Capstone 1 - House Price Prediction

by Jose Miguel Montoro Costela

Problem statement:

Introduction

The main goal of this project is to create a model that predicts the price of a house, given a set of features of the property.

Background and significance

The real estate industry is one of the largest in the US, and property prices one of the main pieces of information that actors in the space need to consider. Big real estate companies offer some version of this prediction. Zillow has the Zestimate, based on property prices in the neighborhood and expected growth. Valuation companies, like CoreLogic, offer market and property insight as their main value proposition.

However, there are multiple factors that affect the final asking price of a property. Location is not the only one, and in this dataset, I'll be able to take into consideration multiple observations. The most common information is included, like size, neighborhood, year built and renovations made. But other property factors, like materials used, proximity to amenities, or type of road access, are also part of the dataset and can be explored for their predictive power.

The value of this research is to bring all available property information into the model and investigate which features have a stronger predictive value. When a house is for sale, buyers and sellers will be able to pay attention to those features in order to better understand the price and adjust their bid accordingly.

Preliminary Suppositions and Implications

The main hypothesis of the study is that we can predict selling price from house features with a high level of accuracy.

From the historical data exploration and feature engineering, we'll be able to build a model that takes into account the features that are most relevant for prediction, and leaves out the ones that are not. In other words, the hypothesis is that some of the features will be good predictors (for example, "Neighborhood" or "Year built") and others won't (like "Roof material" or "Height of basement" could be).

This will offer help for buyers on which property features to examine when buying a house, and it's a tool for negotiation. They'll be able to compare the asking price with the price predicted by the model, and understand if the asking price is "fair". Also, it will help them see if the actual house features are underpriced, overpriced or fairly priced. All this information will be very valuable in the bidding and negotiation part of the house buying process.

Conclusion

This is an exciting project in the dynamic real estate market. It will include data cleaning and preparation, encoding, feature engineering and exploration, and the implementation and evaluation of several regression models.

The result will be a model that can be used in real applications to predict the selling price of a house, given a certain set of features. This will help customers make decisions in the house buying process, and help any actors involved understand the real estate market better.

Data Wrangling

Project Review

This project uses the data from a Kaggle competition. The goal is to predict the value of a house, given a set of features of that property.

Overview

In general, this Capstone Project doesn't require too much cleaning and data wrangling, as it's part of a Kaggle competition and the data is already clean and organized.

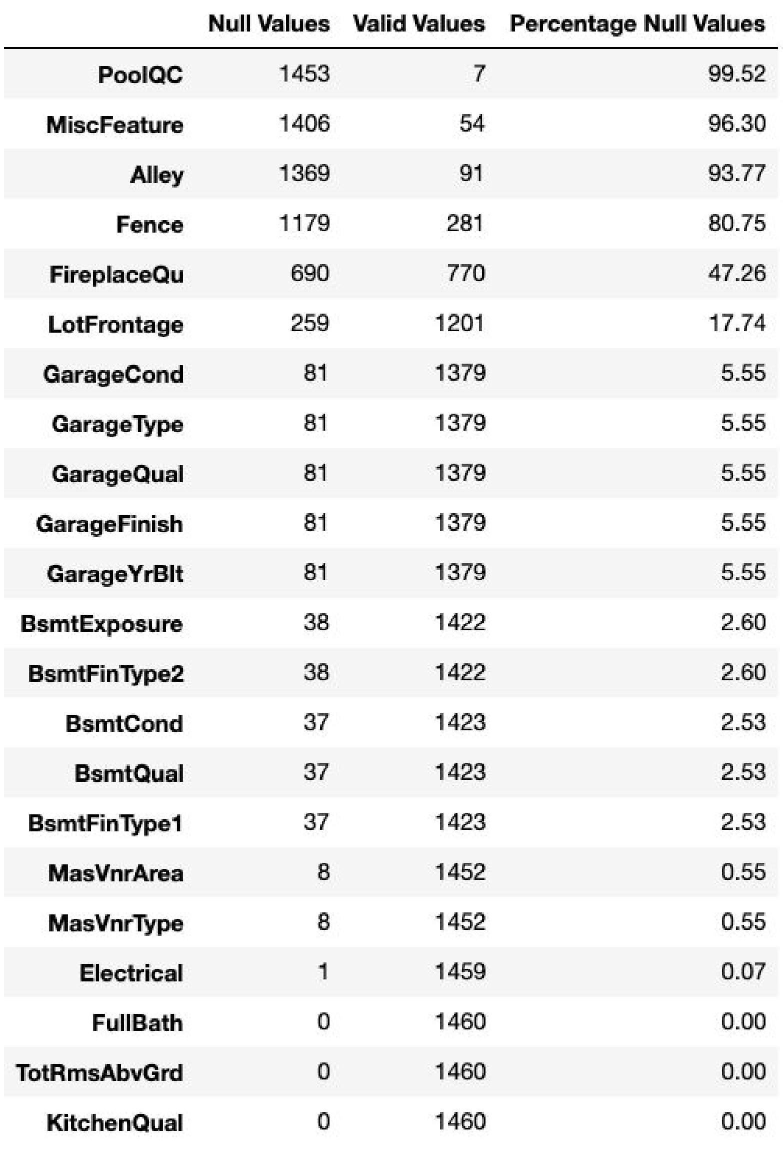
However, some steps are still necessary. In particular, investigating missing values, looking at potential outliers, and doing a first general description of the data.

Missing Values

Data is available in the related Notebook in this same folder.

For a first review, I created a new DF with the total number of null values per column, the total valid values, and the percentage of null values from the total observations.

See the dataframe below. 'Null Values' is a count of NaNs, 'Valid Values' is a count of not null values, and percentage is calculated comparing null values with total count.



