IBM Coursera Advanced Data Science Capstone Project

- Predicting House Prices

by Ioana Bucuri

USE CASE

Establishing a clearly defined question starts with understanding the goal of the person who is asking the question.

The problem in this case sounds quite simple to understand, we want to build a model to predict the price of the houses.

The challenge is choosing the right predictors out of many possible options. I have to find out which features can affect the house prices.

DATA SET

- The dataset used in this notebook is an open dataset that can be found on the next link: https://www.kaggle.com/achyutanandaparida/dataset%20from%20%20house%20sales%20in%20king%20county,%20usa/notebooks
- This dataset contains house sale prices for King County. It includes homes sold between May 2014 and May 2015.
- This dataset contains 21613 samples and 22 columns, having 21 possible features.

QUALITY ASSESSMENT

- Using head() method, I have an overall look on how data is looking like.
- I use the method describe to obtain a statistical summary of the dataframe.
- Overall, the quality of data is quite ok, only few changes had to be done.
- I check for the missing values in the data set and, depending on how many are there, I am deciding whether to delete rows or replace them.
- Because I don't want to lose data, I decide to replace the missing values with the mean of the respective column.
- I check for the data types, I see they are OK, with the exception of date, which is of type object and I change that.

DATA EXPLORATION

- As data scientist or wannabe data scientists, we all know that Data does not mean information
- Therefore, data exploration is a very important step to see in which matter we can use this data set to help us gain information out of it. And also, how can we transform it.

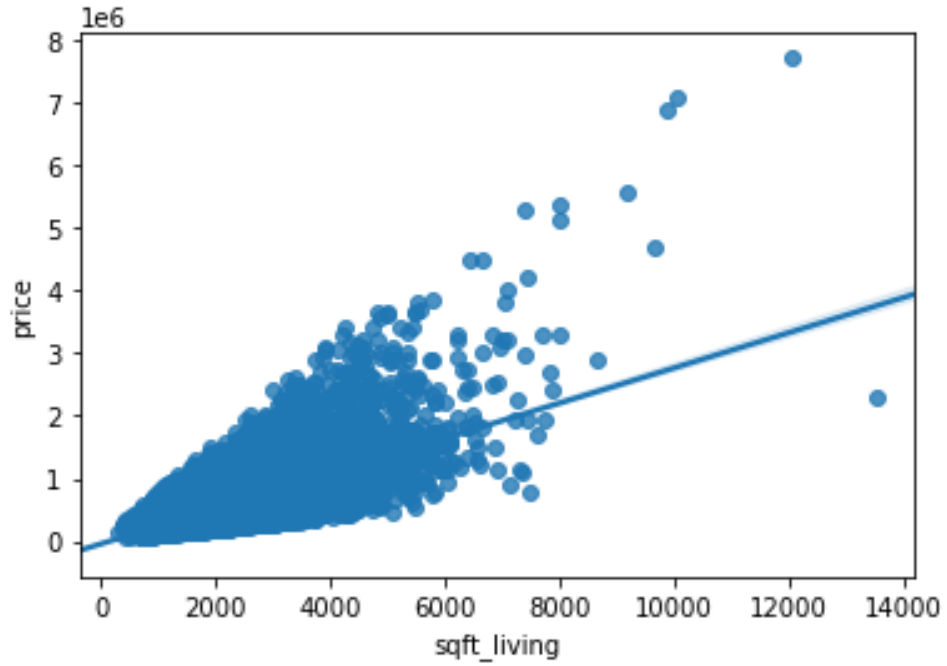
DATA EXPLORATION (continued)

