Predicting Housing Prices Using Machine Learning

SpringBoard DSC:
Capstone Project 1- Final Report

Matthew Wong
April 2019

Section 1: Introduction

Overview

In this project, I applied basic machine learning concepts on data collected for housing prices in the King County, USA. I first wrangled the data and obtained import features and statistics on the dataset. Next, I analyzed each feature and how they were related to the target variable, price. Once I had a good read and feel for the data set, I split the data into testing and training subsets and determined the best performance metric for the problem. I then tested data on different algorithms. Finally, I analyzed each model's performance metrics, this enabled me to choose the model that best generalizes with new data.

Proposal

House prices increases every year, so the need for a model that can predict future prices is in demand. A model that can help predict housing price can benefit potential homeowners, real estate agents and investors.

The scenario: real-estate agents are looking to get ahead of the competition. The market is highly competitive in order to compete with my peers,as a data science consultant, I can leverage a few basic machine learning concepts to assist my clients with finding the best price for the homes they are selling. My task is to build an optimal model based on the data science process. The data science process consists of the use of data wrangling, exploratory data analysis, data visualizations, statistical analysis and data modeling. In the end the model would give the best model to estimate the selling price for my clients selling homes.

Goal

Create a machine learning model to predict housing prices

Data

The King County, USA housing dataset consists 21,000 observations and 21 features. This dataset can be found here: https://www.kaggle.com/harlfoxem/housesalesprediction

Results

- The Random Forest Regressor model is the best model for predicting housing prices. It holds an accuracy score of **~87%**.
- RMSE is 78,053. Indicating our model was able to predict the value of every house in the test set within $78,053 of the mean price.
- Latitude and Square feet living are the best features that influence the model.

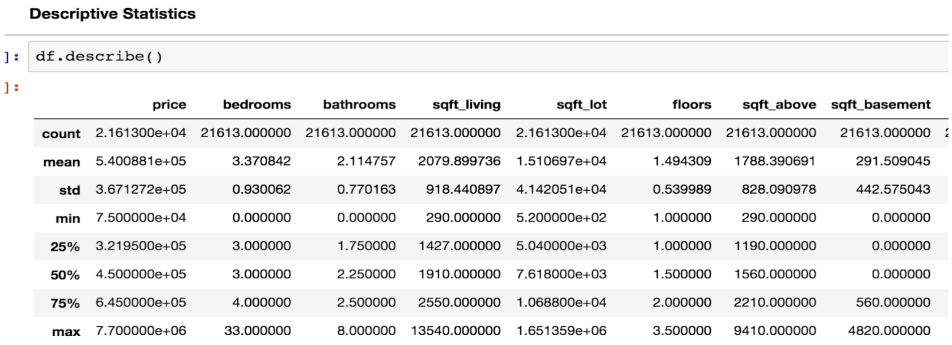
Section 2 : Approach

2.1- Data acquisition and Wrangling

Data science is an iterative process. This means the workflow is non-linear. While exploring one aspect of our data set we may need to go back and change some other aspect of our data. For example, the data wrangling process consist of importing our data but it may also consist of some exploratory data analysis. Data Wrangling for the most part is the process of transforming and mapping our data. It is the step we take before anything else. Before performing analysis, or running algorithms, we need to collect,clean, and transform messy data to create usable data.

The first step of this process is to export the data into the analytical environment. My environment of choice was Jupyter Notebook. In order to accomplish this, I had to import the necessary packages, which included pandas. Pandas is a software library written for the Python programming language for data manipulation and analysis. If offers data structures and operations for manipulating tables. This library was used on the data in order to fill missing values, correct data types and give an overall usable dataset for our analysis. Some functions include:

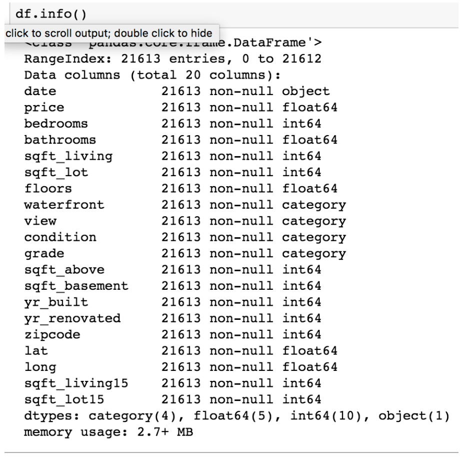
- **Info**: prints information about a DataFrame including index, data type, column type, null-values, number of observations, number of features.
- **Describe**: Generate descriptive statistics that summarize the central tendency, dispersion and shape of the dataset's distribution.
- **Head**: This returns the first few rows of the dataset. It is useful for quick testing.



Descriptive Statistics

```
]: df.describe()
]:
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	sqft_above	sqft_basement
count	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.000000
mean	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	1.494309	1788.390691	291.509045
std	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	0.539989	828.090978	442.575043
min	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	1.000000	290.000000	0.000000
25%	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	1190.000000	0.000000
50%	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	1560.000000	0.000000
75%	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.000000	2210.000000	560.000000
max	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	9410.000000	4820.000000

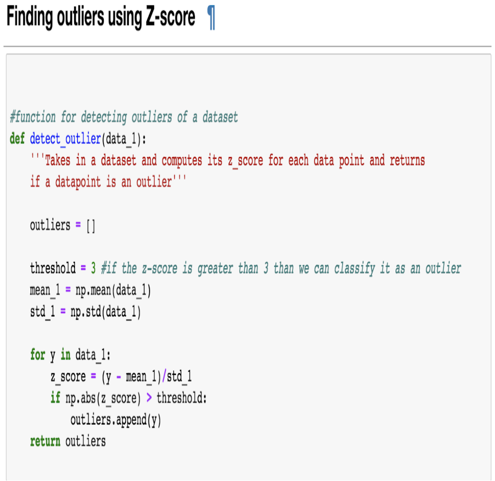