	Null Values	Valid Values	Percentage Null Values
PoolQC	1453	7	99.52
MiscFeature	1406	54	96.30
Alley	1369	91	93.77
Fence	1179	281	80.75
FireplaceQu	690	770	47.26
LotFrontage	259	1201	17.74
GarageCond	81	1379	5.55
GarageType	81	1379	5.55
GarageQual	81	1379	5.55
GarageFinish	81	1379	5.55
GarageYrBlt	81	1379	5.55
BsmtExposure	38	1422	2.60
BsmtFinType2	38	1422	2.60
BsmtCond	37	1423	2.53
BsmtQual	37	1423	2.53
BsmtFinType1	37	1423	2.53
MasVnrArea	8	1452	0.55
MasVnrType	8	1452	0.55
Electrical	1	1459	0.07
FullBath	0	1460	0.00
TotRmsAbvGrd	0	1460	0.00
KitchenQual	0	1460	0.00

Then, I looked at each of the columns to see which type of NaN we have. In some of the columns, a NaN is actually a meaningful observation - mostly the absence of the observed feature. For example, in the ‘PoolCQ’ column (‘Pool quality’), the NaN values denote the absence of a pool whatsoever. This is also confirmed by the ‘data_description.txt’ file that comes with the dataset. For this same column, the possible values are

Ex	*Excellent*
Gd	*Good*
TA	*Average/Typical*
Fa	*Fair*
NA	*No Pool*

Since NaN values are not computable, I transformed all null values to either a string, if they are part of an ‘object’ column, or to a 0 if they’re numerical. In the 3 following cases, however, the NaN values were inputted with the most common value for that variable.

‘True’ Missing Values

These are missing observations in the dataset - values that are expected but they didn’t appear. We can assume these are MCAR types of missing values.

They are not very frequent, here are the cases:
Electrical - 1 missing value - index 1379
BsmtExposure - 1 missing value - index 948
BsmtFinType2 - 1 missing value - index 332

For these 3 cases, I used imputation to fill the observations with the most common value for that variable. It can be seen in the notebook, cell 211. Here are the most common values for each variable that were imputed.

Electrical - 'SBrkr' (Standard)
BsmtExposure - ‘Av’ (Average)
BsmtFinType2 -‘Unf' (Unfinished)

Converting to categorical values

Convert NaN values to categorical text values ‘None’. The columns affected are:
PoolQC
MiscFeature
Alley
Fence
FireplaceQu
GarageCond
GarageType
GarageQual
GarageFinish
GarageYrBlt
MasVnrType
BsmtExposure
BsmtFinType2
BsmtCond
BsmtQual
BsmtFinType1

Converting to numerical values 0

Some other NaN values should be converted to 0, as they are part of numerical observations. For example, the NaN values in the column LotFrontage ('Linear feet of street connected to property') are equivalent to 0 feet of street connected to property.
Here are the columns with NaN values converted to 0:
LotFrontage
MasVnrArea

At the end of all these cleaning steps, there's a new dataframe ready to be consumed with the name 'filled'.

Other notes on data

By exploring all the unique values for the object dtype columns, looks like no typos are present in the categorical values.

The Year remodeled variable is the same as the YearBuilt in 764 cases. That means in 764 houses there was no remodeling done after buying.

Description / Exploration

An additional step was conducted, to do a first exploring pass where we can compare basic statistics like mean, median, max and min values, etc.

This was done using the pandas .describe() method, as well as the pandas profiling package.

Only the relevant columns were included in the .describe() method. For example, the ID column or the MSZoning column were excluded, because they're categorical even though they're numerical.

It all looks good using the describe() method. No apparent outliers come up from the basic statistical methods performed.

Likewise, no actionable warnings come up from the results of the profiling package.

Notes for next steps

Further EDA will be done in the next submission of the capstone project. Some ideas to explore are the following:

- Explore the possibility of aggregating all binary features in one column (Garage/No Garage, Pool/No Pool, Alley/No Alley…) to reduce variables.

Data Visualization

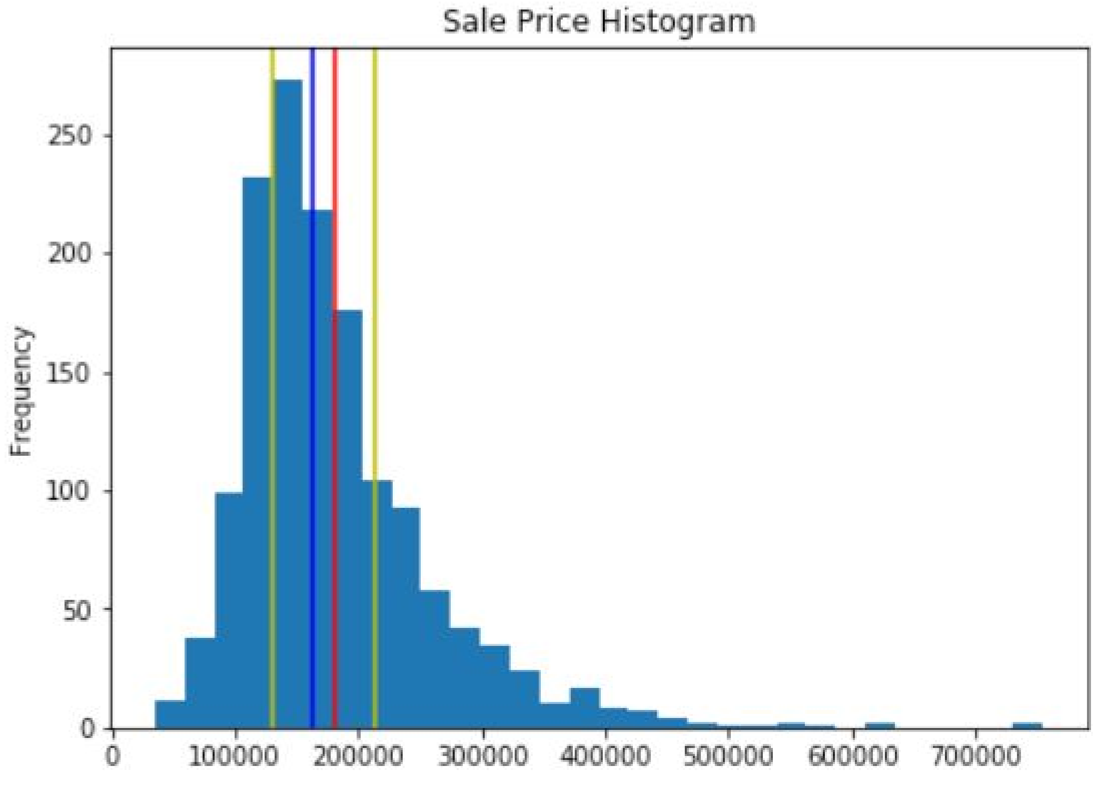
After the data is clean, it’s time to visualize different variables to see how they are related.

