NTNU
Kunnskap for en bedre verden

Department of Computer Science

TDT4173 - Assignment 3

Supervised learning

Implementation and Comparison of prediction algorithms

Group:
Group 6

Authors:
Eivin Emil Floer (eivinef)
Brage Lytskjold (bragelyt)
Sindre Johan Imenes Sivertsen (sjsivert)

November 26, 2020

Abstract

The goal of this paper is to look at the limitations and strengths of linear regression as well as two other supervised learning prediction methods. With modifications to the dataset to make relations more linear, the performance of linear regression will be compared to the other two supervised learning methods. The three methods will be compared on how they are at predicting the housing prices in a certain area, after being trained on data of previously sold houses in this area. The paper will conclude that the decision tree model was outperformed by both logistic regression and decision tree. This indicates that even with modifications the attributes of the data set do not have a completely linear relationship with the price. The group made a web site on top of the decision tree model, in order to predict new house prices: https://house-pricing.nikzy.no/

Table of Contents

List of Figures

List of Tables

1 Introduction

This project will look into the limitations and strengths of linear regression as well as two other supervised learning prediction methods. The paper will explore if a simple method like linear regression can compete with the other supervised learning methods given proper modifications to the data set to make relations more linear. The three methods will be compared on how they are at predicting the housing prices in a certain area, after being trained on data of previously sold houses in this area.

The motivation behind this project is to study whether simpler methods can be used to solve complex problems with some restructuring of the data. Thus saving time and resources as well as lowering the skill cap required to use machine learning.

This will be achieved by using simple machine learning methods on a relatively complex data set. By applying different methods, which each work best with different types of data and analyzing how they perform, this report hopes to get a better understanding of the data set, its impact on the results, and the strengths and weaknesses of each method.

In addition to linear regression, decision trees, and logistic regression were chosen as the other two supervised learning methods. These methods were chosen because they are relatively simple, but can represent relations linear regression can not. Scikit Learn was used for the decision tree model while linear and logistic regression was modeled from scratch. Linear regression from Scikit Learn was also used to verify the accuracy of the self-made method.

As part of this analysis different supervised learning methods are used to predict the sale price, given data about the house. The data set used for this comparison contains information about houses sold in King County. The dataset contains 21 different features, including price, square feet living,floors and view. The values are strictly numerical and the data has 21 613 entries.

2 Related Work

The research of this paper does not directly build upon previous research, but previous papers were still taken into account. These papers revolve around the dataset and its fit for the paper. When looking at previous attempts at house price prediction the first thing that stood out was the complexity of the problem and the number of factors that can influence the price of a house. Banerjee and Dutta [4] discusses the complexity of the housing market and reflects on what features are most influential to the development of price. Likewise, Varma et. [7] reflects upon the complexity and difficulties within house prediction and sees the need to apply four different machine learning methods in tandem to achieve a desirable accuracy.

This further solidified our belief that using house prediction as the measurement of the methods would prove interesting as the complexity and amount of hidden factors do not overtly appeal to a specific regression model, hopefully resulting in a more impartial comparison.

3 Data

The dataset contains house sale prices for King County in the USA and was retrieved from Kaggle [5]. It includes homes sold between May 2014 and May 2015 alongside a range of attributes about the sold house. The dataset has 21613 entries, and 21 different attributes for each entry, each of which is described in Table 2, found in Appendix A.2.

In Table 3, also found in Appendix A.2, the standard deviance, mean, the minimum and maximum value of each attribute is presented, as well as the value of quantiles for the distribution for each attribute. Immediately some anomalies are apparent, for instance, the maximum number of bedrooms being 33. This specific listing had only 1620 square feet of space. These many bedrooms are most likely the result of human error, which means there is a high probability of more errors

in the data. After these findings removing outliers from the data, as described in section 3.2.1, seemed reasonable. This will at least mitigates the negative effects of such errors.

3.1 Analyzing the dataset

As preparation and quality assurance, the data set was checked for repetitions. It appeared that some house ID's were entered in the database twice, though upon further inspection this was a product of some houses being resold. As the constant features, such as size and floors, were unchanged, price and date were changed. In the data set, some of the attributes had a value of 0. In some cases, this made sense numerically, for instance in the case of the square footage of the basement being zero if the house does not have a basement. However, in the case of "yr_renovated", the attribute is zero when it has never been renovated. This is misleading as the house was not renovated 2000 years ago. The house is probably recently built and has not needed to be renovated yet. A solution to this is addressed in section 3.2.

3.1.1 Scatterplot

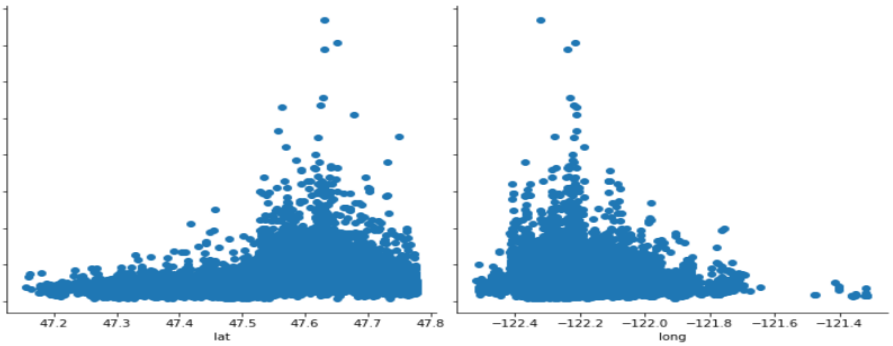


Figure 1: Scatter plots of latitude and longitude, plotted against price

One of the first things done when analyzing the dataset was to look at each feature's scatter plot when plotting against the price. This revealed that both latitude and longitude seemed to have a non-linear correlation with the price, as seen in Figure 1. This led us to set up a heat map of the price with latitude as x-coordinate and longitude as y-coordinate, seen in Figure 2.

