produce an undefined score if the summation of the actual values (y_n) gives 0 (Kim and Kim, 2016). This is not the case for real estate; therefore, property prices always have a positive value ($y > 0$). Desirable outcomes are achieved when the MAPE score is less than 20%.

Mean Absolute Error

$$\textbf{MAE}\ (y, \hat{y}) := \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}$$

MAE is a linear error metric, meaning larger errors are not penalized more than smaller ones, which is a significant advantage over MSE and RMSE. The MAE score is measured as the average of the absolute error values. Therefore, the difference between an actual (y_i) and a predicted value ($\hat{y}_i$) can be either positive or negative and will necessarily be positive when calculating the MAE (Schneider and Xhafa, 2022). The optimal attainable result is reflected in a score of 0.0.

For this thesis, R^2, MSE, RMSE, and MAE will be used. R^2 will serve as a universal reference point due to its widespread usage across various sources; MSE to follow up

the most used forecasting model as indicated by James et al. (2013), RMSE, because it is the error expressed in the unit of measure of the price of houses in Berlin. RMSE will then serve as a reference, in euros, of how well or poorly the model is fitting the training data and predicting the selling price, and finally, MAE, because it does not penalize large errors over small ones. A similar performance logic was already implemented by Mora-Garcia et al. (2022) and Aljohani (2021).

5.2 Resampling Methods

Resampling methods arise from theoretical distributions; they involve iteratively drawing samples from a training set and training a model on each sample. This process aims to gain extra insights into the fitted model to reduce the error rate (Yu, 2002). For the purpose of this thesis Cross Validation (CV) and Bootstrap are briefly explained.

Cross Validation

CV is a resampling technique used to assess the adequacy of a model. CV aims to segregate the dataset into a training and a test set to avoid testing a model with the same observations where it was previously trained (Chernick, 2012). CV is also used to find the appropriate degree of smoothing to determine how much to prune a regression tree on the Classification and Regression Trees (CART) algorithm, among others (Chernick, 2012).

The Bootstrap

Bootstrap is a statistical method that uses computational power to gain insights from the uncertainty within a wide range of AI models, including some for which a measure of variability is otherwise costly and time-expensive to obtain (Hastie, Tibshirani, Friedman and Friedman, 2009). Due to computational power it is possible to generate new samples from the original population sample, without getting new observations. Where the training dataset is represented by $Z = (z_1, z_2, \ldots, z_N)$, with $z_i = (x_i, y_i)$. Therefore, selecting random observations of the training dataset is possible to create B Bootstrap Samples, where $Z^1, Z^2, \ldots, Z^B$. The Bootstrap sample is conducted with replacement, allowing for the same observation to appear multiple times in the Bootstrap dataset. The bootstrap method is applied when calculating the standard deviation of a particular quantity is challenging or not feasible through direct means. (James et al., 2013).

Chapter 6

Feature Selection

The literature lacks consensus regarding the optimal group of attributes for training a ML algorithm in the real estate domain. The objective of this chapter, feature selection, is to strike a balance between selecting too many or too few predictors for the model. If too few features are chosen, the information within that selection will likely be limited. Conversely, if too many irrelevant features are included, the effects of the noise often found in real-world data may overshadow the valuable information (Aljohani, 2021). Consequently, this trade-off is a fundamental consideration for any feature selection technique (Piramuthu, 2004). It is referred to as the curse of dimensionality (Köppen, 2000).

Three feature selection methods will be used to understand the correlation between the dependent and independent variables. However, it is essential to mention that hundreds of feature selection algorithms were already proposed Li, Cheng, Wang, Morstatter, Trevino, Tang and Liu (2017); despite of this, for the aim of this thesis, the focus is on three of these methods. The first one is **Heat map**, the second one is **PFI**, and the third one is **PDP**. In this context, it is necessary to clarify that correlation does not necessarily imply causation. Nevertheless, correlation frequently serves as a suggestive indicator of causation (Barrowman, 2014). For this reason, the methods to be presented should be used with caution, and one should not fall into the trap of allowing correlation observations to imply causality without first uncovering all the hidden variables (Pearl et al., 2000).

6.1 Heat Map

Heat map is a widely used approach to understand which predictors correlate strongly with the dependent attribute. This procedure was implemented by Özöğür Akyüz et al. (2022), Choy and Ho (2023), Aljohani (2021), Sanyal et al. (2022), Iliev and Anand (2023), among others. Heat maps visually represent two-dimensional numerical data, employing a spectrum of colors to convey correlation. They excel at handling large datasets. They can accommodate numerous rows and columns within a single screen. Heat maps are an invaluable tool for facilitating intuitive data interpretation. They achieve this through the strategic use of color-coding and deliberate reordering of rows and columns, facilitating the discovery of underlying patterns and insights within the data, making them particularly valuable for research purposes (Gehlenborg and Wong, 2012). It is a simple and practical tool for visualizing data and correlations, and for this reason, it is used by many researchers related to real estate

using ML. The main limitation of this technique is that it does not work on real-time data.

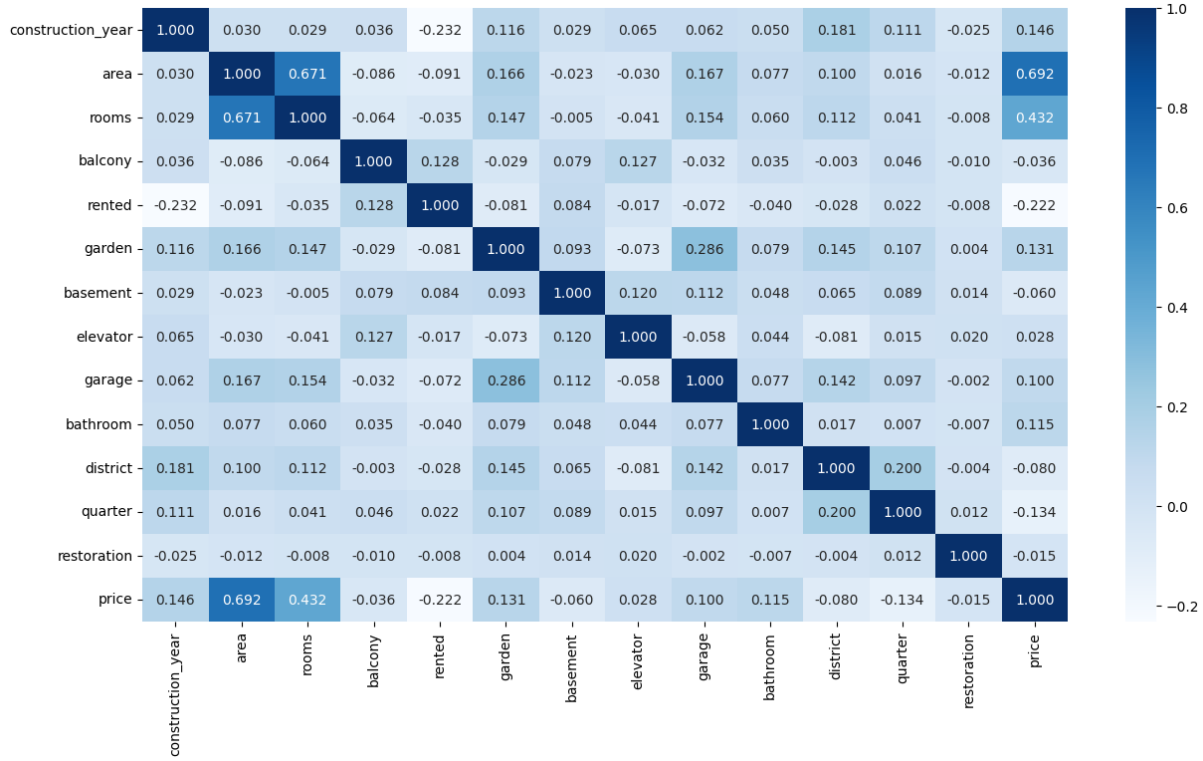


Figure 6.1: Heat map on real estate training set Berlin

For the Heat map, all 13 independent variables and one dependent variable from the training dataset were used. The independent variables are area, balcony, bathroom, construction year, district, elevator, garage, garden, quarter, rented, restoration, and rooms, while the dependent variable is the real estate price. Thanks to the Heat map, it is possible to corroborate that the most correlated predictors are area, with a 69% correlation, and rooms, with a 43% correlated price. The Heat map shows that if a house is rented, its value is expected to be 22% less than if it is not rented. The presence of a garden can lead to a 13% increase in the property value, an extra bathroom to 11%, and a garage by 10%. Contrary to expectations, the presence of a balcony is indicative of a 3% less value of the housing price.

6.2 Permutation Feature Importance

The second approach was used by Mora-Garcia et al. (2022), Khosravi, Arif, Ghasem-inejad, Tohidi and Shabanian (2022), and Chen, Liu, Arribas-Bel and Singleton (2022). This strategy is called PFI. PFI is defined as a model's performance decrease when a single feature is randomly shuffled. Therefore, the performance results of the models were compared using two different datasets, one original dataset and another identical dataset but with this specific feature randomly shuffled. The difference between the error in the performance of both datasets shows the importance of that particular feature associated with a specific model. If the increment in the error is not pronounced, the model does not depend on this predictor. In this case, that feature can be left outside the model to avoid overfitting (Molnar, 2020). A threshold is set to define whether

the increase in the error is pronounced. If the error exceeds the selected threshold, the feature should be included in the algorithm's training data. The acceptance of the chosen threshold depends on the user's subjective perception, and there are no objective parameters for determining it and maximizing the utility of the attributes. The advantages of this approach are the following: This strategy is agnostic, meaning it can be used with any algorithm, allowing us to compare the results of any model. Those, it is useful for comparison and for making observations and conclusions. The metric obtained is easy to interpret and allows for comparing results between different algorithms. It is much more computationally lightweight than other strategies (Mora-Garcia et al., 2022). By eliminating certain features, it is possible to achieve a dual benefit: enhancing the model's performance and interpretability. Using these methods can help dispel the perception of ML algorithms as inscrutable 'black boxes' and facilitate the discovery of new insights from the ML algorithms (Mooney, 2019). On the other side, there are some disadvantages: the PFI is related to the model's error and not related to how robust the model's output is. PFI depends on random noise; when the permutation is repeated, the results might vary (Molnar, 2020). Losing objectivity in this way.

The PFI was measured using the R^2 Score, which allows for identifying the most predictive features. It is observable that the area is by far the most relevant feature, followed by the quarter, the year of construction, the district, and the rooms. The area's importance hardly changes if the ML algorithm varies. The quarter attribute retains its second place for each model but is far from the attribute area. The year of construction obtains a third place for all the models, except for the LR, where it obtains the fourth place and the district obtains the third place. Figure 6.2 summarizes the result of LR, DT, RF, and AB using PFI for each feature on the R^2.

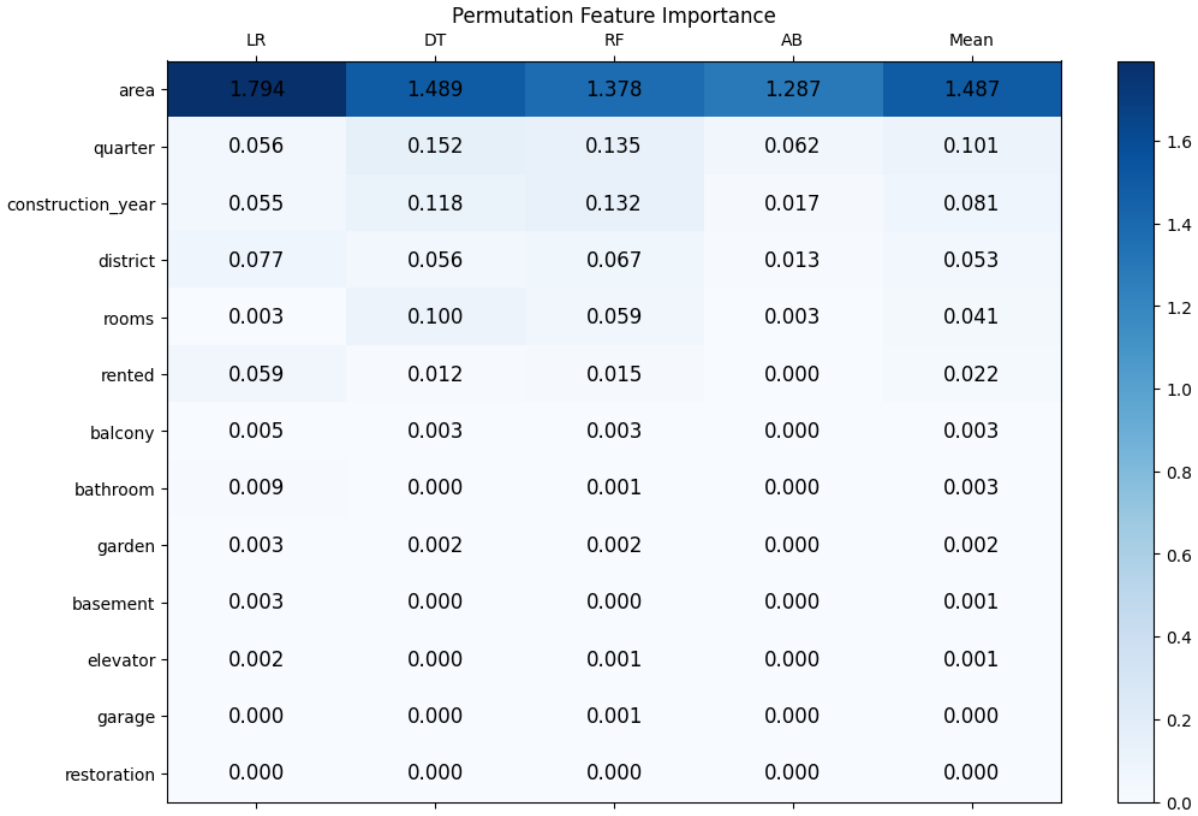
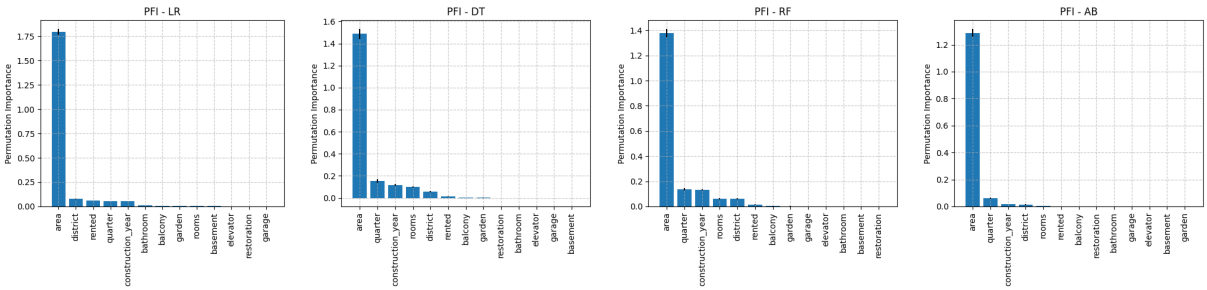


Figure 6.2: Overview Scores Permutation Feature Importance on the Berlin test set.