- I check the correlation between the columns related to our target value which is the price, using pandas method corr():

```
: df.corr()['price'].sort_values(ascending = False)
```

```
: price            1.000000
  sqft_living      0.702035
  grade            0.667434
  sqft_above       0.605567
  sqft_living15    0.585379
  bathrooms        0.525738
  view             0.397293
  sqft_basement    0.323816
  bedrooms         0.308797
  lat              0.307003
  waterfront       0.266369
  floors           0.256794
  yr_renovated     0.126434
  sqft_lot         0.089661
  sqft_lot15       0.082447
  yr_built         0.054012
  condition        0.036362
  Unnamed: 0       0.027372
  long             0.021626
  id              -0.016762
  zipcode         -0.053203
  Name: price, dtype: float64
```

```
]: sns.regplot(x="sqft_living", y="price", data=df)

]: <AxesSubplot:xlabel='sqft_living', ylabel='price'>
```

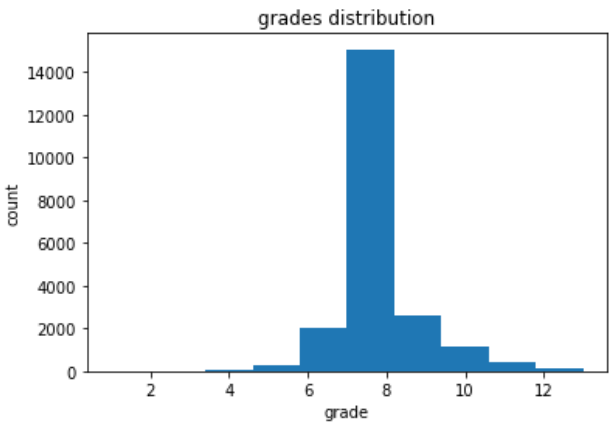


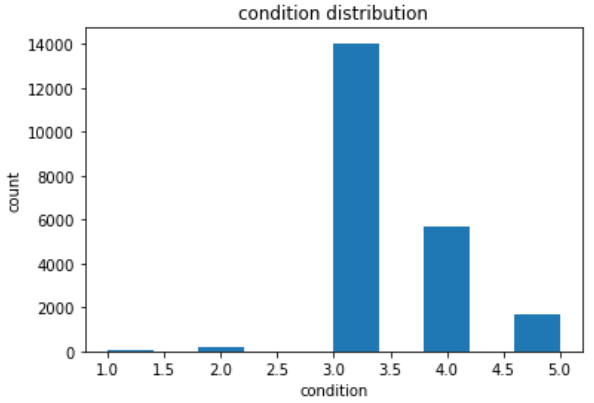
DATA EXPLORATION (cont'd)

I SEE THAT THE STRONGEST CORRELATION RELATED TO PRICE IS THE VALUE OF SQFT_LIVING AND I DECIDE TO PLOT IT.

Data Visualization

- I use plots to see the grades and conditions distribution.





Feature Engineering

- I am using as features the values related to the size of the house, the condition and number of bathrooms.
- Checking them out using describe method, I see that the values showing the surface have big values comparing to the others and will influence the result more due to its larger value. But this doesn't necessarily mean it is more important as a predictor. So we normalize the data to bring all the variables to the same range. I use a formula so the values will range from 0 to 1. The result can be seen on the next slide.

Feature Engineering
(continued)

```
df['sqft_living'] = df['sqft_living']/df['sqft_living'].max()
```

```
df['sqft_basement'] = df['sqft_basement']/df['sqft_basement'].max()
df['sqft_above'] = df['sqft_above']/df['sqft_above'].max()
df['sqft_living15'] = df['sqft_living15']/df['sqft_living15'].max()
df['sqft_lot'] = df['sqft_lot']/df['sqft_lot'].max()
```

