THE CATHOLIC UNIVERSITY OF AMERICA

House Prices Prediction

Rafael A. Moreno Contreras
Introduction to Data Science and Python DA501
Catholic University of America

Introduction

This project was based on a dataset from the Kaggle competition: "House Prices: Advanced Regression Techniques," where participants are encouraged to predict the final price of each home on the basis of over 79 base predictors.

The overall idea of this project is to apply exploratory data analysis, feature engineering, and data preparation to better fit the data set into a linear regression model and a decision tree model, or one of their variations, to ultimately predict the final price of each home, and to test the performance of each prediction.

Problem Statement

As mentioned before, the goal is to predict the house prices, and to archive this, linear regression and decision tree models have to be deployed.

Deploying the models is fairly simple as there are scikit learn packages specially designed for this purpose. The real challenge is to clean and fit the data in a way that not only makes sense for the deployment of the models, but that also makes sense in an overall context of what house features truly mean and how they affect how houses are priced.

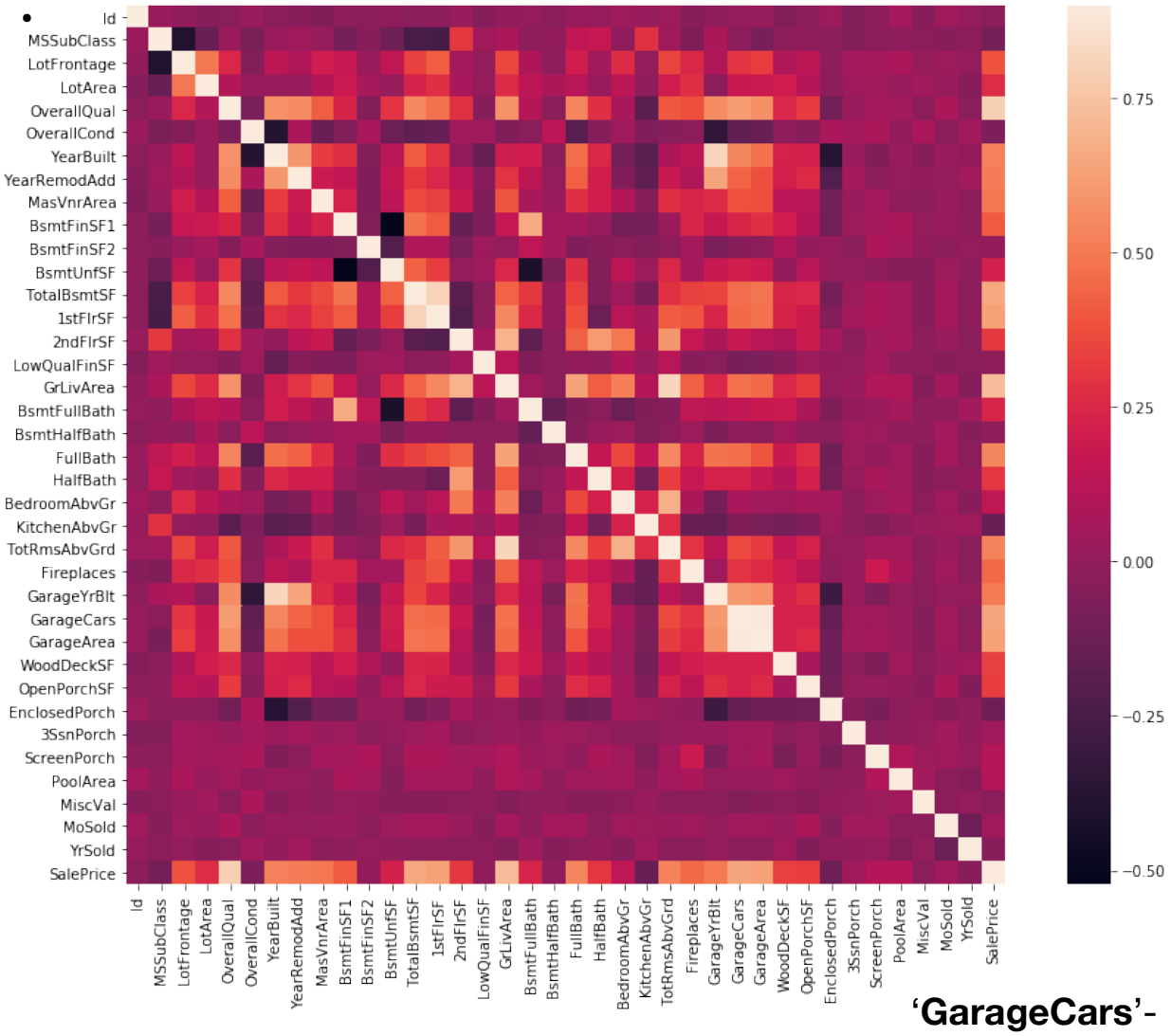
Dataset

Unlike the Kaggle competition which provides the Train and Test data set already split, we count the data set data.csv as a whole, which is meant to be split later into data_train.csv and data_test.csv.