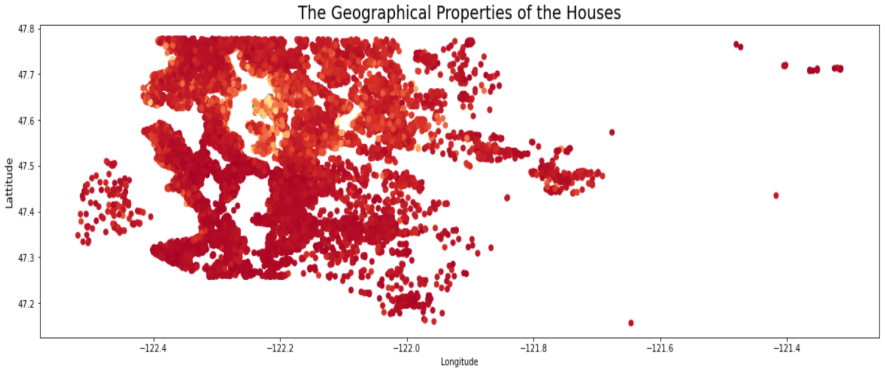


Figure 2: Geographical prices

This revealed a geographical point, where the closer a house is to this point, the higher the price. This finding is used to great advantage later in section 3.2.

3.1.2 Correlation

The correlation between each attribute and the price was plotted, as shown in Figure 3. This made it clear that both the id and date had almost no correlation with the price. The correlation between sqft_lot and the price was also seemingly negligible. This might be caused by houses in the city, as houses in the city had a higher price. The variables with the highest correlation with price are sqft_living, sqft_above, grade, sqft_living15, and bathrooms.

date not

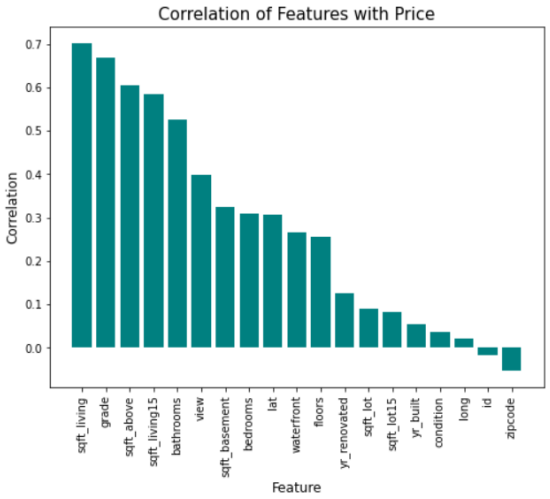


Figure 3: Correlation with price

Looking at the correlation matrix, Figure 6 found in Appendix C, most high correlations can be easily explained. sqft_living correlates highly with sqft_living15 and sqft_lot correlates highly with and sqft_lot15. This might be a result of sqft_living15 and sqft_lot15 are the average square footage of the nearest 15 houses, and houses in the same neighborhood tend to be of similar sizes. Sqft_living also has a high correlation with sqft_above as well as grade, bathroom, and bedrooms. This is a result of these being some of the variables that had the highest correlation with the price. There are also some variables with a negative correlation like for instance condition correlates negatively with yr_built, caused by the higher value denoting a newer building, which would make the condition of the house higher in most cases.

3.2 Preprocessing of the data

The discoveries in section 3.1 led to the combining of some features, intended to replace the original features. The first features that were combined were *year_built* and *year_renovated*. The two reasons behind this are that it can be argued the combination of the two features best described the state of the house, as well as the data set indicated a lack of renovation by setting the year renovated to year 0, resulting in incorrect calculations by the model. The new feature is called $year\_fixed$ and is simply the max value of year renovated and year built. This variable is meant to represent the last time work was done on the house and could be representative of how modern the interior of the house is, as well as the amount of damages from extensive wear on the house.

Next, the features *latitude* and *longitude* were combined to a new feature called *centerdistance*, which is the distance a house is from the geographical point with the highest density of expensive houses, found in section 3.1. This resulted in a new feature determining how far from this optimal location a house is, which gave a higher correlation to the price than both longitude and latitude.

It is however worth noting that this change was done to try and make the relation between position and price more linear. This change might result in a worse model for the decision tree, as some data is lost when going from two dimensions, to one.

3.2.1 Removing features

After a close inspection of the data set and each attribute's correlation with price, house id, and the date was dropped, both of which had a negligible correlation. The house id was dropped as it is a pseudo-randomly generated key with no correlation to the price of the house, and keeping it would not give us any valuable prediction data. The removal of the date was discussed, but as the data only spans one year the trends by demands given season would be cyclical and not linearly explainable. Larger trends like an increase in the house market value could be argued for, but unknown, external factors could skew the overall trend, given the relatively short time span of the data, resulting in an over-fitted model.

3.2.2 Normalization

After having determined all the attributes average, and standard deviation within each attribute was standardized. This was done by subtracting the average of the attribute and dividing by the standard deviation. The reason for this is to make the coefficient in the fitted model more interpretable, as this makes it easier to see which attributes are used the most in the prediction.