```
df.info()
click to scroll output; double click to hide
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 20 columns):
date             21613 non-null object
price            21613 non-null float64
bedrooms         21613 non-null int64
bathrooms        21613 non-null float64
sqft_living      21613 non-null int64
sqft_lot         21613 non-null int64
floors           21613 non-null float64
waterfront       21613 non-null category
view             21613 non-null category
condition        21613 non-null category
grade            21613 non-null category
sqft_above       21613 non-null int64
sqft_basement    21613 non-null int64
yr_built         21613 non-null int64
yr_renovated     21613 non-null int64
zipcode          21613 non-null int64
lat              21613 non-null float64
long             21613 non-null float64
sqft_living15    21613 non-null int64
sqft_lot15       21613 non-null int64
dtypes: category(4), float64(5), int64(10), object(1)
memory usage: 2.7+ MB
```

2.2: Exploratory Data Analysis and Inferential Statistics

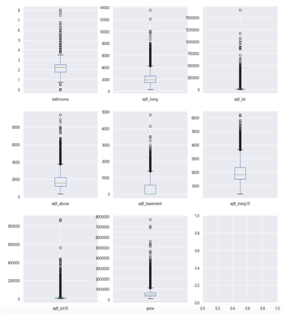
Z-Score Analysis on Outliers

To get rid of outliers I did a z-score analysis of the features. First I created a function that will detect outliers. The function takes in a dataset and computes the z-score for each data point and it returns a datapoint if it is an outlier. I then filtered the data set using the minimum outlier that is given by the function.

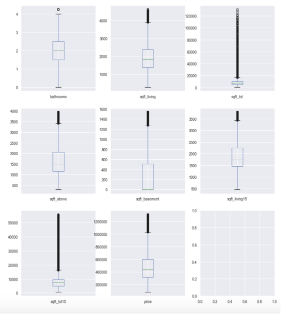


Finding outliers using Z-score ¶

```
#function for detecting outliers of a dataset
def detect_outlier(data_1):
    '''Takes in a dataset and computes its z_score for each data point and returns
    if a datapoint is an outlier'''

    outliers = []

    threshold = 3 #if the z-score is greater than 3 than we can classify it as an outlier
    mean_1 = np.mean(data_1)
    std_1 = np.std(data_1)

    for y in data_1:
        z_score = (y - mean_1)/std_1
        if np.abs(z_score) > threshold:
            outliers.append(y)
    return outliers
```



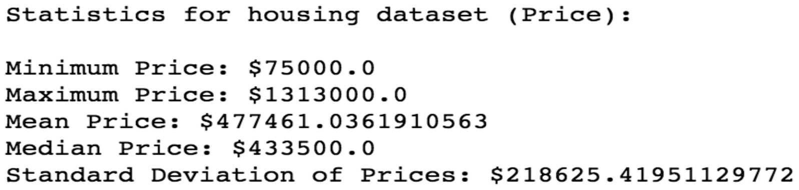
With Outliers



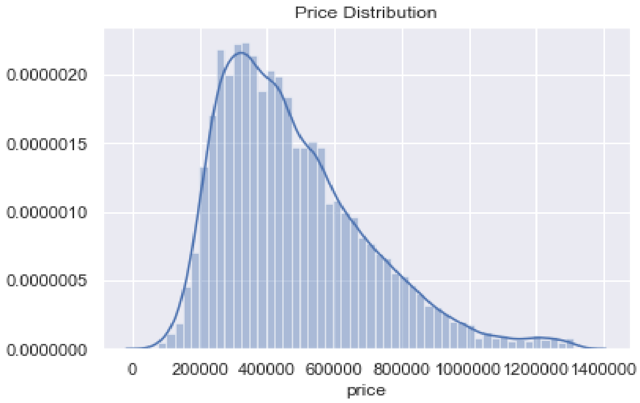
Without Outliers

Exploring features against target variable (price)

Using NumPy, another library in the python programming language that supports mathematical operations, I compute the statistics for the target variable.



```
Statistics for housing dataset (Price):