The data set “data.csv” features the target variable ‘SalePrice’ with no missing values, which will be use later on to test the performance of the predictions, comparing the actual price with the predicted price.

Since those are the main differences between the project data set and the competition data set, the data description file from the competition can be used to better understand the contents of all variables. Some significant variables that should be mentioned are:

- ‘**SalePrice**’- This is the target variable that contains the sale price of the property in dollars.
- ‘**OverallQual**’- This variable rates the overall material and finish quality of the house from 1 to 10, with 1 corresponding to ‘Very Poor’ and 10 to ‘Very Excellent.’ This categorical variable is expressed quantitatively.
- ‘**OverallCond**’- This metrics rates the overall condition of the property. Like, ‘OverallQual’ it has rankings from 1 to 10 with the same classes.
- ‘**YearBuilt**’- This variable provides the year when the original construction of the property was done.
- ‘**GrLivArea**’- This is one of the many area-related parameters available. This one specifically indicates the above ground (grade) living area (in square feet) available in the house.
- ‘**TotalBsmtSF**’- As the name suggests, this metric contains the total area of the basement in square feet.
- ‘**1stFlrSF**’ and ‘**2ndFlrSF**’- These variables represent the areas of the first and the second floor (in square feet) respectively.
- ‘**GarageArea**’- This variable represents the size of garage in square feet.



- **'GarageCars'**- This column contains the size of the garage in terms of car capacity. Therefore all values in this metric should be integers.

- **'TotRmsAbvGrd'**- This parameter contains the total number of rooms above ground. The data description file clearly indicates that this does not take into account the bathrooms.

There are several other metrics important as well that indicate the presence and quality of different features in the property. The data dictionary can provide more insights in how to investigate and use them.

Finally it's important to mention that he data.csv file has 1460 rows and 81 columns, from which 3 are float64, 35 are int64, and 43 are object.

Data Analysis

Splitting the data

The first step was to split the data randomly on a 80 / 20 percent ratio, aligning the 80% to the Train data set exporting it on a csv file as data_train.csv and the remaining 20% to the Test data set exporting it as data_test.csv. This split is necessary to adjust the values on the train set un such way that the prediction would be more accurate while testing it with the test set.

Exploring the data

A true understanding of the behavior and correlation among the variables and the target variable is crucial to deal with them in a proper manor. A good place to start is with a correlation matrix of the train data-set, shown below.

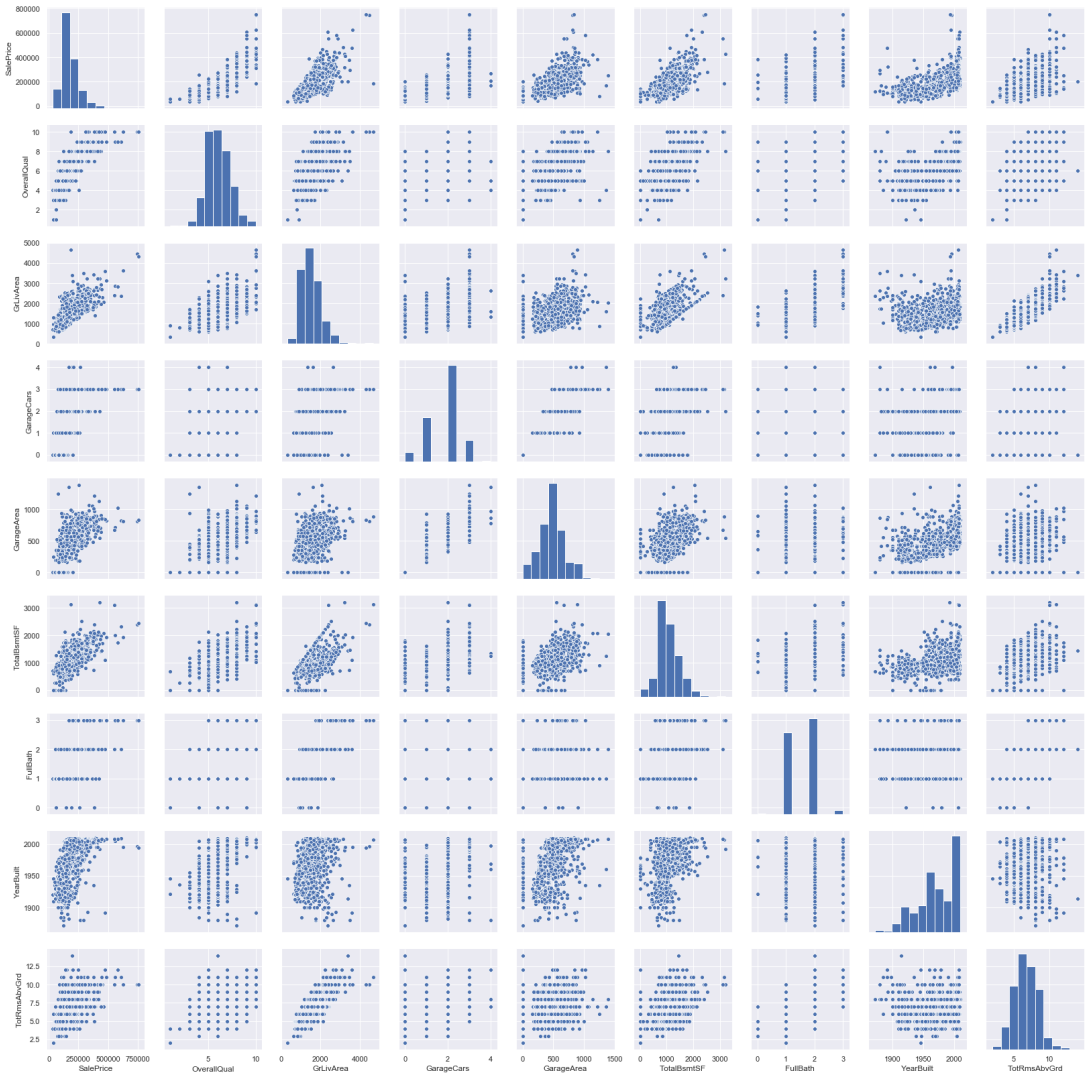
Some basic takeaways from the correlation matrix are:

- All lighter shade squares on the heat map indicate high correlation between the corresponding variables.
- ‘GarageCars’ and ‘GarageArea’ show high correlation among each other. This might be due to the fact that the bigger the garage, more cars can fit on it. Furthermore, both seem to have a similar (and relatively high) correlation with ‘SalePrice’. This shows a possible case of multicollinearity.

- ‘TotalBsmtSF’ and ‘1stFlrSF’ show high correlation among each other so as ‘SalePrice,’ again indicating multicollinearity. But looking out closely the correlation between ‘TotalBsmtSF’ and ‘SalePrice,’ there is almost a white square indicating high correlation. This means that ‘TotalBsmtSF’ should be looked with special attention.

- ‘YearBuilt’ seems to have a decent (around 0.5) correlation with ‘SalePrice’. Which seems quite logical, since the year when the house was built indicates how old the house is.

- ‘OverallQual’, ‘GrLivArea’, ‘FullBath’ and ‘TotRmsAbvGrd’, are other variables that seem to have good correlation with ‘SalePrice’ therefore should be taken in count further on.

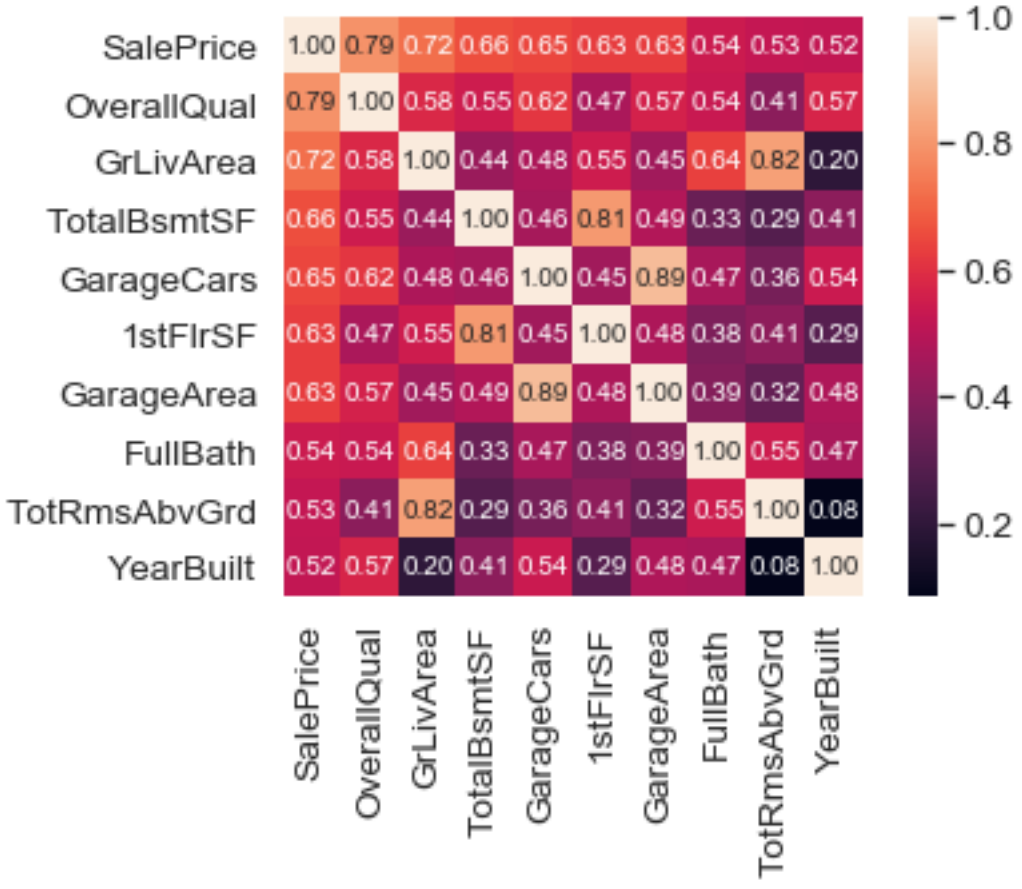