3.2.3 Removeing outliers

After the discoveries of possible human errors in section section 3.1 we decided to remove outliers in the data set. We removed all entries that had a value 3 times the standard deviation, relative to the given attribute. This removes the outer 0.3% of the listings in every feature, resulting in extremal values being removed, mitigating some of the impacts from errors in the data set.

3.3 Splitting the dataset

The data set was split into 70 % training data and 30 % test data. We tried 80 % training and 20 % testing data as a starting point but found that the coefficient of determination for the test set and the training set increased by a few percentage points by increasing the test set to 30 %. Using a test set of 40 % resulted in a lower score. These results can be seen in Table 7 in Appendix D.

4 Methods

4.1 Choice of methods

The methods in this paper were chosen because they are all regression models within supervised learning. The paper aims to take a closer look at some of the more lightweight regression models and how they compare to each other when applied to a specific problem. Furthermore, the chosen models should also achieve different results depending on the type of relationship between variables in the data set. A decision tree builds on the assumption that the data set is separable along with different values of the independent variables, linear regression builds on the assumed linear relations between the independent and dependent variables. Logistic regression was chosen as a wild card, as it is mostly used discreetly on values between 0 and 1. Multinomial logistic regression has a structure that might allow it to represent data sets with diminishing results, this might be good for a price estimation due to the law of diminishing maiginal utity [2].

The dataset was chosen as it was seen as sufficiently complex as not to overly appeal to a single method. Choosing a dataset with an e.g. stricter, more linear relation between dependant and independent variables would probably result in better predictions, though less room for studying the methods and their weaknesses.

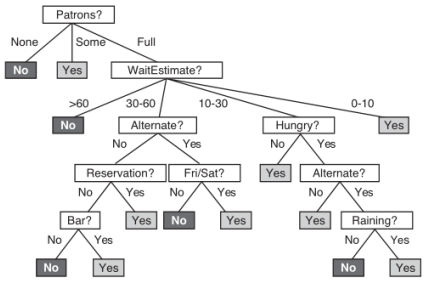
4.2 Decision Tree

For the decision tree modeling, a third party implementation from scikit-learn was used. The library used is called *sklearn.tree.DecisionTreeRegressor*. This library supports multiple algorithms for measuring the quality of splits, or the strategy used to choose the split at each node.

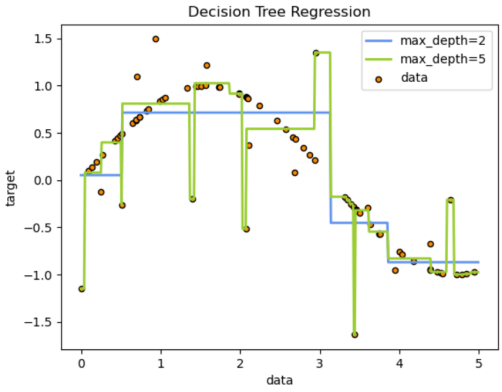
4.3 The main concepts

"Decision tree induction is one of the simplest and yet most successful forms of machine learning" Russel and Norvig [6]

A decision tree is a function that takes an input vector of attribute values and returns a single output value. The output value is determined by traversing a binary tree where each internal node in the tree corresponds to a test of the value of one of the input attributes. At the end of the traverse, a leaf node is reached, representing a decision, or an output value. An example of a simple, discreet decision tree can bee seen in Figure 4a.



(a) Descrete decision tree, Russel and Norvig [6]



(b) Decision tree regression

Figure 4: Decision tree

4.3.1 Building the decision tree

The decision tree is constructed top-down, starting with the root node. To determine the root node the model starts by observing one feature, sorted by its numerical value, and picking out a pair of adjacent observed data points. The average between these two points is used as a potential decision node, splitting the observed data into two sets. The goal of the split is to make the outputs of the two sets as different as possible from each other while keeping the deviation, or cost, in the output values low, to maximize the information gained.

To calculate the cost, the mean square error (MSE) is used. The error is calculated by finding the average of the points in both sets and determining the sum of squared distances each point has to the average. The cost, or evaluation of how good a decision node is, is the sum of errors in the subsets.

This process is repeated over all features, splitting between all adjacent data points. The decision node with the lowest cost is chosen as the first decision node, splitting the dataset into two subsets. This whole process is repeated in each of the new subsets, as well as all subsequent subsets until all subsets can not be divided into two new subsets larger than the minimum threshold.

4.3.2 Strength and weaknesses

One of the main advantages of decision trees is that it can fit any possible function given enough internal nodes, allowing the model to reflect different forms of relationships between the dependent and independent variables. As the decision tree iteratively divides the dataset into multiple, smaller subsets datasets, a dataset that is not otherwise linearly separable, can be separated.

The downside of this is that it makes the model vulnerable to overfitting, e.g. if the amount of leaf nodes approaches the number of entries in the dataset. Figure 4b shows how a too high max depth can create an overfitted decision tree. To counter this weakness a minimum threshold can be put for the size of a data subset within a node, turning nodes not able to be split into two sets larger than this threshold into leaf nodes. As the return value of a leaf node is the average of all dependent variables within its subset, the models of single outliers are reduced.

4.3.3 Hyperparameters

Table 4 in Appendix B.1. gives a summary of the tuneable hyperparameters for the decision tree model. For more in-depth reading, check out the documentation: https://scikit-learn.org/stable/modules/generated/sk

4.4 Linear and Logistic regression

Linear and logistic regression was also applied to the dataset. The method was implemented from scratch using mean squared error (MSE) as a cost function and gradient descent for optimization. As a comparator, scikit learns linear regression model was used in order to proof check the self made model. A scikit learn model for logistic regression was not implemented as the model did not support multinomial logistic regression with MSE.

