**Project Name**: Housing Price Prediction

Introduction: The goal of this project is to produce a model for predicting housing prices given detailed information. The model can be useful for many purposes. From estimating the worth for a house that is not on market, to figuring out which component factors the most into a house, which can help decide if a remodel makes sense financially.

Objective: In this project, we will develop and evaluate the performance and the predictive power of several models trained and tested on data collected from houses in King County.

Our task is to predict the housing price taking input feature as the combination of one or more available in the data.

A house value is simply more than location and square footage. Like the features that make up a person, an educated party would want to know all aspects that give a house its value. We are going to take advantage of all of the feature variables available to use and use it to analyse and predict house prices.

We are going to break everything into logical steps that allow us to ensure the cleanest, most realistic data for our model to make accurate predictions from.

1. Data Exploration and Anomaly detection
2. Modelling and Predictions
3. Conclusion

**Data Exploration and Anomaly detection**

1. Load Data and Packages: We load the data downloaded from Kaggle and check for the sizes.

Training data: Consists of 9761 rows and 21 columns.

Test Data : Consists of 2217 rows and 21 columns.

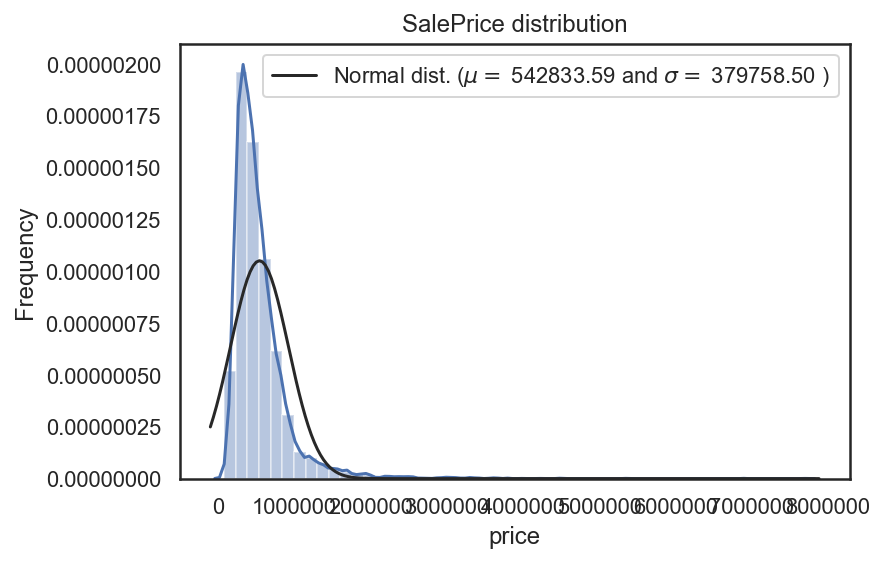
Validate data : consists of 9635 rows and 21 columns.

We are using Pandas, Numpy, scipy and Scikitlearn,seaborn & matplotlib packages for this project.