A paired scatter-plot among some of these selected variables could show more in depth trends and correlation between each other and ‘SalePrice.’

From the above plot we can infer that:

- ‘GrLivArea’ and ‘TotRmsAbvGrd’ show high linear relationship.
- ‘GrLivArea’ and ‘TotalBsmtSF’ show a linear relationship with almost a boundary defining the plot. This basically indicates that ‘GrLivArea’ sets the higher limit for the ‘TotalBsmtSF’ (Basement area). Not many houses will have basements larger than the ground floor living area.
- ‘SalePrice’ shows almost a steep increase with ‘YearBuilt’, basically indicating that prices increase (almost exponentially) as the houses decrease in age. Most recent houses seem to be highly priced.

Now that the key variables are located, a more focused correlation matrix can provide a ver overview of the interaction among them.

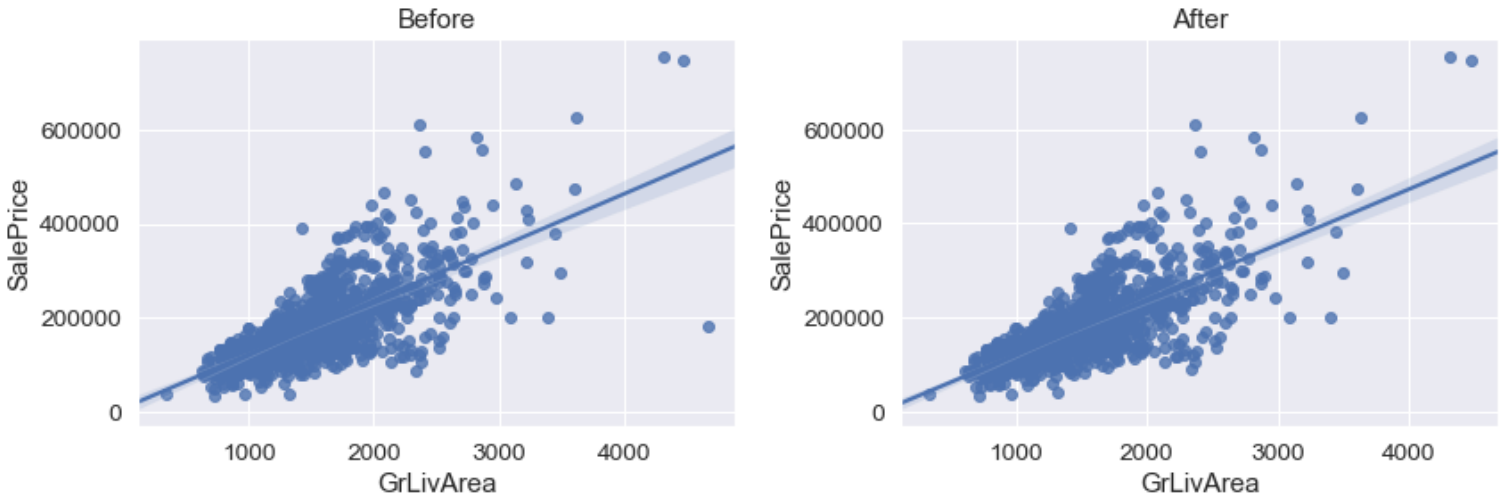


Dealing with outliers

Focusing on the variables that cause the most variance of the target variable ‘SalePrice’, seems logical to examine them as well. The idea is that they should not be influenced by outliers that could ultimately affect the final prediction.

