***PREDICTING HOUSE PRICES***

***USING***

***MACHINE LEARNING***

1.Introduction

2.Importing Modules, Reading the Dataset and Defining an Evaluation Table

3.Defining a Function to Calculate the Adjusted R2

4.Creating a Simple Linear Regression

Let’s Show the Result

5.Visualizing and Examining Data

Checking Out the Correlation Among Explanatory Variables

6.Data Preprocessing

Binning

7.Multiple Regression

Multiple Regression – 1

Multiple Regression – 2

Multiple Regression – 3

Multiple Regression – 4

8.Regularization

Ridge Regression

Lasso Regression

9.Polynomial Regression

10.k-NN Regression

11.Evaluation Table

12.Conclusion

INTRODUCTION:

Welcome to my Kernel! In this kernel, I use various regression methods and try to predict the house prices by using them. As you can guess, there are various methods to suceed this and each method has pros and cons. I think regression is one of the most important methods because it gives us more insight about the data. When we ask why, it is easier to interpret the relation between the response and explanatory variables.

I start with a very simple model and continue with more complex ones after visualizing some features and a data mining process. I try to find the best regression for this dataset.

IMPORTING MODULES, READING THE DATASET AND DEFINING AN EVALUATION TABLE:

In order to make some analysis, we need to set our environment up. To do this, I firstly imported some modules and read data. The below output is the head of the data but if you want to see more details, you might try removing # signs in front of the df.describe() and df.info().

Further, I defined an empty dataframe. This dataframe includes Root Mean Squared Error (RMSE), R-squared, Adjusted R-squared and mean of the R-squared values obtained by the k-Fold Cross Validation, which are the important metrics to compare different models. Having a R-squared value closer to one and smaller RMSE means a better fit. In the following sections, I will fill this dataframe with my results.

DEDINING A FUNCTION TO CALCULATE THE ADJUTED R2:

The R-squared increases when the number of features increase. Because of this, sometimes a more robust evaluator is preferred to compare the performance between different models. This evaluater is called adjusted R-squared and it only increases, if the addition of the variable reduces the MSE. The definition of the adjusted R2

Is:

R2¯=R2−k−1n−k(1−R2)

Where n

Is the number of observations and k Is the number of parameters.

CREATING A SIMPLE LINEAR REGRESSION:

When we model a linear relationship between a response and just one explanatory variable, this is called simple linear regression. I want to predict house prices and then, our response variable is price. However, for a simple model we also need to select a feature. When I look at the columns of the dataset, living area (sqft) seemed the most important feature. When we examine the correlation matrix, we may observe that price has the highest correlation coefficient with living area (sqft) and this also supports my opinion. Thus, I decided to use living area (sqft) as feature but if you want to examine the relationship between price and another feature, you may prefer that feature.

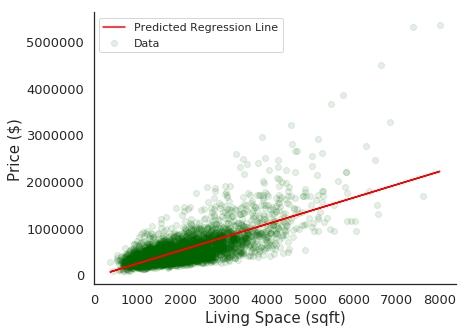
Average Price for Test Data: 539744.130

Intercept: -47235.811302901246

Coefficient: [282.2468152].