4.4.1 The main concepts

The goal of both linear and logistic regression is to try and represent the relation between the dependent and independent variables as a simple function. The main difference is, as their name suggests, that linear regression tries to fit a linear function, Equation 1, and logistic regression a logistical curve, Equation 2, to the relation in the dataset. Even though the model uses different functions, the tuning of the functions is done using the same method.

$$h_{lin}(x) = \alpha + \beta x \qquad (1) \qquad\qquad h_{log}(x) = \frac{1}{1 + e^{-(\alpha+\beta x)}} \qquad (2)$$

4.4.2 Fitting the function to the dataset

When trying to optimize how well a function fits the dataset, a cost function is used as a parameter for determining the error in the model, compared to the actual findings in the dataset. The cost function used, mean squares error takes the sum of the squared distances between the estimated output value and the actual value in the model. The result is the squared offset of the function to all points, or the cost of the model. To optimize the model it is assumed that the lower the cost, the better the model.

Gradient descent was used to minimize the cost function. Gradient descent starts with a set of parameters for the regression function, Equation 1 or Equation 2, and puts them into the cost function derivative. A vector, called the gradient, along with the negative derivative of the cost function, given the current parameters, is then calculated. This gradient is applied to the original parameters and a new set of parameters is then returned, altered to give a lower cost.

Gradient descent is recursively called again with the new parameters, and will continue until the length of the gradient is below a set length ϵ. The final parameters returned will then correspond to

a local minimum for the cost function. Luckily the problem space of gradient decent for Equation 1 or Equation 2, is strictly convex, as the functions are strictly either growing or shrinking and MSE squared of these functions.

4.4.3 Hyperparameters

The last thing remaining before the implementation of the method is to have an understanding of the hyperparameters α and ϵ. α is a guide for the step length of each iteration of gradient descent, or a step size multiplier. It is a constant used to alter the coefficients by a fraction of the length of the gradient. Determining an appropriate α is important to get optimal results. If α is too high, the function will diverge, as each step done by the coefficient is too large, resulting in the new value being higher than the original one. If α is too small on the other hand the training phase is unnecessarily long, as each alteration to the coefficients is minuscule.

ϵ is the error margin and the target value for when tuning of the method should stop. It is defined as the desired length of the gradient, and is implemented by having gradient descent check the length of the gradient, only stopping if the gradient is shorter than ϵ. Here too it is important to determine an appropriate value, as having a too high ϵ results in the function stopping too early, resulting in a higher error than necessary, and a suboptimal solution. A too low ϵ on the other hand forces an unnecessarily large amount of iterations.

5 Results

5.1 Description of the experimental setup

To represent the data the library *pandas*' method *pandas.DataFrame* was used. This allowed for easy access to the different attributes, as well as quick extraction of plots and figures. DataFrame is also the preferred data structure of scikit-learn

As mentioned in section 4, Decision Tree modeling, was implemented from scikit-learn using *sklearn.tree.DecisionTreeRegressor*. Linear regression and logistic regression with gradient descent were on the other hand implemented from scratch using basic python functions. Linear regression from Scikit Learn was also implemented. This was done to verify that the self made method did not have any major errors.

Another thing to note is the fact that the price was scaled down for the logistic regression. The reason for this was because the domain of the logistic function is between 0 and 1. This scaling was done linearly, resulting in the lowest price in the dataset being zero and the most expensive being 1. As was done linearly, the operation can easily be reverted after a prediction to output the actual price. As a consequence, the model can not represent or predict output values higher than the maximum price or lower than the minimum price in the training set.

5.2 Metrics

To evaluate the different models two metrics were used: Mean absolute error (MAE) and mean squared error (MSE). Though both methods are based on the sum of offsets between predictions, $\hat{y}_i$, and actual values,y_i. They have quite different outcomes as MSE looks at the square of the offsets.

$$MSE = \frac{1}{N}\sum_{i=0}^{N}(\hat{y}_i - y_i)^2 \qquad (3) \qquad\qquad MAE = \frac{1}{N}\sum_{i=0}^{N}|\hat{y}_i - y_i| \qquad (4)$$

The main difference between the results of the two models is their relation to outliers and extremal values, as the exponential growth in cost given by MSE generates a higher pull the further away a prediction is from the actual value. The result of this is an error metric that is highly sensitive

to major errors in predictions, while MAE is the average distance of all predictions and thus less sensitive to large errors. MAE is a good description of the overall accuracy of the model, while MSE gives a better metric for how precise the model is.

5.3 Hyperparameter tuning

5.3.1 Decision tree

In order to tune the hyperparameters for the decision tree model, *GridSearchCV* was used. GridSearchCV is a function from scikit-learn's *model_selection* package. The function loops through a dictionary of predefined values for each hyperparameter and chooses the value with the best fit. The given parameter dictionary with these possible values was used as input shown under "possible values" in Table 6 in Appendix B.3. The GridSearchCV algorithm found that parameters under "final value" in gave the best results.

5.3.2 Linear and logistic regression

The hyperparameters for linear and logistic regression were manually tuned. This was done by setting a α value, running gradient descent, and printing out the length of the starting gradient vector after one iteration. As discussed in section 4.4.2, the problem space for both linear and logistic regression are convex. This means that the shorter the gradient-vector of a certain point the closer that point is to the optimal cost function. A good α should therefore result in the greatest decrease in value each iteration.