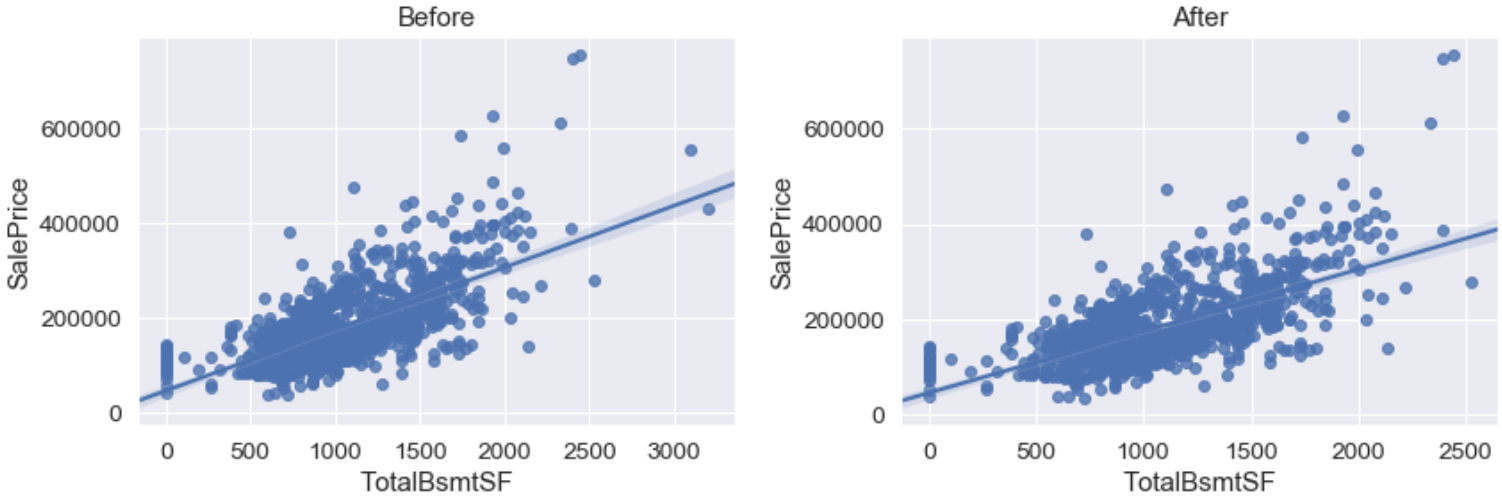
An heuristic method of treating the outliers could be just plotting these variables vs ‘SalesPrice’, in order to find those extreme values. This can be done through a visualization, and then later on they can be removed and analyzed again through a plot.

‘**GrLivArea**' vs '**SalePrice**'



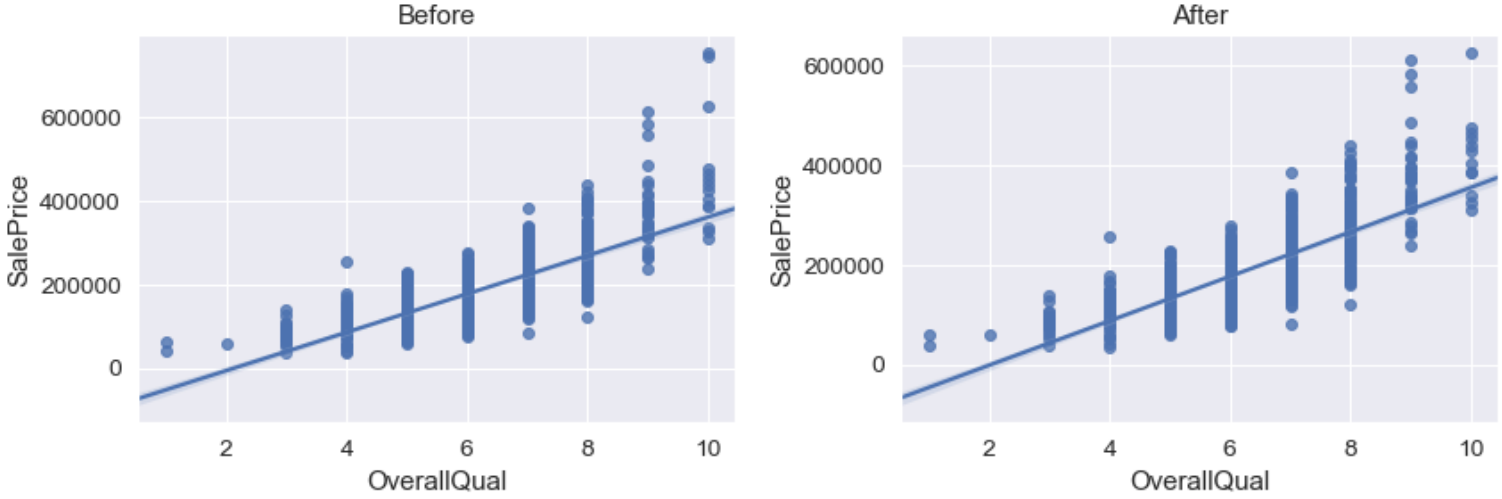
Here it seems to be only one outlier which can be removed by removing the points at ‘GrLivArea’ greater than 4,000 sq. ft. and ‘SalePrice’ less than 300,000 USD.