(a) LR (b) DT (c) RF (d) AB

Figure 6.3: Permutation Feature Importance for Linear Regression, Decision Tree, Random Forest, and Adaptative Boosting models.

6.3 Partial Dependence Plot

Partial Dependence Plots is a method to visualize the impact of one or two features on a model's dependent variable. They enable us to gain insights into the relationship between a feature and the dependent variable, shedding light on whether this relationship is linear or exhibits more intricate patterns (Friedman, 2001). Mora-Garcia et al. (2022) and Kagie and Wezel (2007) used this feature selection strategy.

Among the advantages, it is worth noting that the resulting plot is intuitive and provides a clear interpretation of how the average prediction changes when the independent feature is changed. The calculation for the PDP may have a causal interpretation for that specific model but not necessarily for the real world. This model does have some limitations. It only allows comparing a maximum number of two features

at a time. Some PDPs do not show the feature distribution, though this limitation is easily solved by plotting and displaying a histogram of the dataset. It is crucial to highlight that the accuracy of PDP interpretation hinges on the feature independence assumption. When features are correlated, the interpretation of PDPs may be biased by unrealistic data and misleading conclusions (Molnar, 2020). For example, it can be considered a property dataset that includes the number of rooms, square footage, and price. The objective is to construct a PDP for the number of rooms to identify its effect on house prices. The independence assumption inherent in the PDP means that in constructing such a plot for the number of rooms, it is assumed that there is no correlation with square footage. However, empirical evidence often shows a correlation between these variables, with larger houses having more rooms and smaller houses generally having fewer rooms. Consequently, constructing a PDP for the number of rooms under the assumption of independence could erroneously suggest a positive relationship between the number of bedrooms and house prices, neglecting the confounding influence of house size (area). In summary, relying on the independence assumption within the PDP can lead to misinterpretation in cases where there are correlations between characteristics, such as in the housing context where the number of bedrooms and square footage are often related.

Among the observations due to the charts on section 11.5, it can be said that the characteristics depending on dummy variables do not provide as rich observations as those where the results are more variable, such as the predictors of area, year of construction, quarter, rooms, or district. It can be mentioned that the correlations obtained by LR have allways a linear correlation. LR is an excessively simple model, very different from DT, RF, and AB. So PDP allows us to observe the flexibility of each statistical model.

Chapter 7

Statistical Learning Methods

In this chapter, ML algorithms of LR, DT, RF, and AB are introduced, trained, and tested. To implement the algorithms, the Python library scikit-learn Pedregosa et al. (2011) is used to build the supervised models. The Python framework Optuna Akiba et al. (2019) is used to optimize the mentioned algorithms. The Matplotlib Hunter (2007) and Seaborn Waskom (2021) libraries are implemented to create and display the scatterplots presented in this thesis.

7.1 Linear Regression

The LR is a primary method used to model the linear relationship between dependent (y) and independent (x) variables. In LR, the relationships are represented through linear predictor functions, and the model's unknown parameters (β) are estimated from the dataset **X**. Every dataset with more than one feature is a matrix of row-vectors, where **n** = represents the total of **observations in the statistical sample**, with ($\boldsymbol{i = 1, 2, \ldots, n}$), and **p = represents the variables, features, or predictors** (such as area, year of construction, rooms, garden, among others.), and where ($\boldsymbol{j = 1, 2, \ldots, p}$). At least $\boldsymbol{y}$ is a vector, composed of the observations of the target variable, where $\boldsymbol{y_i}$ $\boldsymbol{(i = 1, 2, \ldots, n)}$ (Yan and Su, 2009).

$$\mathbf{X} := \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}$$

$$\mathbf{y} := \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

$$y_i := \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} + \epsilon_i$$

Where β_0 is **the intercept** (James et al., 2013) also named **the bias** (Hastie et al., 2009). This is the expected value of y when $X = 0$; β_1 is the slope of the LR. **The slope** is the average increase in y for a one-unit increment in X, and the Epsilon (ϵ) acts for

the random error independent of X (James et al., 2013). The acLR model assumes a linear relationship between the predictors (X) and the target (y), where the task is to estimate the coefficients of β_0 and β_1 such that the error term ϵ is minimized, and the LR the training data as accurately as possible (Yan and Su, 2009). There exist many different ways to fit the LR, such as gradient descent (Géron, 2019), and least squares (Hastie et al., 2009) & (James et al., 2013).

Based on the literature, it is not expected to find the target function of the house price prediction using this model; thus, LR has only 2 degrees of freedom. This algorithm does not have the necessary flexibility to meet the target function; hence, the resulting price prediction $\hat{Y}$ is expected to be biased. However, using LR as a benchmark input to compare the results obtained with the other proposed ML algorithms is interesting to shed more light on the research question: Whether a LR can estimate the target value better than DT, RF, or AB.

7.2 Decision Trees

A DT is a supervised learning algorithm. It comprises a hierarchical upside-down tree structure, consisting of a root node on the top, branches or internal nodes, and leaf nodes or leaves. Arrows are pointing to branches, and arrows pointing away from them. Leaves have arrows pointing to them, but no arrows are pointing away. DT represents data as a tree where each internal node denotes a condition, each branch is regarded as an outcome of the test, and each leaf node keeps a class label for classification trees or a continuous value for regression trees.

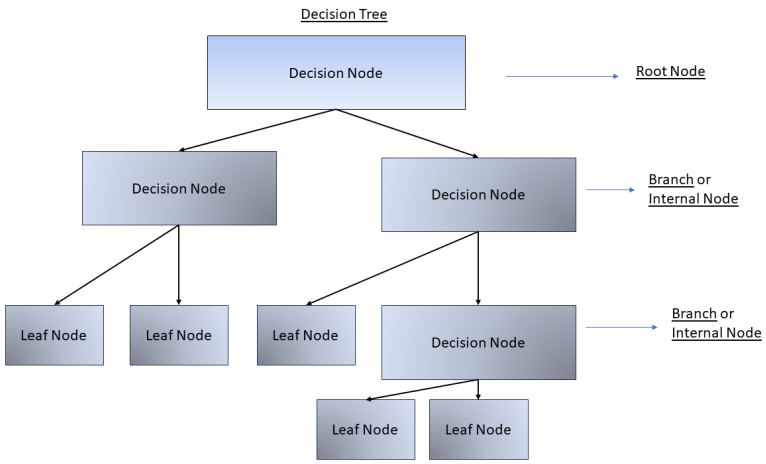


Figure 7.1: Decision Tree Structure

Splitting the data: The way to split the observations with regression and classification trees is different. This thesis focuses on regression trees. Thus, the objective is to predict the house price for real estate in Berlin, which is a continuous value.

DT splits recursively the feature space into binary rectangles, based on a split point (s) and a predictor X_j, which results in the most considerable possible reduction on the residual sum of squares. The observations that accomplish that condition are fitted into each feature space rectangle. For fitting the observations on each rectangle, the output value is the mean of y (Hastie et al., 2009). This procedure is a greedy process;

the algorithm selects the best possible combination of (X_j, s) at each split without planning for a future step where a better tree might be constructed.

For this purpose, the CART algorithm from the framework scikit-learn Pedregosa et al. (2011) has been used. It allows the recursively split of the feature space in a binary way (Hastie et al., 2009).

The sample dataset consists of a matrix X and a target function y, with n observations, where $i = 1, 2, \ldots, n$. X is composed of p variables, so $j = 1, 2, \ldots, p$. Examples of these variables are area and construction year, among others. The input space, also known in the literature as predictor space or feature space, is composed of M regions, where $R_1, R_2, \ldots, R_M$. There are no overlapping regions. A different constant value c_m (Hastie et al., 2009) belongs to each region. The element | represents an indicator function, where if the condition (s) is reached, the output value is one; otherwise, it is zero.

$$f(x) := \sum_{m=1}^{M} c_m I(x \in R_m).$$

The complete dataset sits at the top of the tree. The Observations that satisfy the condition (s) at each junction are assigned to the left branch and the others to the right. The splitting variable is j, the split point or threshold is s, and the definition of the pair half-planes is as follows.

$$R_1(j, s) = \{X | X_j \leq s\}$$
$$R_2(j, s) = \{X | X_j \leq s\}.$$

The algorithm seeks the splitting variable j and split point s that achieves the most significant minimization of the residual sum of squares at each step:

$$min_{j,s}[min_{c1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2] + [min_{c2} \sum_{x_i \in R_2(j,s)} (y_i - c_1)^2].$$

For any choice j and s, the inner minimization is solved by

$$\hat{c}_1 = \text{avg}(y_i | x_i \in R_1(j, s))$$
$$\hat{c}_2 = \text{avg}(y_i | x_i \in R_2(j, s)).$$

Where $\hat{c}_1$ is the mean response for the training observations in $R_1(j, s)$ and $\hat{c}_2$ is the mean response for the training observations in $R_2(j, s)$. This process is repeated until a condition is reached or until all the observations are grouped (James et al., 2013).

Pruning: The regression tree is a non-parametric model because the number of parameters is not determined before the training process. Therefore, the model structure has the freedom to closely adhere to the data (Géron, 2019). It is for this reason that if the algorithm remains unconstrained, the training data is going to be overfitted. The proposed alternative in DT to avoid the overfitting phenomenon is to apply regularization in the model, also called pruning the tree. The framework scikit-learn allows the possibility to adjust attributes to apply regularization; for example, the attribute max_depth allows restricting the maximum depth of the Regression Tree. Min_samples_leaf restricts the minimum number of samples a leave must have (Géron, 2019). There are other hyperparameters for the regularization, such as min_samples_split, which is the minimal number of samples to split a node, and

max_features, which is the number of features to consider for the split (Pedregosa et al., 2011).

7.3 Bagging, and Random Forest

The RF is a statistical model that departs from a technique called **Ensemble Methods**. Ensemble Methods combines weak learner models to work together on the same predictive task. It results in a better predictive algorithm with a more accurate predictive capacity than each weak learner by itself (James et al., 2013). A weak learning algorithm is solely an algorithm that predicts better than random guessing (Schapire et al., 1999). For this thesis, regression trees were used as weak learners.

RF is based also on the ML algorithm **Bootstrap aggregation** also referred as **Bagging**. Bagging is a procedure for reducing the variance of a statistical learning method (James et al., 2013). This affirmation is based on the premise that averaging a series of observations ($\overline{Z}$) reduces the variance (Breiman, 1996). Where the training data is represented as $Z = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, and where each observation has its own variance (θ^2). Therefore, a natural way to reduce the variance and increase the test set accuracy of a statistical learning method is to take many training sets from the population, build a separate prediction model using each training set, and average the resulting predictions. However, this procedure is not feasible because, under normal circumstances, it is impossible to access multiple training sets (James et al., 2013). In its place, it is possible to use the resampling method The Bootstrap explained in chapter 5.1 to produce B different Bootstrap datasets, $Z^{*1}, Z^{*2}, \ldots, Z^{*B}$.

For each Bootstrapped dataset (B), a regression tree is fitted to perform over it, given by $\hat{f}^1(x), \hat{f}^2(x), \ldots, \hat{f}^B(x)$. Lastly, all resulting predictions are averaged to reduce the variance from the estimated prediction function and, therefore, to improve the quality of fit (Breiman, 1996).

$$\hat{f}\text{bag}(x) := \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)$$

Applying bagging to regression trees involves generating B regression trees trained in bootstrapped training sets. The outputs from these models are then averaged; the deep growth of these trees without pruning results in each model having high variance but low bias. The average of the models reduces the variance. Combining hundreds or even thousands of trees into a unified procedure, as seen in bagging, has significantly improved predictive accuracy at the cost of model interpretability (James et al., 2013).

The difference between RF and bagging is that by RF, the trees are built on a subset **(m)** of the predictors **(p)** from the dataset. At each split in building a tree, only m predictors are considered as the split candidates (James et al., 2013). This modification triggers a **decorrelation** among the weak learners, leading to a significant reduction in the variance due to bagging. If an RF algorithm is built using m = p, this amount equals bagging (Hastie et al., 2009), where T_b refers to the random forest tree to the bootstrapped data.

$$\hat{f}_{rf}^{B}(x) := \frac{1}{B}\sum_{b=1}^{B} T_b(x)$$

7.4 Boosting, and Adaptative Boosting

Boosting, as Bagging, is a general approach that can be applied to various statistical algorithms for regression or classification tasks (James et al., 2013). It arises from discovering numerous weak hypotheses or "rule-of-thumb" and combining them, which can be easier than identifying a highly accurate rule. Although such a weak hypothesis is rough and inaccurate, combining a bunch of weak hypotheses leads to a very accurate prediction rule (Schapire, 2003).