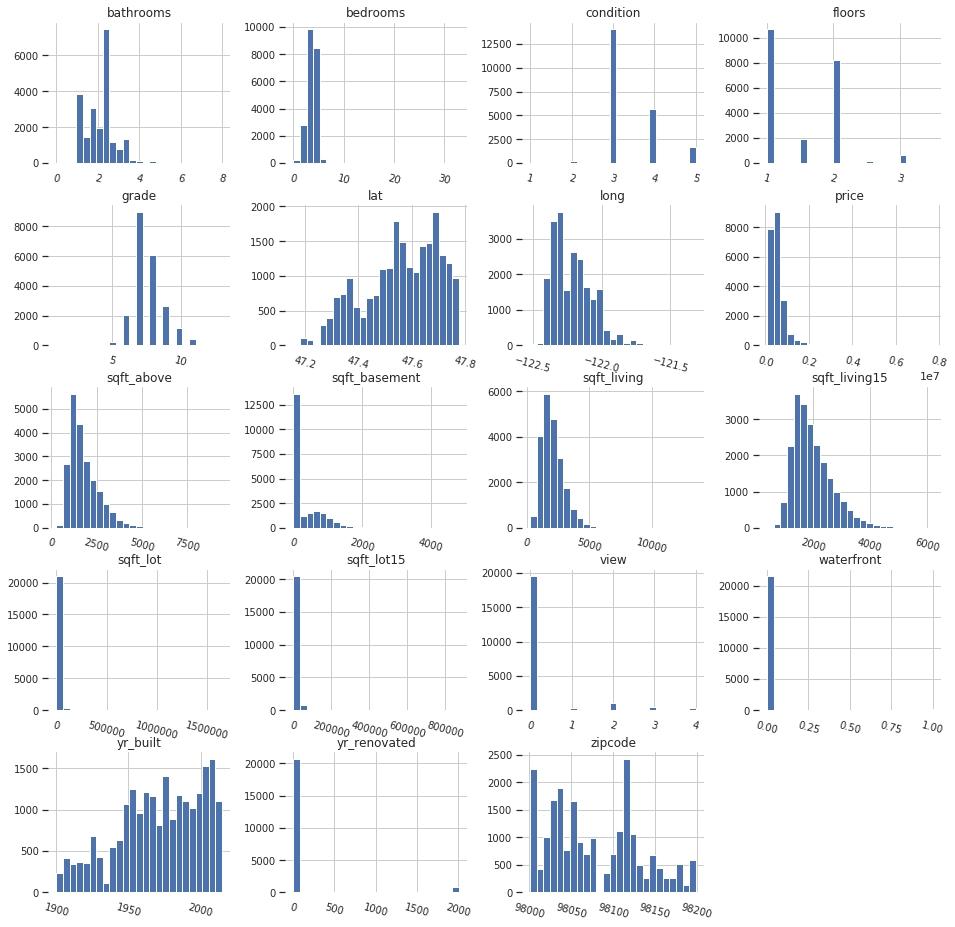
LETS SHOW THE RESULT:

Since we have just two dimensions at the simple regression, it is easy to draw it. The below chart determines the result of the simple regression. It does not look like a perfect fit but when we work with real world datasets, having a perfect fit is not easy.



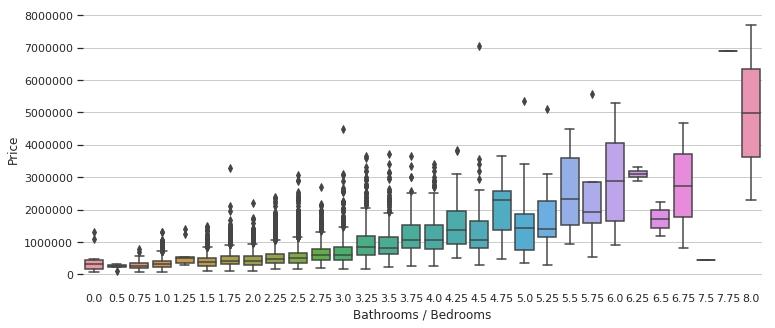
VISUALISING AND EXAMINING DATA:

This is not a very big data and we do not have too many features. Thus, we have chance to plot most of them and reach some useful analytical results. Drawing charts and examining the data before applying a model is a very good practice because we may detect some possible outliers or decide to do normalization. This is not a must but get know the data is always good. Then, I started with the histograms of dataframe.



