HOUSING PROJECT

SUBMITTED BY:

NAINCY JOSHI

INTRODUCTION

• Business Problem Framing

Houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy. It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases. Predictive modelling, Market mix modelling, recommendation systems are some of the machine learning techniques used for achieving the business goals for housing companies. Our problem is related to one such housing company. A US-based housing company named Surprise Housing has decided to enter the Australian market. The company uses data analytics to purchase houses at a price below their actual values and flip them at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia. The data is provided in the CSV file below. The company is looking at prospective properties to buy houses to enter the market. You are required to build a model using Machine Learning in order to predict the actual value of the prospective properties and decide whether to invest in them or not. For this company wants to know:

Which variables are important to predict the price of variable?

How do these variables describe the price of the house?

Conceptual Background of the Domain Problem

You are required to model the price of houses with the available independent variables. This model will then be used by the management to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market

• Technical Requirement

Data contains 1460 entries each having 81 variables. • Data contains Null values. You need to treat them using the domain knowledge and your own understanding. • Extensive EDA has to be performed to gain relationships of important variable and price. • Data contains numerical as well as categorical variable. You need to handle them accordingly. • You have to build Machine Learning models, apply regularization and determine the optimal values of Hyper Parameters. •

DATASET:

- MSSubClass: Identifies the type of dwelling involved in the sale.
- MSZoning: Identifies the general zoning classification of the sale.
- LotFrontage: Linear feet of street connected to property.
- LotArea: Lot size in square feet.
- Street: Type of road access to property.
- Alley: Type of alley access to property.
- LotShape: General shape of property.
- LandContour: Flatness of the property
- Utilities: Type of utilities available
- LotConfig: Lot configuration
- LandSlope: Slope of property
- Neighborhood: Physical locations within Ames city limits
- Condition1: Proximity to various conditions
- Condition2: Proximity to various conditions (if more than one is present)
- BldgType: Type of dwelling and so on…..

- We have two dataset i.e train_data and the test_data.
- In train_data ‘SalePrice’ of the house is given but in test_data,no SalePrice is given,so we have to predict the SalePrice of the test_data.
- We have 1168 rows × 75 columns in the train_data while in test_data we have

  292 rows × 74 columns.

- We have used train_data.info() and the test_data.info() to check the datatype of the feature and train_data.isna().sum() will give us the missing value of the dataset.
- There are lot of missing value in the dataset and there are lot of features which are categorical value which has to be converted into the numeric feature.
- Train_Data.describe() and the test_data.describe() will give us the count value and the mean,standard deviation,and the Quantile value.

In [595]:
```
train_data.describe()
```

Out[595]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	...	WoodDeck
count	1168.000000	954.00000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1161.000000	1168.000000	1168.000000	...	1168.000(
mean	56.767979	70.98847	10484.749144	6.104452	5.595890	1970.930651	1984.758562	102.310078	444.726027	46.647260	...	96.206:
std	41.940650	24.82875	8957.442311	1.390153	1.124343	30.145255	20.785185	182.595606	462.664785	163.520016	...	126.158
min	20.000000	21.00000	1300.000000	1.000000	1.000000	1875.000000	1950.000000	0.000000	0.000000	0.000000	...	0.000(
25%	20.000000	60.00000	7621.500000	5.000000	5.000000	1954.000000	1966.000000	0.000000	0.000000	0.000000	...	0.000(
50%	50.000000	70.00000	9522.500000	6.000000	5.000000	1972.000000	1993.000000	0.000000	385.500000	0.000000	...	0.000(
75%	70.000000	80.00000	11515.500000	7.000000	6.000000	2000.000000	2004.000000	160.000000	714.500000	0.000000	...	171.000(
max	190.000000	313.00000	164660.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	1474.000000	...	857.000(

8 rows × 37 columns

In [596]:
```
test_data.describe()
```

Out[596]:

	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	...	GarageAre
count	292.000000	247.000000	292.000000	292.000000	292.000000	292.000000	292.000000	291.000000	292.000000	292.000000	...	292.00000
mean	57.414384	66.425101	10645.143836	6.078767	5.493151	1972.616438	1985.294521	109.171821	439.294521	46.157534	...	457.45890
std	43.780649	21.726343	13330.669795	1.356147	1.063267	30.447016	20.105792	175.030021	429.559675	152.467119	...	210.78559
min	20.000000	21.000000	1526.000000	3.000000	3.000000	1872.000000	1950.000000	0.000000	0.000000	0.000000	...	0.00000

- Nulls: The data have 19 features with nulls, five of then area categorical and with more then 47% of missing ration. They are features to drop or use them to create another more interesting feature:
    - PoolQC
    - MiscFeature
    - Alley
    - Fence
    - ID
    - Electrical

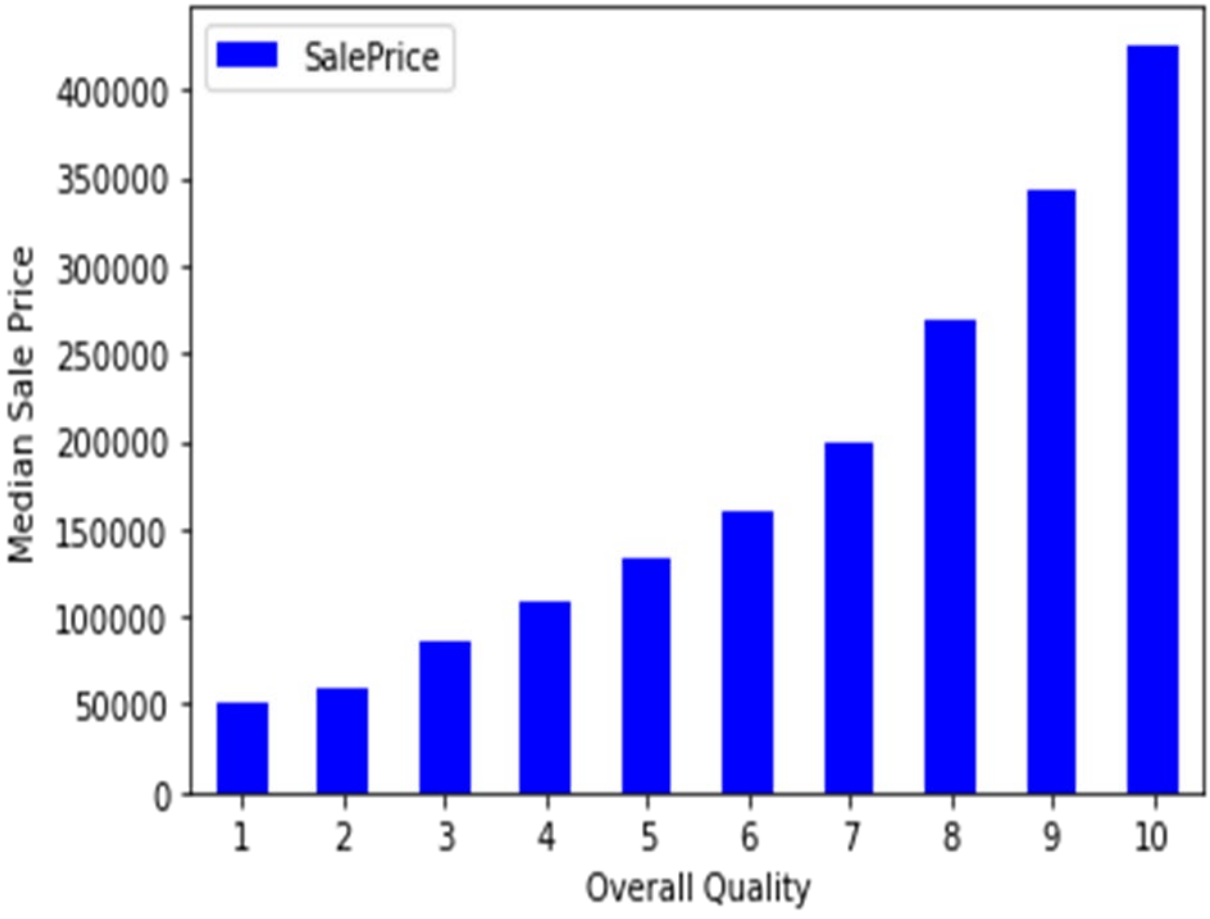
Features ***high skewed right***, ***heavy-tailed distribution***, and with ***high correlation*** to Sales Price.