Predicted variable

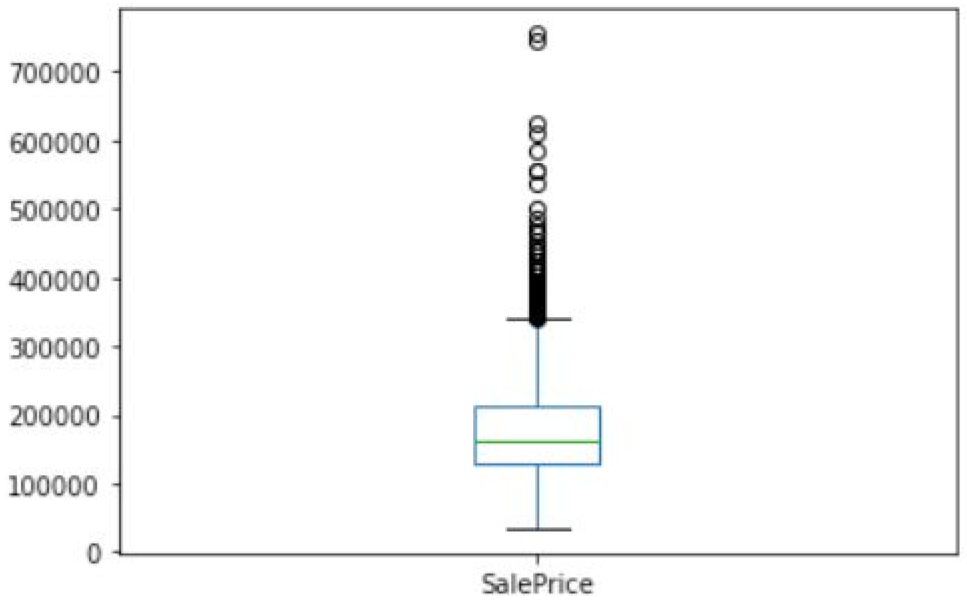
As part of the training data, I was curious to know more about the SalePrice variable. We have 1460 observations in the training data. Here’s the distribution of the SalePrice.

The cheapest house costs $34,900 and the most expensive one costs $755,000. The mean value is $180921 and the median $163000, relatively close. Here’s a histogram with those values plotted (mean in red, median in blue, 25% and 75% quantiles in yellow).

You can see that most houses’ price is between $100,000 and $250,000.



Here’s a boxplot, too. We can see that there are two very clear outliers. We’ll probably discard those so they don’t interfere with our linear regression model.



Feature selection

This dataset has 81 variables. They are too many to keep track of. For that reason, one goal of the visualization part of the project will be to reduce the number of variables to the most significant ones.

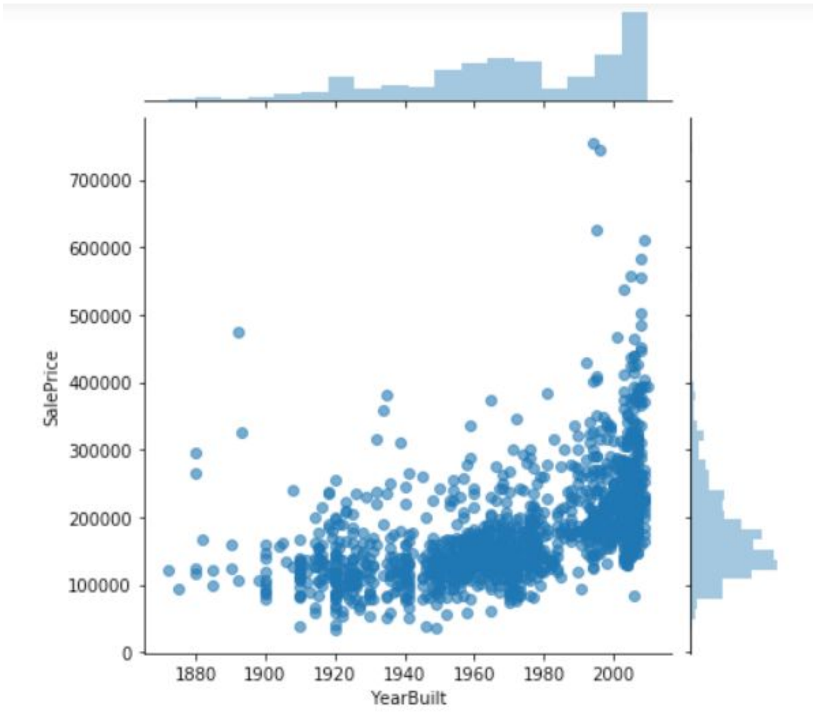
Since we're trying to predict Sale price, the ones with high correlation are good candidates.

Numerical Variables Scatterplots

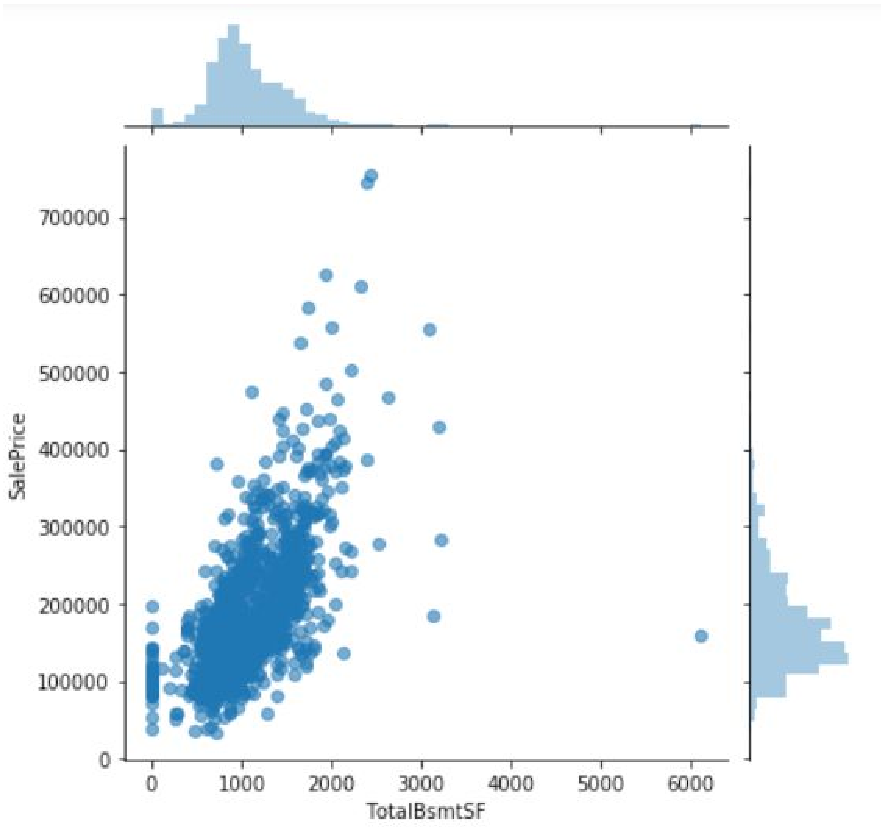
I plotted scatterplots of the numerical variables vs the SalePrice variable. Here are some examples.