This resulted in the tuning being done by doing a "manual binary search", by looking at gradient lengths after the first step of gradient descent for different α's, the optimal α could be approximated. This was mostly a process of trial and error and would ideally be done automatically. The process was however not very time-consuming and yielded tolerable results, so implementing this was not prioritized.

ϵ was then set to a low enough value for the gradient descent to finish within a reasonable amount of time. The final values of ϵ and α can be seen in Table 5 in Appendix B.2. Something to note however is that due to the recursion limit of python at 1000, the optimal model was iteratively fitted by manually running the algorithm multiple times with lower ϵ each time.

5.4 Results

When evaluating the results a worst-case baseline was created to help contextualize the accuracies of the models. As a baseline, the average price of the training set was used as a prediction for each price in the test set. As seen in Table 1 all methods outperformed this baseline and the decision tree seems to be the most accurate. Logistic regression performed somewhat better than linear regression. The self-made linear regression performed approximately equal to the one from scikit learn. This slight difference is so small that it can be a result of chance and is negligible.

Model	MAE Score	MSE Score
Guess average price (baseline)	180503.24683608094	55912359681.44628
Linear regression (sklearn)	99377.72959950761	18178442326.462067
Linear regression (self made)	99377.7295995079	18178442326.462067
Logistic regression (self made)	94821.71714334025	16636233345.902458
Decision tree (sklearn)	79251.11452372445	13192253517.004978

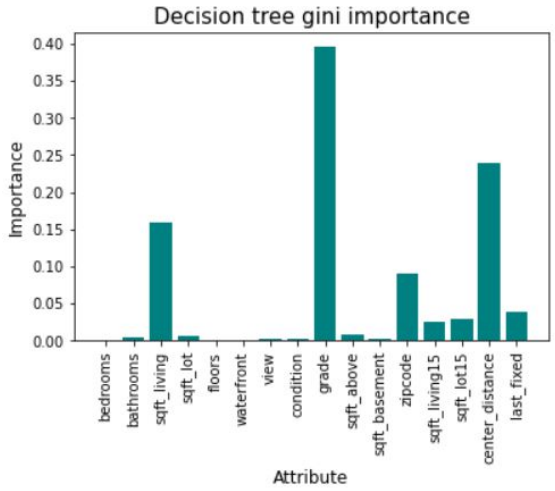
Table 1: Results of all the different models

5.4.1 Importance of features in the models

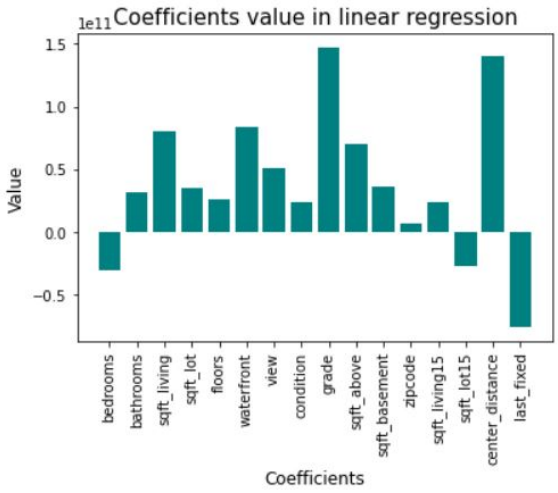
For linear and logistic regression the value of the coefficients in the trained model was used to analyze the importance of each attribute. Since the variables were normalized before training, the value of the coefficient, as a result, signifies how important a certain attribute is for the prediction model. As illustrated in Figure 5c and Figure 5b sqft_living, grade and center distance seem to be the most important variables for prediction both for linear and logistic regression.

To evaluate how important each attribute is for the prediction in the decision tree the Gini importance was used [1]. As seen in Figure 5a grade, sqft_living and center distance seem to be the most important attributes for this model as well. Figure 7 in Appendix D.3 shows how the decision tree when built using a smaller training set.

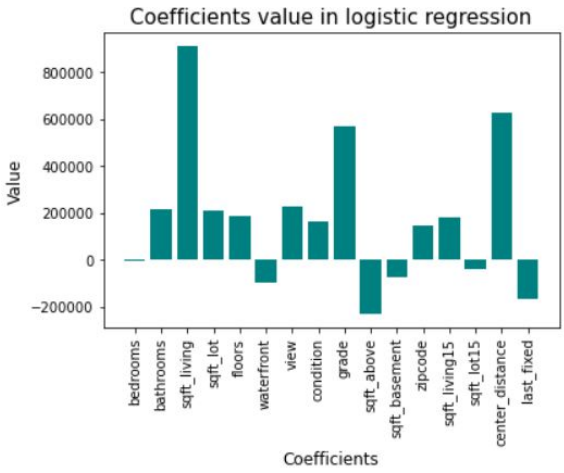
The actual decision tree is not shown as it is too large to be illustrated in a meaningful way. The tree reached a depth of 14, and it had 1015 number of leaves, as seen in Table 9 in Appendix D.4. The importance of each attribute is calculated and shown in Figure 5a. The values are the computed Gini importance [1].



(a) Decision tree Gini importance



(b) Linear regression coefficients



(c) Logistic regression coefficients

Figure 5: Importance of attributes to models prediction

5.5 Interpretation of results

In the preprocessing of the data was focused on making each attribute's correlation with price as linear as possible, by among other things introducing the variable center distance which had a much more linear relationship with price than latitude and longitude. This was done in an attempt to give linear regression the best possible chance of success. However, even with these steps linear regression still was outperformed by both logistic regression and decision trees. This could seem to indicate that the underlying assumption of linearity seemingly does not seem to hold for this particular dataset and the relation between price and the attributes is not linear in nature.