1. Analysing the Test Variable (Sale Price): Let's check out the most interesting feature in this study: Sale Price.

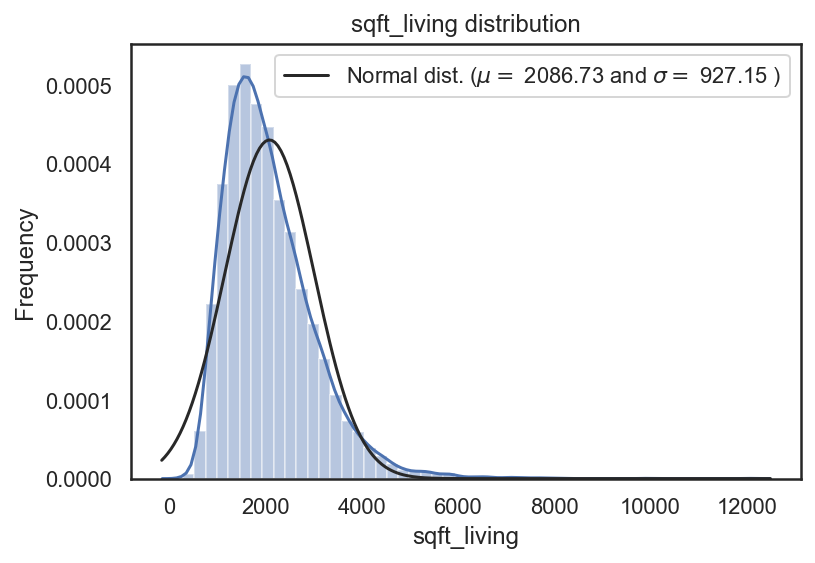
Statistics for Housing Price 🡪

* Minimum price: $8.000000e+04
* Maximum price: $7700000.0
* Mean price: $5.428336e+05
* Median price $450000.0
* Standard deviation of prices: $ 3.797779e+05
* 25% 3.200000e+05
* 50% 4.500000e+05
* 75% 6.490000e+05



Skewness: 4.296023 & Kurtosis: 38.871048

Does not Look like a normal distribution, looking at the kurtosis score, we can see that there is a very nice peak. Also looking at the skewness score, we can see that the sale prices deviate from the normal distribution.



**Multivariable Analysis**

Let's check out all the variables! There are two types of features in housing data, categorical and numerical.

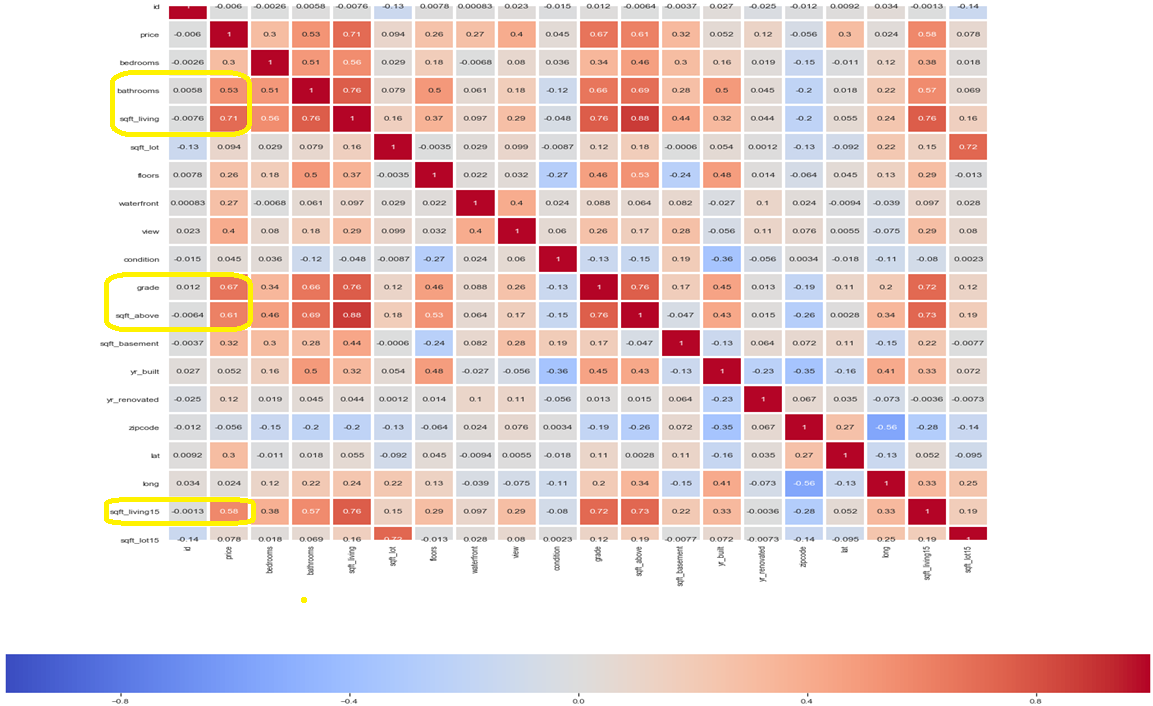
Categorical data is just like it sounds. It is in categories. It isn't necessarily linear, but it follows some kind of pattern. For example, take a feature of "Zipcode".

Numerical data is data in number form. These features are in a linear relationship with each other. For example, a 2,000 square foot place is 2 times "bigger" than a 1,000 square foot place. Plain and simple. Simple and clean.