These features present a strong correlation with SalePrice, therefore are good candidates to keep.

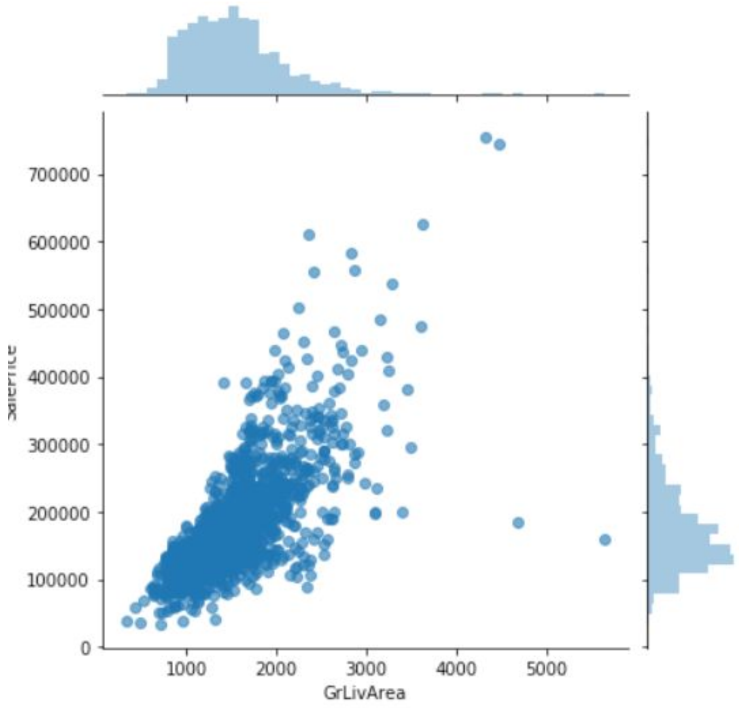
- Year Built. We can also see in the histogram that a lot of the houses are new, built in the 2000s.



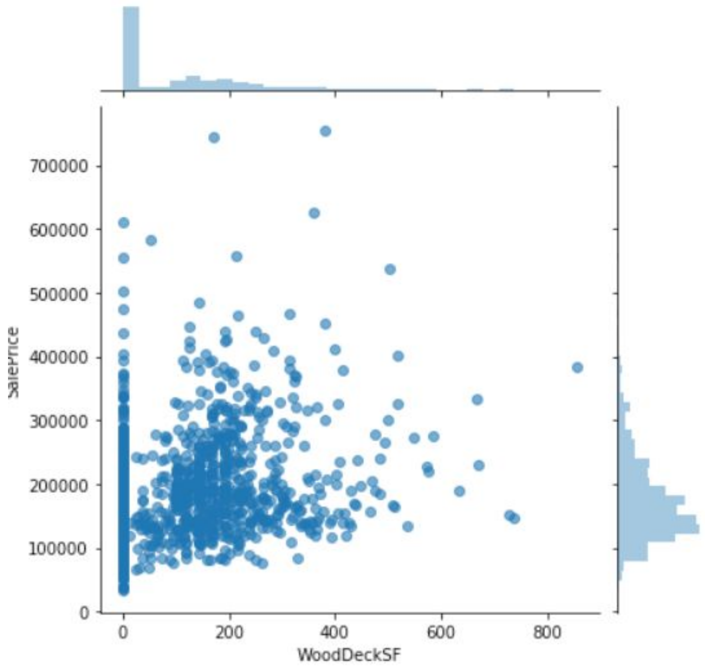
- Total Basement Square Footage: How big the basement is. Again, we have a huge outlier here with +6K sqf. We'll have to investigate.



- General Living Area: sqf of the house itself. This seems like a great predictor. Some outliers there, again.



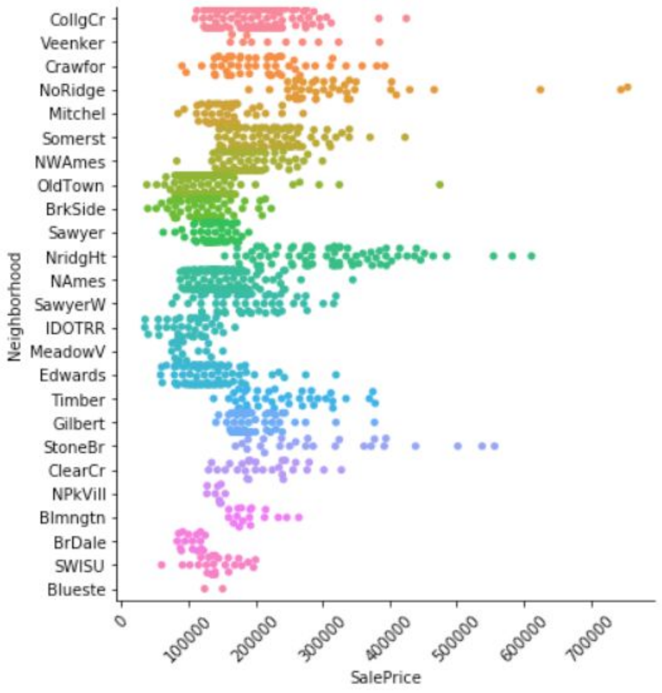
- Wood Deck sqf: Not all houses have a wood deck. A lot of values here are 0. But if they do, it seems related to the house price.