An observation to note is the fact that linear regression was outperformed by logistic regression. This could be due to the real-world phenomena of the marginal utility of a commodity, decreasing with an increase in the quantity possessed of that commodity [2] . For instance the larger the square footage of a house the less valuable is another square foot of space. This relationship with the price is better represented by a logistic function than by a linear function.

The three most important attributes for all three model predictions were sqft_living, grade, and center distance. This result aligned with what could be expected as the size, location, and condition of a house might be seen as the three most important factors when it comes to price.

Another thing to interpret from the importance of attributes is the fact that zipcode as an attribute is the fourth most important variable for the decision tree. Whereas it is of relatively low importance to the other two models. This makes sense as houses within the same zip codes tend to be similarly priced but there is no necessary relation between the number in the zip code and the price. This means that linear and logistic regression struggle with representing this link. Decision trees however have, as discussed in section 4.3.2, the ability to represent links between sets of attribute values and price.

Lastly, it is important to note that the differences in performance between the models are too close to say anything with certainty. And this interpretation of the results needs to be further investigated to say anything conclusively.

6 Conclusion

In this project, linear regression, logistic regression, and decision trees were utilized to try to predict the house prices in king county. The goal was to try to restructure the data set in such a way that the linear regression model performed as well as the other two.

To try and achieve this data was preprocessed and new variables were created to try to make correlations to price more linear. After fitting the models all models performed better than the baseline, the decision tree however proved to yield the best results based on MAE and MSE scores. The fact that linear regression did better than the baseline proves that there at least is some approximation to linear correlations in the preprocessed data. However, the fact that it was outperformed by the other two models indicates that the relation between attributes and price are better described by other non-linear models.

The goal of making this problem be solved as good as or better than the other models by the linear model was not reached. However, the linear regression method had a MAE of 99377 and the average price is 540088. Meaning that the average offset is around. 18% (99377/540088) of the average of from the price. This is still a decent prediction and not much worse than the decision tree which is around 15% (79251/540088).

The group also made a web application to predict housing prices using the models developed in this project. The application is built to React and Flask and uses the decision tree to predict housing prices. The web application is hosted here https://house-pricing.nikzy.no/

Bibliography

[1] Gini impurity (with examples). https://bambielli.com/til/2017-10-29-gini-impurity/.

[2] The law of diminishing returns. "https://www.investopedia.com/terms/l/lawofdiminishingutility.asp".

[3] Scikit learn decision tree documentation. "https://github.com/scikit-learn/scikit-learn/blob/bf24c7e3d/sklearn/metrics/_regression.py#L396".

[4] D. Banerjee and S. Dutta. Predicting the housing price direction using machine learning techniques.

[5] harlfoxem. House sales in king county, usa. https://www.kaggle.com/harlfoxem/housesalesprediction.

[6] S. J. Russel and P. Norvig. *Artificial Intelligence a modern approach third Edition*. 2010.

[7] A. Varma, A. Sarma, S. Doshi, and R. Nair. House price prediction using machine learning and neural networks.

Appendix

A Variable tables

A.1 Description

Variable	Description	Unit
id	Unique Id for house (repeated if house sold more than once)	number
date	The date the house is sold	date (string)
price	Price the house is sold for	number
bedrooms	Number of bedrooms in the house	number
bathrooms	Number of bathrooms in the house	number
sqft_living	Square footage inside the house	number
sqft_lot	Square footage of the the house lot	number
floors	Number of floors in the house	number
waterfront	Does the house have a waterfront	boolean(1/0)
view	Rating of view from the house	number(0-4)
condition	Rating of the condition of the house	number(1-5)
grade	Classification by construction quality which refers to the types of materials used and the quality of workmanship. Buildings of better quality (higher grade) cost more to build per unit of measure and command higher value	number(1-5)
sqft_above	Square feet above ground	number
sqft_basmt	Square feet below ground	number
yr_built	The year the house was built	number
yr_renovated	The year the house was last. '0' if never renovated	number
zip_code	5 digit zip code	number
lat	Latitude of house location	number
long	Longitude of house location	number
sqft_liv15	Average size of interior housing living space for the closest 15 houses	number
sqft_lot15	Average size of land lost for the closest 15 houses	number
sqft_basmt	Square feet below ground	number

Table 2: Variables, description and units

A.2 Statistics

	count	mean	std	min	25%	50%	75%	max
price	21613	540088.14	367127.196	75000	321950	450000	645000	7700000
bedrooms	21613	3.370842	0.930062	0	3	3	4	33
bathrooms	21613	2.114757	0.770163	0	1.750	2.25	2.5	8
sqft_living	21613	2079.9	918.44089	290	1427	1910	2550	13540
sqft_lot	21613	15106.97	41420.5	520	5040	7618	10688	1651000
floors	21613	1.494309	0.539989	1	1	1.5	2	3.5
waterfront	21613	0.007542	0.086517	0	0	0	0	1
view	21613	0.234303	0.766318	0	0	0	0	4
condition	21613	3.409430	0.650743	1	3	3	4	5
grade	21613	7.656873	1.175459	1	7	7	8	13
sqft_above	21613	1788.39	828.1	290	1190	1560	2210	9410
sqft_basement	21613	291.51	442.575043	0	0	0	560	4820
yr_built	21613	1971	29.37	1900	1951	1975	1997	2015
yr_renovated	21613	84.4	401.68	0	0	0	0	2015
zipcode	21613	98077.94	53.51	98001	98033	98065	98118	98200
lat	21613	47.56	0.138564	47.156	47.471	47.5718	47.678	47.7
long	21613	-122.214	0.1408	-122.52	-122.33	-122.23	-122	-121
sqft_living15	21613	1986.55	685.4	399	1490	1840	2360	6210
sqft_lot15	21613	12768.45	27304.18	651	5100	7620	10083	871200