The booster is provided with a set of labeled training examples $(x_1, y_1), \ldots, (X_N, Y_N)$, where y_i is the label associated with instance x_i. Where x_i is the observable data associated with a particular random fact, for example, a horse race, and y_i is the outcome of that random fact, for example, the winner of that race. On each period $t = 1, 2, \ldots, T$ the booster devises a distribution D_t over the set of examples and request from an unspecified weak learner (weak hypothesis or "rule-of-thumb") h_t with low error ϵ_t concerning D_t, which is $\epsilon_t = Pr_{i \sim D_t}[h_t(x_i \neq y_i)]$. Thus, distribution D_t specifies the relative importance of each example for the current period (t). After T periods (horse races), the booster must combine the weak hypothesis into a single prediction rule. Thus, this bound shows that if it is possible to get consistently a weak hypothesis that is moderately better than random guessing, then the error of the final hypothesis will drop. So it is noticed that the final model accuracy improves when any of the weak hypotheses are improved (Freund and Schapire, 1997).

The Boosting Algorithm **AdaBoost**, also known as **Adaptative Boosting** (β) requires as input a training set composed of a sequence labeled examples $(x_1, y_1), \ldots, (X_N, Y_N)$, with a distribution ($D$) over the examples ($N$). The algorithm uses Boosting to find a consistent hypothesis h_f with most observations in the training dataset. On the other hand, $\mathscr{P}$ is the distribution over the XxY population. At the beginning of the iteration ($t = 1, 2, \ldots, T$), the distribution (D) will be set to be uniform for every different strategy ($i = 1, 2, \ldots, N$) so $D(i) = 1/N$. The algorithm assigns weights (w^t) over the training examples on each iteration. On the iteration (t), a distribution p^t is computed by normalizing the weights.

$$p^t = \frac{w^t}{\sum_{i=1}^{N} w_i^t}$$

The distribution p^t is fed to the weak learner, which outputs a hypothesis (h_t), which is expected to have a slight error (ϵ_t) regarding the distribution (D).

$$h_t : \epsilon_t = \sum_{i=1}^{N} p_i^t |h_t(x_i) - y_i|$$

The algorithm AB is defined as $\beta_t = \epsilon_t / (1 - \epsilon_t)$.

The application of the new hypothesis (h_t) is used to generate the new weights (w_i^{t+1}), and the process repeats.

$$w_i^{t+1} = w_i^t \beta_t^{1-|h_t(x_i)-y_i|}$$

After T iterations, the output is the final hypothesis (h_f). It combines the outputs of the T weak hypotheses using a weighted majority vote (Freund and Schapire, 1997).

$$h_f(x) = \begin{cases} 1, & \text{if } \sum_{t=1}^{T} \left(\frac{\log 1}{\beta_t} \right) h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \frac{\log 1}{\beta_t} \\ 0, & \text{otherwise.} \end{cases}$$

7.5 Hyperparameters Optimization

Until now, the implemented ML algorithms were introduced and explained. In this chapter, the optimization process will be described. Hyper-parameters command the operation of the algorithm and therefore, they influence on the score (Wu, Chen, Zhang, Xiong, Lei and Deng, 2019). For this step, the open-source hyper-parameter optimization framework Optuna is used. This framework is a versatile next-generation optimization software. The framework raises the optimization hyper-parameters as a minimizing cost function that takes a set of given parameters as the function's input and measures the resulting output as the score. Based on the obtained results, the algorithm is designed to delimit the search space of the hyper-parameters dynamically. Each optimization process receives the name of a *study*, composed of a bunch of *trials*. The user suggests minimum and maximum values for the search space. The framework is based on the result of the trials and, in a dynamic way, delimits the search space (Akiba et al., 2019).

For the optimization process Optuna will train the model using the training set; and automatically testing it on the test set. When the training is done, the hyperparameters which achieved the highest scores will be printed to be replicated.

The number of trials in the study, and the search space should be given manually by the user. For each algorithm described in this thesis, the total number of trials was set to 150. The test set is composed of 25% of the observations of the data frame, and the used random state is the number 42.

The Optuna Code is shown in section 11.8. It is also possible to check it online on the following url: `https://gitfront.io/r/gonzalez-sonnenberg/cbvDMonp1t8j/Master-s-Thesis-Gonzalez-Sonnenberg-gitfront/`

As an example for the training of the RF, the parameter n_estimators is used for delimiting the number of trees that each forest has. For this thesis, the suggested search space for this estimator was having a minimum of 10 and a maximum of 350. Based on the scikit learn framework the default value is 100. Despite of this, the optimum founded by the Optuna framework is 286. R^2 obtained using n_estimators = 100: 83,34%; R^2 obtained using n_estimators = 286: 83,54%. Improvement: 0,2%.

The final result of the hyper-parameters of the RF were n_estimators = 286, criterion = squared error, max_depth = 17, min_samples_leaf = 2, max_features = 9, max_leaf_nodes = 4997.

Chapter 8

Results

This thesis aims to help home buyers in the challenging task of getting a better house for the lowest possible price. This challenge was segregated into four steps to work on. Picking up the best error metrics; selecting the promising features to improve the search quality on the houses due to the heat map, PFI, and PDP approaches; presenting and explaining the four ML algorithms LR, DT, RF, and AB; and optimizing them to get the best predictive results.

To achieve this aim, it is required to develop a model that can estimate the selling price of a house in the city of Berlin. In that case, if the estimated house price is above the actual price of the house, it is a sign not to buy the property. On the contrary, if the predicted price is below the selling price, it is a purchasing sign. Another important key question to answer is whether the LR achieves a better predictive score than the other machine learning algorithms such as DT, RF, and AB. According to the carried study of Ragapriya et al. (2023) LR tends to outperform DT, and AB. According to Soltani et al. (2022), Gokalani et al. (2022), and Manasa et al. (2020) LR achieved the worst compared result.

Therefore, this chapter is divided into two phases: the first one corresponds to the results of the feature selection methods, and the second one is dedicated to the predicted results on the statistical learning models.

8.1 Feature Selection

Three techniques were discussed, described, and presented in the Feature Selection phase: Heat Map, PFI, and PDP.

The heat map Figure 6.1 is an indicator to analyze the correlation between a predictor and the target. According to the Heat map, area (69%), rooms (43%), garden (13%), garage (10%), and elevator (3%) show to have a positive correlation with the price. Contrary, basement (-6%), balcony (-3,6%), and restoration (-1,5%) show negative correlation.

PFI shows the error reduction that a feature achieves on a certain model. Therefore, it helps to objectively quantify a predictor's importance. Figure 6.2 displays that for the four researched algorithms, the area is the most important predictor in the dataset. Nevertheless, from the second predictor onward, there are some discrepancies. LR includes district in second place, while DT, RF, and AB display quarter on the second place. The same occurs with the third position, where for DT, RF, and AB, the choice remains with construction year, and for LR, it is placed for feature rented. Already in

fourth place, the differences between the models become even more pronounced. On the one hand, RF and AB have a common predictor: district. LR choices is quarter and DT is rooms. It can also be observed that garage and restoration could be deleted from the dataset because their relative importance is almost nil.

The implementation of PDP allows collecting information regarding each model on the relation of two features in a visual way. These predictors must not be correlated. Otherwise, this tool will lead to a misunderstanding and false correlations (Friedman, 2001). The images on section 11.5 display that LR presented only linear relationships between the target and the predictors. It is also possible to see the flexibility (RF) or inflexibility (LR) of a model. The greater the flexibility of a model, the greater the tendency of the model to overfit the training data (Hawkins, 2004). Thanks to PDP, it is also possible to note that not all algorithms agree on how to interact with a predictor. For example, area; in this case Figure 11.5 shows that all the algorithms agree that more area means a higher price. Conversely, with the feature restoration in Figure 11.15, and Figure 11.19, it is noticeable that LR shows that the price of a property will decrease if it has a restoration, for DT, and AB restoration has any relevance; and for RF it increments the price of a property.

8.2 Supervised Machine Learning Algorithms

In this thesis, the supervised ML algorithms of LR, DT, RF, and AB were explained, trained, and tested. In this section, the training and test results are displayed and compared.

Model:	**Training Set**				**Test Set**			
	R^2	MSE[1]	RMSE	MAE	R^2	MSE[1]	RMSE	MAE
LR	54,97%	95.710	309.371	192.306	57,56%	92.995	304.951	189.383
DT	85,47%	31.116	176.397	100.124	77,34%	48.677	220.630	127.526
RF	94,68%	11.297	106.289	58.827	83,90%	35.275	187.817	105.120
AB	65,61%	73.097	270.365	178.948	67,50%	71.209	266.850	174.539

[1] MSE values are expressed in millions.

Table 8.1: Performance results on the trained algorithms over the training and the test set

The algorithm with the best performance on the test set is RF. However, it is also noticeable that it has a large amount of overfitting, as shown in the Table 8.2 since the difference between the training and test set results is over ten perceptual points. The algorithm with the lowest performance is LR. The results of Table 8.1 are reflected in Figures 11.20. In the scatter plots, it can be observed the relationship between actual values and predicted values for the test and training set. It is observable, that the better the price estimate, the smaller the error generated, and therefore the closer the prediction is to the line. The more expanded the points of the plot are, the worse the performance of the model. The Figures 11.21 visualize the Residuals for each predicted value. The Residuals are composed of the difference between actual and

predicted values (James et al., 2013). This analysis was also used by Jha, Babiceanu, Pandey and Jha (2020), Choy and Ho (2023), Crosby et al. (2016), Mora-Garcia et al. (2022), among others.

Model	Training Set	Test Set	Difference
	R^2	R^2	R^2
LR	54,97%	57,56%	2,59%
DT	85,47%	77,34%	-8,13%
RF	94,68%	83,9%	-10,78%
AB	65,61%	67,5%	1,89%

Table 8.2: Score Difference between Training and Test Set

With the employed hyper-parameter optimization framework, the results on the testing set may be better than the results on the training set. This is the case for LR and AB. Conversely, DT and RF tend to overfit the training set.

None of the proposed algorithms can estimate the price of a property with a lower threshold than 187.817 RMSE or Euros because RMSE is also expressed in the unit of measurement.

8.3 Concluding summary

To conclude, the three feature selection approaches helped evaluate each feature's relative importance for the predictive task in the Berlin housing data frame. Heat map constructed an overview of the data frame; PFI served to quantify the relative importance of each feature on each model; and PDP showed the relation of the chosen algorithm with each feature compared to the dependent variable, this allowed to estimate how flexible an algorithm is, and how the model interacts with the feature.

The RF model outperformed AB, DT, and LR. RF presented the highest overfitting. However, its results were the best over the test set. Conversely, the LR model achieved the worst performance among the analyzed models. While DT and RF overfit the training set, LR and AB underfit the training set.

Chapter 9

Discussion

In this thesis, all studied algorithms agree that area is the most essential feature for the pricing task. Starting from the second feature in Figure 6.2, there are different ratings and assessments between the reasons for the choice of the relative importance for the attributes according to LR, and DT, RF, and AB. This last group of algorithms has prioritized quarter and construction year for the second and third positions. In contrast, LR choices were for district and rented, respectively.

In the research of Mora-Garcia et al. (2022) it is possible to observe that the algorithm Gradient Boosting gives the highest relative importance to the feature area. In contrast, in the same study, External Trees found as the best predictor the net household income.

The tools of Heat map,PFI, and PDP were critical in analyzing and interpreting the resulting outputs and concluding that every algorithm evaluates each predictor differently. Therefore, there is no 'rule of thumb' to simplify the task of selecting certain features to estimate the selling price of a property. However, the attributes that helped the algorithms the most with this task were area, quarter, construction year, district, rooms, rented, balcony, and garden. Conversely, the least relevant attributes for the prediction score are garage, and restoration.

Therefore, its possible to conclude that the first hypothesis is false, since each algorithm assesses to what extent an attribute determines the house prices. Thus, the answer to the first research question is, the essential attributes depend on each particular case, and the tools of Heat map, PFI, and PDP will contribute to make "black boxes" explainable.

Regarding the feature selection tools, it is noticed that multiple tools can have different results over the attributes; for example, the Heat map, and the PDP over the balcony feature have contrary statements. In Figure 6.1 is observed, that the Heat map presents a negative correlation between balcony and price (-3,6%). Conversely, PDP shows in Figure 11.18, that for each studied algorithm exists a positive relation between price and balcony. This relation seems to be slighter regarding to AB, but still positive. The cause of this is still undefined, and it deserves future investigation.

LR prioritizes the importance of features different than DT, RF, and AB based on their linear limitation. This LR limitation can be exemplified with the Figure 11.17 which shows that LR simplifies the relationship between area and price. The simplification leads to losses in the insights. Contrary to LR, DT, RF, and AB estimate similarly. After the research, and the displayed results, it is possible to affirm that AB, DT, and RF tend to outperform LR.

LR uses only linear relationships, which causes valuable information to get lost. Therefore, this study supports the idea exposed by Mora-Garcia et al. (2022), and supported by the examples of Soltani et al. (2022), Gokalani et al. (2022), and Manasa et al. (2020). Conversely, this research also differs from the results presented by Rahman et al. (2019). It is expected that the presented experiments provide a new literature reference into the regression relations between the ML algorithms of LR, DT, RF, and AB. In view of the above it is possible to affirm that for this thesis, the second proposed hypothesis is correct.

The best performing model is RF with an R^2 of 83,9% on the test set, and a RMSE of 187.817. The research of Choy and Ho (2023) evaluate the algorithms RF, Extra Trees, k-Nearest Neighbors, and Ordinary Least Squares. Gokalani et al. (2022) compare LR, DT, and RF. Koktashev, Makeev, Shchepin, Peresunko and Tynchenko (2019) use RF, LR, and Lasso Regression. Choy and Ho (2023), Gokalani et al. (2022), and Koktashev et al. (2019) demonstrate that RF get the best comparative result. Adversely the researchers Ho, Tang and Wong (2021), and Soltani et al. (2022) found that Gradient Boosting Tree achieves better scores than RF.

Regarding the scope of this thesis, it is possible to affirm that RF predicts the most accurate property prices. Nevertheless, the proposed models of this thesis do not achieve to predict under the threshold of 30.000 Euros difference. The reached minimum error is about 187.817 Euros. Therefore, it is possible to conclude that the third hypothesis is also correct.

