FLIP ROBO

HOUSE PRICE PREDICTION PROJECT

Submitted By:

Girija Chandra Mohan

ACKNOWLEDGMENT

I would like to express my deepest gratitude to my SME (Subject Matter Expert) Khushboo Garg as well as Flip Robo Technologies who gave me the opportunity to do this project on House Price Prediction, which also helped me in doing lots of research wherein I came to know about so many new things.

Also, I have utilized a few external resources that helped me to complete the project. I ensure that I have learned from the samples and modified things according to my project requirement. All the external resources that were used in creating this project are listed below:

1) https://www.google.com/
2) https://www.youtube.com/
3) https://scikit-learn.org/stable/user_guide.html
4) https://github.com/
5) https://www.kaggle.com/
6) https://medium.com/
7) https://towardsdatascience.com/
8) https://www.analyticsvidhya.com/

INTRODUCTION

Business Problem Framing

Houses are one of the necessary needs 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 modeling, Market mix modeling, 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.

We 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.

Conceptual Background of the Domain Problem

Real estate is the least transparent industry in our ecosystem. Housing prices keep changing day by day and sometimes are hyped rather than being based on valuation. Predicting housing prices with real factors is the main objective of our project. Here we aim to make our evaluations based on every basic parameter that is considered while determining the price

With a large amount of unstructured resources and documents, the Real estate industry has become a highly competitive business. The data science process in such an industry provides an advantage to the developers by processing those data, forecasting future trends and thus assisting them to make favorable knowledge-driven decisions.

House prices increase every year, so there is a need for a system to predict house prices in the future. House price prediction can help the developer determine the selling price of a house and can help the customer to arrange the right time to purchase a house.

Predicting house prices are expected to help people who plan to buy a house so they can know the price range in the future, and then they can plan their finance well. In addition, house price predictions are also beneficial for property investors to know the trend of housing prices in a certain location.

There are three factors that influence the price of a house which include physical conditions, concept and location. There are several approaches that can be used to determine the price of the house, one of them is the prediction analysis. We use various regression techniques in this pathway.

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 a variable?
- ➢ How do these variables describe the price of the house?

Review of Literature

Based on the sample data provided to us from our client database where we have understood that the company is looking at prospective properties to buy houses to enter the market. The data set explains it is a regression problem as we need 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. Also, we have other independent features that would help to decide which all variables are important to predict the price of the variable and how do these variables describe the price of the house.

Motivation for the Problem Undertaken

House Price prediction is important to drive Real Estate efficiency. As earlier, House prices were determined by calculating the acquiring and selling price in a locality. The House Price prediction model is very essential in filling the information gap and improves Real Estate efficiency. The goal of the paper is to predict the efficient house pricing for real estate customers with respect to their budgets and priorities. By analyzing previous market trends and price ranges, and also upcoming developments future prices will be predicted. This model will help customers to invest in an estate without approaching an agent. It also decreases the risk involved in the transaction.

Nowadays, e-education and e-learning is highly influenced. Everything is shifting from manual to automated systems. The objective of this project is to predict the house prices so as to minimize the problems faced by the customer. The present method is that the customer approaches a real estate agent to manage his/her investments and suggest suitable estates for his investments. But this method is risky as the agent might predict wrong estates and thus leading to loss of the customer’s investments.

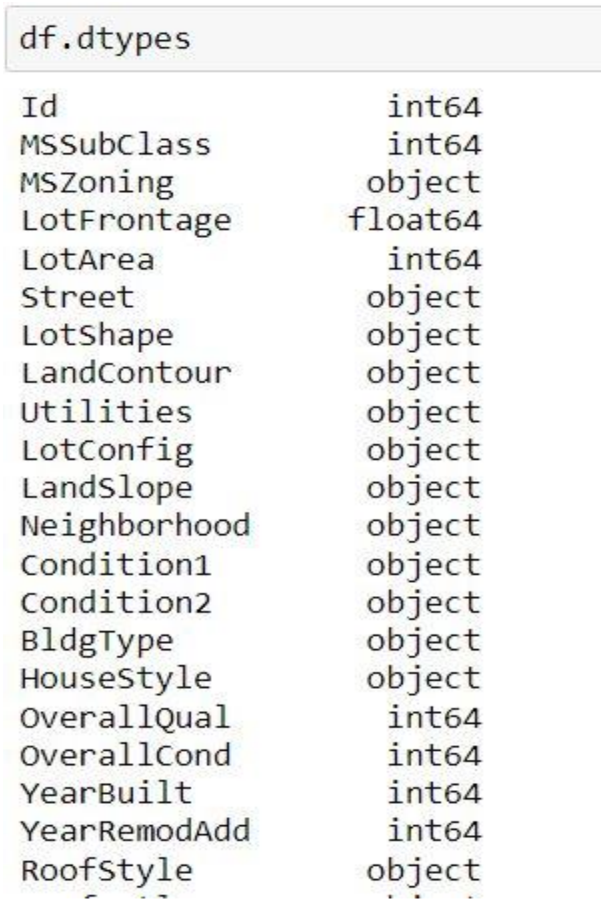
In this project, the main focus is on developing a model which not only predicts the sale price of properties for a customer according to his\her interests, but also recognizes the most preferred location of real estate and other utilities and features in any given area.

Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

We are building a model in Machine Learning to predict the actual value of the prospective properties and decide whether to invest in them or not. So, this model will help us to determine which variables are important to predict the price of variables & also how do these variables describe the price of the house. This will help to determine the price of houses with the available independent variables.