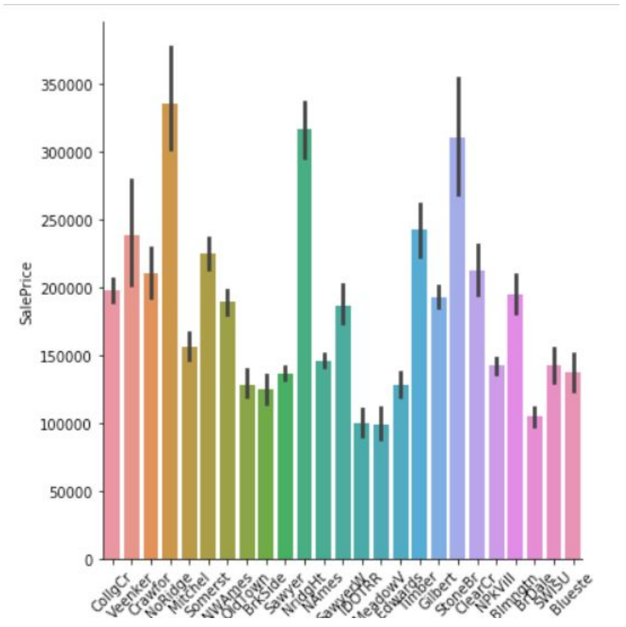
Categorical Variables

These are plots that show the correlation between SalePrice and a categorical variable. They also show, to some extent, how many datapoints are in each variable.

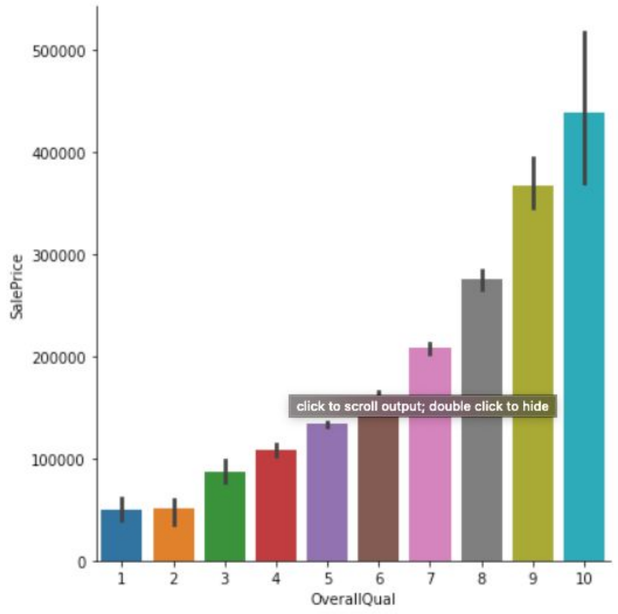
- Neighborhood: Seems to be a good predictor, some of them have higher house prices than the others. Also, some have many more houses, too.



Here’s a barplot of the same data, showing the means and the CIs.



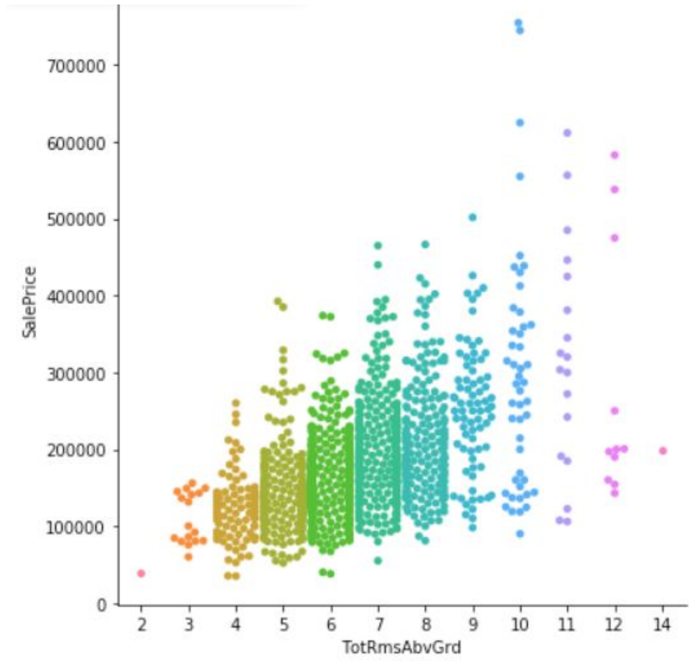
- Overall quality: Seems to be highly related to the SalePrice. Here it is:



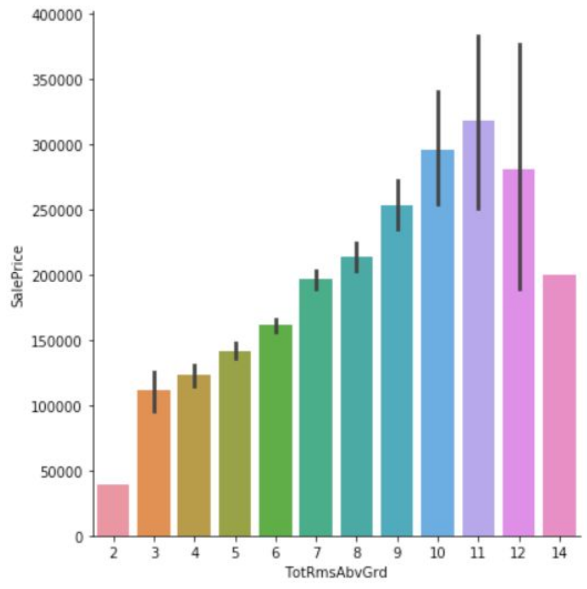
However, Overall Condition doesn't seem to be a very good prescriptor, therefore I'm not showing it here.

- Number of rooms: More rooms mean a higher house price, as expected.

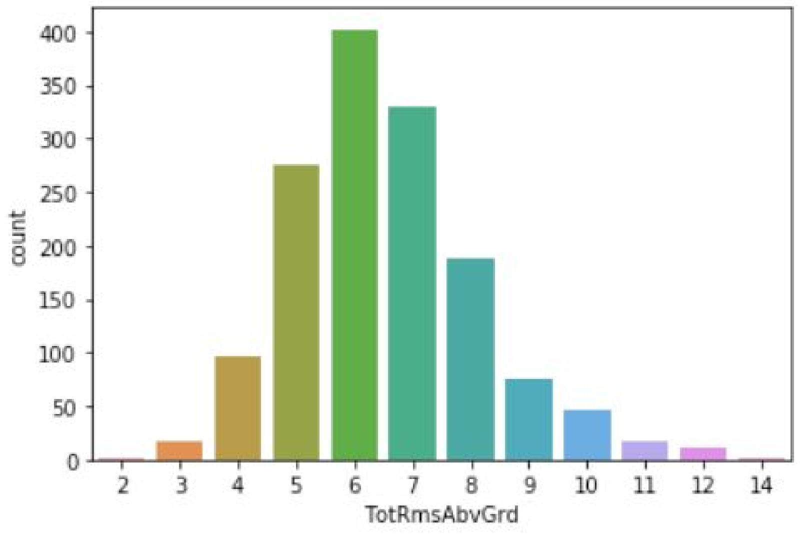
Here's a swarm plot.