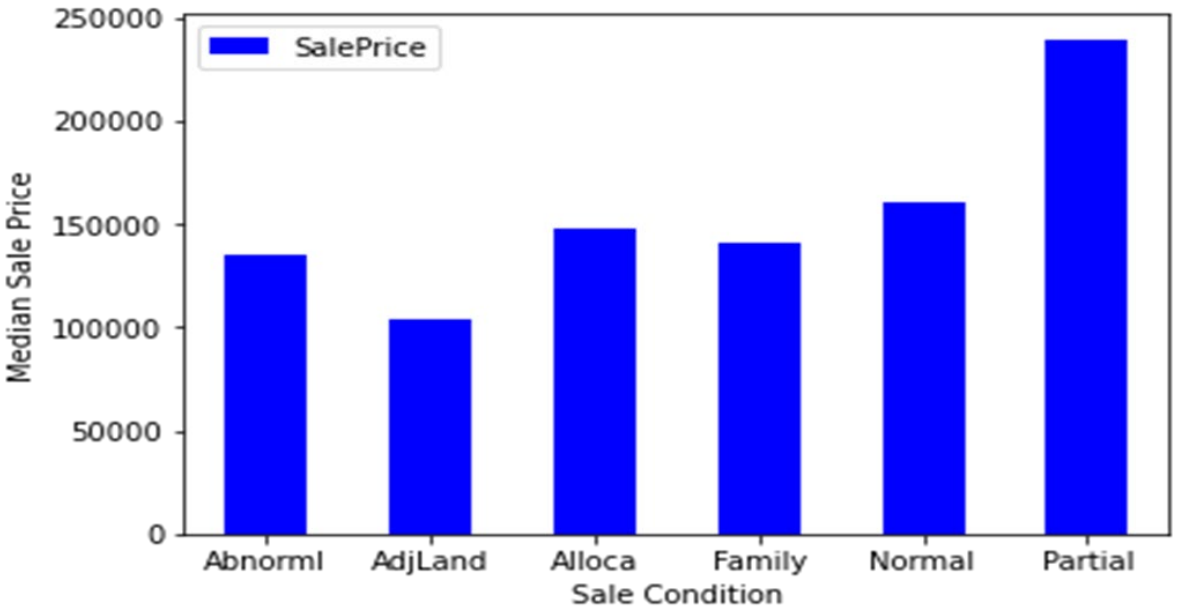
- OverallQual: Rates the overall material and finish of the house (1 = Very Poor, 10 = Very Excellent)
- GrLivArea: Above grade (ground) living area square feet
- GarageCars: Size of garage in car capacity
- GarageArea: Size of garage in square feet
- TotalBsmtSF: Total square feet of basement area
- 1stFlrSF: First Floor square feet