It is noticed in Table 8.2 that RF, and DT tend to overfit the dataset; in this thesis, the score difference between the training and the test set is about 10,78% for RF, and of 8,13% for DT. Contrarily, the algorithms of LR, and AB achieved a better score in the test set, than in the training set. The difference is about 2,59% for LR and 1,89% for AB. Mora-Garcia et al. (2022) exposed that, those algorithms that use bagging have overfiting problems, and those that use boosting do not suffer this problem; and perform better. Thus, it is expected that increasing the number of training trials for AB model can solve this. However, further research is needed.situation

9.1 Limitations of this Thesis

This thesis presents some limitations. Firstly, it has to be written within a fixed period of five months. Secondly, for the creation of the data set, the selection of attributes external to the house was not taken into account. Mora-Garcia et al. (2022) present examples of these, as net household income, distance of the property to educational centers, distance of the property to urban green spaces, among others. Thirdly, no allowance was made for possible fluctuations in property prices. Fourth, there were technical limitations to crawl more websites to increase the gathered observations. Fifth, this thesis is the first research experience of the author in the topic of ML.

9.2 Further research

Further research is required within the realm of ML.

1. The analysis of the difference between Heat map, and PDP referred to the balcony attribute.

2. The analysis of the obtained difference between training, and test set.
3. The use of more algorithms, such as Gradient Boosting Trees, Catboost, Support Vector Machine, Artificial Neural Networks, Convolutional Neural Networks, Generative Adversarial Networks, among others to improve the scores.
4. Develop a reliable application to help home buyers.
5. Create a data set using time series, to evaluate the price fluctuations according to time.
6. Optimize the hyperparameters using more techniques such as manual search, Search Grid CV, Randomized Search, Bayesian Optimization, among others.
7. The use of price binning technique to optimize algorithms scores.

Chapter 10

Conclusion

In this thesis, a data set of the of city Berlin is created using web scraping. The dataframe has more than 14.000 house examples. The ML algorithms scope of this research is LR, DT, RF, and AB. These algorithms are trained and tested using Optuna, an hyper-parameter optimization framework.

The research aim is to develop a ML model, which can estimate house prices in the city of Berlin with high accuracy to help home buyers in the task of purchasing properties.

For this the house attributes are area, balcony, basement, bathroom, construction year, district, elevator, garage, garden, quarter, rented, restoration, and rooms. The independent variables are analyzed using three different techniques, Heat map, Permutation Feature Importance, and Partial Dependence Plot. The results show that area is the most relevant feature for the predictive aim. The second, and third important features are questionable between LR and DT, RF, and AB. Nevertheless, each algorithm has its own relative priorities regarding the importance of the features. Therefore, no evidence of a general rule for the predictor selection phase is found.

This thesis found a contradiction between two papers. The first one shows that LR outperforms DT and AB. Conversely, in other paper a general rule contrary to this first statement is exposed. Therefore, this thesis aims to shed light on this controversy. The results of this thesis show that RF outperformed the other algorithms. Whereas, LR has the worst performance for price forecasting.The final score is not considered suitable to help home-buyers in their purchasing task.

RF and DT use bagging; those algorithms that use bagging tend to overfit the data set. On the other hand, that algorithms that use boosting do not overfit.

References

Abdi, H., Williams, L. J. et al. (2010), 'Normalizing data', *Encyclopedia of research design* **1**.

Aho, A. V. (1991), 'Algorithms for finding patterns in strings, handbook of theoretical computer science (vol. a): algorithms and complexity'.

Akiba, T., Sano, S., Yanase, T., Ohta, T. and Koyama, M. (2019), Optuna: A next-generation hyperparameter optimization framework, *in* 'Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining'.

Aljohani, O. (2021), Developing a stable house price estimator using regression analysis, *in* 'The 5th International Conference on Future Networks & Distributed Systems', pp. 113–118.

Alzubi, J., Nayyar, A. and Kumar, A. (2018), Machine learning from theory to algorithms: an overview, *in* 'Journal of physics: conference series', Vol. 1142, IOP Publishing, p. 012012.

Barnston, A. G. (1992), 'Correspondence among the correlation, rmse, and heidke forecast verification measures; refinement of the heidke score', *Weather and Forecasting* **7**(4), 699–709.

Barrowman, N. (2014), 'Correlation, causation, and confusion', *The New Atlantis* pp. 23–44.

Baur, K., Rosenfelder, M. and Lutz, B. (2023), 'Automated real estate valuation with machine learning models using property descriptions', *Expert Systems with Applications* **213**, 119147.

Bechwati, N. N. and Morrin, M. (2003), 'Outraged consumers: Getting even at the expense of getting a good deal', *Journal of Consumer Psychology* **13**(4), 440–453.

Berlin-Brandenburg, A. S. (2023), 'Wachstum durch zuzug', https://www.statistik-berlin-brandenburg.de/141-2023.

Breiman, L. (1996), 'Bagging predictors', *Machine learning* **24**, 123–140.

Bundesamt, S. (2023), 'Pressemitteilung nr. 245 vom 23. juni 2023', https://www.destatis.de/DE/Presse/Pressemitteilungen/2023/06/PD23_245_61262.html.

Chen, M., Liu, Y., Arribas-Bel, D. and Singleton, A. (2022), 'Assessing the value of user-generated images of urban surroundings for house price estimation', *Landscape and Urban Planning* **226**, 104486.

Chernick, M. R. (2012), 'Resampling methods', *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **2**(3), 255–262.

Choy, L. H. and Ho, W. K. (2023), 'The use of machine learning in real estate research', *Land* **12**(4), 740.

Crosby, H., Davis, P., Damoulas, T. and Jarvis, S. A. (2016), A spatio-temporal, gaussian process regression, real-estate price predictor, *in* 'Proceedings of the 24th ACM

SIGSPATIAL International Conference on Advances in Geographic Information Systems', pp. 1–4.

Eckardt, F. (2005), 'In search for meaning: Berlin as national capital and global city', *Journal of Contemporary European Studies* **13**(2), 189–201.

Freund, Y. and Schapire, R. E. (1997), 'A decision-theoretic generalization of on-line learning and an application to boosting', *Journal of computer and system sciences* **55**(1), 119–139.

Friedman, J. H. (2001), 'Greedy function approximation: a gradient boosting machine', *Annals of statistics* pp. 1189–1232.

Gehlenborg, N. and Wong, B. (2012), 'Heat maps', *Nature Methods* **9**(3), 213.

Gokalani, L. B., Das, B., Ramnani, D. K., Kumar, M. and Shah, M. A. (2022), 'House price prediction of real time data (dha defence) karachi using machine learning', *Sir Syed University Research Journal of Engineering & Technology* **12**(2), 75–80.

Géron, A. (2019), *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*, 2nd edition edn, O'Reilly Media, Inc.

Hahn, G. J. (1973), 'The coefficient of determination exposed', *Chemtech* **3**(10), 609–612.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. and Oliphant, T. E. (2020), 'Array programming with NumPy', *Nature* **585**(7825), 357–362.
URL: *https://doi.org/10.1038/s41586-020-2649-2*

Hastie, T., Tibshirani, R., Friedman, J. H. and Friedman, J. H. (2009), *The elements of statistical learning: data mining, inference, and prediction*, Vol. 2, Springer.

Hawkins, D. M. (2004), 'The problem of overfitting', *Journal of chemical information and computer sciences* **44**(1), 1–12.

Ho, W. K., Tang, B.-S. and Wong, S. W. (2021), 'Predicting property prices with machine learning algorithms', *Journal of Property Research* **38**(1), 48–70.

Hunter, J. D. (2007), 'Matplotlib: A 2d graphics environment', *Computing in Science & Engineering* **9**(3), 90–95.

Iliev, A. I. and Anand, A. (2023), Huber loss and neural networks application in property price prediction, *in* 'Future of Information and Communication Conference', Springer, pp. 242–256.

James, G., Witten, D., Hastie, T., Tibshirani, R. et al. (2013), *An introduction to statistical learning*, Vol. 112, Springer.

Jha, S. B., Babiceanu, R. F., Pandey, V. and Jha, R. K. (2020), 'Housing market prediction problem using different machine learning algorithms: A case study', *arXiv preprint arXiv:2006.10092* .

Kagie, M. and Wezel, M. V. (2007), 'Hedonic price models and indices based on boosting applied to the dutch housing market', *Intelligent Systems in Accounting, Finance & Management: International Journal* **15**(3-4), 85–106.

Kholodilin, K. and Michelsen, C. (2020), 'Wohnungsmarkt in deutschland: Trotz krise steigende immobilienpreise, gefahr einer flächendeckenden preisblase aber gering', *DIW Wochenbericht* **87**(37), 684–693.

Khosravi, M., Arif, S. B., Ghaseminejad, A., Tohidi, H. and Shabanian, H. (2022), 'Performance evaluation of machine learning regressors for estimating real estate

house prices'.
Kim, S. and Kim, H. (2016), 'A new metric of absolute percentage error for intermittent demand forecasts', *International Journal of Forecasting* **32**(3), 669–679.
Koktashev, V., Makeev, V., Shchepin, E., Peresunko, P. and Tynchenko, V. (2019), Pricing modeling in the housing market with urban infrastructure effect, *in* 'Journal of Physics: Conference Series', Vol. 1353, IOP Publishing, p. 012139.
Köppen, M. (2000), The curse of dimensionality, *in* '5th online world conference on soft computing in industrial applications (WSC5)', Vol. 1, pp. 4–8.
Kouzis-Loukas, D. (2016), *Learning Scrapy*, Packt Publishing Ltd.
Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J. and Liu, H. (2017), 'Feature selection: A data perspective', *ACM computing surveys (CSUR)* **50**(6), 1–45.
Manasa, J., Gupta, R. and Narahari, N. (2020), Machine learning based predicting house prices using regression techniques, *in* '2020 2nd International conference on innovative mechanisms for industry applications (ICIMIA)', IEEE, pp. 624–630.
Mohd, T., Jamil, N. S., Johari, N., Abdullah, L. and Masrom, S. (2020), An overview of real estate modelling techniques for house price prediction, *in* 'Charting a Sustainable Future of ASEAN in Business and Social Sciences: Proceedings of the 3 International Conference on the Future of ASEAN (ICoFA) 2019—Volume 1', Springer, pp. 321–338.
Molnar, C. (2020), *Interpretable machine learning*, Lulu. com.
Mooney, P. (2019), 'Feature selection with permutation importance', https://www.kaggle.com/code/paultimothymooney/feature-selection-with-permutation-importance.
Mora-Garcia, R.-T., Cespedes-Lopez, M.-F. and Perez-Sanchez, V. R. (2022), 'Housing price prediction using machine learning algorithms in covid-19 times', *Land* **11**(11), 2100.
Özöğür Akyüz, S., Eygi Erdogan, B., Yıldız, Ö. and Karadayı Ataş, P. (2022), 'A novel hybrid house price prediction model', *Computational Economics* pp. 1–18.
O'brien, R. M. (2007), 'A caution regarding rules of thumb for variance inflation factors', *Quality & quantity* **41**, 673–690.
pandas development team, T. (2020), 'pandas-dev/pandas: Pandas'.
URL: *https://doi.org/10.5281/zenodo.3509134*
Pearl, J. et al. (2000), 'Models, reasoning and inference', *Cambridge, UK: CambridgeUniversityPress* **19**(2), 3.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011), 'Scikit-learn: Machine learning in Python', *Journal of Machine Learning Research* **12**, 2825–2830.
Piramuthu, S. (2004), 'Evaluating feature selection methods for learning in data mining applications', *European journal of operational research* **156**(2), 483–494.
Plevris, V., Solorzano, G., Bakas, N. P. and Ben Seghier, M. E. A. (2022), Investigation of performance metrics in regression analysis and machine learning-based prediction models, *in* '8th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS Congress 2022)', European Community on Computational Methods in Applied Sciences.
Ragapriya, N., Kumar, T. A., Parthiban, R., Divya, P., Jayalakshmi, S. and Raman, D. R.

(2023), 'Machine learning based house price prediction using modified extreme boosting', *Asian Journal of Applied Science and Technology (AJAST)* **7**(1), 41–54.

Rahman, S. N. A., Maimun, N. H. A., Razali, M. N. M. and Ismail, S. (2019), 'The artificial neural network model (ann) for malaysian housing market analysis', *Planning Malaysia* **17**.

Sanyal, S., Biswas, S. K., Das, D., Chakraborty, M. and Purkayastha, B. (2022), Boston house price prediction using regression models, *in* '2022 2nd International Conference on Intelligent Technologies (CONIT)', IEEE, pp. 1–6.

Schapire, R. E. (2003), 'The boosting approach to machine learning: An overview', *Nonlinear estimation and classification* pp. 149–171.

Schapire, R. E. et al. (1999), A brief introduction to boosting, *in* 'Ijcai', Vol. 99, Citeseer, pp. 1401–1406.

Schneider, P. and Xhafa, F. (2022), *Anomaly Detection and Complex Event Processing Over IoT Data Streams: With Application to EHealth and Patient Data Monitoring*, Academic Press.

Soltani, A., Heydari, M., Aghaei, F. and Pettit, C. J. (2022), 'Housing price prediction incorporating spatio-temporal dependency into machine learning algorithms', *Cities* **131**, 103941.

Thamarai, M. and Malarvizhi, S. (2020), 'House price prediction modeling using machine learning', *International Journal of Information Engineering & Electronic Business* **12**(2).

Waskom, M. L. (2021), 'seaborn: statistical data visualization', *Journal of Open Source Software* **6**(60), 3021.
URL: *https://doi.org/10.21105/joss.03021*

Wu, J., Chen, X.-Y., Zhang, H., Xiong, L.-D., Lei, H. and Deng, S.-H. (2019), 'Hyperparameter optimization for machine learning models based on bayesian optimization', *Journal of Electronic Science and Technology* **17**(1), 26–40.

Yağmur, A., Kayakuş, M. and Terzioğlu, M. (2022), 'House price prediction modeling using machine learning techniques: a comparative study', *Aestimum* **81**.

Yan, X. and Su, X. (2009), *Linear regression analysis: theory and computing*, world scientific.

Yu, C. H. (2002), 'Resampling methods: concepts, applications, and justification', *Practical Assessment, Research, and Evaluation* **8**(1), 19.