Minimum Price: $75000.0
Maximum Price: $1313000.0
Mean Price: $477461.0361910563
Median Price: $433500.0
Standard Deviation of Prices: $218625.41951129772
```

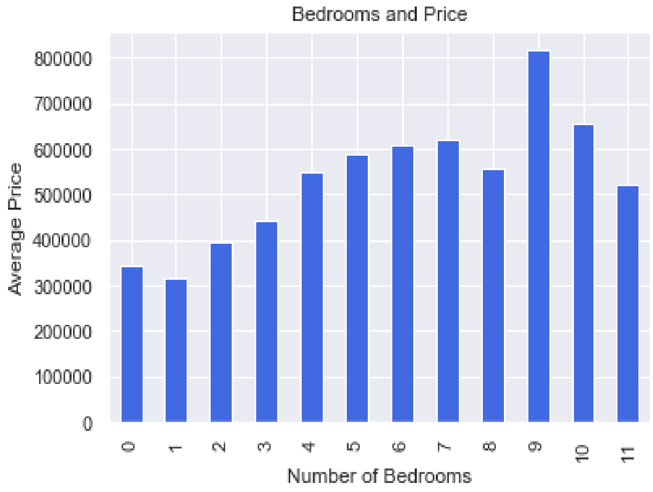


Data science is the process of making assumptions and hypothesis on the data, and testing them by performing some task. Before analyzing the feature I had initial assumptions regarding features that might influence price, these assumptions include

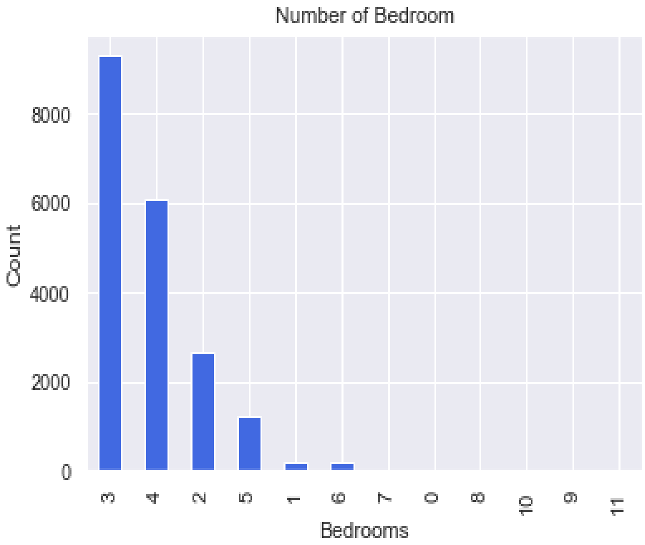
- Houses with more bedrooms will be worth more.
- Larger houses will be worth more
- Location of houses will affect price

These assumptions are useful because they give me a sense of direction, and the opportunity to test these hypotheses.

Bedrooms

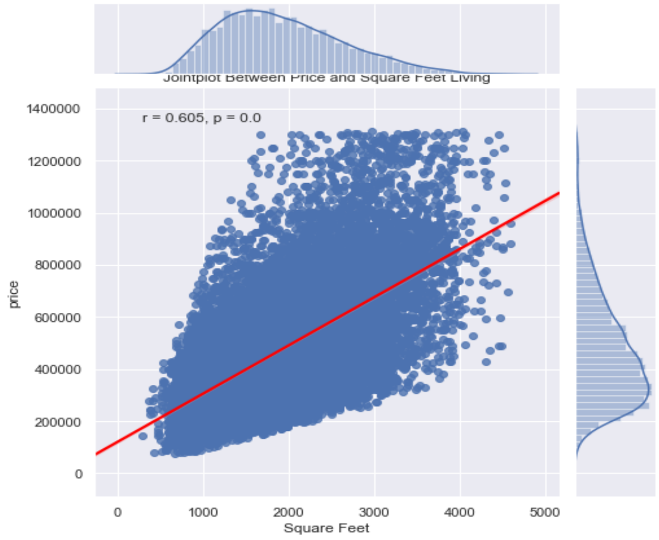


This plot shows the number of bedrooms against average house price. This gives us a good initial understanding the price of a house given the number of bedrooms. How is it useful? For a real estate agents having this data, they can make a quick inference on the housing price based of the feature of bedrooms.



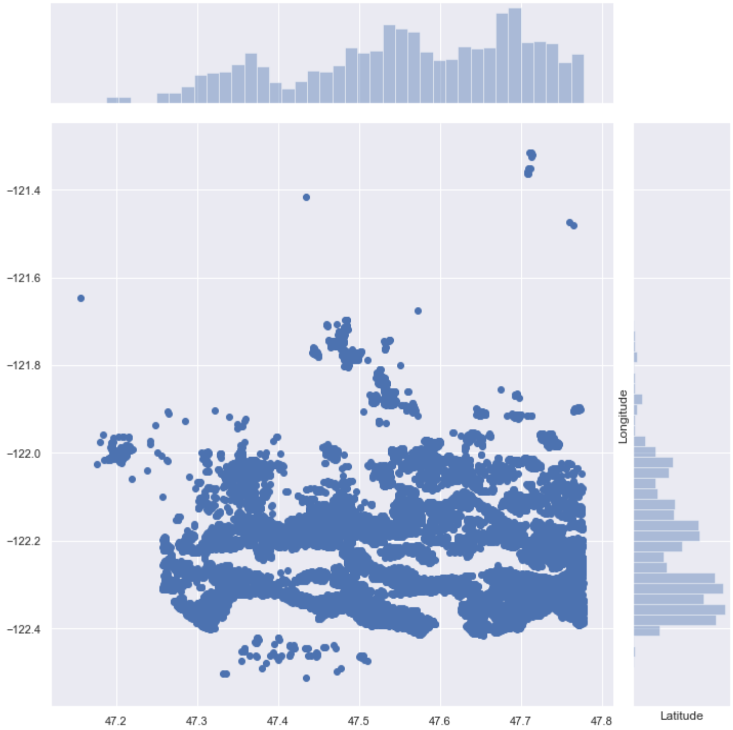
This plot shows the most common bedroom number. 3 Bedroom houses are most commonly sold followed by 4 bedrooms. This is useful for us, we can see which houses will attract more buyers.

Square Feet



As we can see from the graph, square feet and the price of the house are positively correlated, which means as square feet increases price increases. The correlation coefficient represented by r, is a numeric representation of how strong the relationship is on a scale of -1 to 1. The p value is 0, this indicates the relationship is statistically significant.

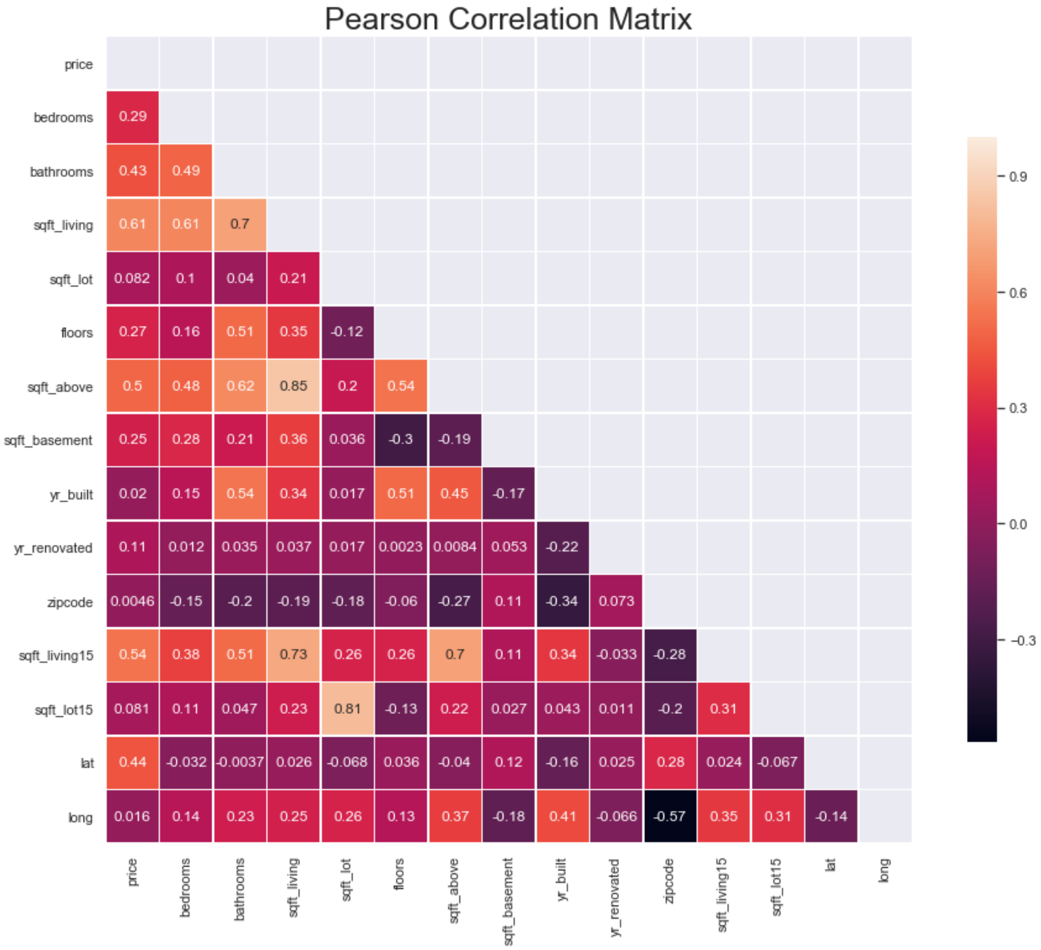
Location



Using a joint plot, we can understand the distribution and concentration of the data. In this case the distribution of latitude and longitude of houses. For latitude between 47.7 and 47.8 there are many houses and for longitude a high concentration can be found -122.2 to -122.4. These could indicate the ideal location of houses, because most buy's has been from these coordinates.

Correlation Matrix