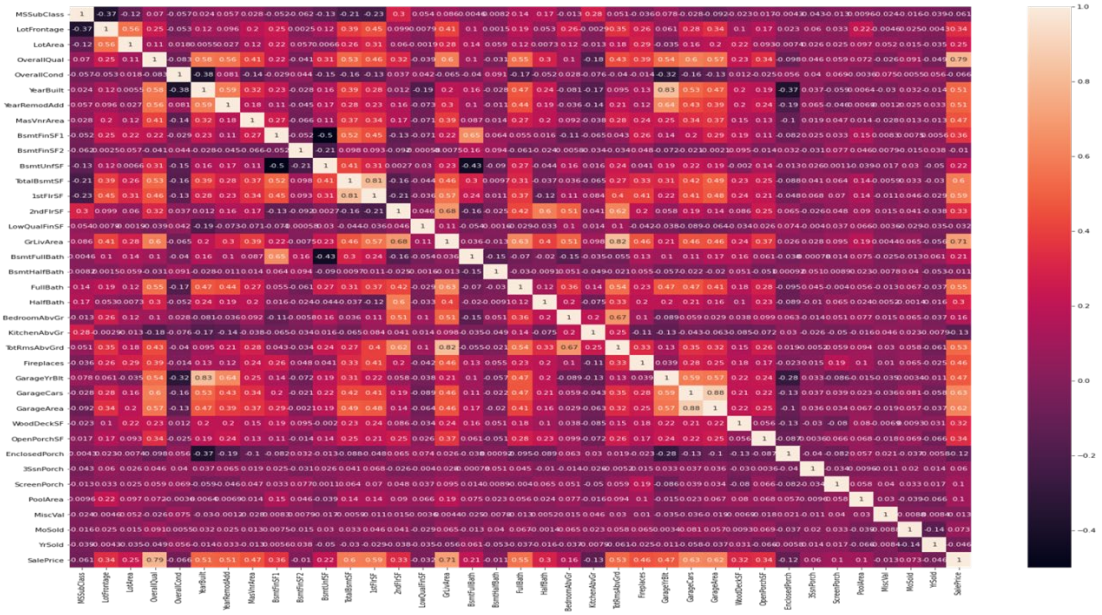
    - FullBath: Full bathrooms above grade
    - TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
    - YearBuilt: Original construction date

- Features ***high skewed***, ***heavy-tailed distribution***, and with ***low correlation*** to Sales Price.
    - MiscVal
    - TSsnPorch
    - LowQualFinSF
    - BsmtFinSF2
    - BsmtHalfBa

VISUALIZATION:




SalePrice
250000
200000
150000
100000
50000
0
Median Sale Price
Abnorml
AdjLand
Alloca
Family
Normal
Partial
Sale Condition

- 

The numeric data which are present in the dataset:

- MSSubClass
- LotFrontage
- LotArea
- OverallQual
- OverallCond
- YearBuilt
- YearRemodAdd
- MasVnrArea
- BsmtFinSF1
- BsmtFinSF2
- BsmtUnfSF

- TotalBsmtSF
- 1stFlrSF
- 2ndFlrSF
- LowQualFinSF
- GrLivArea
- BsmtFullBath
- BsmtHalfBath
- FullBath
- HalfBath
- BedroomAbvGr
- KitchenAbvGr
- TotRmsAbvGrd
- Fireplaces
- GarageYrBlt

- GarageCars
- GarageArea
- WoodDeckSF
- OpenPorchSF
- EnclosedPorch
- 3SsnPorch
- ScreenPorch
- PoolArea
- MiscVal
- MoSold YrSold
- SalePrice

The categorical data which are present in the dataset

- MSZoning
- Street
- LotShape
- LandContour
- Utilities
- LotConfig
- LandSlope
- Neighborhood
- Condition1
- Condition2
- BldgType
- HouseStyle

- RoofStyle
- BsmtQual
- BsmtCond
- BsmtExposure
- BsmtFinType1
- BsmtFinType2
- Heating
- Foundation
- RoofMatl
- Exterior 1st
- Exterior2nd object
- MasVnrType
- ExterQual
- ExterCond

- HeatingQC
- CentralAir
- KitchenQual
- Functional
- FireplaceQu
- GarageType
- GarageFinish
- GarageQual
- GarageCond
- PavedDrive
- SaleType
- SaleCondition

Visualization observation:

- As the “overallQuality” increases, the SalePrice of the houses also increases.
- There is a skewness in the SalePrice.
- There are lot of skewness and outlier in many columns
- We will be using LabelEncoder() for encoding the dataframe of string label.