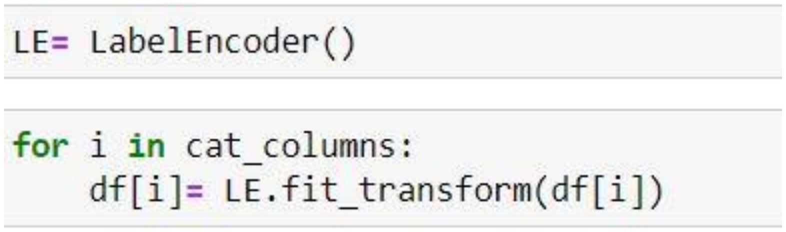
First, the dataset is subjected to different methods of analysis before letting it to a model. The features are first analyzed for its data type to get a clear insight of the features.



```
df.dtypes
```

```
Id                 int64
MSSubClass         int64
MSZoning          object
LotFrontage      float64
LotArea            int64
Street            object
LotShape          object
LandContour       object
Utilities         object
LotConfig         object
LandSlope         object
Neighborhood      object
Condition1        object
Condition2        object
BldgType          object
HouseStyle        object
OverallQual        int64
OverallCond        int64
YearBuilt          int64
YearRemodAdd       int64
RoofStyle         object
```

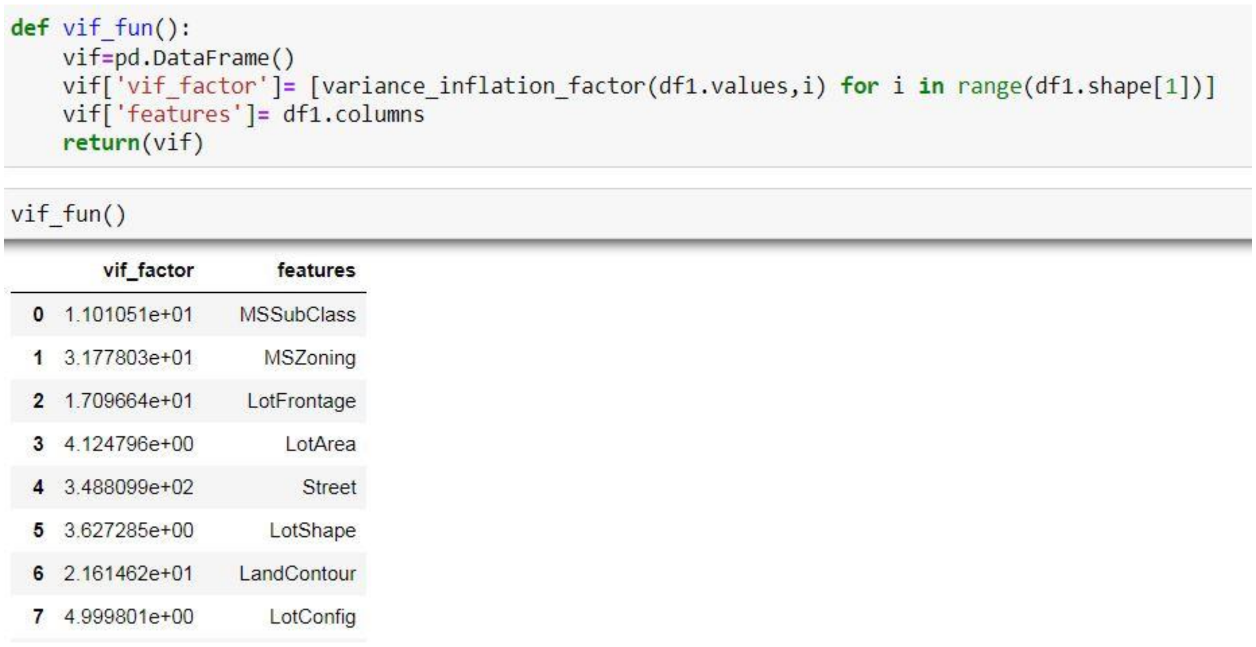
The statistical analysis is carried out on each numeric columns using the dataframe.describe () method which gives all the statistical information such as mean of the feature, median, standard deviation, maximum and minimum count, before which the categorical columns are converted to numerical columns using Label Encoder.



```
LE= LabelEncoder()
```

```
for i in cat_columns:
    df[i]= LE.fit_transform(df[i])
```

Now we shall look for the statistical data and correlation. The data set is then analyzed for multicollinearity with VIF.



```
def vif_fun():
    vif=pd.DataFrame()
    vif['vif_factor']= [variance_inflation_factor(df1.values,i) for i in range(df1.shape[1])]
    vif['features']= df1.columns
    return(vif)
```

```
vif_fun()
```

	vif_factor	features
0	1.101051e+01	MSSubClass
1	3.177803e+01	MSZoning
2	1.709664e+01	LotFrontage
3	4.124796e+00	LotArea
4	3.488099e+02	Street
5	3.627285e+00	LotShape
6	2.161462e+01	LandContour
7	4.999801e+00	LotConfig

Features that has multicollinearity has been removed depending on its contribution towards the target variable.

Z-score is a method used to make analyses of the outliers present.
The skewness and distribution of various data is also studied using visualizations.

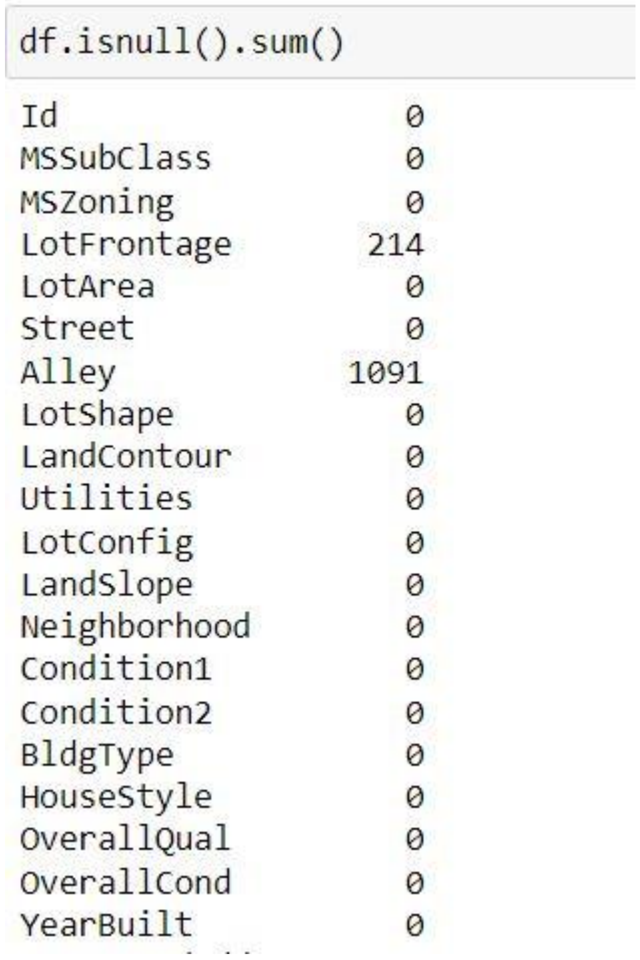
Data Sources and their formats

- A US-based housing company named Surprise Housing has decided to enter the Australian market and has collected a data set from the sale of houses in Australia.
- The data is provided in the CSV file.
- The entire dataset is divided into train and test datasets.
- The train dataset has 81 columns and a total of 1168 records (rows).Where there are 80 inputs/features and 1 target column (label) .Sale price is our target variable.
- The test dataset has 80 columns and a total of 292 records (rows) for which we need to make the predictions.
- Data contains Null values. Which needs to be treated using the domain knowledge and own understanding.
- The train data had 3 float64 features,35 columns with int64 and 43 features with object data type
- Almost 18 features have Nan values
- Data contains numerical as well as categorical variable. We need to handle them accordingly.
- We will have to build Machine Learning models, apply regularization and determine the optimal values of Hyper Parameters, find important features which affect the price positively or negatively
- We will train on train.csv dataset and predict on test.csv file.
- There are few outliers (about 9%) in the data that needs to be handled.