I created a correlation matrix to quantify and summarize the relationships between the features. This matrix contains the pearson's r correlation coefficient. Which determines the relationship of the features. This gives us a better understanding how each feature relates to one another.



2.3: Baseline Analysis

In this section of the project, I developed the tools and techniques necessary for a model to make a prediction.

Performance Metrics

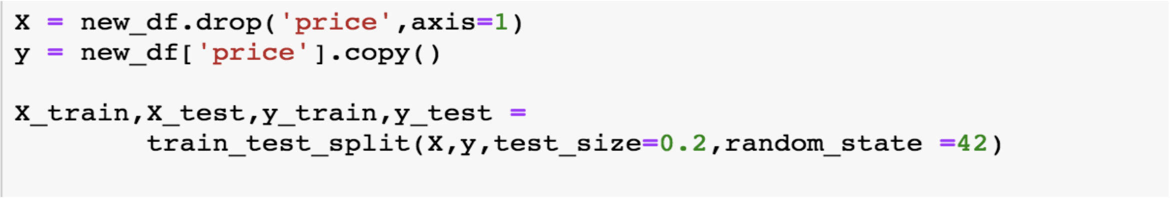
In order to quantify the performance of the model on the training and testing sets we need a performance metrics. This is usually done by calculating some type of error, goodness of fit, or other measurement. For this model our performance metrics are R^2 , RMSE, and MAE.

- **R-Squared (R^2)**: also known as the coefficient of determination for a model is a useful statistic in regression analysis, describes how ‘good’ that model is at making predictions. The value of this metric ranges from 0 to 1, and captures the percentage of squared correlation between the predicted and actual values of the target variable.
- **Mean Absolute Error** (MAE) : Measures the average magnitude of the errors in a set of predictions, without considering their direction. It’s the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.
- **Root Mean Squared Error** (RMSE): is a quadratic scoring rule that also measures the average magnitude of the error. It’s the square root of the average of squared differences between prediction and actual observation.

Training and testing sets

A machine learning algorithm needs to be trained on a set of data to learn the relationships between different features on how these features affect the target variable. In order to do this I needed to divide the data into two sets. One set for training and another for testing. The training set is the set which I will train the algorithm to build a model, the test set is the model I’ll use to see how accurate the predictions are. We want to know if the model has learned properly from a training split.

- Underfitting: The model didn’t learn well on the data, and can’t predict even the outcomes of the training set, high bias.
- Overfitting: The model learned too well from the data, which prevents it from being able to generalize to new data, high variance.
- The model has the right balance between variance and bias.