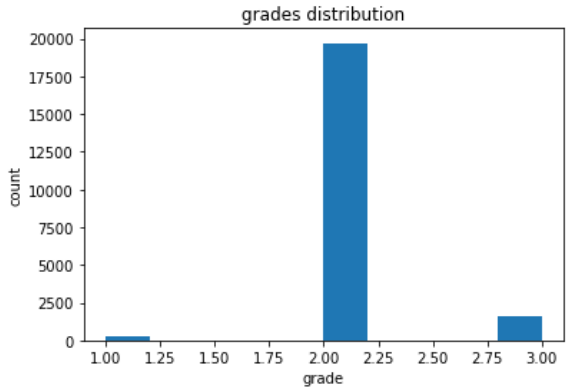
```
df['sqft_living15'] = df['sqft_living15']/df['sqft_living15'].max()
```

```
df[['sqft_living','sqft_basement', 'condition', 'grade', 'sqft_above', 'sqft_living15', 'bathrooms', 'sqft_lot']].head()
```

	sqft_living	sqft_basement	condition	grade	sqft_above	sqft_living15	bathrooms	sqft_lot
0	0.087149	0.000000	3	7	0.125399	0.215781	1.00	0.003421
1	0.189808	0.082988	3	7	0.230606	0.272142	2.25	0.004385
2	0.056869	0.000000	3	6	0.081828	0.438003	1.00	0.006056
3	0.144756	0.188797	5	7	0.111583	0.219002	3.00	0.003028
4	0.124077	0.000000	3	8	0.178533	0.289855	2.00	0.004893

Feature Engineering (continued)

- I notice that the grades have 13 unique values. I use binning to transform the values into categorical "bins". I split the grade data into 3 bins and I will have 3 groups: 1 for low grade, 2 for medium grade and 3 for good grade. I use a graph to see the distribution of the values now:



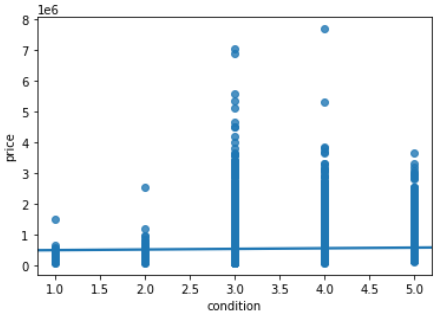

```
bins = np.linspace(min(df['grade']), max(df['grade']), 4)
group_names = [1,2,3]
df['grade'] = pd.cut(df['grade'], bins, labels=group_names, include_lowest=True )
```

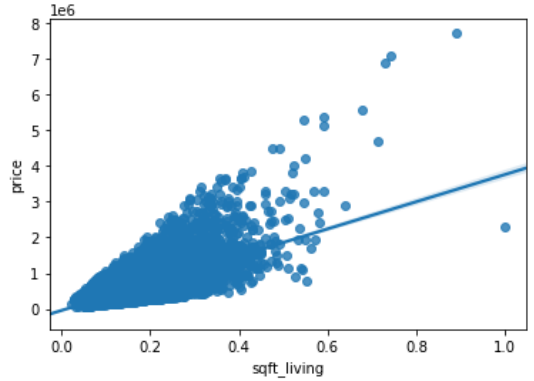
```
df.grade.unique()
```

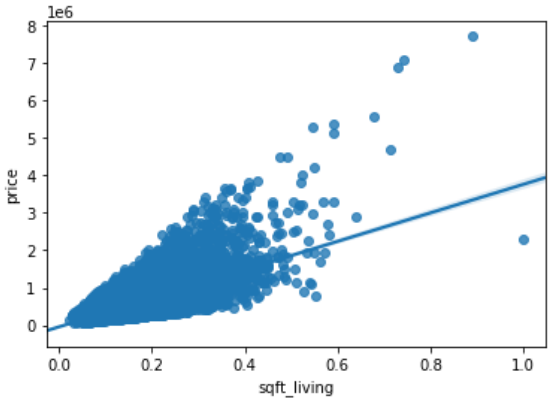
```
[2, 3, 1]
Categories (3, int64): [1 < 2 < 3]
```

Analyze feature patterns using Visualization

- I use regression plots to visualize the correlation between different features and the price. can see the difference between correlated features (sqft_living and sqft_above) and features which are not correlated, so I decide to remove from my list (condition).