‘**TotalBsmtSF**' vs '**SalePrice**'



In this case it seems to be 2 outliers which can be removed by removing the points at ‘TotalBsmtSF’ greater than 3,000 sq. ft.

’**OverallQual**' vs '**SalePrice**'

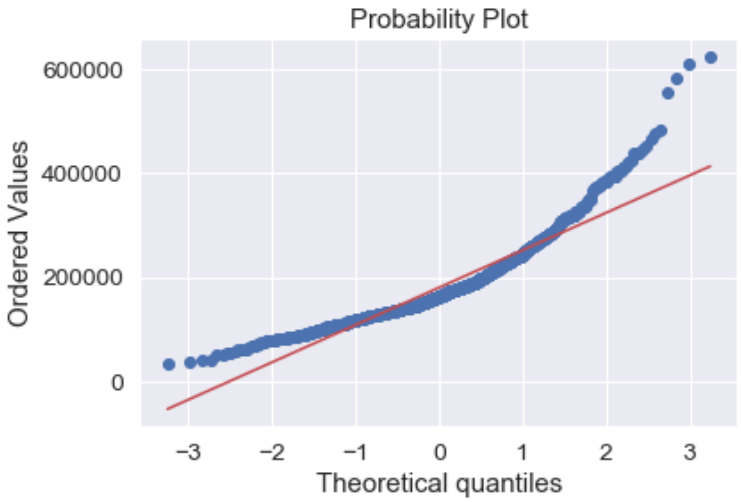


Lastly it seems to be 2 outliers which can be removed by removing the points at ‘OverallQual greater than 9 & at ‘SalePrice’ greater than 700000

Normalizing the Target Variable

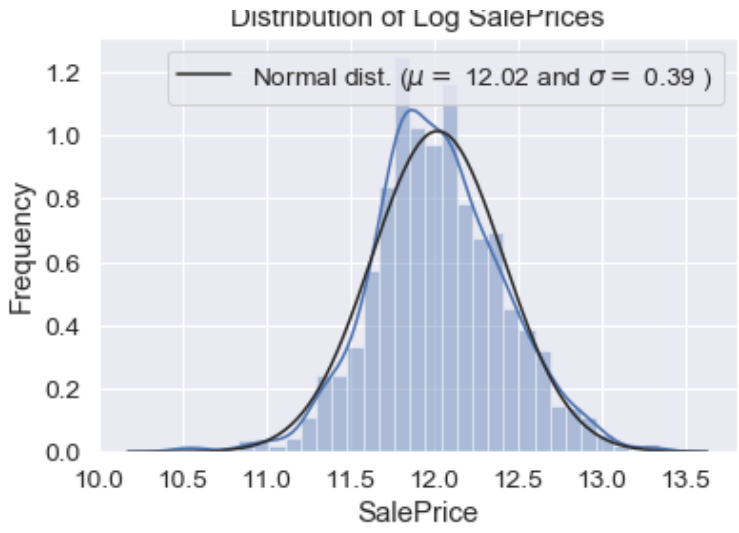
Finding the target variable 'SalePrice' distribution can open options on how to deal with it, in case it is not normally distributed, since this would be the distribution that will suit to the models in the best possible manner.

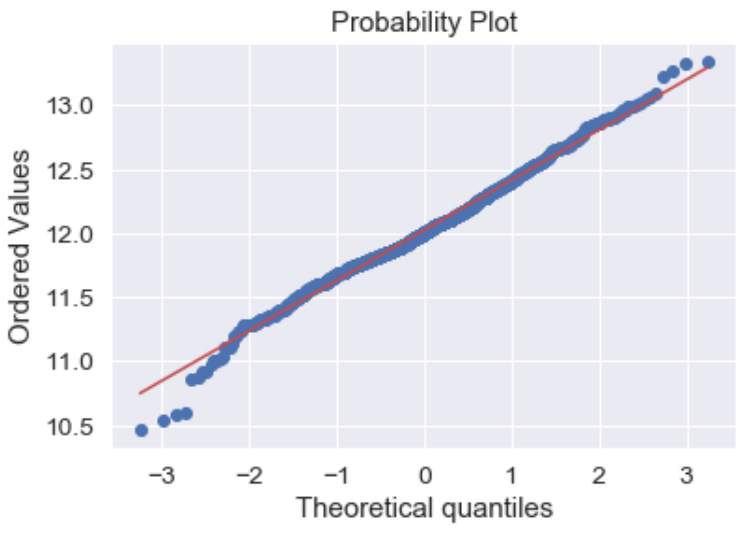
One way to find the distribution is to plot a histogram, alongside with its probability plot.