Data Pre-processing Done

Data pre-processing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. Data pre-processing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we pre- process our data before feeding it into our model. Therefore, it is the first and crucial step while creating a machine learning model. I have used some following pre-processing steps:

- Loading the training dataset as a dataframe
- Used pandas to set display to ensure we do not see any truncated information
- Checked the number of rows and columns present in our training dataset
- Checked for missing data and the number of rows with null values

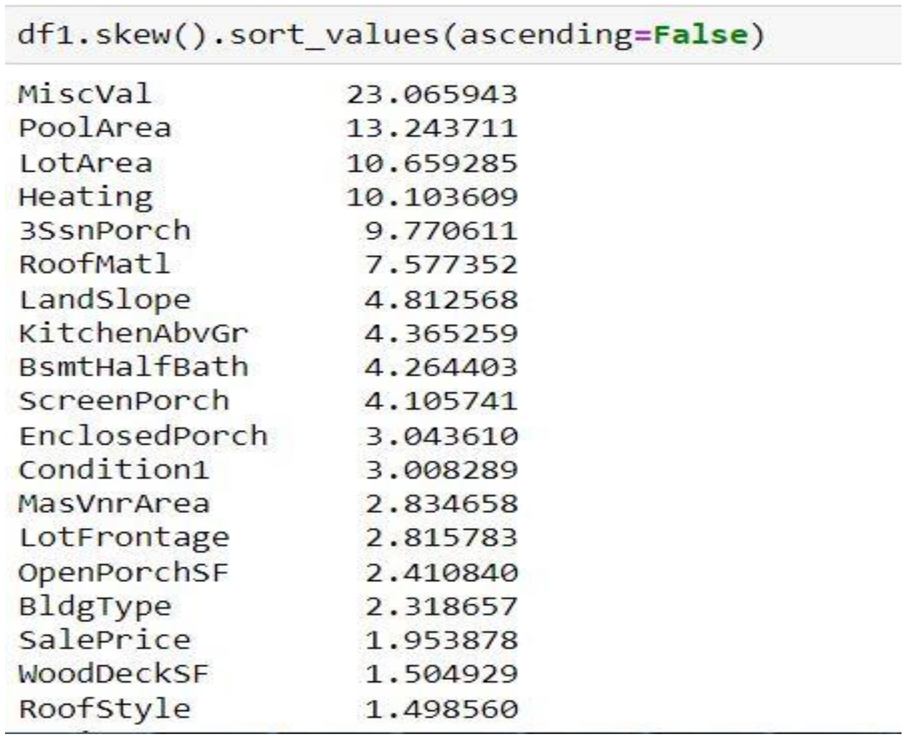


```
df.isnull().sum()
```

```
Id                0
MSSubClass        0
MSZoning          0
LotFrontage     214
LotArea           0
Street            0
Alley          1091
LotShape          0
LandContour       0
Utilities         0
LotConfig         0
LandSlope         0
Neighborhood      0
Condition1        0
Condition2        0
BldgType          0
HouseStyle        0
OverallQual       0
OverallCond       0
YearBuilt         0
```

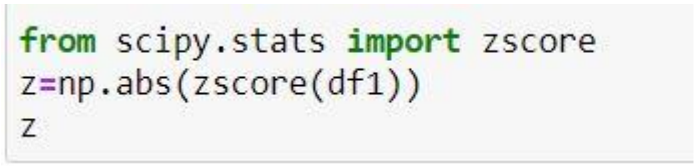
- Verified the percentage of missing data in each column and decided to discard the one’s that have more than 50% of null values
- Dropped all the unwanted columns and duplicate data present in our dataframe

- Separated categorical column names and numeric column names in separate list variables for ease in visualization
- Checked the unique values information in each column to get a gist for categorical data
- Performed imputation to fill missing data using mean on numeric data and mode for categorical data columns
- Used Pandas Profiling during the visualization phase along with pie plot, count plot, scatter plot and the others
- With the help of label encoding technique converted all object datatype columns to numeric datatype
- Using VIF factor the columns with multi collinearity is ruled out.
- Thoroughly checked for outliers and skewness information



```
df1.skew().sort_values(ascending=False)
```

```
MiscVal          23.065943
PoolArea         13.243711
LotArea          10.659285
Heating          10.103609
3SsnPorch         9.770611
RoofMatl          7.577352
LandSlope         4.812568
KitchenAbvGr      4.365259
BsmtHalfBath      4.264403
ScreenPorch       4.105741
EnclosedPorch     3.043610
Condition1        3.008289
MasVnrArea        2.834658
LotFrontage       2.815783
OpenPorchSF       2.410840
BldgType          2.318657
SalePrice         1.953878
WoodDeckSF        1.504929
RoofStyle         1.498560
```



```
from scipy.stats import zscore
z=np.abs(zscore(df1))
z
```

- With the help of heatmap, correlation bar graph was able to understand the Feature vs Label relativity and insights on multicollinearity amongst the feature columns

- Separated feature and label data to ensure feature scaling is performed avoiding any kind of biasness
- Checked for the best random state to be used on our Regression Machine Learning model pertaining to the feature importance details
- Finally created a regression model function along with evaluation metrics to pass through various model formats.