Chapter 11

Appendix

11.1 Attributes Abbreviations

Feature	Abbreviation
Price	*pr*
Construction Year	*cy*
Area	*ar*
Rooms	*ro*
Balcony	*bal*
Rented	*ren*
Garden	*gan*
Basement	*bas*
Elevator	*el*
Garage	*gar*
Bathroom	*bat*
District	*di*
Quarter	*qu*
Restoration	*res*

Table 11.1: Feature Abbreviations

Quarter N	Quarter	District N	District
1	Mitte	1	Mitte
2	Tiergarten	1	Mitte
3	Wedding	1	Mitte
4	Moabit	1	Mitte
5	Hansaviertel	1	Mitte
6	Gesundbrunnen	1	Mitte
7	Friedrichshain	2	Friedrichshain-Kreuzberg
8	Kreuzberg	2	Friedrichshain-Kreuzberg
9	Prenzlauer	3	Pankow
10	Rosenthal	3	Pankow

Quarter N	Quarter	District N	District
11	Französisch	3	Pankow
12	Weißensee	3	Pankow
13	Karow	3	Pankow
14	Pankow	3	Pankow
15	Niederschönhausen	3	Pankow
16	Heinersdorf	3	Pankow
17	Blankenburg	3	Pankow
18	Blankenfelde	3	Pankow
19	Buch	3	Pankow
20	Prenzlauer Berg	3	Pankow
21	Französisch Buchholz	3	Pankow
22	Wilhelmsruh	3	Pankow
23	Buchholz	3	Pankow
24	Grunewald	4	Charlottenburg-Wilmersdorf
25	Wilmersdorf	4	Charlottenburg-Wilmersdorf
26	Charlottenburg	4	Charlottenburg-Wilmersdorf
27	Schmargendorf	4	Charlottenburg-Wilmersdorf
28	Halensee	4	Charlottenburg-Wilmersdorf
29	Westend	4	Charlottenburg-Wilmersdorf
30	Staaken	5	Spandau
31	Kladow	5	Spandau
32	Spandau	5	Spandau
33	Haselhorst	5	Spandau
34	Siemensstadt	5	Spandau
35	Gatow	5	Spandau
36	Dahlem	6	Steglitz-Zehlendorf
37	Nikolassee	6	Steglitz-Zehlendorf
38	Zehlendorf	6	Steglitz-Zehlendorf
39	Wannsee	6	Steglitz-Zehlendorf
40	Steglitz	6	Steglitz-Zehlendorf
41	Lichterfelde	6	Steglitz-Zehlendorf
42	Lankwitz	6	Steglitz-Zehlendorf
43	Friedenau	7	Tempelhof-Schöneberg
44	Tempelhof	7	Tempelhof-Schöneberg
45	Schöneberg	7	Tempelhof-Schöneberg
46	Marienfelde	7	Tempelhof-Schöneberg
47	Lichtenrade	7	Tempelhof-Schöneberg
48	Mariendorf	7	Tempelhof-Schöneberg
49	Neukölln	8	Neukölln
50	Buckow	8	Neukölln
51	Britz	8	Neukölln
52	Rudow	8	Neukölln
53	Gropiusstadt	8	Neukölln
54	Altglienicke	9	Treptow-Köpenick
55	Rahnsdorf	9	Treptow-Köpenick
56	Köpenick	9	Treptow-Köpenick
57	Bohnsdorf	9	Treptow-Köpenick

Quarter N	Quarter	District N	District
58	Müggelheim	9	Treptow-Köpenick
59	Baumschulenweg	9	Treptow-Köpenick
60	Grünau	9	Treptow-Köpenick
61	Friedrichshagen	9	Treptow-Köpenick
62	Niederschöneweide	9	Treptow-Köpenick
63	Adlershof	9	Treptow-Köpenick
64	Johannisthal	9	Treptow-Köpenick
65	Oberschöneweide	9	Treptow-Köpenick
66	Treptow	9	Treptow-Köpenick
67	Schmöckwitz	9	Treptow-Köpenick
68	Plänterwald	9	Treptow-Köpenick
69	Hessenwinkel	9	Treptow-Köpenick
70	Wilhelmshagen	9	Treptow-Köpenick
71	Alt-Treptow	9	Treptow-Köpenick
72	Karolinenhof	9	Treptow-Köpenick
73	Mahlsdorf	10	Marzahn-Hellersdorf
74	Biesdorf	10	Marzahn-Hellersdorf
75	Kaulsdorf	10	Marzahn-Hellersdorf
76	Marzahn	10	Marzahn-Hellersdorf
77	Hellersdorf	10	Marzahn-Hellersdorf
78	KGA Neues Leben	10	Marzahn-Hellersdorf
79	KGA Einigkeit	10	Marzahn-Hellersdorf
80	Lichtenberg	11	Lichtenberg
81	Alt	11	Lichtenberg
82	Wartenberg	11	Lichtenberg
83	Friedrichsfelde	11	Lichtenberg
84	Lübars	11	Lichtenberg
85	Karlshorst	11	Lichtenberg
86	Falkenberg	11	Lichtenberg
87	Malchow	11	Lichtenberg
88	Neu	11	Lichtenberg
89	Alt-Hohenschönhausen	11	Lichtenberg
90	Neu-Hohenschönhausen	11	Lichtenberg
91	Frohnau	12	Reinickendorf
92	Konradshöhe	12	Reinickendorf
93	Tegel	12	Reinickendorf
94	Heiligensee	12	Reinickendorf
95	Hermsdorf	12	Reinickendorf
96	Reinickendorf	12	Reinickendorf
97	Wittenau	12	Reinickendorf
98	Waidmannslust	12	Reinickendorf
99	Borsigwalde	12	Reinickendorf
100	Märkisches Viertel	12	Reinickendorf

Table 11.2: Quarter and District Abbreviations

For more information the entire dataset is hosted on github at the following url: https://

gitfront.io/r/gonzalez-sonnenberg/cbvDMonp1t8j/Master-s-Thesis-Gonzalez-Sonnenberg-gitfront/

11.2 Frame of the Real Estate Data Frame over the city of Berlin

*pr**	*cy*	*ar*	*ro*	*bal*	*ren*	*gan*	*bas*	*el*	*gar*	*bat*	*di*	*qu*	*res*
503	1931	89	3,5	0	0	0	0	0	0	0	4	25	0
455	1908	87	3	0	0	0	0	0	0	0	4	25	0
185	1911	55	2	0	0	0	0	0	0	0	4	25	0
229	1955	34	1	0	1	0	0	0	0	0	4	25	0
495	1912	107	3	1	0	0	0	0	0	0	4	25	0
389	1905	65	2	0	0	0	0	0	0	0	4	25	0
329	1905	52	2	0	0	0	0	0	0	0	4	25	0
219	1898	39	1	0	0	0	0	0	0	0	4	25	0
698,5	1898	131	4	1	1	0	0	1	0	0	4	25	0
974	2023	113	4	1	0	0	0	0	0	0	4	25	0
259,9	1956	58	2,5	1	1	0	0	0	0	0	4	25	0
418	1954	66	2,5	0	0	0	0	0	0	0	4	25	0
155	1970	36	1	1	0	0	0	0	0	0	4	25	0
250	1969	49	2	1	0	0	0	1	0	0	4	25	0
770	1912	187	5	1	0	0	0	1	0	0	4	25	0

Table 11.3: Fifteen first items from the data frame
*The price displayed in Table 11.3 is expressed in thousands of Euros.

*pr**	*cy*	*ar*	*ro*	*bal*	*ren*	*gan*	*bas*	*el*	*gar*	*bat*	*di*	*qu*	*res*
189,9	1.996	34	1	0	1	0	0	0	0	0	9	71	0
528	1.900	73	2	1	0	0	0	0	0	0	9	71	0
179,9	1.908	34	1	0	1	0	0	0	0	0	9	71	0
459	1.910	75	3	0	0	0	0	0	0	0	9	71	0
325	2.000	75	2	1	1	0	0	0	0	0	9	71	0
335	2.000	77	2	1	1	0	0	0	0	0	9	71	0
189	1.910	52	2	0	0	0	0	0	0	0	9	71	0
2.600	1.900	966	30	1	1	0	0	0	0	0	9	71	0
2.600	1.900	1.098	30	1	1	0	0	0	0	0	9	71	0
365	1.920	79	3	0	0	0	1	0	0	0	9	71	0
364	1.910	65	2	1	1	0	0	0	0	0	9	71	0
649	1.919	128	4	1	1	0	0	0	0	0	9	71	0
382	1.960	150	3	0	0	1	0	0	1	0	10	79	0
729	2.019	117	5	0	0	0	0	1	0	0	10	78	0
644	1.950	140	1	0	0	0	0	0	0	0	10	78	0

Table 11.4: Fifteen last items from the data frame
*The price displayed in Table 11.4 is expressed in thousands of Euros.

For more information the entire dataset is hosted on github at the following url: https://

gitfront.io/r/gonzalez-sonnenberg/cbvDMonp1t8j/Master-s-Thesis-Gonzalez-Sonnenberg-gitfront/

11.3 Data Frame Exploration

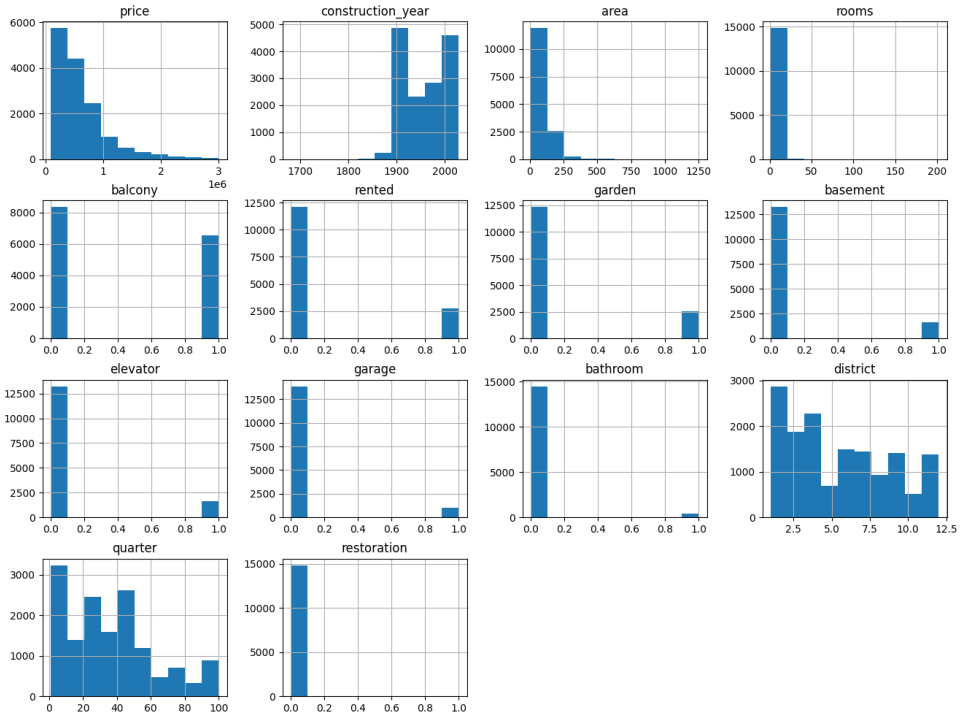
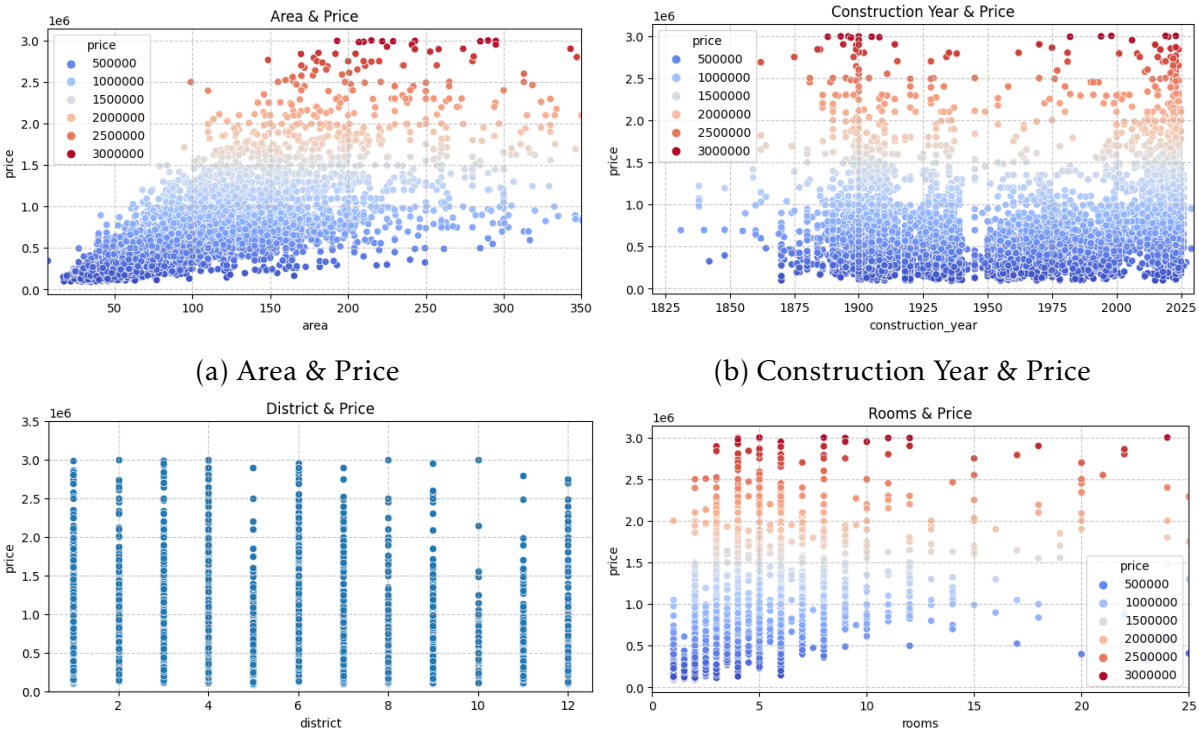


Figure 11.1: Histogram of the Data Frame over the city of Berlin.