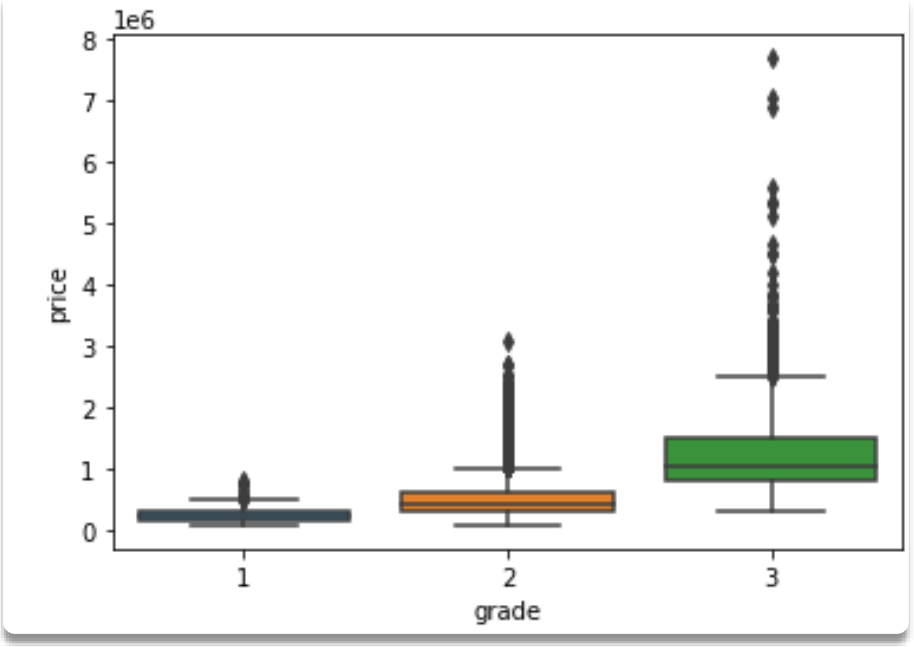






Analyze feature patterns using Visualization (continued)

- For the categorical variables like grade in my dataset, a boxplot is a good way to visualize.
- Considering the distributions of price between the different grades don't overlap, the grade is a good predictor.



Feature Engineering
Conclusion

- After conducting feature engineering, I include in my features list the following numerical variables:
- Square footage of the home
- Living room area in 2015
- Square footage of the basement
- number of bathrooms
- And the following categorial variable:
- grade

Algorithm used

- Regression is the process of predicting a continuous value, which is the case of this project, because I want to predict the price.
- In regression, there are two types of variables, a dependent variable, and one or more independent variables. In my case, I have one dependent variable, which is price, and more independent variables which are the features selected in the previous step.
- The dependent variable being a continuous value, not categorical, I use the linear regression.
- Considering there are more than one feature to be used as dependent variables, multiple linear regression is suitable.