```
X = new_df.drop('price',axis=1)
y = new_df['price'].copy()

X_train,X_test,y_train,y_test =
        train_test_split(X,y,test_size=0.2,random_state =42)
```

Baseline Model- Linear Regression

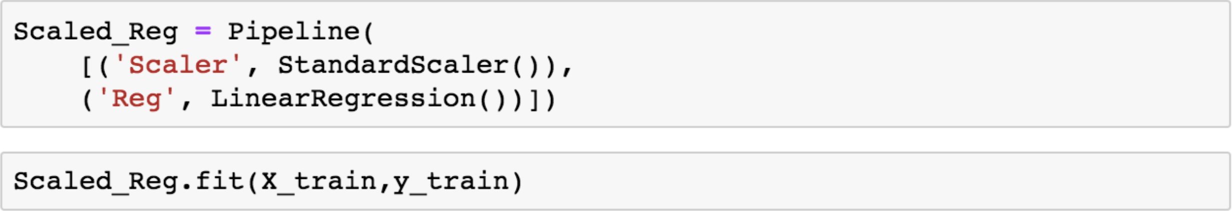
Our Y (a vector) is our target variable, the data that we are trying to predict. For this project, our Y is the price of the house. Our X (a matrix) will be all other features that will be used as variables that influence Y.

- Y = housing prices (Dependent variable, response variable)
- X = all other features (Independent variables, predictors, explanatory variables)

When there is only one predictor variable, the prediction method is simple regression, and for multiple predictor variables it is multiple regression. For the baseline model we will do a multiple regression on our data. I used a linear regression classifier on the training data. The goal for machine learning is to train the data and compute the accuracy to determine the success of the model.

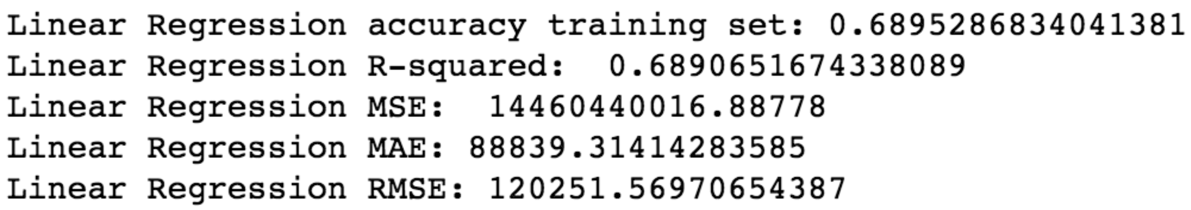
The first step is to import all necessary packages for our model. This includes the Linear Regression classifier. Next, I created a pipeline which normalized the data and called the classifier. After, I trained the training data on the pipeline and then computed its result using a 5 fold cross validation. Finally, I computed the performance metrics of the training and testing data. The results from the baseline model are:

- ~68.9% of the variability in Y can be explained using X.
- The accuracy for the training set is ~68.9% while the score for the testing set is also ~68.9%. This is relatively low, however this also indicates no overfitting or underfitting for our models.
- MAE and RMSE are 88,839 and 120,251 respectively. Indicating our model was able to predict the value of every house in the test set within $120,251 of the mean price.



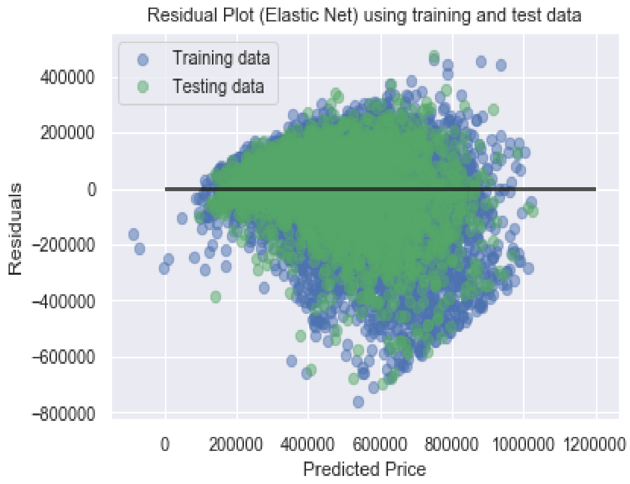
```
Scaled_Reg = Pipeline(
    [('Scaler', StandardScaler()),
    ('Reg', LinearRegression())])
```

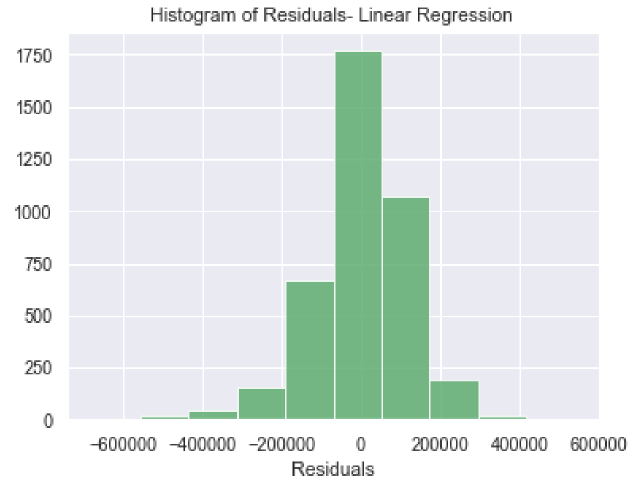
```
Scaled_Reg.fit(X_train,y_train)
```



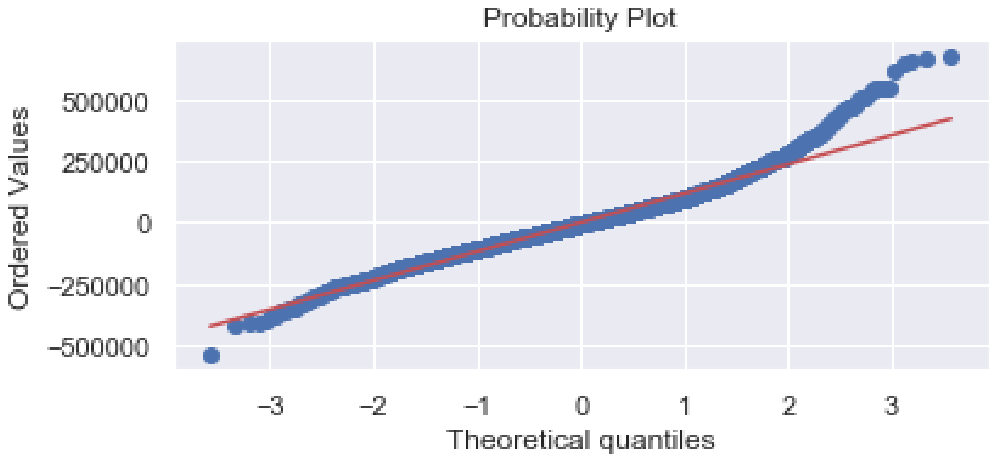
```
Linear Regression accuracy training set: 0.6895286834041381
Linear Regression R-squared:  0.6890651674338089
Linear Regression MSE:  14460440016.88778
Linear Regression MAE: 88839.31414283585
Linear Regression RMSE: 120251.56970654387
```

Residual Analysis for Linear Regression





Residuals, in the context of regression models, are the difference between observed value of the target variable and the predicted value, also known as the error of the prediction. A common use of the residual plots is to analyze the variance of the error of the regressor. If the points are randomly dispersed around the horizontal axis, a linear regression model is more appropriate. In the case above, we see a fairly random distribution. We can also see from the histogram that our error is normally distributed around zero, which generally indicates a well fitted model. From the above residual scatter plot we can see the corresponding point is reasonably random, meaning a linear regression model is appropriate. An alternative way to check for normality is using a quantile plot.



The good fit indicates that normality is a reasonable approximation.

2.4: Extended Analysis

Random Forest

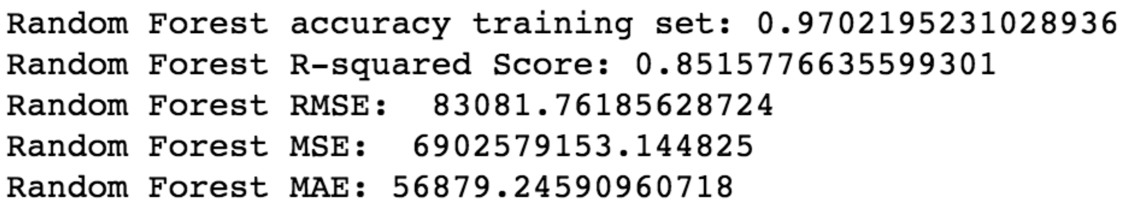
Random forest is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. Random forest is an example of a ensemble method. Ensemble are machine learning techniques that combine several base models in order to produce one optimal predictive model, in this case the base models are decision trees. Decision trees are intuitive ways to classify or label objects: you simply ask a series of questions that contain a binary answer. The binary splitting makes the algorithm extremely efficient. To prevent a decision tree classifier from overfitting we use a ensemble method, Random Forest. Random forests models create sets of trees on randomly selected data samples, get a prediction from each tree and selects the best solution. The advantages of a random forest algorithm are as follows:

- Highly accurate
- Does not suffer from overfitting
- Used for both classification and regression problems
- Can handle missing values
- You can compute feature importance.

The first step in building the random forest model was to create a pipeline. The pipeline contained a normalization classifier and a random forest classifier. Next, I trained the training data on the pipeline by using the fit method. After, I computed the performance metrics for the training data, and the testing data. Finally, I did some analysis on the metrics, and residual by the using graphs. The results from the Random Forest model are:

- ~85.15% of the variability in Y can be explained using X.
- The accuracy for the training set is ~97% while the score for the testing set is also ~85.15%.
- MAE and RMSE are 56,879 and 83,081 respectively. Indicating our model was able to predict the value of every house in the test set within $83,081 of the mean price.

These results are an improvement from our baseline model.