(a) Area & Price

(b) Construction Year & Price

(c) District & Price

(d) Rooms & Price

Figure 11.2: Data Frame Exploration

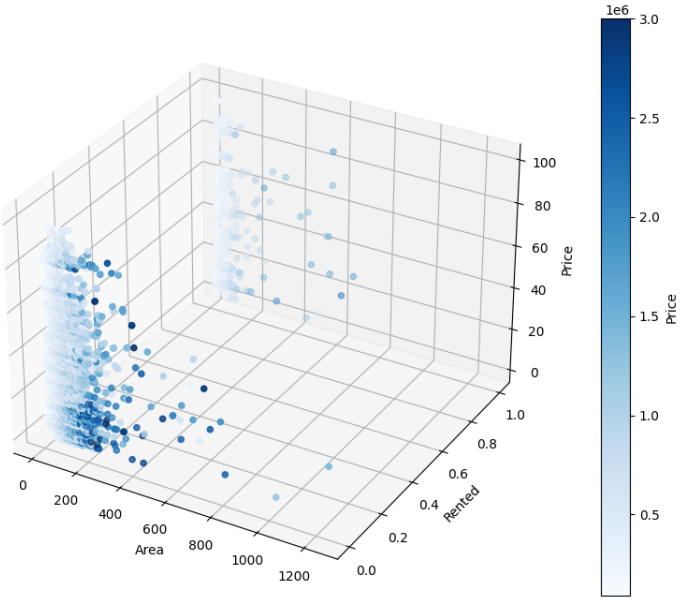


Figure 11.3: Relation between the predictors Area, Rented, and Price

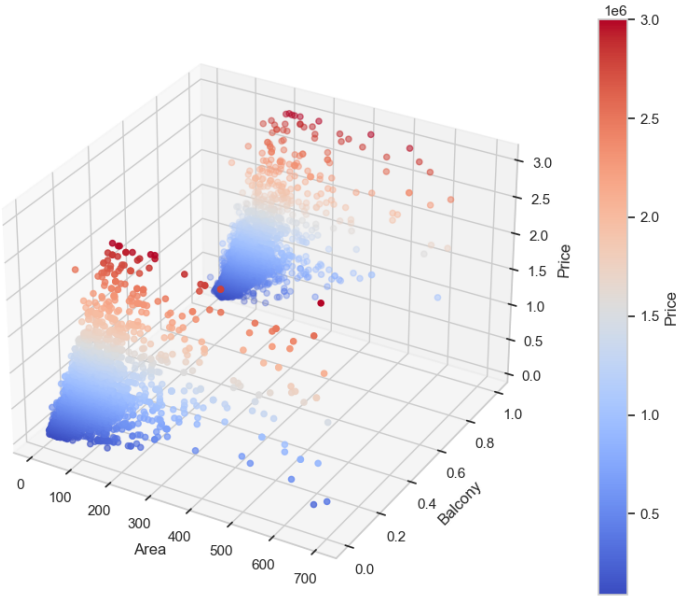


Figure 11.4: Relation between the predictors Balcony, Area, and Price

11.4 Normalized Data Frame

	pr	*cy*	*ar*	*ro*	*bal*	*ren*	*gan*	*bas*	*el*	*gar*	*bat*	*di*	*qu*	*res*
mean	0,18	0,80	0,08	0,01	0,44	0,19	0,17	0,11	0,11	0,07	0,03	0,41	0,36	0
std	0,16	0,14	0,06	0,01	0,50	0,39	0,38	0,31	0,31	0,25	0,16	0,29	0,26	0,05
min	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25%	0,07	0,66	0,04	0	0	0	0	0	0	0	0	0,18	0,13	0
50%	0,13	0,80	0,06	0,01	0	0	0	0	0	0	0	0,36	0,31	0
75%	0,23	0,92	0,09	0,01	1	0	0	0	0	0	0	0,64	0,49	0
max	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 11.5: Descriptive Statistics of Normalized Data Set