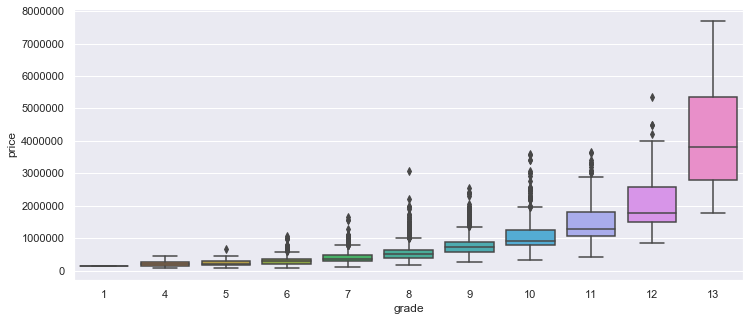
With 21 features, how could we possibly tell which feature is most related to house prices? Good thing we have a correlation matrix. Let's do it!

Correlation Matrix Heatmap🡪

Highlighted features below sqft\_living, grade, sqft\_above, sqft\_living15, bathrooms have high correlation with price.



Relationship between the Grade & Sale Price 🡪



Clearly shows higher the grade 🡪 higher the price.

ZIPCODE :

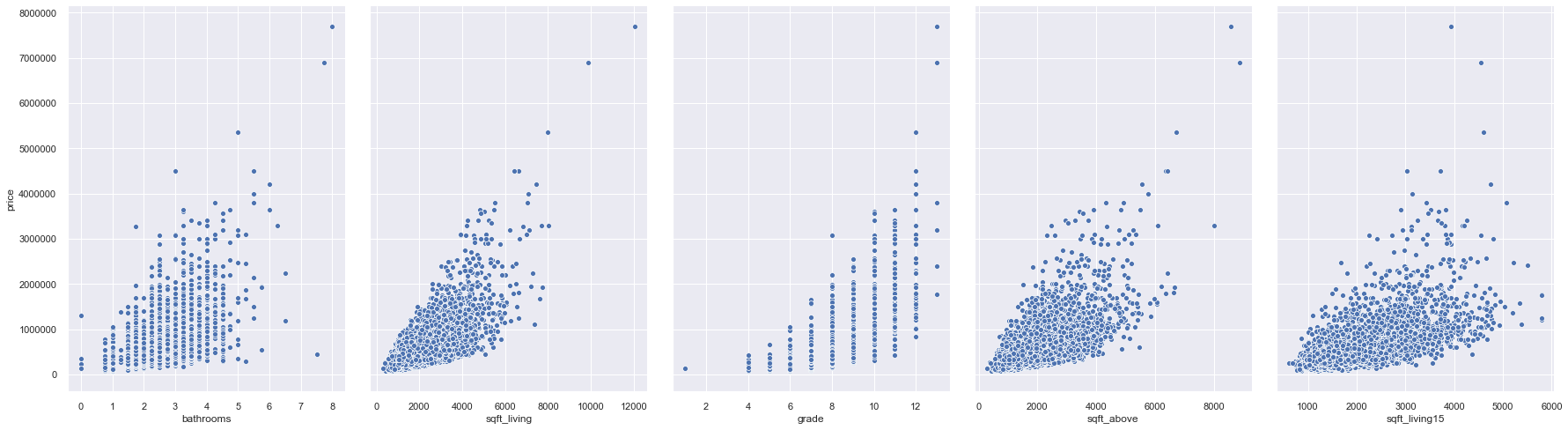
max houses in a particular zipcode :

98115 278

min houses in a particular zipcode :

98039 20

Target vs high correlation features🡪



We are going to use the above high correlation features as :

feature\_cols=['bathrooms', 'sqft\_living','grade','sqft\_above', 'sqft\_living15']

There is no missing data in the given data.