The distribution from the histogram plot doesn't seem to be normal since it has a right skewness. This can also be appreciated on the probability plot since the values don't seem to follow the linear trend.

When it comes to right skewness, a logarithmic transformation is usually a good option. Applying a log function on these values should give a linear trend and convert the set of values into 'normally distributed' values.





As we see here, the distribution now looks normal and a linear trend appears to be ore clear in the probability plot.

Merging the data sets, and dealing with missing values

Now that the Train data set seems to be in line with what better suits the models, the next step is treating the missing values and those that make little or no sense for the prediction model.

Missing values in the dataset, especially in the training dataset, can deeply affect the model performance. There are different options when it comes to deal with missing values and we will see some of them latter-on. The bellow tables represent the variables that have missing values in the dataset and their rate respectively.

	Train	Test
LotFrontage	213	45
Alley	1089	275
MasVnrType	7	1
MasVnrArea	7	1
BsmtQual	30	7
BsmtCond	30	7
BsmtExposure	31	7
BsmtFinType1	30	7
BsmtFinType2	30	8
Electrical	1	0
FireplaceQu	547	143
GarageType	64	17
GarageYrBlt	64	17
GarageFinish	64	17
GarageQual	64	17
GarageCond	64	17
PoolQC	1160	289
Fence	932	243
MiscFeature	1122	279

	Missing Ratio
PoolQC	99.587629
MiscFeature	96.288660
Alley	93.745704
Fence	80.756014
FireplaceQu	47.422680
LotFrontage	17.731959
GarageYrBlt	5.567010
GarageType	5.567010
GarageFinish	5.567010
GarageQual	5.567010
GarageCond	5.567010
BsmtFinType2	2.611684
BsmtExposure	2.611684
BsmtFinType1	2.542955
BsmtCond	2.542955
BsmtQual	2.542955
MasVnrArea	0.549828
MasVnrType	0.549828
Electrical	0.068729

Variables like ‘Alley’, ‘Fence’, and ‘PoolQC’ among many others indicate the presence or absence of features in the house. The missing values basically mean that these features are not present in the house being spoken about here.

After this point, it’s useful to locate unique values like: nan, ‘Shed', ‘Othr', ‘Gar2', ‘TenC’, so as checking for null values on every column, which in this case, the amount is zero.

Treating categorical variables

To execute a linear regression model, and a decision tree regression, all values must be numerical, so those categorical values that are not expressed on numerical terms, must be converted to numerical so the model can be deployed.

There is a function on pandas that helps on this type on situation, it is called get_dummies and what it does is to convert categorical variables into dummy/indicator variables.

Modeling

Splitting the data

Now the data fits the requirements to be modeled, it should be once again split the into train and test. It’s at this point when the ‘SalePrice’ target or dependent variable should be dropped from the test set, since it’s the value that will be predicted through the models.

Here, the four parameters that will fit into the model are defined as: X_train, X_test, y_train, y_test. The training set is what the model is trained on, and the test set is used to see how well that model performs on unseen data.

Modeling

The actual model is just a few lines of code, from scikit-learn where the LinearRegression() package will do the work for the linear regression model so as the DecisionTreeRegressor() for the decision tree regression model.

Once the partitioned data set is fitted into these packages, the model is deployed and the predictions are available to be further tested and analyzed, not before the logarithmic transformation is reversed by an exponential one.