Table 3: Variable statistics

B Hyperparameters

B.1 Decision tree hyperparameters

Parameter name	Description	Default
criterion	Function to measure the quality of splits	MSE
splitter	Strategy used to choose the split at each node	Best
max_depth	Maximum depth of the tree	None
min_samples_split	Minimum number of samples required to split and internal node	2
min_samples_leaf	Minimum number of samles required to be at a leaf node	1
min_weight_fraction_leaf	Minimum weighted fraction of the sum total weights required to be a leaf node.	0
max_features	Number of features to consider when looking for the best split	None
random_state	Randomness of the estimator	None
max_leaf_nodes	Grow nodes in best-first fashion	None
min_impurity_decrease	A node will be split if this split induces a decrease of impurity greater than or equal	0.0
ccp_alpha	Complexity paramater used for minimal cost-complexity pruning	0.0

Table 4: Hyperparameters for the decision tree model

B.2 Hyperparameters for linear and logistic regression

Method	α	ϵ
Linear regression	0.00003	0.000001
Logistic regression	0.00004	0.3

Table 5: Hyperparameters for the linear and logistic regression

B.3 Values for hyperparameters used

Parameter name	Possible values	final value
min_samples_split	[10, 20, 40]	20
max_depth	[1, 2, ..., 20]	14
min_samples_leaf	[1, 5, 10, 20, 50, 100]	10

Table 6: Hyperparameters for the decision tree model

C Correlation Matrix

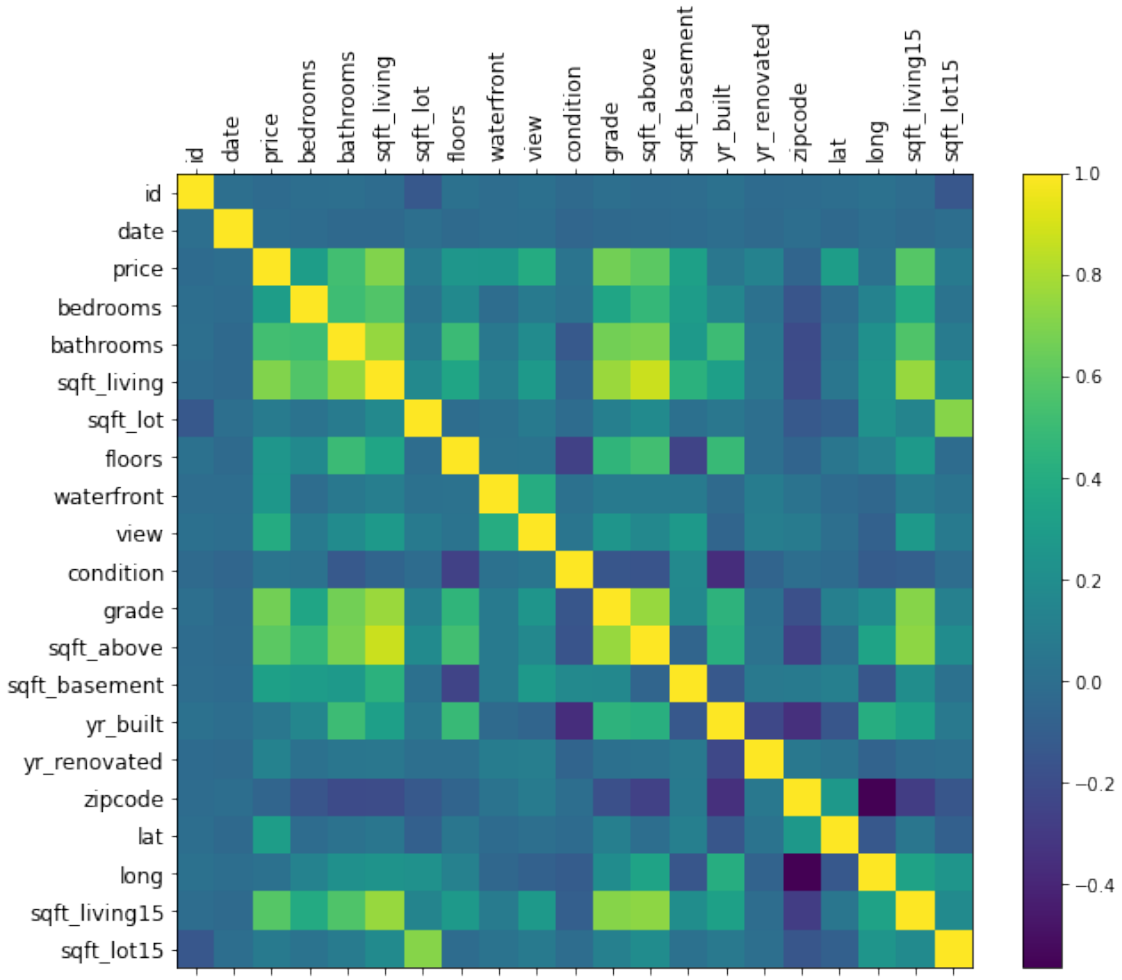


Figure 6: Correlation matrix with new attributes

D Decision tree accuracy given training-testing split

% of dataset	procedure	R^2 score
80%	Training	0.8515253048394
20%	Testing	0.7748776059995517
70%	Training	0.8472397475704189
30%	Testing	0.80502757
60%	Training	0.8385862034279422
40%	Testing	0.7541815675079168