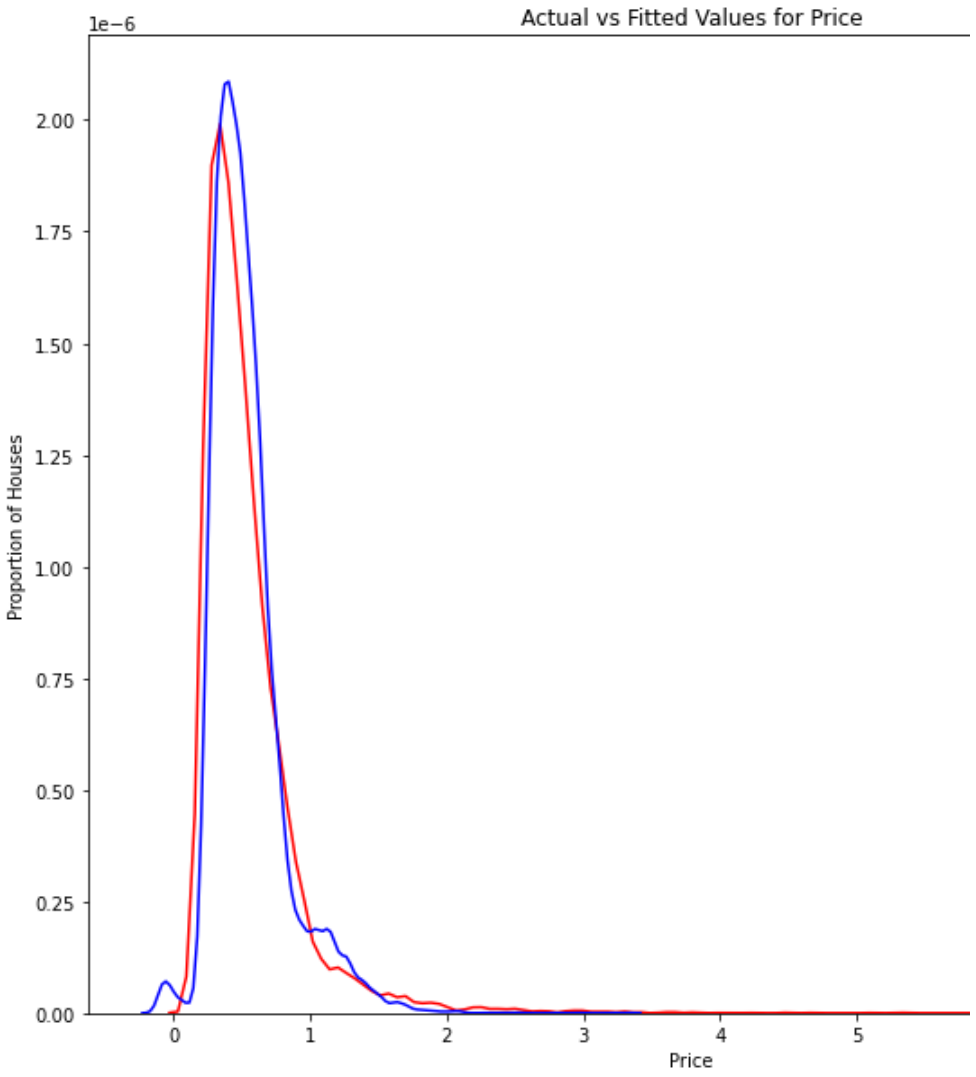
Algorithm used (continued)

- I create a linear regression instance and I fit the linear regression using the selected features. Then I output a prediction.

```
In [166]: lm = LinearRegression()
          lm

Out[166]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                           normalize=False)

In [171]: X=df[['sqft_living', 'bathrooms', 'sqft_living15', 'sqft_basement','grade']]
          Y=df['price']

In [172]: lm.fit(X,Y)

Out[172]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                           normalize=False)

In [173]: Yhat=lm.predict(X)
          Yhat[0:5]

Out[173]: array([301062.99123683, 603780.52696187, 303589.17977965, 499018.71544533,
                 420142.57970592])
```

Algorithm used (continued)

- Next I use a plot to see the actual versus fitted values in respect to the price.
- The fitted values are close to the actual values, as the distributions overlap, but it is room for improvement.



Model Performance

- Evaluation metrics, provide a key role in the development of a model, as it provides insight to areas that require improvement.
- R squared (R2) is not an error per se but is a popular metric for the accuracy of your model. It represents how close the data values are to the fitted regression line.
- Separating data into training and testing sets is an important part of model evaluation.
- We use the test data to get an idea how our model will perform in the real world.
- So, for my model, I use the metrics above.