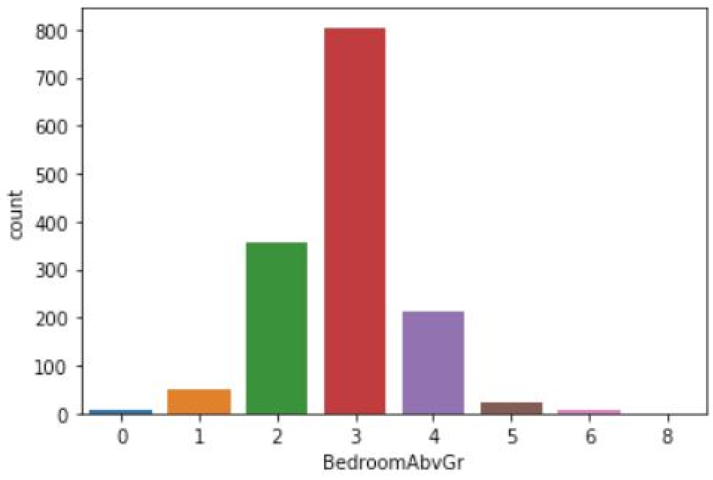
And below is a barplot with the mean of each group.



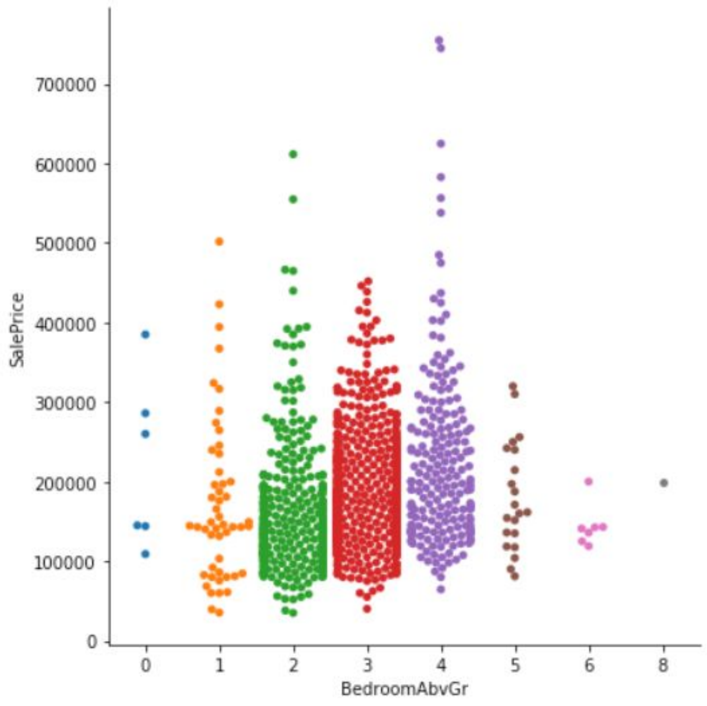
I’m curious to see the distribution of the variable itself. Below is a count plot of TotRmsAbvGrd. It looks like a normal distribution, with most values in the “middle ground”: between 5 and 8 rooms.



Note that this variable refers to all rooms, including kitchen, bathrooms, etc (not just bedrooms). If we only include bedrooms, this is the distribution:



And the correlation with price. Surprisingly, doesn't seem as strong as all rooms included.



Conclusion

After the data exploration and visualization, I have a better understanding of the distribution of the variable I'm trying to predict. I'm also more familiar with the different predicting variables, and I can decide which ones to keep for my first baseline model that appear to be the most informative.

There were some surprises, like LotArea or number of bedrooms not having a strong correlation with price. Other variables were related as expected, like total number of rooms, or livable square feet.

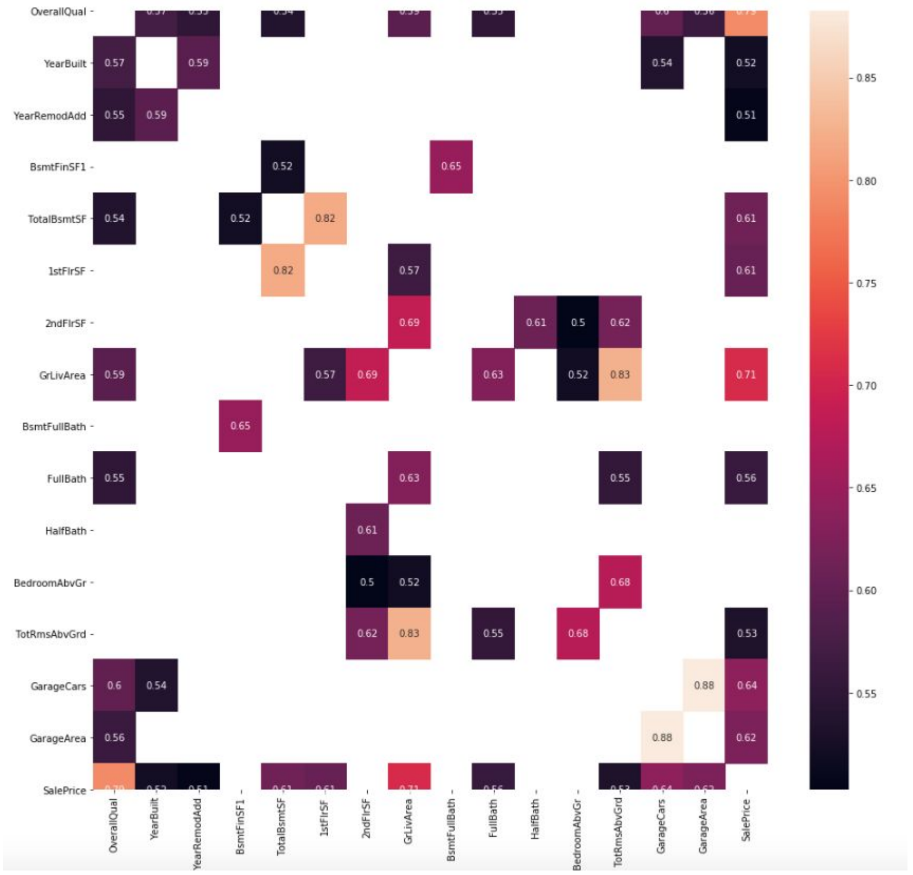
Statistics

The Visualization techniques give us a good understanding of the data. Additionally, we need to do some statistics-based exploration.