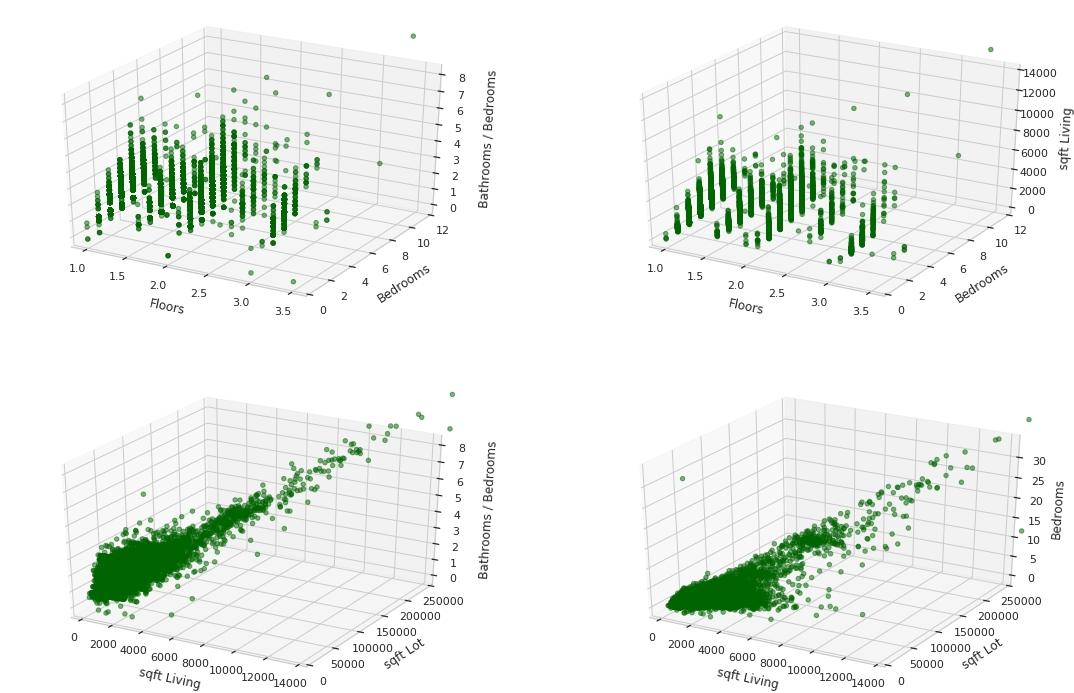
To determine bedrooms, floors or bathrooms/bedrooms vs price, I preferred boxplot because we have numerical data but they are not continuous as 1,2,… bedrooms, 2.5, 3,… floors (probably 0.5 stands for the penthouse).

From the below charts, it can be seen that there are very few houses which have some features or price appears far from others like 33 bedrooms or price around 7000000. However, determining their possible negative effect will be time consuming and in the real data sets there will always be some outliers like some luxury house prices in this dataset. That’s why I am not planning to remove outliers.

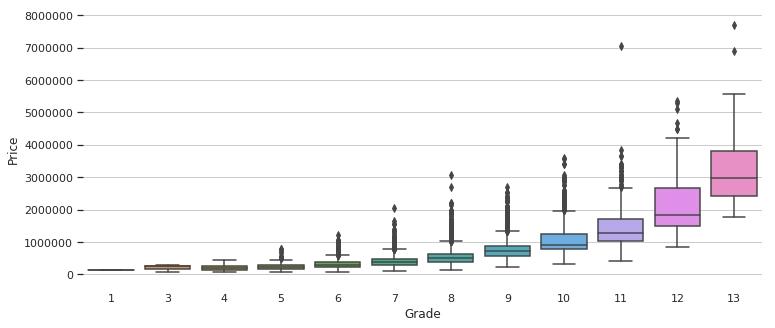


I drew the price vs some features and it seems that there is not a perfect linear relationship between the price and these features. On the other hand, what about the relationship among each other? To show this, I preferred 3D plots. Also, I used light green as the point color. Dark green parts mean high density, many light green points overlap and become darker.