Table 7: R^2 scores given the split ratio of the data set

D.1 Calculating R-squared

Scikit learn uses R-squared as the default metric for many of their models [3]. The R-squared is a statistical measure for how close the data points are fitted to the regression line. The definition of R-squared is

$$R^2 = 1 - \frac{Explained\_variance}{Total\_variance} \tag{5}$$

The explained variance is one minus the variance over the residual divided by the price variance

$$Explained\_variance\_score = 1 - \frac{Var[\hat{y} - y]}{Var[y]} \tag{6}$$

The variable y is the price from the test set and $\hat{y}$ is the predicted value.

D.2 Feature importance in the decision tree

Attribute	Importance
'bedrooms'	8.55538782966807e-05
'bathrooms'	0.0023443507014980023
'sqft_living'	0.2695266789643382
'sqft_lot'	0.008880529542567153
'floors'	0.0010189197775407463
'waterfront'	0.020019702883619685
'view'	0.036717679760204416
'condition'	0.0013909826945257813
'grade'	0.3960731655065039
'sqft_above'	0.010238096854995148
'sqft_basement'	0.0016268056946709666
'zipcode'	0.043321857015085036
'sqft_living15'	0.01734512855322961
'sqft_lot15'	0.018065124453830107
'center$_d istance'$	0.15835386745176142
'last_fixed'	0.014991556267333114]

Table 8: Feature importance of the decision tree

Sort table

D.3 Decision tree built on small trainingset

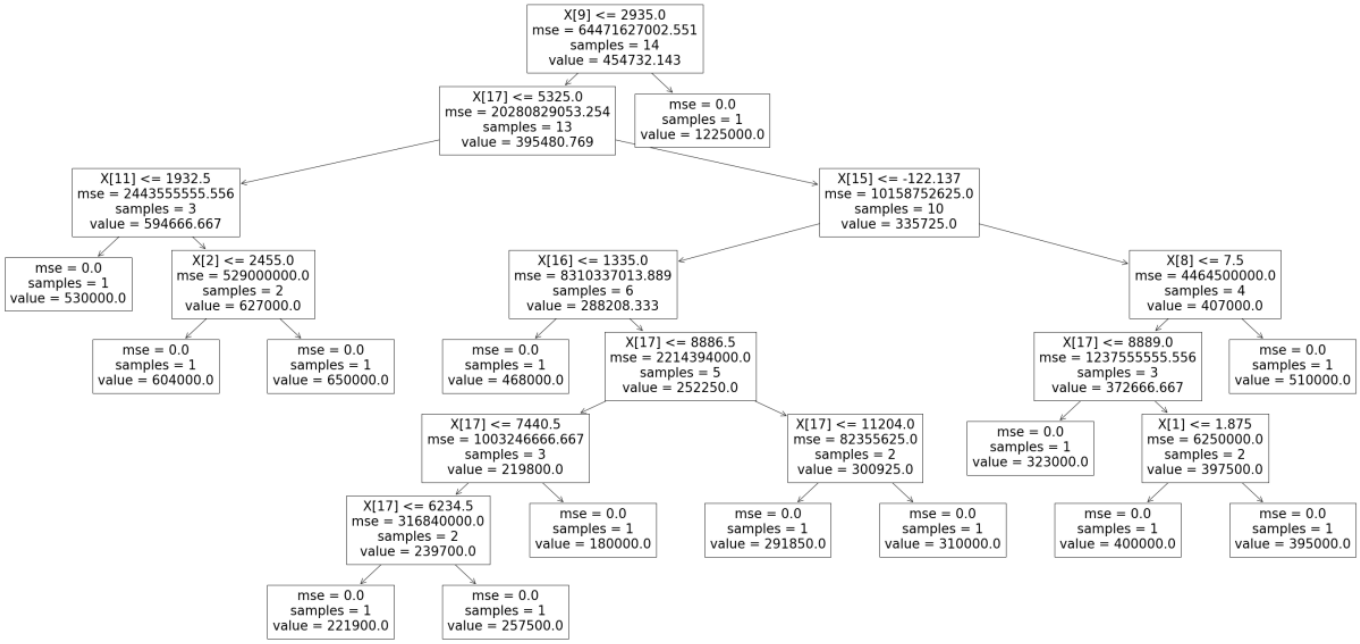


Figure 7: Illustraded decision tree with a reduces training set.

D.4 Depth of decision tree

Tree depth	14
Tree number of leaves	1015

Table 9: Depth and number of leaves of decision tree

D.5 Functions

$$Var[y] = \frac{1}{N}\sum_{i=0}^{N}(y_i - \bar{y})^2 \tag{7}$$

$$\bar{y} = \frac{1}{N}\sum_{i=0}^{N} y_i \tag{8}$$

$$Var[\hat{y} - y] = \frac{1}{N}\sum_{i=0}^{N}(\hat{y}_i - y_i - E[\hat{y} - y])^2 \tag{9}$$

$$E[\hat{y} - y] = \frac{1}{N}\sum_{i=0}^{N}\hat{y}_i - y_i \tag{10}$$