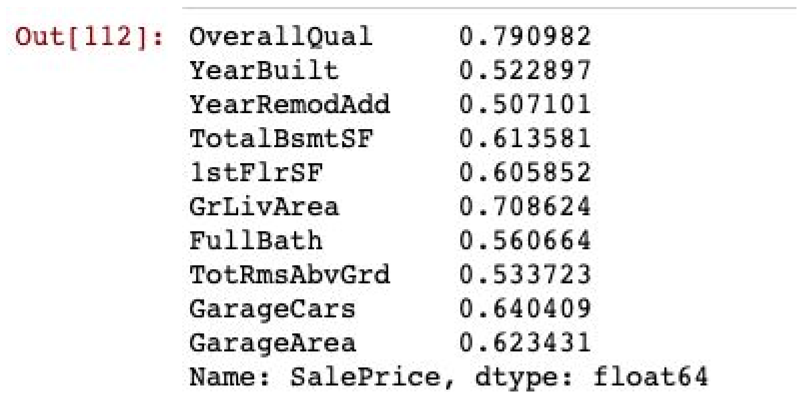
Correlation

One thing we'd like to explore is the correlation between variables. Linear Regression doesn't do very well with variables highly correlated. We'll want to do Principal Component Analysis on the ones that are.

Below is a heatmap of correlations between numerical variables.

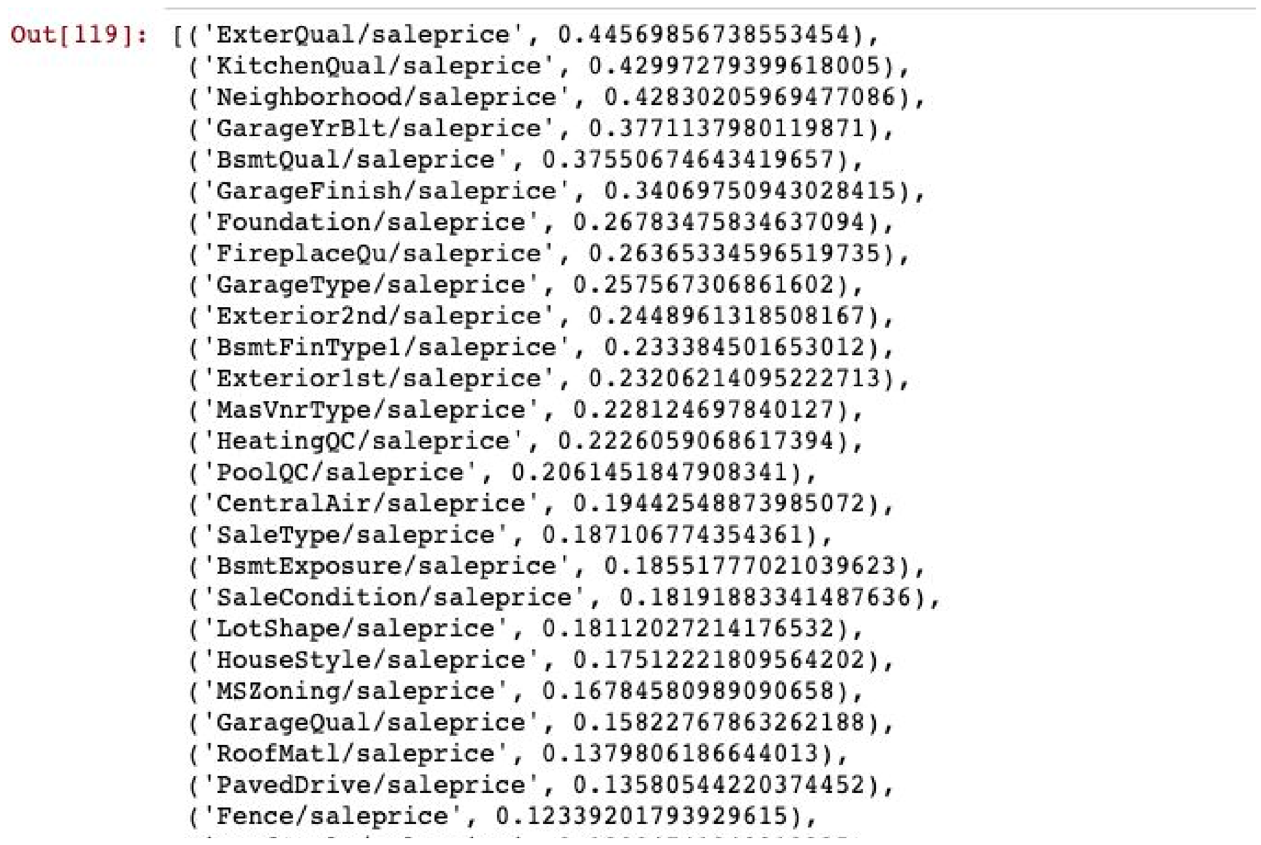


And we can also explore the correlation between the target variable and the predictors. The highest correlated variables are shown here. We can see that OverallQuality and YearBuilt are the top 2 ones, which intuitively makes sense.



```
Out[112]: OverallQual     0.790982
          YearBuilt       0.522897
          YearRemodAdd    0.507101
          TotalBsmtSF     0.613581
          1stFlrSF        0.605852
          GrLivArea       0.708624
          FullBath        0.560664
          TotRmsAbvGrd    0.533723
          GarageCars      0.640409
          GarageArea      0.623431
          Name: SalePrice, dtype: float64
```