Relationship between Sqft\_living Sale Price 🡪

99.44% of the data lies between sqft\_living [0-6000] & price[100000 – 3000000 ]

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **sqft\_living\_grp** | **(0, 1000]** | **(1000, 2000]** | **(2000, 3000]** | **(3000, 4000]** | **(4000, 5000]** | **(5000, 6000]** | **(6000, 7000]** | **(7000, 8000]** | **(8000, 9000]** | **(9000, 10000]** |
| **price\_grp** |  |  |  |  |  |  |  |  |  |  |
| (0, 100000] | 11 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| (100000, 200000] | **146** | **213** | **4** | **0** | **0** | **0** | **0** | 0 | 0 | 0 |
| (200000, 300000] | **211** | **1311** | **193** | **2** | **0** | **0** | **0** | 0 | 0 | 0 |
| (300000, 400000] | **188** | **1120** | **565** | **48** | **0** | **0** | **0** | 0 | 0 | 0 |
| (400000, 500000] | **98** | **923** | **539** | **82** | **10** | **1** | **0** | 0 | 0 | 0 |
| (500000, 600000] | **24** | **546** | **552** | **114** | **11** | **0** | **0** | 0 | 0 | 0 |
| (600000, 700000] | **8** | **288** | **467** | **121** | **9** | **2** | **0** | 0 | 0 | 0 |
| (700000, 800000] | **3** | **107** | **281** | **177** | **14** | **3** | **0** | 0 | 0 | 0 |
| (800000, 1000000] | **0** | **71** | **281** | **277** | **51** | **6** | **0** | 0 | 0 | 0 |
| (1000000, 2000000] | **0** | **17** | **166** | **232** | **127** | **36** | **9** | 4 | 0 | 0 |
| (2000000, 3000000] | **0** | **0** | **1** | **11** | **27** | **18** | **3** | 1 | 0 | 0 |
| (3000000, 4000000] | 0 | 0 | 0 | 2 | 7 | 7 | 3 | 4 | 1 | 0 |
| (4000000, 5000000] | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 |
| (5000000, 6000000] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| (6000000, 7000000] | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** |

Methods Used : Regression is a machine learning tool that helps one make predictions by learning – from the existing statistical data – the relationships between target parameter and a set of other parameters.

Splitting Data : One benefit of splitting a dataset into some ratio of training and testing subsets is that it prevents 'overfitting', a problem where the model has become too finely tuned to the data its been given so that it is unable to create accurate predictions on data it hasn't been trained on.

**Modelling and Predictions:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Features used** | **R^2** | **Package** | **Comments** |
| Simple Linear Regression | ['sqft\_living'] | **0.5** | **scikitlearn** |  |
| Multiple Linear Regression | ['bathrooms', 'sqft\_living','grade','sqft\_above', 'sqft\_living15'] | **0.55** | **scikitlearn** |  |
| Multiple Linear Regression | ['bedrooms', 'bathrooms', 'sqft\_living',  'sqft\_lot', 'floors', 'waterfront', 'view', 'condition', 'grade',  'sqft\_above', 'sqft\_basement', 'yr\_built', 'yr\_renovated', 'zipcode',  'sqft\_living15', 'sqft\_lot15'] | **0.641** | **scikitlearn** |  |
| Polynomial regression | ['sqft\_living'] | **0.53** | **scikitlearn** | By creating polynomial dataframes for sqft\_living up to 15th order. Found that R^2 for 3rd order polynomial yields highest R^2 1st order: R^2: 0.5001767814877556 2nd order: R^2: 0.5253323667647642 3rd order: R^2: **0.5298187801979004** 4th order: R^2: 0.5220823928462339 |
| CrossValidation | ['bathrooms', 'sqft\_living','grade','sqft\_above', 'sqft\_living15'] | **0.54** | **scikitlearn** | **Highest R2 Training:** 8th split 0.704177  (After split X1\_test has 98 rows & X1\_train has 9663 rows. Split starts at 1112) **Highest R2 in Training+Validation¶** 139th split 0.701195 ( After split X1\_Test has 75 rows & X1\_Train has 19321 rows. Split starts at 19321) |
| Interaction | ['sqft\_living','interaction'] | **0.696** | **scikitlearn** | Interaction = Bedrooms \* Bathrooms ['interaction\_grade\_sqft\_above'] = ['sqft\_above'] \* ['grade'] ['interaction\_grade\_sqft\_living']= ['sqft\_living'] \* ['grade'] |
| OLS regression | ['bathrooms', 'sqft\_living','grade','sqft\_above', 'sqft\_living15'] | **0.544** | **statsmodel** |  |
| OLS regression | ['bedrooms', 'bathrooms', 'sqft\_living',  'sqft\_lot', 'floors', 'waterfront', 'view', 'condition', 'grade',  'sqft\_above', 'sqft\_basement', 'yr\_built', 'yr\_renovated', 'zipcode',  'sqft\_living15', 'sqft\_lot15'] | **0.654** | **statsmodel** |  |
| DecisionTreeRegressor | ['bathrooms', 'sqft\_living','grade','sqft\_above', 'sqft\_living15'] | **0.64** | **scikitlearn** |  |
| DecisionTreeRegressor | ['bedrooms', 'bathrooms', 'sqft\_living',  'sqft\_lot', 'floors', 'waterfront', 'view', 'condition', 'grade',  'sqft\_above', 'sqft\_basement', 'yr\_built', 'yr\_renovated', 'zipcode',  'sqft\_living15', 'sqft\_lot15'] | **0.7** | **scikitlearn** | Optimizing the dataset as 99.44% of the data lies between : sqft\_living [0-6000] & price[100000 – 3000000 ] R^2 of the Training.: 0.9938343027722922 Test Variance score: 0.70 RMSE: 195989.56 |