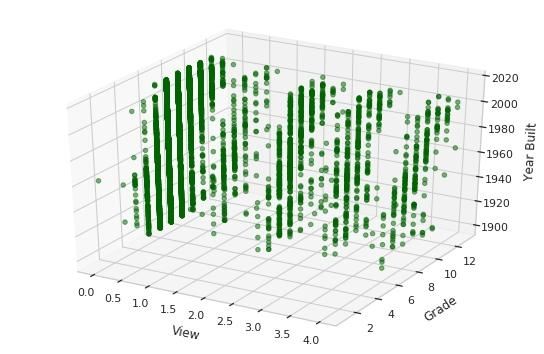
The below charts show that when the sqrt\_living increases, sqrt\_lot and bedrooms or bathrooms/bedrooms increases. However, the floors, bedrooms and bathrooms/bedrooms or sqrt\_living does not have a similar relationship.



Let’s visualize more features. When we look at the below boxplots, grade and waterfront effect price visibly. On the other hand, view seem to effect less but it also has an effect on price.



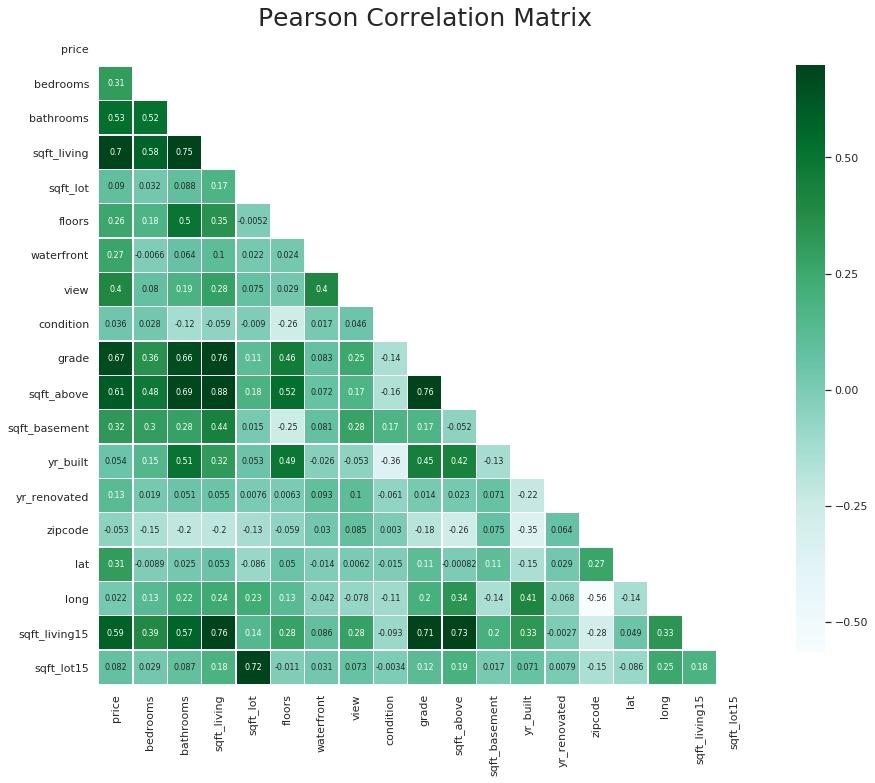
Further, I drew the 3D plot to determine the relation between the view, grade and year built. The below chart shows that the newer houses have better grades but we can not say much about the change in the view.



In this dataset, we have latitude and longtitude information for the houses. By using lat and long columns, I displayed the below heat map which is very useful for the people who does not know Seattle well. Also, if you select a spesific zip code, you may just see the heat map of this zip code’s neighborhood.

Having too many features in a model is not always a good thing because it might cause overfitting and worser results when we want to predict values for a new dataset. Thus, if a feature does not improve your model a lot, not adding it may be a better choice.

Another important thing is correlation. If there is very high correlation between two features, keeping both of them is not a good idea most of the time not to cause overfitting. For instance, if there is overfitting, we may remove sqt\_above or sqt\_living because they are highly correlated. This relation can be estimated when we look at the definitions in the dataset but to be sure correlation matrix should be checked. However, this does not mean that you must remove one of the highly correlated features. For example: bathrooms and sqrt\_living. They are highly correlated but I do not think that the relation among them is the same as the relation between sqt\_living and sqt\_above.