Data Inputs- Logic- Output Relationships

When we loaded the training dataset, we had to go through various data pre processing steps to understand what was given to us and what we were expected to predict for the project. When it comes to logical part the domain expertise of understanding how real estate works and how we are supposed to cater to the customers came in handy to train the model with the modified input data

With the objective of predicting sale prices accurately we had to make sure that a model was built that understood the customer priorities trending in the market imposing those norms when a relevant price tag was generated. We tried the best to retain as much data possible that was collected but discarding columns that had lots of missing data was good for results.

Hardware and Software Requirements and Tools Used Open source web-application used for programming:

1. Jupyter Notebook

Python Libraries / Packages used were:

1. **Pandas**: pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

We have used pandas to import the csv file using pd.read_csv all data analysis have been done using the pandas and numpy libraries. The data characteristics have been studied using pandas functions like df.shape(), df .dtypes, df.columns etc.

2. **NumPy:** NumPy is an open-source numerical Python library. NumPy contains a multi-dimensional array and matrix data structures. It can be utilised to perform a number of mathematical operations on arrays such as trigonometric, statistical, and algebraic.

We have used the np.where function many times while dealing with the z-scores.np.abs () function has also been used to find the zscore and some mathematical operations like square root.

3. **Matplotlib**: library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

The Matplotlib libraries pyplot function is used for making plots ,plt.show() ,plt.figure(figsize) that has been used is a part of matplotlib library.

4. **Seaborn**: Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

All the visualizations made are built using the seaborn library. Alias used for seaborn is sns.
Sns.boxplot(), sns.heatmap(), sns.distplot(), sns.scatterplot() ,sns.stripplot,sns.swarmplot ,heatmap are few of the libraries used

5. **SciPy**: SciPy, a scientific library for Python is an open source library for mathematics, science and engineering. The SciPy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation. The main reason for building the SciPy library is that, it should work with NumPy arrays.
In this particular project scipy functions such as scipy.stats is used .Zscores are also obtained via scipy.stats library

6. **Sklearn:** Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for

machine learning and statistical modelling including classification, regression, and clustering and dimensionality reduction via a consistence interface in Python.

All the Machine Learning regression algorithms have been imported from the sklearn package. Simple imputer used is also a part of sklearn
The evaluation metrics, RMSE,MSE,MAE functions are also imported from same.

7. **Python:** is a set of Data analysis tools in python which gives us measures of association for categorical features, Plots features correlation and association for mixed data-sets (categorical and continuous features) in an easy and simple way.

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

We have used both statistical and analytical approaches to solve the problem which mainly includes the pre-processing of the data and EDA to check the correlation of independent and dependent features. Also, before building the model, We made sure that the input data is cleaned and scaled before it was fed into the machine learning models.

For this project we need to predict the sale price of houses, means our target column is continuous so this is a regression problem. We have used various regression algorithms and tested for the prediction. By doing various evaluations We have selected RandomForest Regressor as best suitable algorithm for our final model as it is giving good r2-score and least difference in r2-score and CV-score among all the algorithms used. Other regression algorithms are also giving good accuracy but some are over-fitting and some are with under-fitting.

In order to get good performance as well as accuracy and to check my model from over-fitting and under-fitting we made the hyper parameter tuning for the final model.

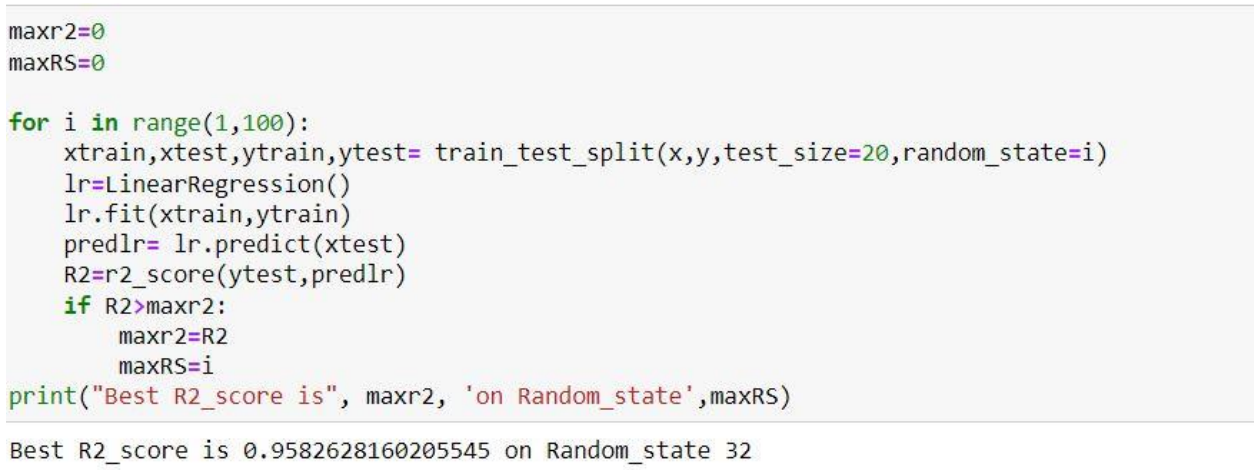
Once we are able to get the desired final model we ensured to save that model before loading the testing data and started performing the data pre-processing as the training dataset and obtaining the predicted sale price values out of the Regression Machine Learning Model.

The algorithms used on training and test data are as follows:

- Linear Regression Model
- Ridge Regularization Regression Model
- Lasso Regularization Regression Model
- Support Vector Regression Model
- Decision Tree Regression Model
- Random Forest Regression Model
- K Nearest Neighbours Regression Model