```
Random Forest accuracy training set: 0.9702195231028936
Random Forest R-squared Score: 0.8515776635599301
Random Forest RMSE:  83081.76185628724
Random Forest MSE:  6902579153.144825
Random Forest MAE: 56879.24590960718
```

Hyperparameter Tuning - Random Forest

I have built a random forest model to solve the machine learning problem. The results are significantly better from the baseline model. The next step is tuning the hyperparameters to get the best optimal random forest model. The hyperparameters to tune are:

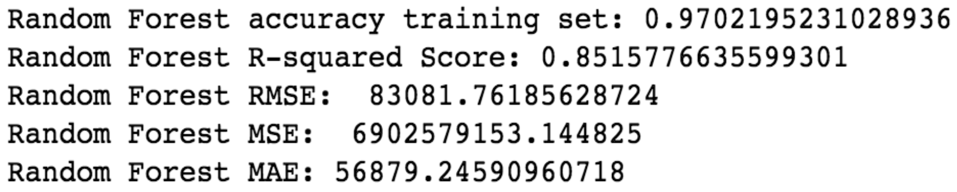
- N_estimators = number of trees in the forest
- Max_features = max number of features considered for splitting a node
- Max_depth = max number of levels in each decision tree
- Min_samples_split = min number of data points placed in a node before the node is split
- Min_samples_leaf = min number of data points allowed in a leaf node
- Bootstrap = method for sampling data points (with or without replacement)

I used RandomizedSearchCV to find the best parameters. To use this regressor I first create a parameter grid. Next, I called the random search classifier on the random forest estimator and the parameter grid, using 3 fold cross validation. After, completing the search I fitted the model on the training data. Then, I used the best parameter method to get the best parameters from the search. Finally, I computed the performance metrics using the best parameters for random forest. The results from the Random Forest model using the best parameters are:

- ~86.89% of the variability in Y can be explained using X.
- The accuracy for the training set is ~97% while the score for the testing set is also ~86.89%.
- MAE and RMSE are 52,810 and 78,053 respectively. Indicating our model was able to predict the value of every house in the test set within $78,053 of the mean price.

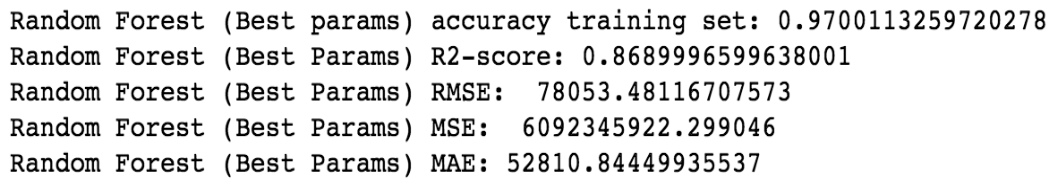
This yield better results from our first random forest model, by ~2%.

Random Forest



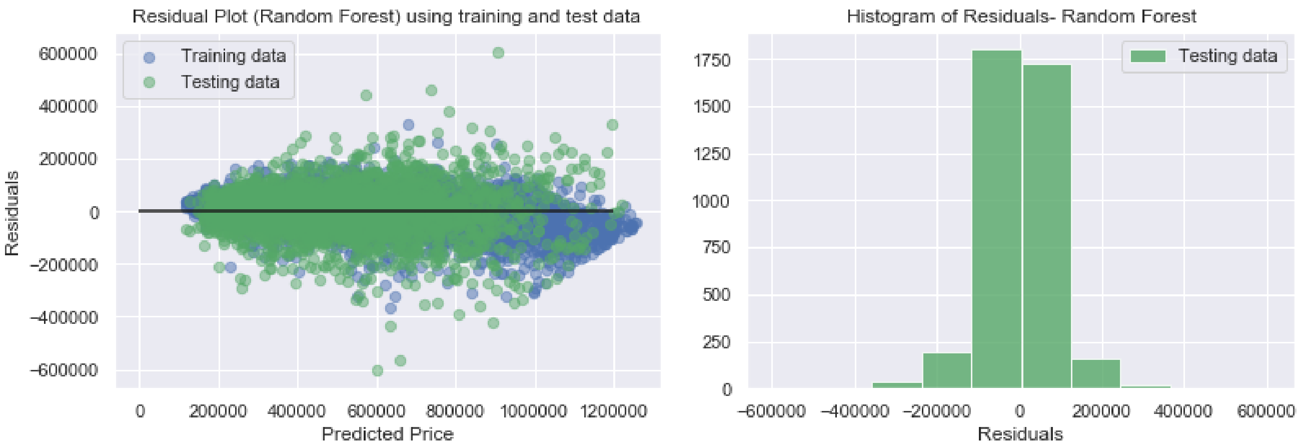
```
Random Forest accuracy training set: 0.9702195231028936
Random Forest R-squared Score: 0.8515776635599301
Random Forest RMSE:  83081.76185628724
Random Forest MSE:  6902579153.144825
Random Forest MAE: 56879.24590960718
```

Random Forest (Best Parameters)

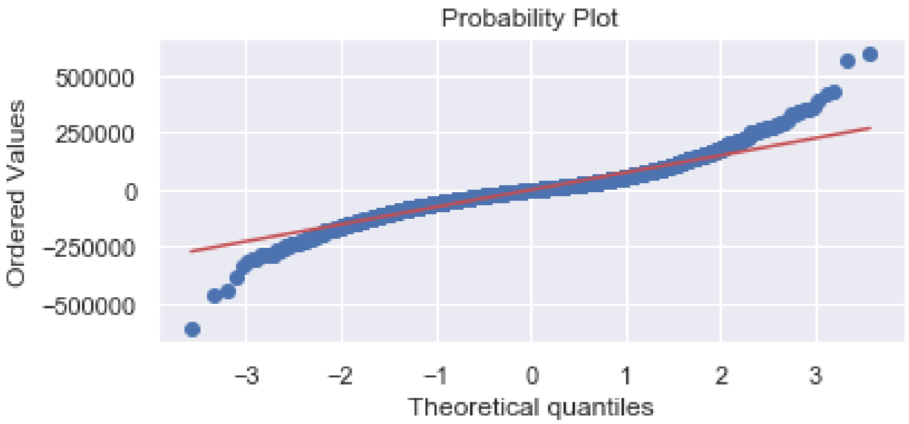