We'd also want to explore the correlation of the categorical variables. To do so, we compute the Cramér's V value, which is based on Pearson's chi-squared statistic, for each categorical variable with the target variable. Here are the results:



```
Out[119]: [('ExterQual/saleprice', 0.44569856738553454),
           ('KitchenQual/saleprice', 0.42997279399618005),
           ('Neighborhood/saleprice', 0.42830205969477086),
           ('GarageYrBlt/saleprice', 0.3771137980119871),
           ('BsmtQual/saleprice', 0.37550674643419657),
           ('GarageFinish/saleprice', 0.34069750943028415),
           ('Foundation/saleprice', 0.26783475834637094),
           ('FireplaceQu/saleprice', 0.26365334596519735),
           ('GarageType/saleprice', 0.257567306861602),
           ('Exterior2nd/saleprice', 0.2448961318508167),
           ('BsmtFinType1/saleprice', 0.233384501653012),
           ('Exterior1st/saleprice', 0.23206214095222713),
           ('MasVnrType/saleprice', 0.228124697840127),
           ('HeatingQC/saleprice', 0.2226059068617394),
           ('PoolQC/saleprice', 0.2061451847908341),
           ('CentralAir/saleprice', 0.19442548873985072),
           ('SaleType/saleprice', 0.187106774354361),
           ('BsmtExposure/saleprice', 0.18551777021039623),
           ('SaleCondition/saleprice', 0.18191883341487636),
           ('LotShape/saleprice', 0.18112027214176532),
           ('HouseStyle/saleprice', 0.17512221809564202),
           ('MSZoning/saleprice', 0.16784580989090658),
           ('GarageQual/saleprice', 0.15822767863262188),
           ('RoofMatl/saleprice', 0.1379806186644013),
           ('PavedDrive/saleprice', 0.13580544220374452),
           ('Fence/saleprice', 0.12339201793929615),
```

With that, we're ready to fit our first baseline model, which will be a Linear Regression with only one variable - the best predictor, OverallQuality.

Machine Learning

We are trying to predict the price of a house, based on a set of known features. Price is a continuous numerical value, so a Linear Regression model is our first choice as a Machine Learning algorithm.

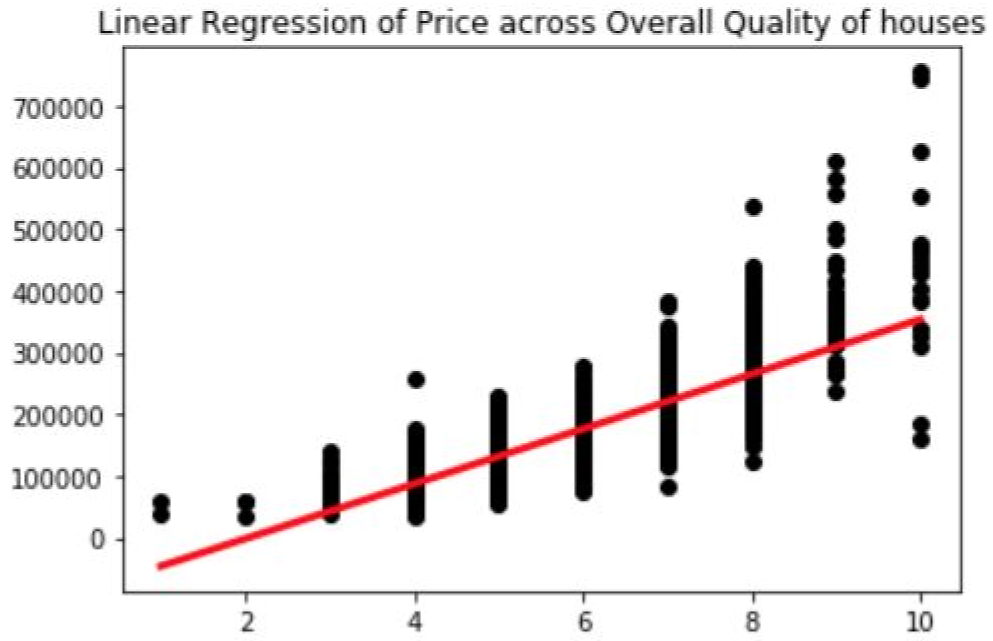
8 models were implemented and tested to see which one was the best one for this dataset.

1. Simple linear regression with only the best predictor

Training score: 0.617
Testing score: 0.650

As expected, the result with only one predictor is not great - but it's not that horrible, either!

Here's a representation of the prediction space:



2. Multiple linear regression with all numerical predictors

Training score: 0.806
Testing score: 0.825

It’s very clear that this dataset needs more complexity. The results with an out-of-the-box linear regression model, using all numerical features without hyperparameter tuning, are pretty decent, and there’s no overfit. But the model is still underfitting, so we’ll continue trying other options.

3. Multiple linear regression with a set of 10 most correlated to target variable predictors

Training score: 0.764
Testing score: 0.796

Again, it’s made obvious that the model needs more complexity, exemplified by this fit. Reducing the features also reduces the accuracy of the linear regression model.

4. Multiple linear regression with highly correlated numerical variables reduced by Principal Component Analysis

Training score: 0.805
Testing score: 0.825

The results are very similar to the second model, so there’s no big gain with the PCA step here.

5. Multiple Linear Regression with one-hot encoded categorical values

Training score: 0.858
Testing score: 0.859

It was very interesting being able to translate the categorical features into numerical ones. It also gave the model a little bump in its accuracy.

Alternative encodings schemes like *Dummy Coding* and *Effect Coding* schemes can be explored too, to expand the project. See here.

6. Regularized Lasso Linear Regression

Training score: 0.858
Testing score: 0.858

Lasso and Ridge regression are techniques to reduce model complexity and prevent overfitting. Even though there’s not too much overfitting in our previous models, it could help us in selecting the right features, and it’s worth exploring.

In this case it doesn’t improve the accuracy score, though.

Lasso regression’s regularization term is absolute, meaning it reduces the “less important” variables to zero, which leads to a less complex model. We’ve seen that doesn’t really help with this dataset. Let’s try Ridge regression now.

7. Regularized Ridge Linear Regression

Training score: 0.858
Testing score: 0.860

Ridge regression doesn’t get rid of the lower beta coefficients, but only reduces them. We do get an increase of the accuracy with this implementation!

8. Single Decision Tree

Training score: 1.0
Testing score: 0.734

When we try a decision tree, we can see it’s clearly overfitting. It’s somewhat expected, as the tree is built specifically for the training dataset, and new unseen data is not well predicted.

9. Decision Tree with GridSearch

Training score: 1.0
Testing score: 0.770

I tried a GridSearch, which was an interesting experiment, but it still didn’t increase the accuracy much.

Conclusion

We implemented two types of algorithms, Linear Regression and Decision Tree. The Linear Regression option was the most obvious, as we’re predicting a continuous numerical variable. And in fact, it’s the one that gave the best results, in particular with Ridge coefficient regularization, in which case the accuracy for the test data was 86%.

Next steps

The project can be expanded in the following ways:

1. Try different encoding schemas for the categorical variables.
2. Find a more appropriate parameter for the Ridge regression lambda parameter.
3. Try more complex models like Random Forests or XGBoosting