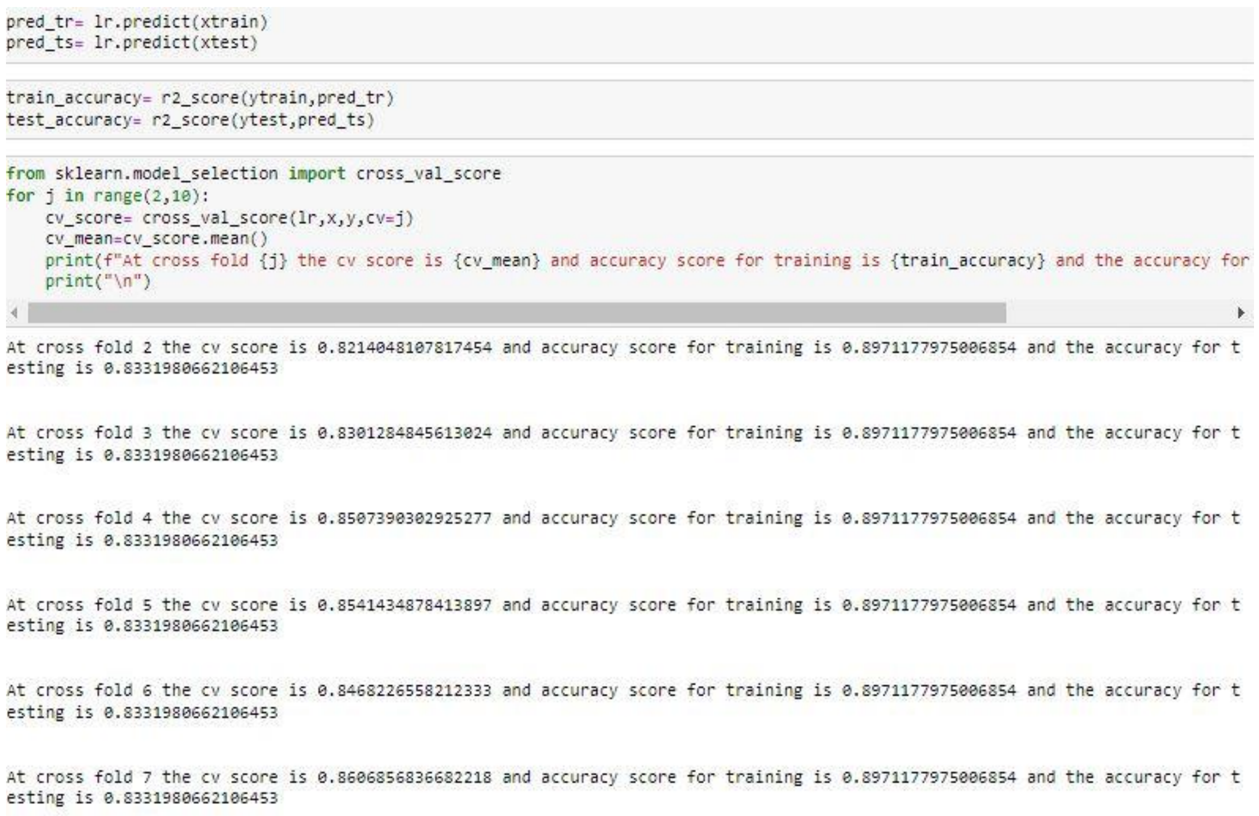
Run and Evaluate selected models

We have used a total of 7 Regression Models after choosing the random state amongst 1-100 number. Then it is checked for best fold for CV using the following codes.



```
maxr2=0
maxRS=0

for i in range(1,100):
    xtrain,xtest,ytrain,ytest= train_test_split(x,y,test_size=20,random_state=i)
    lr=LinearRegression()
    lr.fit(xtrain,ytrain)
    predlr= lr.predict(xtest)
    R2=r2_score(ytest,predlr)
    if R2>maxr2:
        maxr2=R2
        maxRS=i
print("Best R2_score is", maxr2, 'on Random_state',maxRS)
```

```
Best R2_score is 0.9582628160205545 on Random_state 32
```



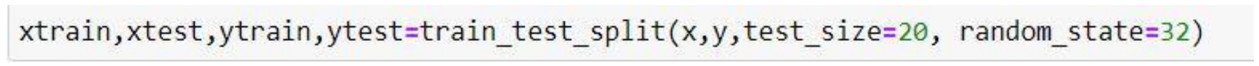
```
pred_tr= lr.predict(xtrain)
pred_ts= lr.predict(xtest)
```

```
train_accuracy= r2_score(ytrain,pred_tr)
test_accuracy= r2_score(ytest,pred_ts)
```

```
from sklearn.model_selection import cross_val_score
for j in range(2,10):
    cv_score= cross_val_score(lr,x,y,cv=j)
    cv_mean=cv_score.mean()
    print(f"At cross fold {j} the cv score is {cv_mean} and accuracy score for training is {train_accuracy} and the accuracy for
    print("\n")
```

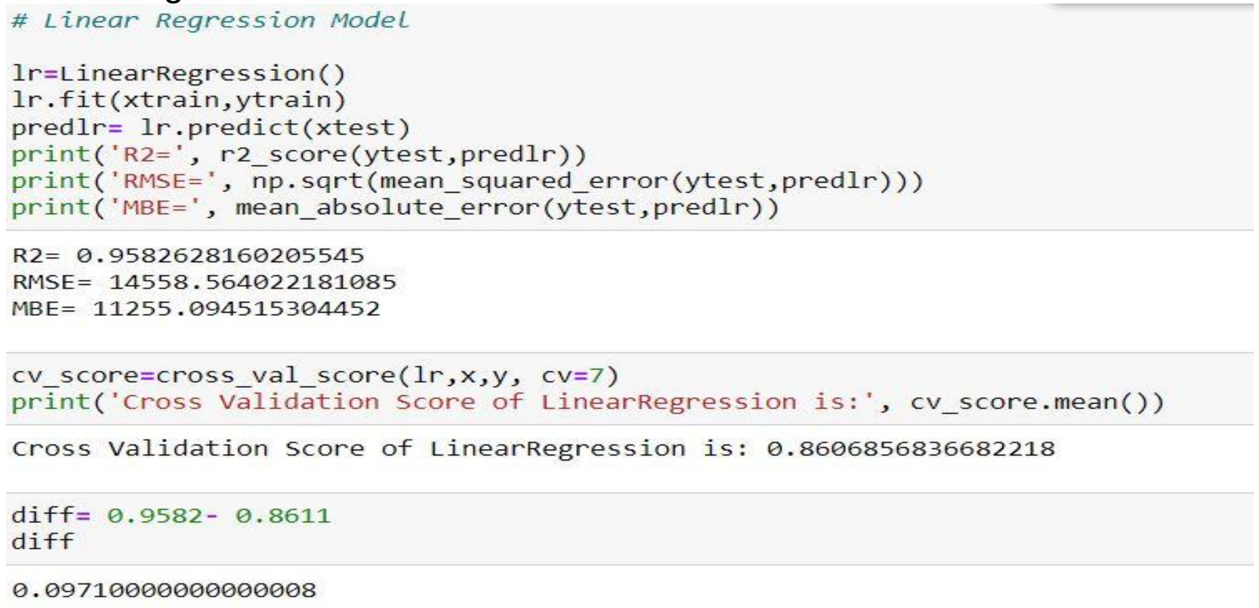
```
At cross fold 2 the cv score is 0.8214048107817454 and accuracy score for training is 0.8971177975006854 and the accuracy for t
esting is 0.8331980662106453


At cross fold 3 the cv score is 0.8301284845613024 and accuracy score for training is 0.8971177975006854 and the accuracy for t
esting is 0.8331980662106453


At cross fold 4 the cv score is 0.8507390302925277 and accuracy score for training is 0.8971177975006854 and the accuracy for t
esting is 0.8331980662106453


At cross fold 5 the cv score is 0.8541434878413897 and accuracy score for training is 0.8971177975006854 and the accuracy for t
esting is 0.8331980662106453


At cross fold 6 the cv score is 0.8468226558212333 and accuracy score for training is 0.8971177975006854 and the accuracy for t
esting is 0.8331980662106453


At cross fold 7 the cv score is 0.8606856836682218 and accuracy score for training is 0.8971177975006854 and the accuracy for t
esting is 0.8331980662106453
```