DATA PREPROCESSING:

A preprocessing on data might improve the model accuracy and make the model more reliable. It does not always have to improve our results but when we are conscious of the features and use a proper input, we might reach some outcomes easier. I tried various data mining techniques like transformation or normalization but in the end, decided to just use binning and created a new dataframe called df\_dm.

BINNING:

Data binning is a preprocessing technique used to reduce the effects of minor observation errors. I think it is worthwhile applying to some columns of this dataset. I applied binning to yr\_built and yr\_renovated. I added the ages and renovation ages of the houses when they were sold. Also, I partitioned these columns to intervals and you can observe this in the below histograms.

MULTIPLE REGRESSION:

I used a simple linear regression and found a poor fit. In order to improve this model I am planing to add more features. When we have more than one feature in a linear regression, it is defined as multiple regression. Then, it is time to create some complex models.

MULTIPLE REGRESSION -1:

I determined features at first sight by looking at the previous sections and used in my first multiple linear regression. As in the simple regression, I printed the coefficients which the model uses for the predictions. However, this time we must use the below definition for our predictions, if we want to make calculations manually.

MULTIPLE REGRESSION -2:

In addition to the previous subsection, I added more features to the features list. Also, I printed the coefficients of the model as in the previous subsection. When we look at the evaluation metrics, they improved significantly.

MULTIPLE REGRESSION -3:

In order to observe differences easily, I created a model with all features without any preprossing. Evaluation metrics improved remarkable again .

MULTIPLE REGRESSION-4:

This time I used the data obtained after preprocessing step.

REGULARIZATUON:

Regularization is designed to address the problem of overfitting and undefitting. Overfitting means high variance and is usually caused by a complicated function that creates a lot of unnecessary curves and angles unrelated to the data. This function fits training data well but might cause poor results for the test set. On the other hand, underfitting means low variance and a very simple model. This might also cause poor results too. Possible remedies are adjusting features manually or using some model selection algoritms which brings an extra workload. Conversely, when we apply regularization, all the features are kept and the model adjusts θj

. This especially works when we have a lot of slightly useful features. There are two widely used regularization types (Ridge and Lasso Regressions) and in this section, I used them.

When to use ridge vs lasso regression:

Many small/medium sized effects: use ridge.

Only a few variables with medium/large effect: use lasso.

Ridge Regression

Ridge regression is called L2 regularization and by adding a penalty, we obtain the below equation

RSSRIDGE=∑i=1m(hθ(xi)−yi)2+α∑j=1nθ2j

By changing the α

value, we can control the amount of the regularization. When we increase α

, regularization increases and the opposite is valid too. As a result of this, I selected different α

values and used a linear regression without regularization in order to observe the differences easily.

LASSO REGRESSION:

Lasso regression is called L1 regularization and it is defined as

RSSLASSO=∑i=1m(hθ(xi)−yi)2+α∑j=1n|θj|

The main difference between ridge and lasso is the penalty but α

Works the same way.

POLYNOMIAL REGRESSION:

For the linear models, the main idea is to fit a straight line to our data. However, if the data has a quadratic distribution, this time choosing a quadratic function and applying a polynomial transformation may give us better results. This time the hypothesis function is defined as

Hθ(X)=θ0+θ1x+θ2x2+…+θnxn

Since there are many variations for the polynomial regression, I prefered to show results by a new table and it can be seen from the below table that polynomial transformation improved the model fit a lot. On the other hand, while using polynomial transformation and deciding to degree, we should be very careful because it migh cause overfitting. Also, in the below table overfitting exists for some models. The 5-fold cross validation metrics are negative or low for these models although they have very high R-squared values for the training set.

K-NN REGRESSION:

I included k-NN regression in this kernel but actually, I did not expect to obtain a good result by using it. Also, k-NN does not give much insight. It is a very simple method and the idea behind the algorithm is similar to the k-NN classification. Briefly, it uses the weighted average, median or another statistic you want of k-nearest instances.