```
Random Forest (Best params) accuracy training set: 0.9700113259720278
Random Forest (Best Params) R2-score: 0.8689996599638001
Random Forest (Best Params) RMSE:  78053.48116707573
Random Forest (Best Params) MSE:  6092345922.299046
Random Forest (Best Params) MAE: 52810.84449935537
```

Residual Analysis for Random Forest



In the case above, we see a fairly random distribution. We can also see from the histogram that our error is normally distributed around zero, which indicates a well fitted model. Reminder, ideally all residuals should be small and unstructured; this would mean the regression analysis has been successful in explaining the variation of the dependent variable. In the above graphs the residuals show no signs of patterns.



Reminder, A a probability plot is a graphical tool to help assess if a set of data possibly came from some theoretical distribution such as a normal or exponential. The plot is a scatterplot created by plotting two set of quantiles against one another. If both sets came from the same distribution, we should see the points forming a line. In this case our data does show fit on the line, not perfect, but enough to show normality.

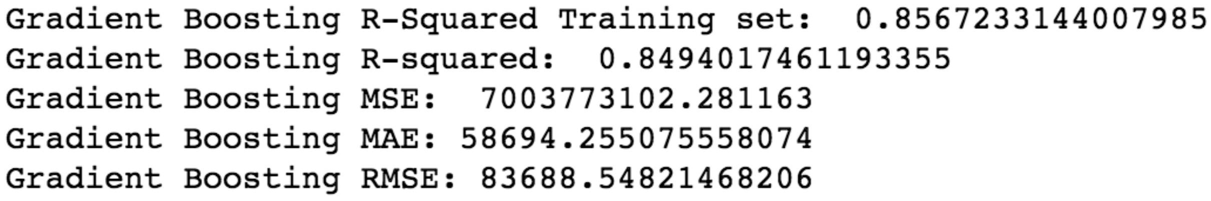
Gradient Boosting

For building a prediction model, many experts use gradient boosting regression. Gradient boosting is a machine learning technique for regression and classification problems. This is another ensemble method just like random forest that work with weaker prediction models, like decision trees. Boosting can be defined as a method of converting weak learners into strong learners. In boosting, each new tree is a fit on a modified version of the original data set. To create this model I used the same pipeline as the previous models. First, I created a pipeline for normalizing the data and classifying the gradient boosting algorithm. Next, I fitted the training data on the pipeline. Then, I computed the performance metrics. Finally, I analyzed the residuals. The results for this model are as follows:

- ~84.94% of the variability in Y can be explained using X.
- The accuracy for the training set is ~85.7% while the score for the testing set is also ~84.94%.
- MAE and RMSE are 58,694 and 83,688 respectively. Indicating our model was able to predict the value of every house in the test set within $83,688 of the mean price.

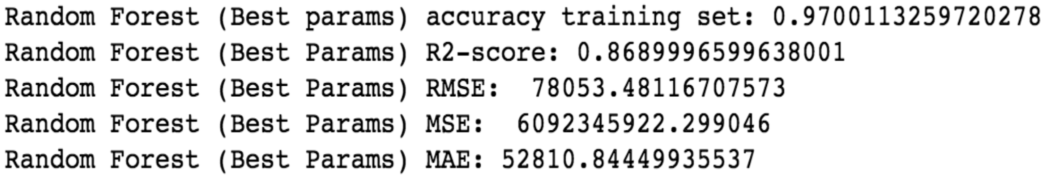
The results indicate a good model, however random forest yielded better results by ~2%

Gradient Boosting



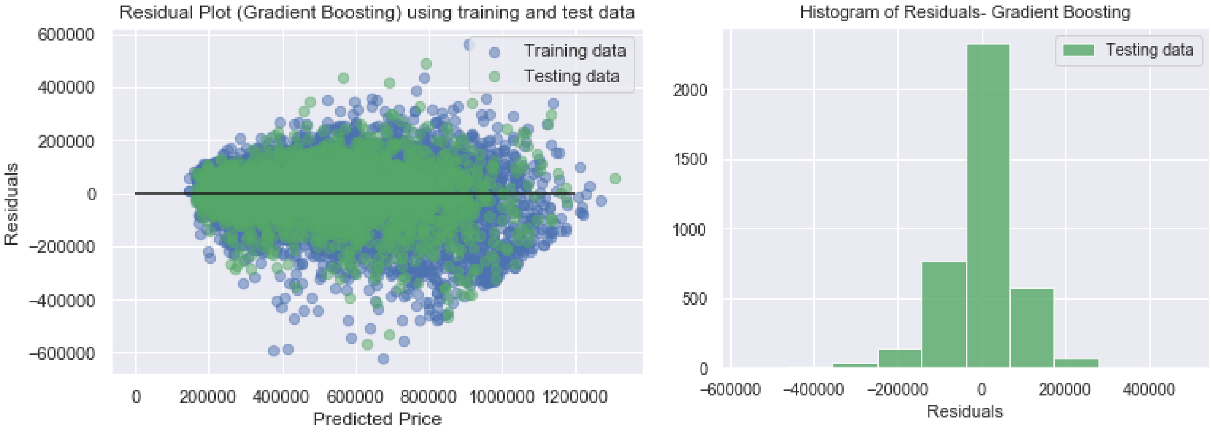
```
Gradient Boosting R-Squared Training set:  0.8567233144007985
Gradient Boosting R-squared:  0.8494017461193355
Gradient Boosting MSE:  7003773102.281163
Gradient Boosting MAE: 58694.255075558074
Gradient Boosting RMSE: 83688.54821468206
```

Random Forest (Best Parameters)

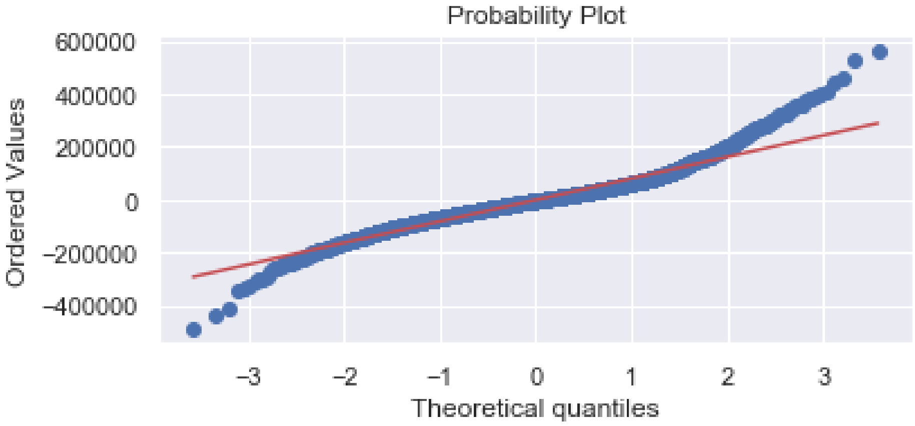