Regression Model Function



```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=20, random_state=32)
```

Linear Regression



```
# Linear Regression Model

lr=LinearRegression()
lr.fit(xtrain,ytrain)
predlr= lr.predict(xtest)
print('R2=', r2_score(ytest,predlr))
print('RMSE=', np.sqrt(mean_squared_error(ytest,predlr)))
print('MBE=', mean_absolute_error(ytest,predlr))
```

```
R2= 0.9582628160205545
RMSE= 14558.564022181085
MBE= 11255.094515304452
```

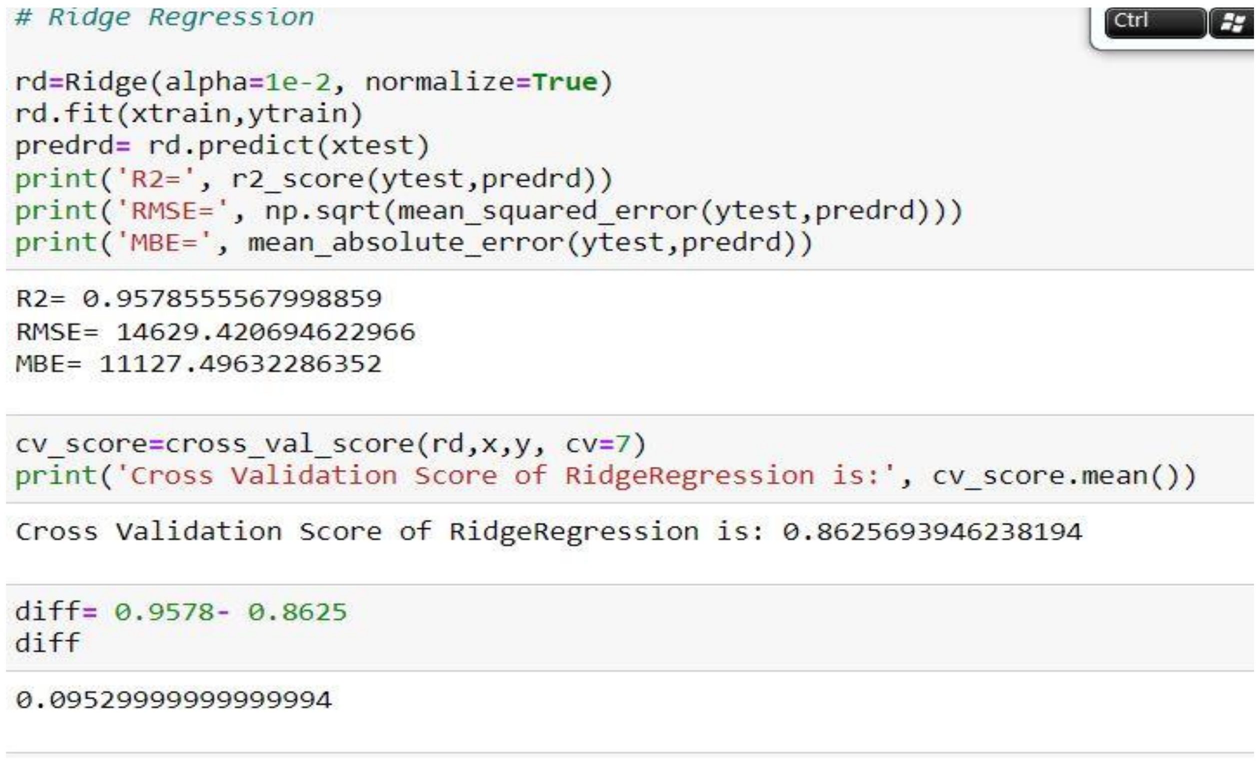
```
cv_score=cross_val_score(lr,x,y, cv=7)
print('Cross Validation Score of LinearRegression is:', cv_score.mean())
```

```
Cross Validation Score of LinearRegression is: 0.8606856836682218
```

```
diff= 0.9582- 0.8611
diff
```

```
0.09710000000000008
```

Ridge Regression



```
# Ridge Regression

rd=Ridge(alpha=1e-2, normalize=True)
rd.fit(xtrain,ytrain)
predrd= rd.predict(xtest)
print('R2=', r2_score(ytest,predrd))
print('RMSE=', np.sqrt(mean_squared_error(ytest,predrd)))
print('MBE=', mean_absolute_error(ytest,predrd))
```

```
R2= 0.9578555567998859
RMSE= 14629.420694622966
MBE= 11127.49632286352
```