Results

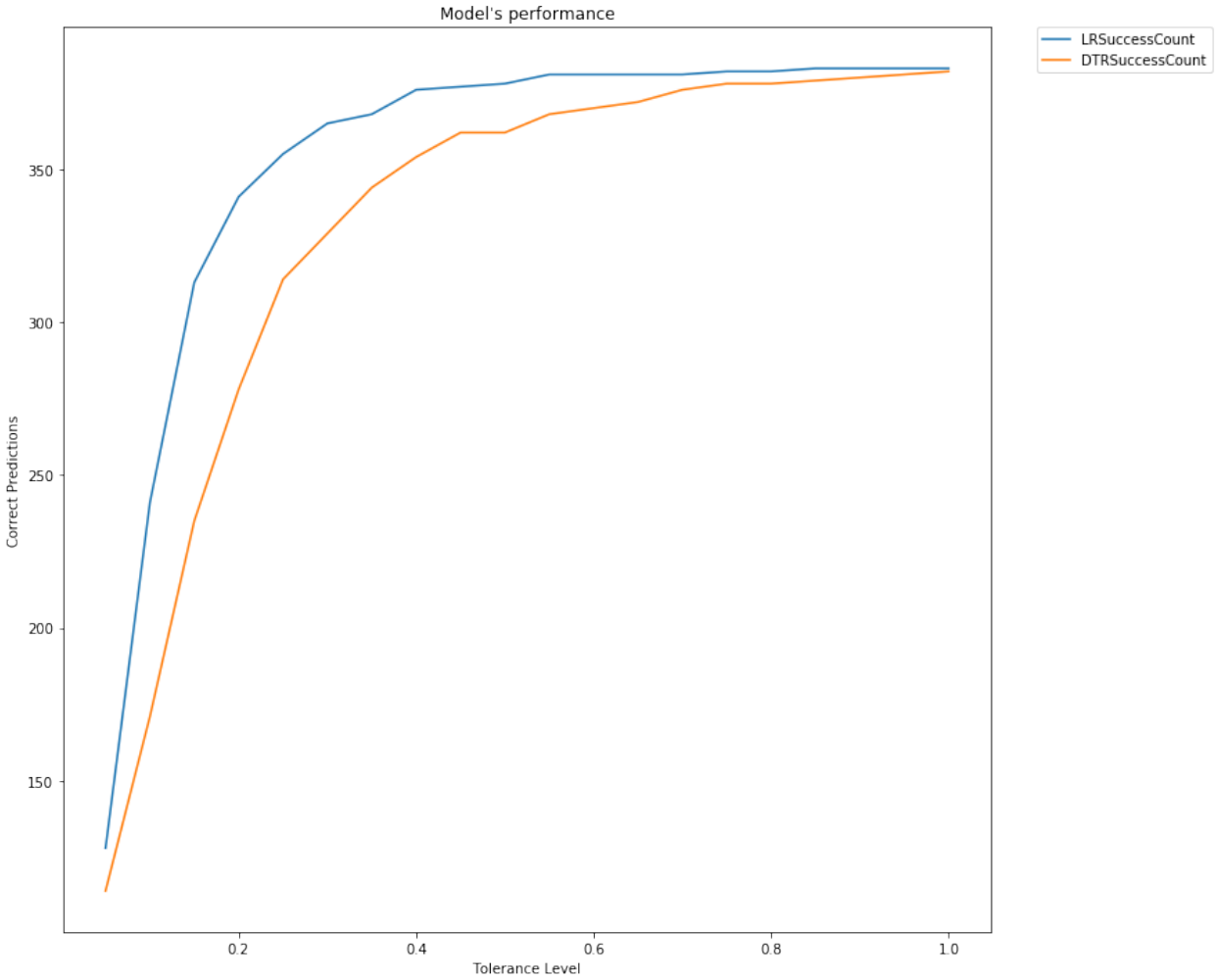
The predicted price for both models and the real price were all exported into a csv file named data_results.csv to be compared and measured in terms of a tolerance, to find out how accurate the prediction model was.

While examining the summation of errors for each model, in other words, the difference between the real price and the predicted price of each model, there seems to be a considerable amount of errors that should be taken into consideration while predicting the prices.

The summation of errors for the linear regression are: 144134.686, while the summation of errors for the decision tree regression are: 149532.544 The difference between these two is noticeable enough 5397.858, to indicate the higher accuracy of the linear regression model.

Now when the performance is shown in terms of a tolerance value which goes from 0.05 to 1, and a success count based on the value of the tolerance, it’s possible to see a clear picture.

Both models seem to be quite able to predict the price with a tolerance value higher than 0.4, and more epscificaly in the case of the linear regression when the tolerance value is higher than 0.3, as shown in the graphic below.



Recommendations

The time spent analyzing and understanding the data is time well spent, since the model can only do so much, but a well cleaned data set can make the difference in terms of performance and reliability of the predictions.

Diving deeper into feature engineering can lead to a better understanding of the overall dataset, and the ability to come with better combinations of the variables that can lead to more useful insights. This can improve the accuracy and performance of the model and therefore the predictions.

On tope of the tolerance and summation of errors, another recommended way to test the models is the one used on the Kaggle competition which is the R^2 score, which in this case is of 0.8616 for the linear regression and the R^2 score for the decision tree regression is 0.6722 which goes in line with the performance graph shown bellow.