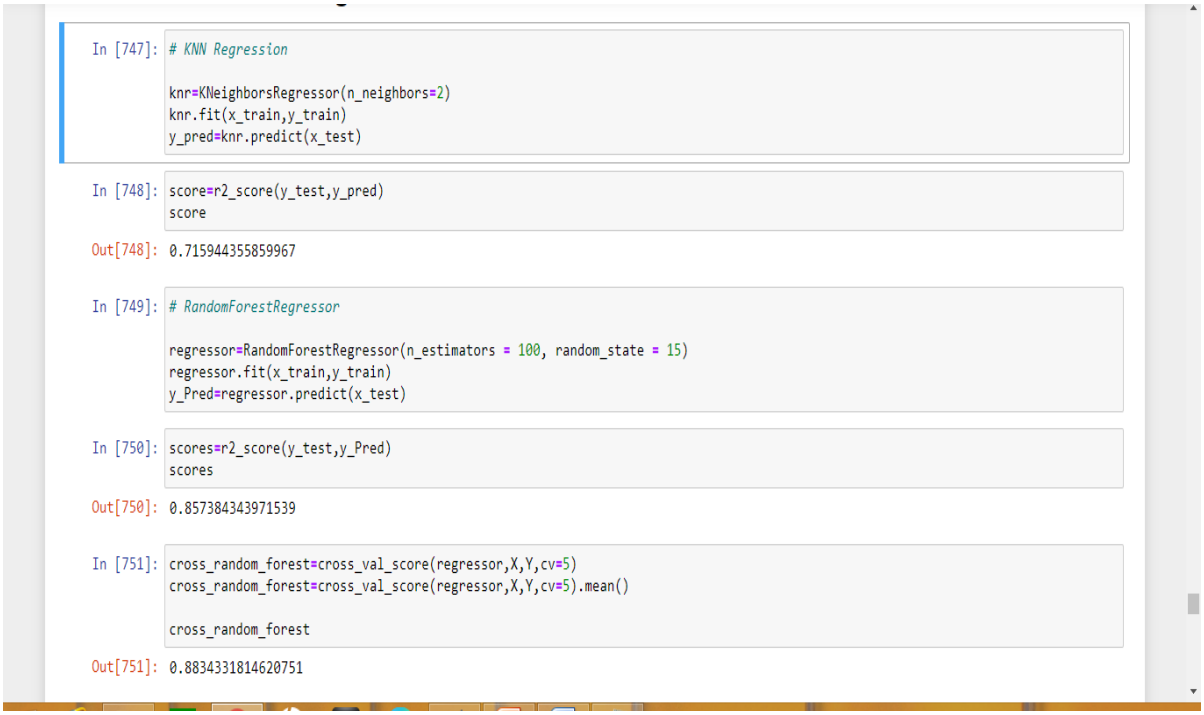
Preprocessing Model:

- Imputing the missing value and the 'NAN' value in the train_data set and the test_data set.
- We will fill the “Nan” value using data.fillna(0),we can also use bfill and ffill to fill the Nan value.
- For imputing the missing value,we can use mean,mode and the median.
- If the missing value is ‘int’ we can use mean/mode,if it is an object,we can use

‘mode’ and the the missing value is a float then we can use ‘mean’.

- The train_data and the test_data should have the same feature in the dataset.
- If they are not “set” to be true, we will make them as a True to set.
- We can use the replace or add the column in the dataset to make it as a True

Model Building:



```
In [747]: # KNN Regression

          knr=KNeighborsRegressor(n_neighbors=2)
          knr.fit(x_train,y_train)
          y_pred=knr.predict(x_test)

In [748]: score=r2_score(y_test,y_pred)
          score

Out[748]: 0.715944355859967

In [749]: # RandomForestRegressor

          regressor=RandomForestRegressor(n_estimators = 100, random_state = 15)
          regressor.fit(x_train,y_train)
          y_Pred=regressor.predict(x_test)

In [750]: scores=r2_score(y_test,y_Pred)
          scores

Out[750]: 0.857384343971539

In [751]: cross_random_forest=cross_val_score(regressor,X,Y,cv=5)
          cross_random_forest=cross_val_score(regressor,X,Y,cv=5).mean()

          cross_random_forest

Out[751]: 0.8834331814620751
```