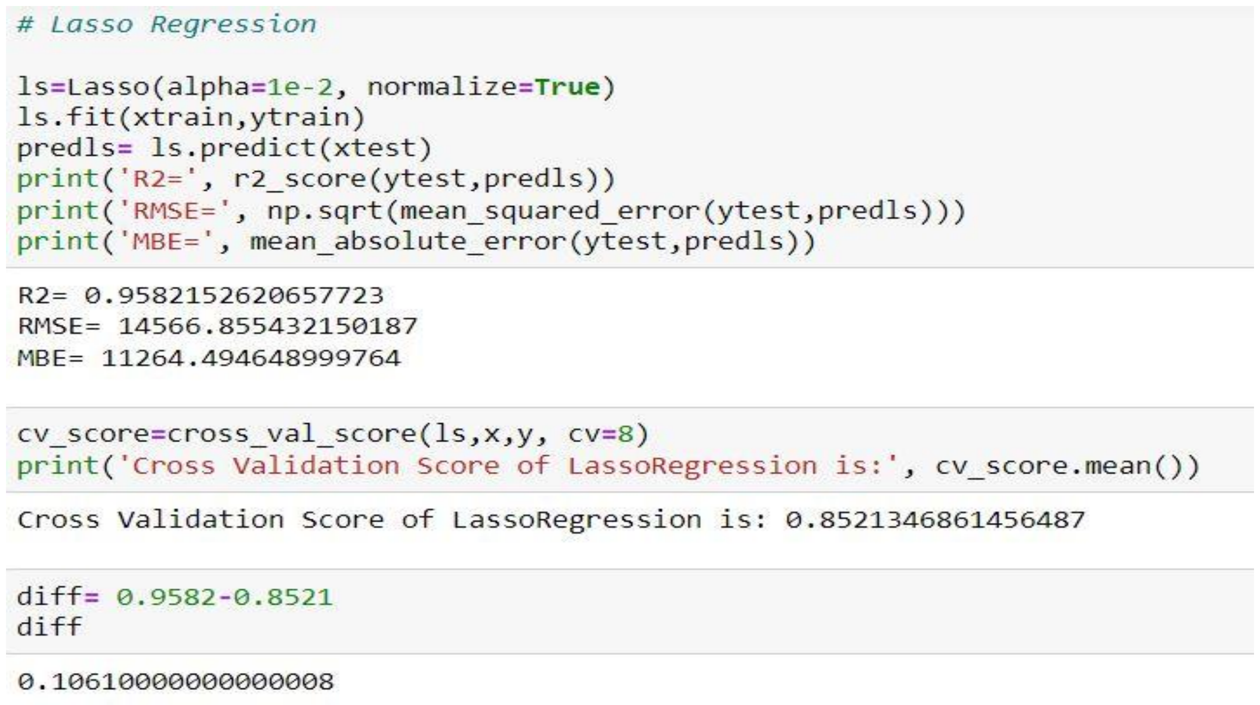
```
cv_score=cross_val_score(rd,x,y, cv=7)
print('Cross Validation Score of RidgeRegression is:', cv_score.mean())
```

```
Cross Validation Score of RidgeRegression is: 0.8625693946238194
```

```
diff= 0.9578- 0.8625
diff
```

```
0.09529999999999994
```

Lasso Regression



```
# Lasso Regression

ls=Lasso(alpha=1e-2, normalize=True)
ls.fit(xtrain,ytrain)
predls= ls.predict(xtest)
print('R2=', r2_score(ytest,predls))
print('RMSE=', np.sqrt(mean_squared_error(ytest,predls)))
print('MBE=', mean_absolute_error(ytest,predls))
```

```
R2= 0.9582152620657723
RMSE= 14566.855432150187
MBE= 11264.494648999764
```

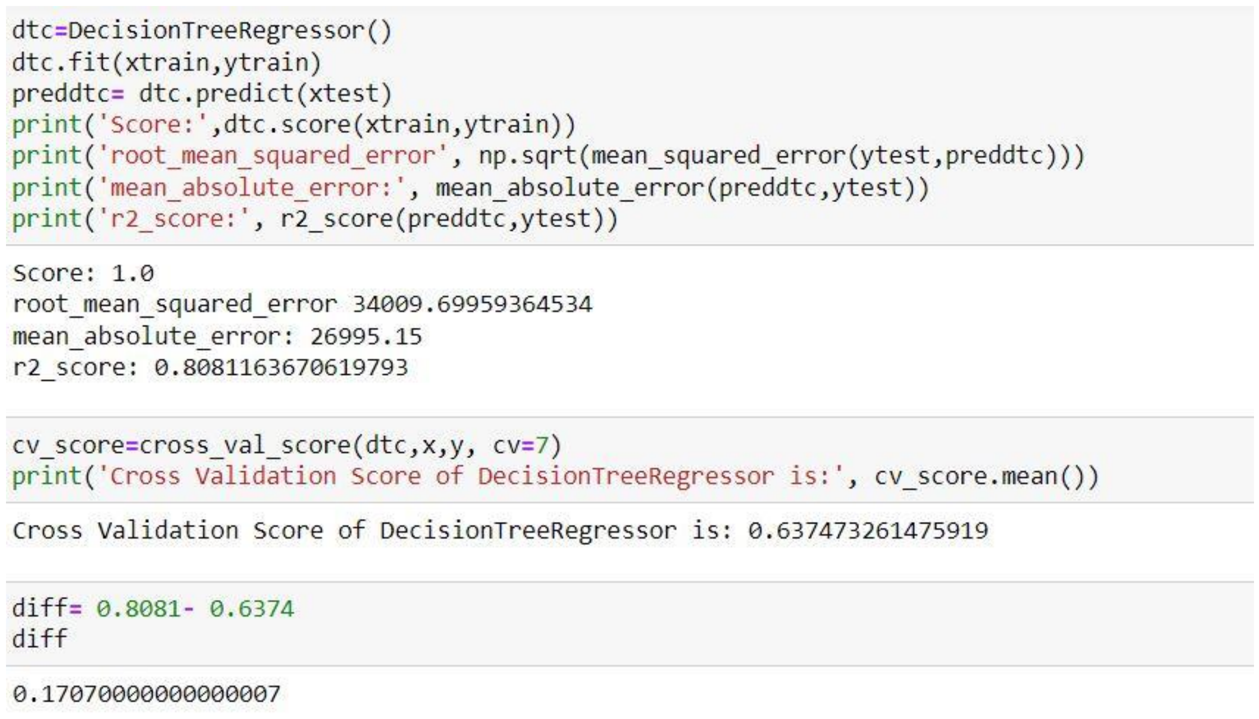
```
cv_score=cross_val_score(ls,x,y, cv=8)
print('Cross Validation Score of LassoRegression is:', cv_score.mean())
```

```
Cross Validation Score of LassoRegression is: 0.8521346861456487
```

```
diff= 0.9582-0.8521
diff
```

```
0.10610000000000008
```