11.5 Partial Dependence Plot

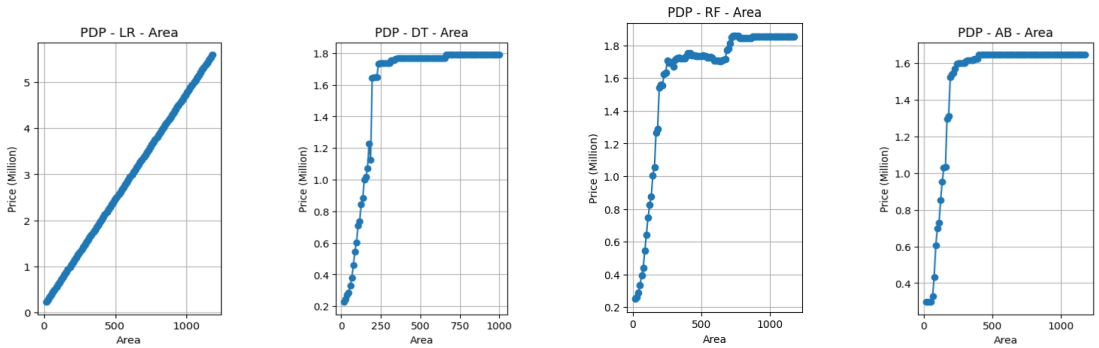


Figure 11.5: PDP - Area

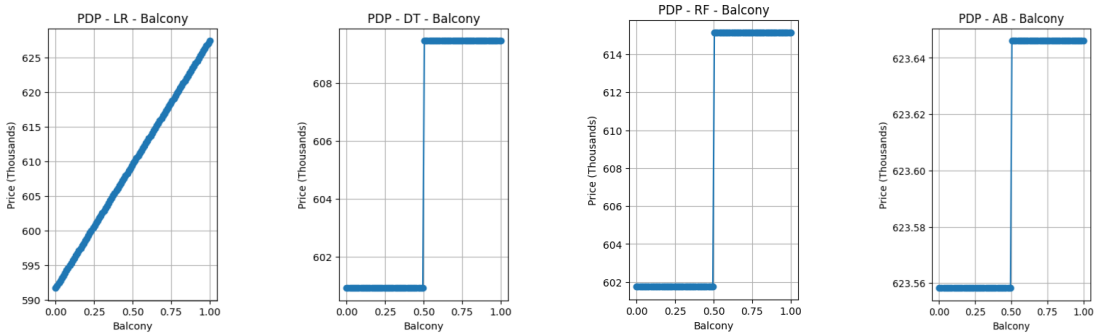


Figure 11.6: PDP - Balcony

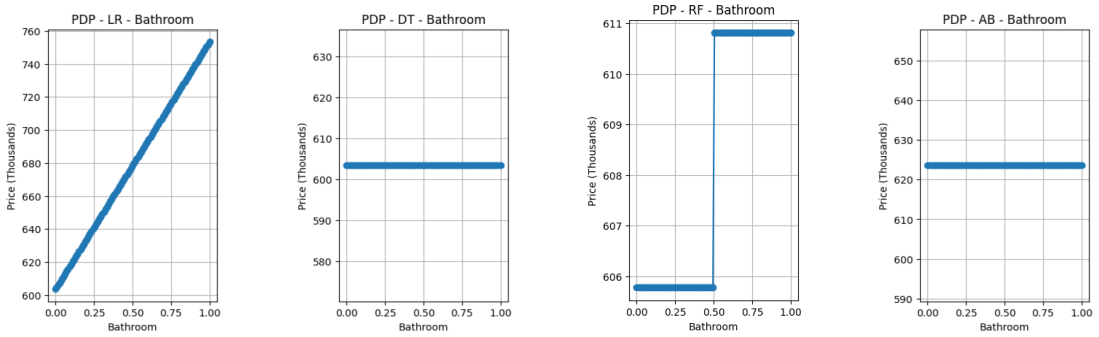


Figure 11.7: PDP - Bathroom

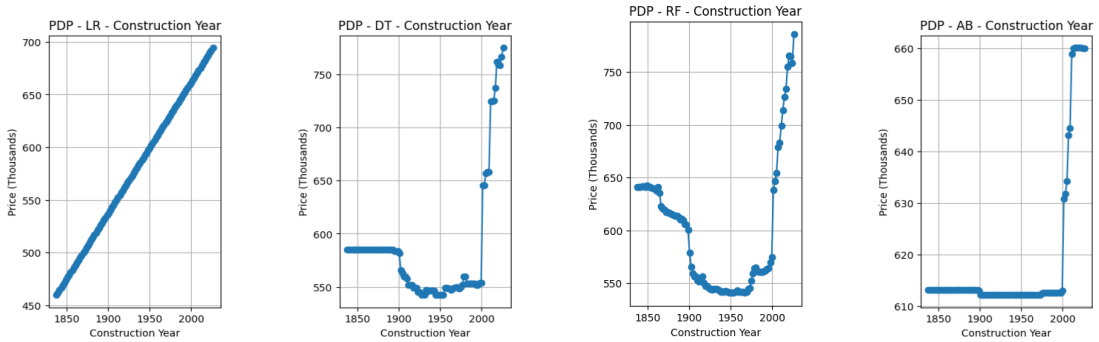


Figure 11.8: PDP - Construction Year

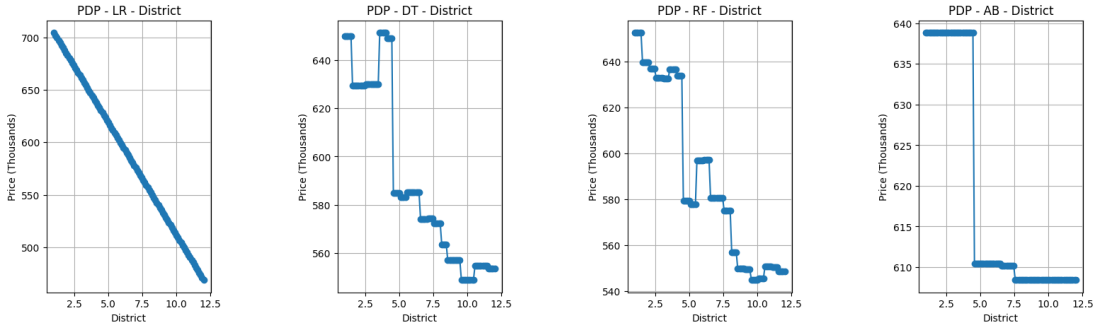


Figure 11.9: PDP - District

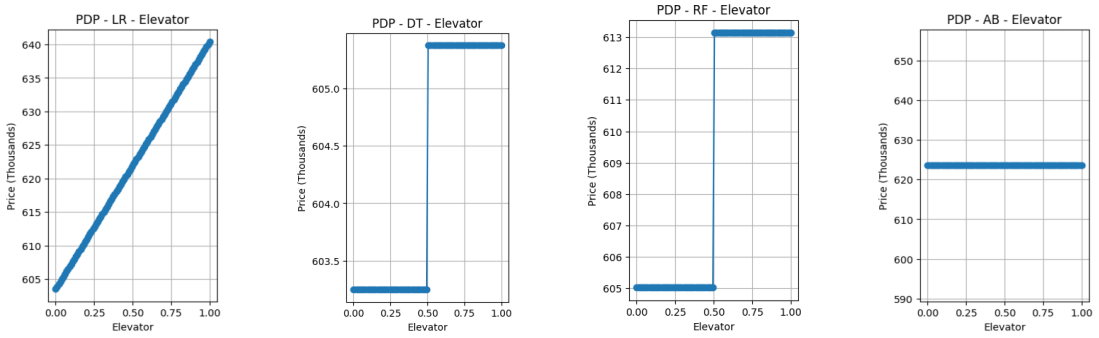


Figure 11.10: PDP - Elevator

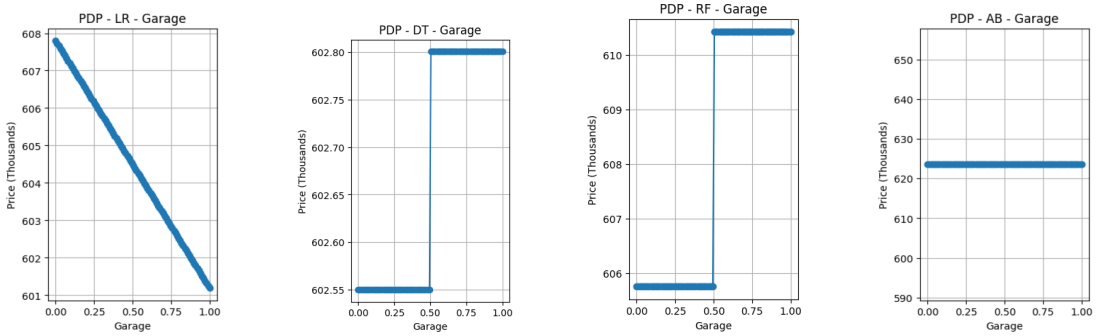


Figure 11.11: PDP - Garage

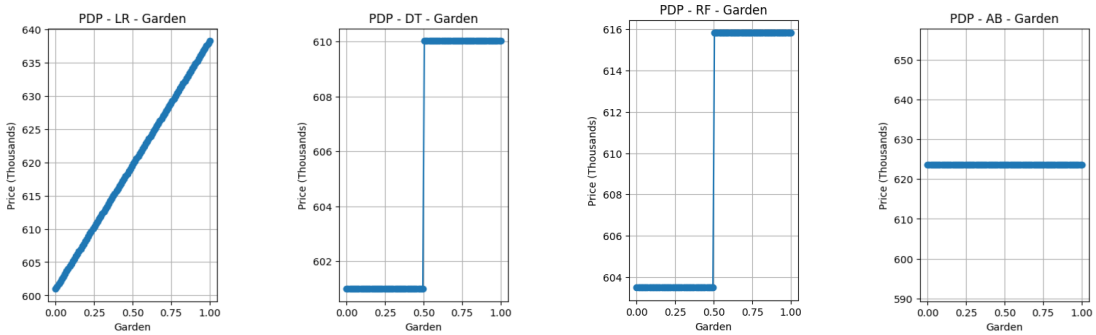


Figure 11.12: PDP - Garden

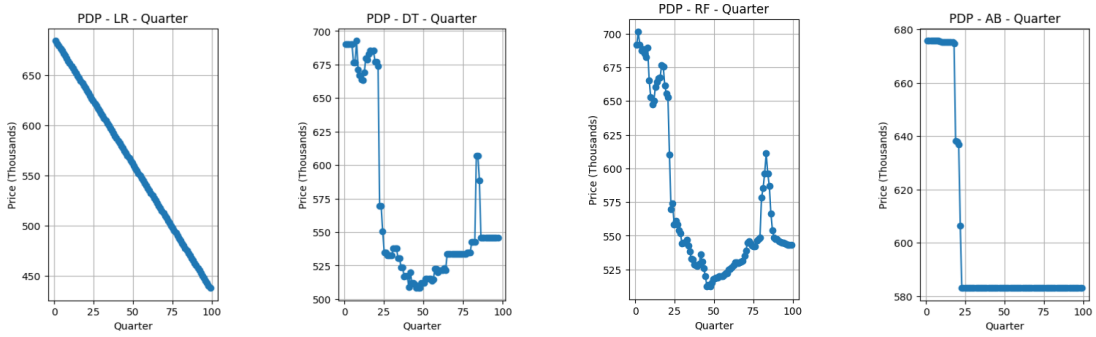


Figure 11.13: PDP - Quarter

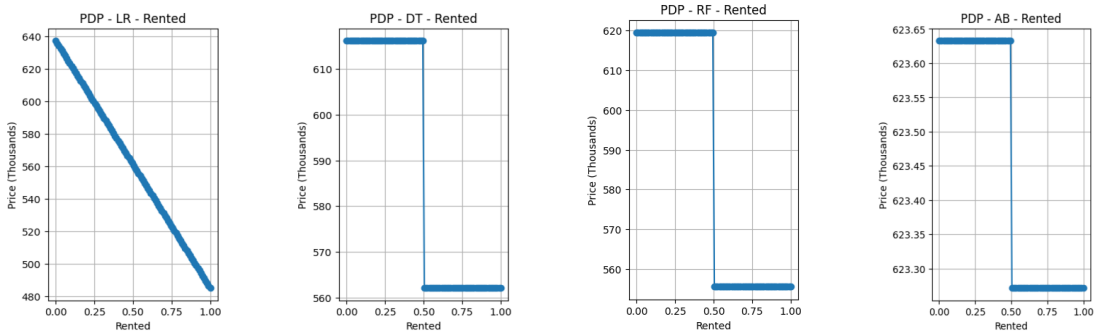


Figure 11.14: PDP - Rented

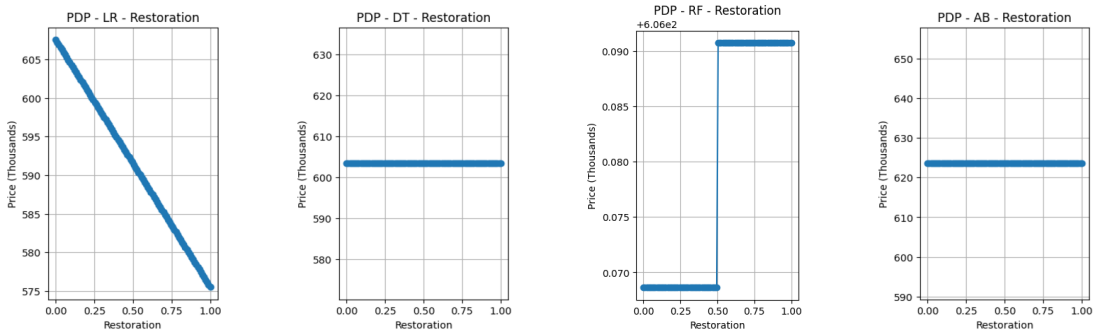


Figure 11.15: PDP - Restoration

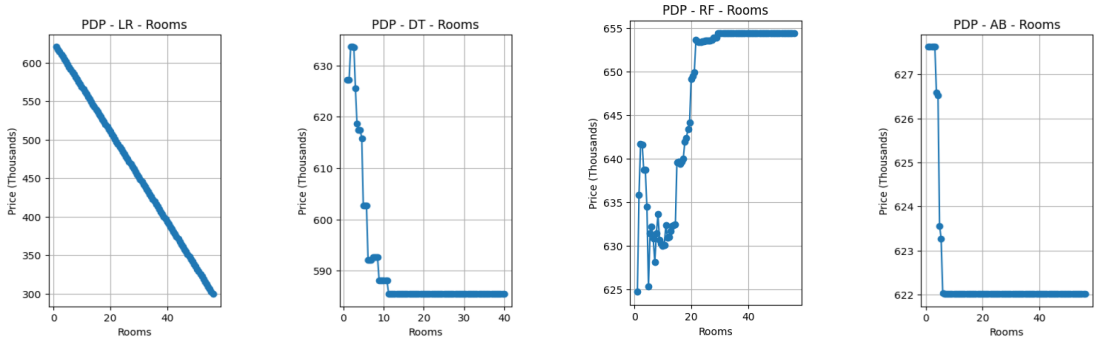


Figure 11.16: PDP - Rooms

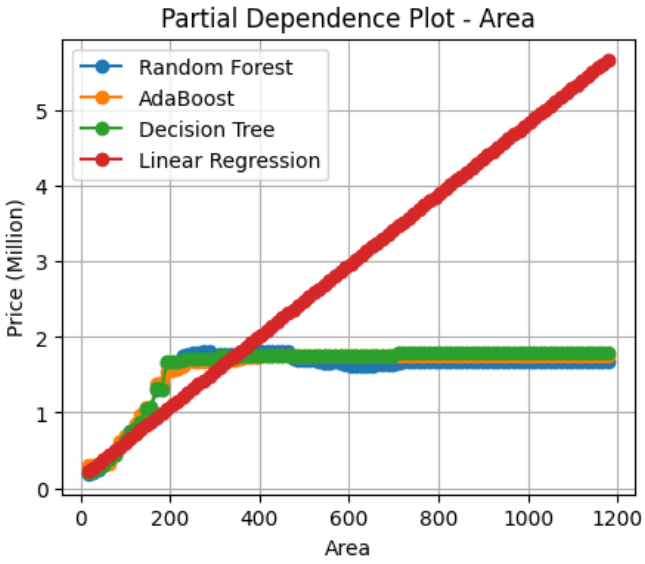


Figure 11.17: Relation between Area and Price

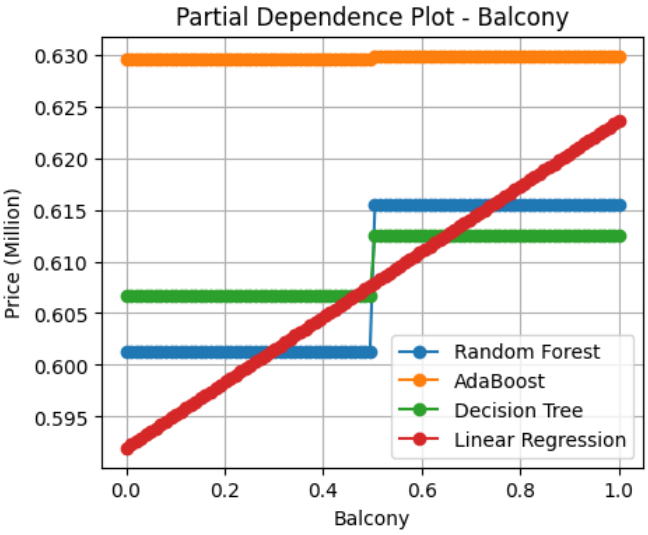


Figure 11.18: Relation between Balcony and Price

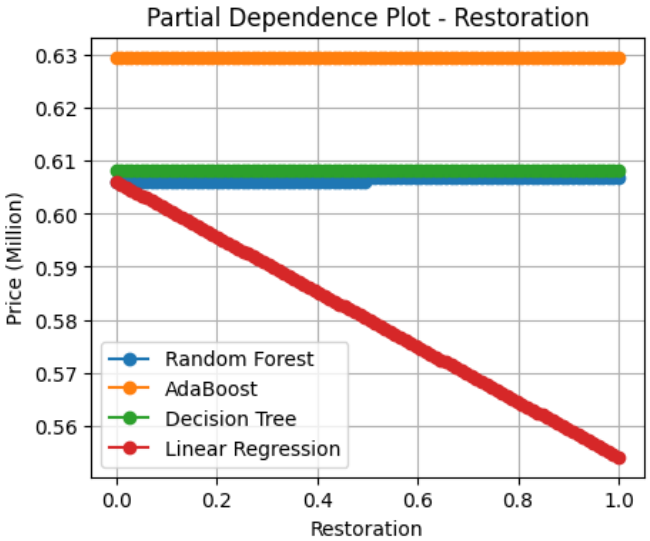
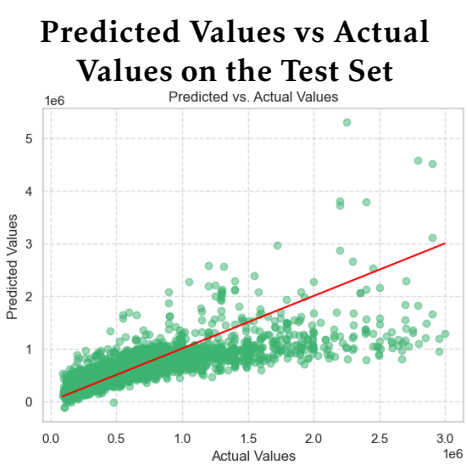
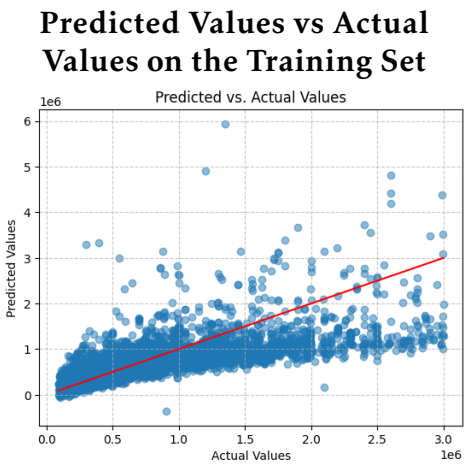


Figure 11.19: Relation between Restoration and Price

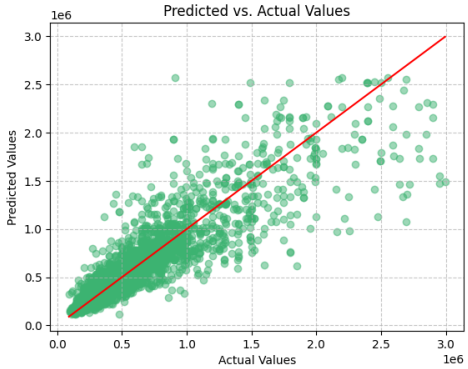
11.6 Prediction Error and Residual Plots



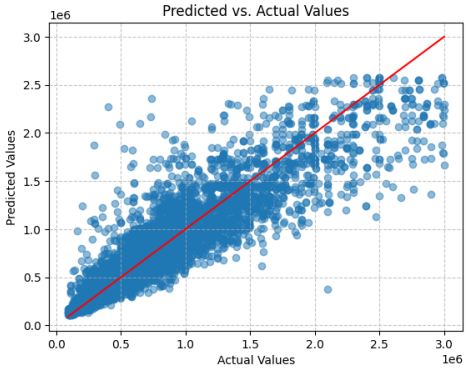
(a) LR on the test set



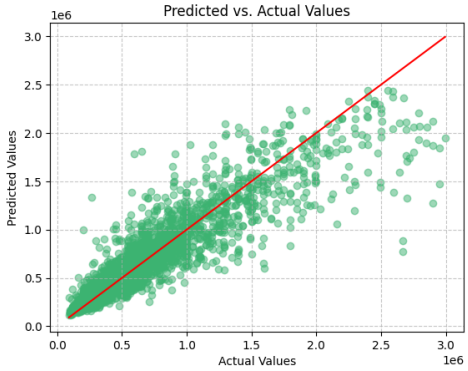
(b) LR on the training set



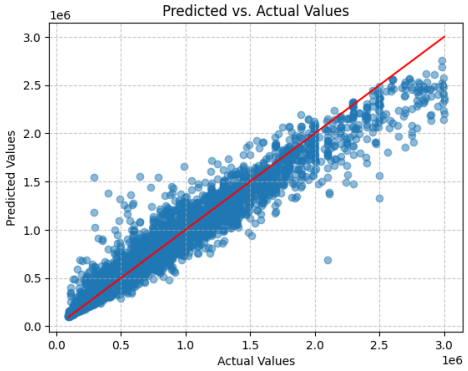
(c) DT on the Test set



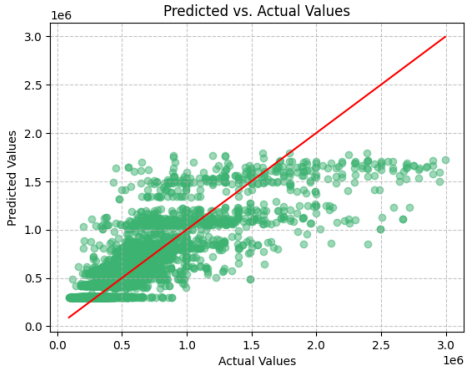
(d) DT on the Training set



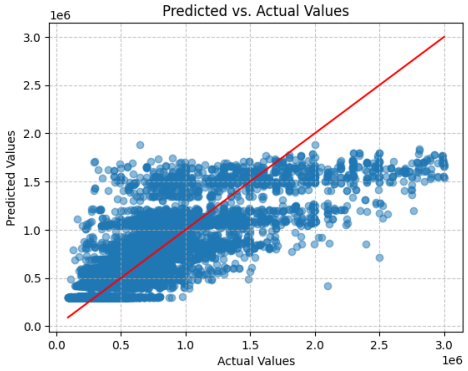
(e) RF on the Test set



(f) RF on the Training set

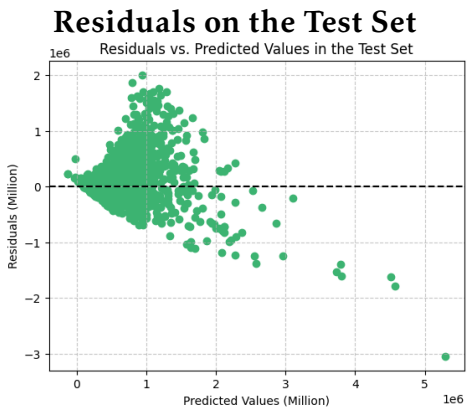


(g) AB on the Test set

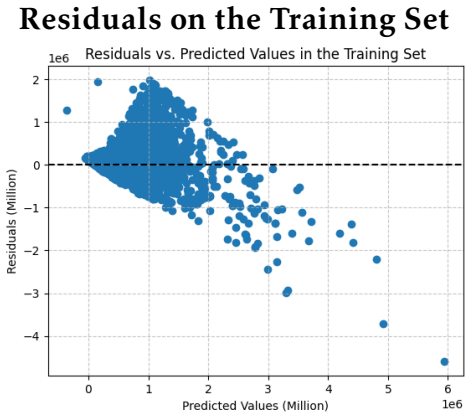


(h) AB on the Training set

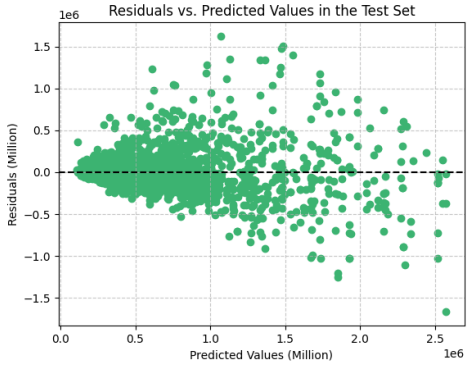
Figure 11.20: Performance on the Test & Training Set