In [752]:
```
# Logistic Regression

lr= LogisticRegression()
lr.fit(x_train, y_train)
Y_pred = lr.predict(x_test)
```

In [753]:
```
score1=r2_score(y_test,y_Pred)
score1
```

Out[753]: 0.857384343971539

In [757]:
```
# SGDRegressor

sgd=SGDRegressor()
sgd.fit(x_train, y_train)
Y_pred = sgd.predict(x_test)
```

In [758]:
```
score2=r2_score(y_test,y_Pred)
score2
```

Out[758]: 0.857384343971539

In [760]:
```
# DecisionTreeRegressor

dt=DecisionTreeRegressor()
dt.fit(x_train, y_train)
Y_pred = dt.predict(x_test)
```

In [761]:
```
score3=r2_score(y_test,y_Pred)
score3
```

Out[761]: 0.857384343971539

In [762]:
```
cross_dt=cross_val_score(dt,X,Y,cv=5)
cross_dt=cross_val_score(dt,X,Y,cv=5).mean()

cross_dt
```

Out[762]: 0.7344939342998975

-
-

Model Building Conclusion:

- We have find that RandomForestRegressor has the least difference between the accuracy score and the cross_validation.
- So we will perform hyper-parameter tunning to this model so that we can improve more to give the best accuracy score.

Hyper_parameter Tunning:

In [763]:
```
# Random Forest Classification
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score


array = [10, 50, 100, 200, 500]

for num_trees in array:
    max_features = 5
    kfold = KFold(n_splits=10)
    model_1 = RandomForestRegressor(n_estimators=num_trees, max_features=max_features)
    results = cross_val_score(model_1, X, Y, cv=kfold)
    print('num_trees : %f (%f)'%(num_trees, results.mean()))
```

```
num_trees : 10.000000 (0.867347)
num_trees : 50.000000 (0.885797)
num_trees : 100.000000 (0.885949)
num_trees : 200.000000 (0.887634)
num_trees : 500.000000 (0.887198)
```

In [764]:
```
max_features = 5
model_2 = RandomForestRegressor(n_estimators=500, max_features=max_features)
model_2.fit(X, Y)
predictions = model_2.predict(test_data)
predictions
```

Out[764]: array([318.16 , 186.896, 249.836, 150.738, 275.482, 34.618, 138.762

Predicting the “SalePrice” of the test_data:

```
In [766]: test_dummy=pd.get_dummies(test_data.iloc[:,0:74])

In [767]: regressor.fit(x_train,y_train)
          predict=regressor.predict(test_dummy)

In [768]: predict.shape

Out[768]: (292,)

In [769]: print(predict)

[297.54 166.22 140.54  79.59 111.42  25.79  67.56 199.27 161.65  81.78
  11.88  70.51  56.67  96.95 172.22  52.04  60.16  72.04 110.1  130.61
 101.67  66.96  83.33  19.59  23.98  59.67  76.28  55.19  89.45  49.69
  67.52 138.93 124.22  91.63  32.45  95.75  87.    37.62  85.77  78.4
  29.81 193.42 118.7  116.36  67.22  69.76  54.83  29.8  162.31 290.5
  71.59 105.85  27.73  16.43 137.65  62.18  70.13  97.35  42.17 151.09
  31.78  97.4   68.58  64.17 135.37  33.43 103.84 122.7   82.62  81.18
 180.04  88.28  96.12  84.57  82.97 146.99 198.98 125.09 188.86  79.06
 119.63  59.33  78.53  82.29  94.92 154.22  53.99 196.7   70.77  76.03
 131.95  57.39  50.35  46.99 102.99  83.65 146.36  93.19 210.62  63.23
 159.66  34.02  46.27  56.11 100.88  81.59 172.52  86.77  92.36 116.7
 115.01  87.34 101.53 121.1   42.53  37.44  67.77 104.17  96.87  43.58
  36.59 106.62 146.33  65.99  72.56 112.3   57.18 100.16  38.33  26.78
```