**Conclusion:**

Among the several models tried and with combination of features for training the models, the one which stood out is Decisiontreeregressor from scikitlearn. Also optimizing the dataset and including the below interaction parameters into the training helped in improving the model training score to 99%. The test R^2 score maximum achieved is 0.7.

Here is the sample output of predicted vs actual test values**:**

|  |  |  |
| --- | --- | --- |
| **Actual Price** | **Predicted Price** | **Residuals** |
| 270000 | 3.66E+05 | -9.60E+04 |
| 487500 | 5.74E+05 | -8.60E+04 |
| 292500 | 3.71E+05 | -7.85E+04 |
| 357500 | 4.25E+05 | -6.75E+04 |
| 500000 | 5.40E+05 | -4.00E+04 |
| 1320000 | 1.36E+06 | -4.00E+04 |
| 206000 | 2.45E+05 | -3.90E+04 |
| 201700 | 2.30E+05 | -2.82E+04 |
| 400000 | 4.25E+05 | -2.50E+04 |
| 295500 | 3.20E+05 | -2.45E+04 |
| 199950 | 2.24E+05 | -2.40E+04 |
| 349950 | 3.71E+05 | -2.14E+04 |
| 327555 | 3.48E+05 | -2.04E+04 |
| 610000 | 6.26E+05 | -1.60E+04 |
| 360000 | 3.76E+05 | -1.56E+04 |
| 280000 | 2.95E+05 | -1.45E+04 |
| 375000 | 3.86E+05 | -1.07E+04 |
| 305000 | 3.10E+05 | -5.00E+03 |
| 387000 | 3.88E+05 | -7.57E+02 |
| 919990 | 9.15E+05 | 5.49E+03 |
| 480680 | 4.75E+05 | 5.61E+03 |
| 788000 | 7.79E+05 | 9.00E+03 |
| 345000 | 3.36E+05 | 9.00E+03 |
| 301000 | 2.90E+05 | 1.10E+04 |
| 349950 | 3.35E+05 | 1.50E+04 |
| 465000 | 4.50E+05 | 1.50E+04 |
| 431000 | 4.10E+05 | 2.10E+04 |
| 776000 | 7.55E+05 | 2.12E+04 |
| 405000 | 3.82E+05 | 2.35E+04 |
| 249500 | 2.17E+05 | 3.30E+04 |
| 740000 | 7.00E+05 | 4.00E+04 |
| 663000 | 6.10E+05 | 5.29E+04 |
| 469000 | 4.13E+05 | 5.58E+04 |
| 466950 | 4.10E+05 | 5.70E+04 |
| 435000 | 3.71E+05 | 6.36E+04 |
| 738000 | 6.70E+05 | 6.80E+04 |
| 479000 | 4.11E+05 | 6.85E+04 |
| 680000 | 5.98E+05 | 8.23E+04 |
| 914600 | 8.32E+05 | 8.29E+04 |