The evaluation metrics for the training, test sets and different k values were given in the below table. It can be seen that, k-NN was not very successful as expected.

CONCLUSUON:

When we look at the evaluation table, 2nd degree polynomial (all features, no preprocessing) is the best. However, I have doubts about its reliability. I would prefer the polynomial ridge regression (alpha=50000, degree=2, all features) but other models might be useful depending to the situation too.

In this kernel, I used scikit-learn which provides me all built in functions I need but if you want to examine the theory behind the logistic regression, which is used for classification and similar to the linear regression, and some other methods, you might also want to check my other kernel k-NN, Logistic Regression, k-Fold CV from Scratch.

CODE:

from subprocess import check\_output

print(check\_output(["ls", "../input"]).decode("utf8"))

import pandas as pd

import numpy as np

import scipy as sp

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn import metrics

%matplotlib inline

Data= pd.read\_csv(“../input/kc\_house\_data.csv”)

Data.head()

Data.describe(include=[np.number])

Data.isnull().sum()

Names=[‘price’,’bedrooms’,’bathrooms’,’sqft\_living’,’sqft\_lot’,’floors’,’waterfront’,’view’,’condition’,’grade’,’sqft\_above’,’sqft\_basement’,’zipcode’,’lat’,’long’]

Df=data[names]

Correlations= df.corr()

Fig=plt.figure()

Ax=fig.add\_subplot(111)

Cax=ax.matshow(correlations,vmin=-1,vmax=1)

Fig.colorbar(cax)

Ticks=np.arange(0,15,1)

Ax.set\_xticks(ticks)

Ax.set\_yticks(ticks)

Ax.set\_xticklabels(names)

Ax.set\_yticklabels(names)

Plt.show()

Data[‘waterfront’] = data[‘waterfront’].astype(‘category’,ordered=True)

Data[‘view’] = data[‘view’].astype(‘category’,ordered=True)

Data[‘condition’] = data[‘condition’].astype(‘category’,ordered=True)

Data[‘grade’] = data[‘grade’].astype(‘category’,ordered=False)

Data[‘zipcode’] = data[‘zipcode’].astype(‘category’,ordered=False)

Data.dtypes

Sns.regplot(x=’sqft\_living’,y=’price’,data=data)

Sns.regplot(x=’sqft\_basement’,y=’price’,data=data)

Sns.regplot(x=’sqft\_above’,y=’price’,data=data)

Sns.stripplot(x=’bedrooms’, y=’price’,data=data)

Sns.stripplot(x=’bathrooms’, y=’price’,data=data, size=5)

Sns.stripplot(x=’grade’, y=’price’,data=data, size=5)

Data=data[data[‘bedrooms’] < 10]

Data=data[data[‘bathrooms’]<8]

Data.head()

C=[‘bedrooms’,’bathrooms’,’sqft\_living’,’sqft\_above’,’grade’]

Df=data[c]

Df=pd.get\_dummies(df,columns=[‘grade’], drop\_first=True)

Y=data[‘price’]

X\_train,x\_test,y\_train,y\_test=train\_test\_split(df,y,train\_size=0.8,random\_state=42)

X\_train.head()

Reg=LinearRegression()

Reg.fit(x\_train,y\_train)

Print(‘Coefficients: \n’, reg.coef\_)

Reg.score(x\_test,y\_test)

Df=pd.get\_dummies(data,columns=[‘waterfront’,’view’,’condition’,’grade’,’zipcode’], drop\_first=True)

Y=data[‘price’]

Df= df.drop([‘date’,’id’,’price’],axis=1)

X\_train,x\_test,y\_train,y\_test=train\_test\_split(df,y,train\_size=0.8,random\_state=42)

reg.fit(x\_train,y\_train)

print(‘Coefficients: \n’, reg.coef\_)

print(metrics.mean\_squared\_error(y\_test, reg.predict(x\_test)))

print(reg.score(x\_test,y\_test))