```
Random Forest (Best params) accuracy training set: 0.9700113259720278
Random Forest (Best Params) R2-score: 0.8689996599638001
Random Forest (Best Params) RMSE:  78053.48116707573
Random Forest (Best Params) MSE:  6092345922.299046
Random Forest (Best Params) MAE: 52810.84449935537
```

Residual Analysis for Gradient Boosting

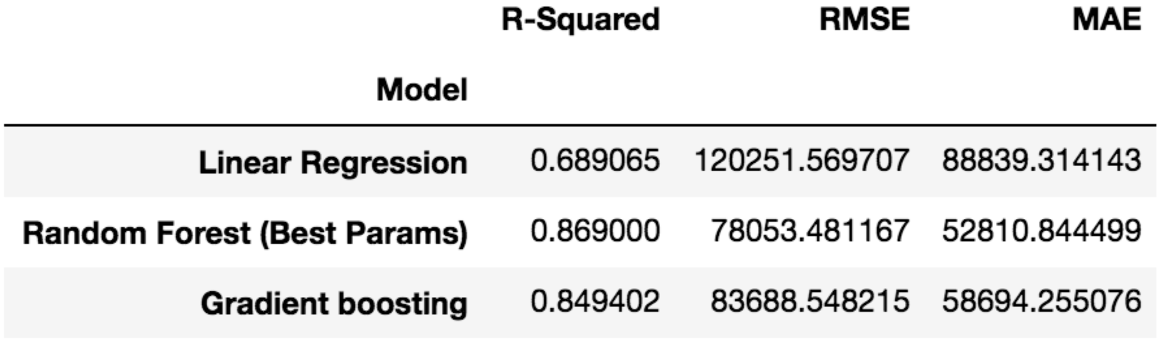


For the residual plots of our gradient boosting model the residuals showcase no patterns, and the histogram that our error is normally distributed around zero, which indicates a well fitted model.



The data points on the line indicates normality.

Section 3 : Results



	R-Squared	RMSE	MAE
Model			
Linear Regression	0.689065	120251.569707	88839.314143
Random Forest (Best Params)	0.869000	78053.481167	52810.844499
Gradient boosting	0.849402	83688.548215	58694.255076

From the analysis, Random Forest with tuning is the best. It had the highest R-Squared, and lowest RMSE and MAE.

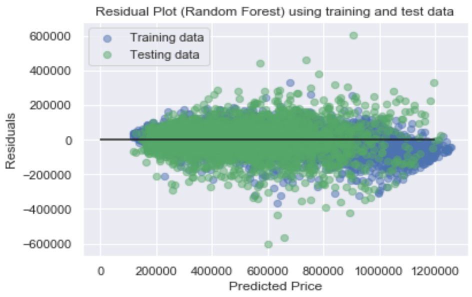
Conclusions

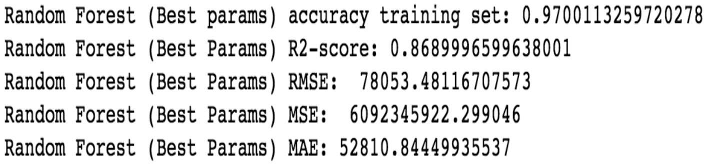
In this project, I have explored the use of different machine learning techniques with relation to housing price. I performed data exploration for the housing data, created data visualization while doing exploratory data analysis, and build machine learning models using three different techniques: linear regression, random forest, and gradient boosting. After training and evaluating the models, I computed the performance metrics of each model and identified the best model. The main goal was to create a model that would be able to predict price given a list of features. I concluded the best model for predicting price was random forest. The results was that the best Mean absolute error was $52,810. The RMSE was $78,053 and the model computed an accuracy score of ~87%.

Future work

- Reevaluate features, perhaps do a feature selection on linear regression to determine which features influence price and discard that ones that don’t
- Study potential collinearity among independent variables to decide which ones not to consider in the analysis
- Curate housing data from different sources. Find other alternative housing data from King County to provide a bigger sample size.
- Analyze other machine learning models and how it compares to the ones analyzed.
- Hyper tune gradient boosting model to find optimal score.
- Conduct additional feature influence analyses for all the implemented models (e.g, linear regression)
- Study carefully the effect of geo-based features such as latitude, longitude, and zip code

Section 4: Recommendation for the Clients

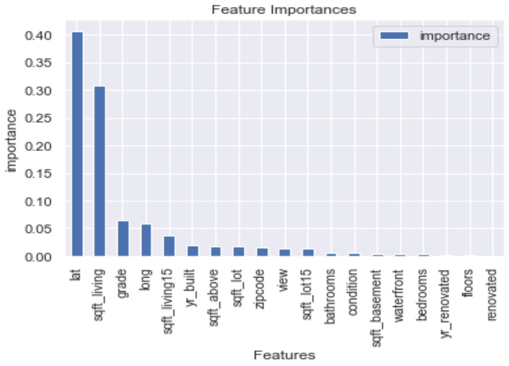




```
Random Forest (Best params) accuracy training set: 0.9700113259720278
Random Forest (Best Params) R2-score: 0.8689996599638001
Random Forest (Best Params) RMSE:  78053.48116707573
Random Forest (Best Params) MSE:  6092345922.299046
Random Forest (Best Params) MAE: 52810.84449935537
```

- Using Random Forest Regressor as the model to predict price you are able to compute the price with an accuracy score of ~87%
- RMSE is 78,053. Indicating our model was able to predict the value of every house in the test set within $78,053 of the mean price.

Feature Importance



- The most valuable features are 'lat' and 'sqft_living'. Both strongly impact model performance: permuting them decreases model performance by ~40% and ~31% respectively, you should take this into consideration.

Model's Applicability

- The dataset is limited, only containing 21,000 observations, and only contains data from King's County. This model can give a great estimate for houses in this area, but may not provide accurate results in other areas. The Random Forest Regressor algorithm however can be used to predict housing prices using different housing datasets but it needs to be analyzed separately.
- There are many features of the house that are important in predicting a house, features like square feet of living, the number of rooms, renovated, and the condition. All of these features are relevant when deciding house prices, this indicates good features for a model.