Model Evaluation & Performance (continued)

- The data is randomly split into training and testing set; the testing set is 20% of the total dataset.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.20, random_state=1)
print("number of test samples :", x_test.shape[0])
print("number of training samples:",x_train.shape[0])
```

```
number of test samples : 4323
number of training samples: 17290
```

- Fitting the train set and checking the score:

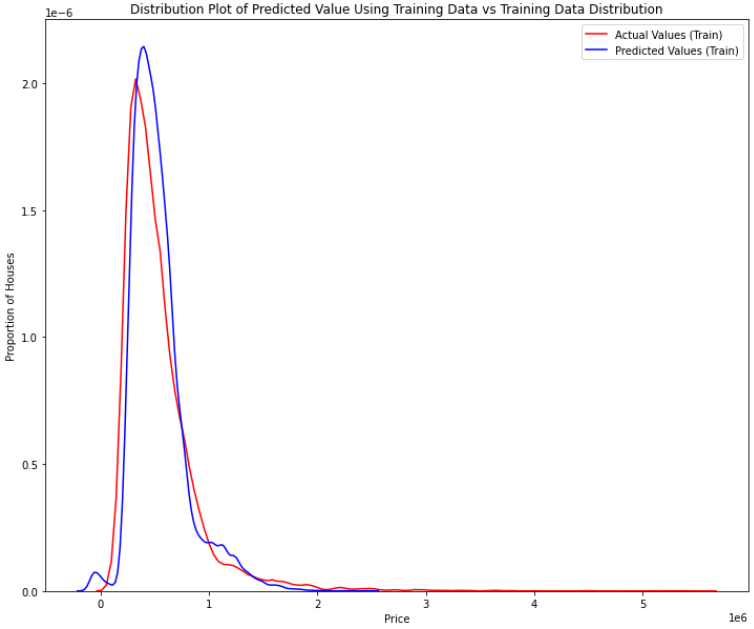
```
lm.fit(x_train[['sqft_living', 'bathrooms', 'sqft_living15', 'sqft_basement','grade']], y_train)
```

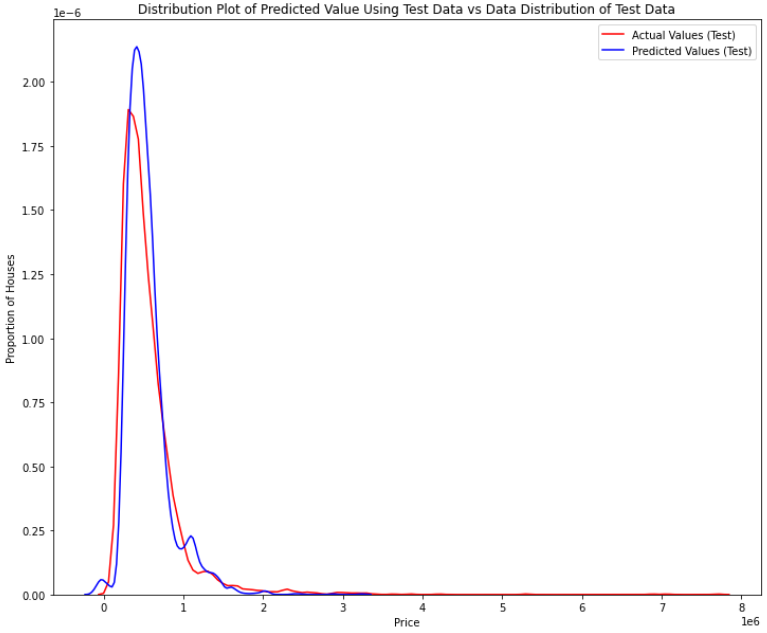
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

```
lm.score(x_test[['sqft_living', 'bathrooms', 'sqft_living15', 'sqft_basement','grade']], y_test)
```

Model Performance (continued)

- Next, I am evaluating the model using the train and test data separately:





Conclusion

- The Linear Regression Model performed good, but it was room for improvement.
- Therefore, I used a a polynomial regression which doesn't improve the accuracy and a Ridge Regression with different alpha parameters which improves the performance of the prediction.