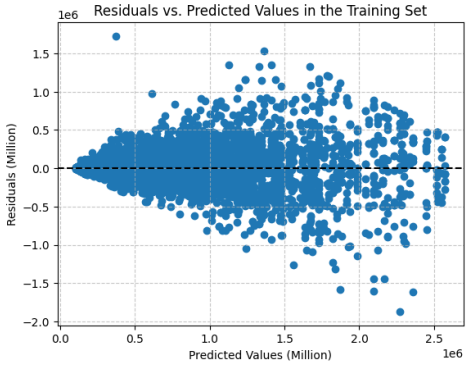
(a) LR Residuals on the Test Set



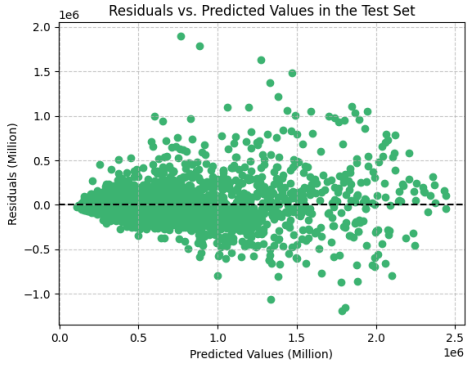
(b) LR Residuals on Training Set



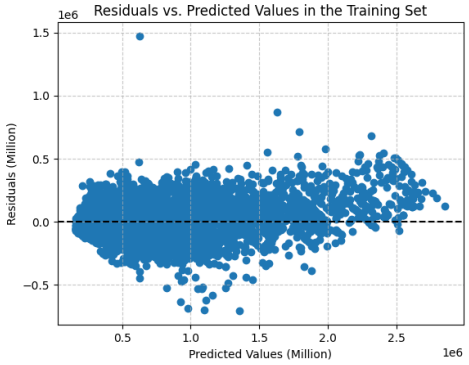
(c) DT Residuals on Test Set



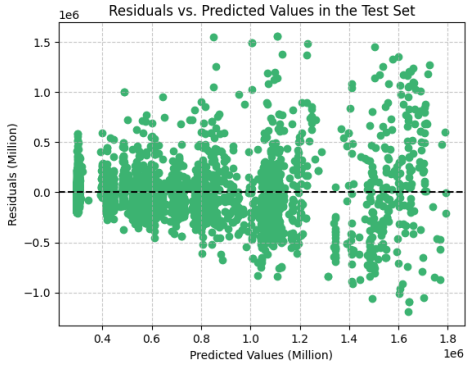
(d) DT Residuals on Training Set



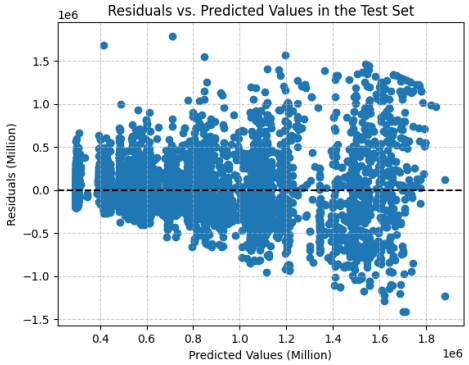
(e) RF Residuals on Test Set



(f) RF Residuals on Training Set



(g) AB Residuals on Test Set



(h) AB Residuals on Training Set

Figure 11.21: Model Residuals on Test & Training Set

11.7 Splitting the Data Frame into Training and Evaluation Set

```
from sklearn.model_selection import train_test_split

#Add the target
X = df.drop(['price'], axis=1)
y = df['price']

#Set the training set as the 75 % and the test set as\
 the 25 % of the data frame.
X_train, X_test, y_train, y_test = train_test_split\
(X, y, test_size = 0.25, random_state=42)

#Print the shape of the training and testing set
print(f"Shape␣X_train:␣{X_train.shape},␣Shape␣y_train:␣\
{y_train.shape}")
print(f"Shape␣X_test:␣{X_test.shape},␣Shape␣y_test:\
{y_test.shape}")

>>>Shape X_train: (11151, 13), Shape y_train: (11151,)
>>>Shape X_test: (3718, 13), Shape y_test: (3718,)

train_data = X_train.join(y_train)

// Code adapted from codemy.com example:
// https://www.youtube.com/watch?v=aV_sRopNTrw
//retrieved in October 2023.
// https://www.youtube.com/watch?v=zAxuIlCBvOw
// retrieved in October 2023.
```

11.8 Linear Regression Code, Training, Evaluation, and Parameter Optimization

Linear Regression training the model:

```
from sklearn.linear_model import LinearRegression
#Create a Linear Regression Model:

lr = LinearRegression(fit_intercept = True,\
 copy_X = False, n_jobs = 2649,\
 positive = False)

#Training the model on the training set
lr.fit(X_train, y_train)
```

```
y_train_predictions = lr.predict(X_train)

// Code adapted from codemy.com example:
// https://www.youtube.com/watch?v=zAxuIlCBvOw
// retrieved in October 2023.
```

Linear Regression evaluating the model on the training set:

```
from sklearn.metrics import r2_score, mean_squared_error, \
mean_absolute_error

#Evaluate the performance of the model:
lr_r2 = r2_score(y_train, y_train_predictions)
print(f"R2 score: {lr_r2}")

lr_mse = mean_squared_error\
(y_train, y_train_predictions)
print(f"MSE score: {lr_mse}")

lr_rmse = np.sqrt(lr_mse)
print(f"RMSE score: {lr_rmse}")

lr_mae = mean_absolute_error\
(y_train, y_train_predictions)
print(f"MAE score: {lr_mae}")

intercept = lr.intercept_
print(f"Intercept: {intercept}")

>>>R2 score: 0.5651421467357032
>>>MSE score: 92432280319.91563
>>>RMSE score: 304026.77566279523
>>>MAE score: 189446.3051427345
>>>Intercept: -2074867.4647747707

// Code adapted from codemy.com example:
// https://www.youtube.com/watch?v=zAxuIlCBvOw
// retrieved in October 2023.
```

Linear Regression evaluating the model on the test set:

```
from sklearn.linear_model import LinearRegression

lr.fit(X_train, y_train)
y_test_predictions = lr.predict(X_test)

#Conclusions:
from sklearn.metrics import r2_score, mean_squared_error, \
mean_absolute_error
```

```
#Evaluate the performance of the model:
lr_r2 = r2_score(y_test, y_test_predictions)
print(f"R2 score: {lr_r2}")

lr_mse = mean_squared_error(y_test, y_test_predictions)
print(f"MSE score: {lr_mse}")

lr_rmse = np.sqrt(lr_mse)
print(f"RMSE score: {lr_rmse}")

lr_mae = mean_absolute_error\
(y_test, y_test_predictions)
print(f"MAE score: {lr_mae}")

intercept = lr.intercept_
print(f"Intercept: {intercept}")

>>>R2 score: 0.588443225318364
>>>MSE score: 90200210380.41759
>>>RMSE score: 300333.498598504
>>>MAE score: 188449.3816457803
>>>Intercept: -2074867.4647747707


// Code adapted from codemy.com example:
// https://www.youtube.com/watch?v=zAxuIlCBvOw
// retrieved in October 2023.
```

Linear Regression Parameter Optimization:

```
import pandas as pd

df = pd.read_excel(r"D:\...\data.xlsx")

from sklearn.model_selection import train_test_split
#Add the target
X = df.drop(['price'], axis=1)
y = df['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, \
test_size = 0.25, random_state=42)

print(X_train.shape)

import optuna
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
#from sklearn.metrics import mean_squared_error

class ModelOptimization:
```

```
def __init__(self, X_train, X_test, y_train, y_test):
self.X_train = X_train
self.X_test = X_test
self.y_train = y_train
self.y_test = y_test

def objective(self, trial):

fit_intercept = trial.suggest_categorical\
('fit_intercept', [True, False])
copy_X = trial.suggest_categorical('copy_X', [True, False])
n_jobs = trial.suggest_int('n_jobs', 2, 5000)
positive = trial.suggest_categorical('positive', [True, False])

model = LinearRegression(fit_intercept = \
fit_intercept, copy_X = copy_X,
n_jobs = n_jobs, positive = positive)

model.fit(self.X_train, self.y_train)
return model.score(self.X_test, self.y_test)

if __name__ == '__main__':
model = ModelOptimization(X_train, X_test, y_train, y_test)
study = optuna.create_study(direction='maximize')
study.optimize(model.objective, n_trials=150)
print(f"Best parameters:{study.best_params}")
print(f"Best value:{study.best_value}")

// Code adapted from Ferneutron example:
// https://www.youtube.com/watch?v=87L5NzwQXLI&t=521s
// retrieved in December 2023.
```

11.9 Decision Tree, Random Forest, and Adaptative Boosting: Hyperparameters

DT, RF, and AB Selected Hyperparameters:

```
#Selected Hyperparameters:

#Decision Tree:
dt_regressor = DecisionTreeRegressor(criterion = "friedman_mse", \
splitter = 'best', max_depth = 420, min_samples_split = 17,
min_samples_leaf = 17, max_features = 9, random_state = 42)

#Random Forest:
rf_regressor = RandomForestRegressor(n_estimators = \
286, criterion = 'squared_error', max_depth = 17,
```

```
min_samples_split = 4, min_samples_leaf = 2, max_features = 9,
max_leaf_nodes = 4997, random_state = 42)

#Adaptative Boosting:
adaboost_regressor = AdaBoostRegressor(loss = "exponential", \
n_estimators = 465,
learning_rate = 0.006647992286399134, random_state = 42)

// Code adapted from codemy.com example:
// https://www.youtube.com/watch?v=zAxuIlCBvOw
//retrieved in October 2023.
```

DT, RF, and AB Hyperparameters Optimization Space:

```
#Hyperparameters optimization space:

#Decision Tree:
criterion = trial.suggest_categorical\
('criterion', ['squared_error', 'friedman_mse',\
 'absolute_error', 'poisson'])
splitter = trial.suggest_categorical('splitter',\
 ['best', 'random'])
max_depth = trial.suggest_int("max_depth", 2, 1000)
min_samples_split = trial.suggest_int('min_samples_split',\
 2, 20)
min_samples_leaf = trial.suggest_int('min_samples_leaf', 2, 20)
max_features = trial.suggest_int('max_features', 2, 10)
random_state = trial.suggest_int('random_state', 42, 42)

#Random Forest:
n_estimators = trial.suggest_int('n_estimators', 10, 350)
criterion = trial.suggest_categorical("criterion", \
['squared_error'])
max_depth = trial.suggest_int('max_depth', 5, 25)
min_samples_split = trial.suggest_int\
('min_samples_split', 2, 7)
min_samples_leaf = trial.suggest_int("min_samples_leaf", 2, 7)
#min_weight_fraction_leaf = trial.suggest_float\
('min_weight_fraction_leaf', 0.1, 0.99)
max_features = trial.suggest_int('max_features', 2, 15)
max_leaf_nodes = trial.suggest_int('max_leaf_nodes', 2, 10000)
#min_impurity_decrease = trial.suggest_float\
('min_impurity_decrease', 0.001, 0.99)
bootstrap = trial.suggest_categorical('bootstrap',\
 [True, True])
oob_score = trial.suggest_categorical('oob_score',\
 [True, False])
#n_jobs = trial.suggest_int('n_jobs', -1, 1)
```

```
random_state = trial.suggest_int('random_state', 42, 42)

#Adaptative Boosting:
loss = trial.suggest_categorical("loss",\
 ['linear', 'square', 'exponential'])
n_estimators = trial.suggest_int('n_estimators', 50, 10000)
learning_rate = trial.suggest_float\
('learning_rate', 0.0001, 0.02)
random_state = trial.suggest_int('random_state', 42, 42)


// Code adapted from Ferneutron example:
// https://www.youtube.com/watch?v=87L5NzwQXLI&t=521s
// retrieved in December 2023.
```

Declaration of Honor

I hereby certify that I have written this thesis independently and that I have not used any sources or tools other than those specified and that the work has not been submitted in the same or a similar form to any other examination authority in the same or a similar form.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt wurde.

Lucas G. Gonzalez Sonnenberg

Berlin, 29.01.2024

Acknowledgment

The author would like to thank Dr. Professor Winfried Pfister and Mr. Raddatz for agreeing to be thesis tutors and for having the time and patience to guide through this process. Sincere thanks to the reviewers of this thesis, Hanna Hodel, and Nicolas Vaz Ferreyra, for their comments and suggestions. Thanks to the author's parents, who always motivated him to start new challenges.