DecisionTree Regression



```
dtc=DecisionTreeRegressor()
dtc.fit(xtrain,ytrain)
preddtc= dtc.predict(xtest)
print('Score:',dtc.score(xtrain,ytrain))
print('root_mean_squared_error', np.sqrt(mean_squared_error(ytest,preddtc)))
print('mean_absolute_error:', mean_absolute_error(preddtc,ytest))
print('r2_score:', r2_score(preddtc,ytest))
```

```
Score: 1.0
root_mean_squared_error 34009.69959364534
mean_absolute_error: 26995.15
r2_score: 0.8081163670619793
```

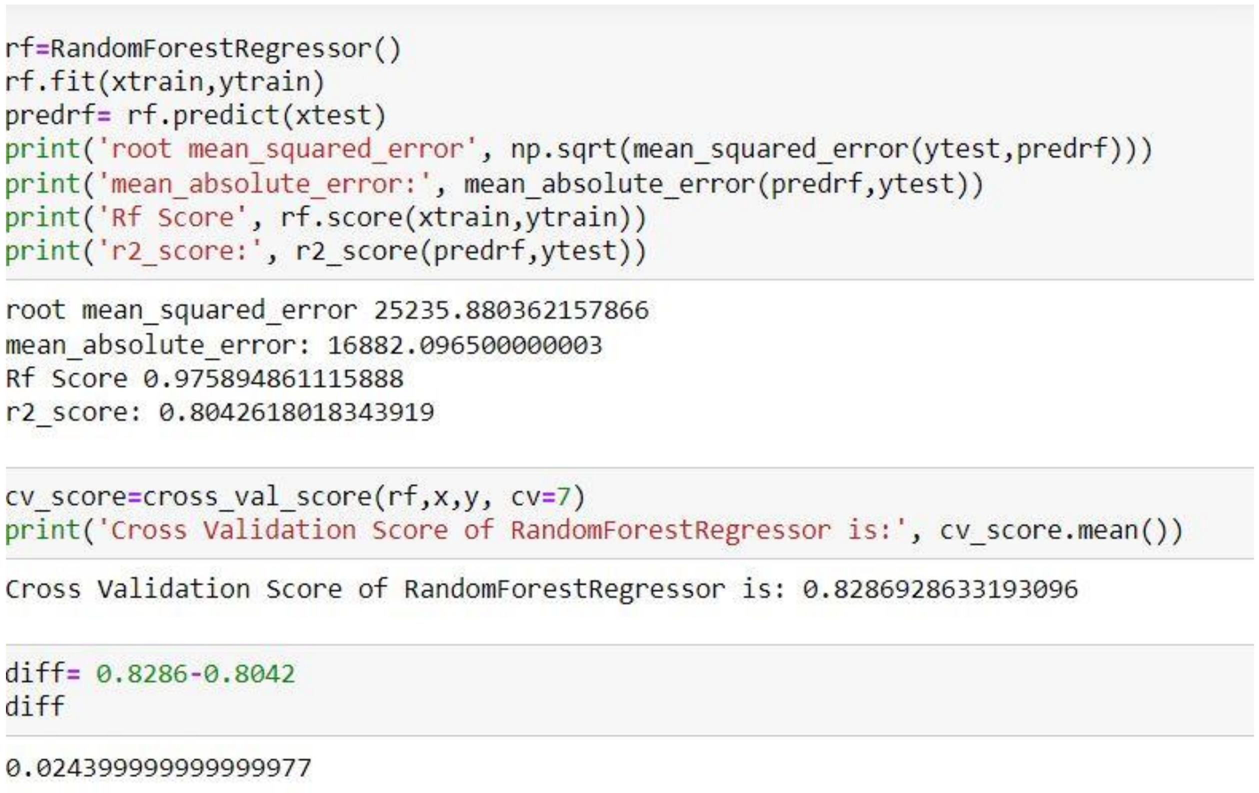
```
cv_score=cross_val_score(dtc,x,y, cv=7)
print('Cross Validation Score of DecisionTreeRegressor is:', cv_score.mean())
```

```
Cross Validation Score of DecisionTreeRegressor is: 0.637473261475919
```

```
diff= 0.8081- 0.6374
diff
```

```
0.17070000000000007
```

Random Forest Regression



```
rf=RandomForestRegressor()
rf.fit(xtrain,ytrain)
predrf= rf.predict(xtest)
print('root mean_squared_error', np.sqrt(mean_squared_error(ytest,predrf)))
print('mean_absolute_error:', mean_absolute_error(predrf,ytest))
print('Rf Score', rf.score(xtrain,ytrain))
print('r2_score:', r2_score(predrf,ytest))
```

```
root mean_squared_error 25235.880362157866
mean_absolute_error: 16882.096500000003
Rf Score 0.975894861115888
r2_score: 0.8042618018343919
```

```
cv_score=cross_val_score(rf,x,y, cv=7)
print('Cross Validation Score of RandomForestRegressor is:', cv_score.mean())
```

```
Cross Validation Score of RandomForestRegressor is: 0.8286928633193096
```

```
diff= 0.8286-0.8042
diff
```

```
0.024399999999999977
```

Support Vector Regression

```python
#SVR

svc=SVR()
svc.fit(xtrain,ytrain)
predsvc= svc.predict(xtest)
print('root mean_squared_error', np.sqrt(mean_squared_error(ytest,predsvc)))
print('mean_absolute_error:', mean_absolute_error(predsvc,ytest))
print('Score:', svc.score(xtrain,ytrain))
print('r2_score:', r2_score(predsvc,ytest))
```

```
root mean_squared_error 75973.72572615088
mean_absolute_error: 64666.49612864862
Score: -0.021439330154186864
r2_score: -11173441.720453784
```

KNN

```python
knn=KNeighborsRegressor()
knn.fit(xtrain,ytrain)
predknn= knn.predict(xtest)
print('root mean_squared_error', np.sqrt(mean_squared_error(ytest,predknn)))
print('mean_absolute_error:', mean_absolute_error(predknn,ytest))
print('Rf Score', knn.score(xtrain,ytrain))
print('r2_score:', r2_score(predknn,ytest))
```

```
root mean_squared_error 28601.791439628396
mean_absolute_error: 21566.880000000005
Rf Score 0.8505748709761425
r2_score: 0.7290407741750269
```

```python
cv_score=cross_val_score(knn,x,y, cv=7)
print('Cross Validation Score of KNN is:', cv_score.mean())
```

```
Cross Validation Score of KNN is: 0.7750910569219766
```

```python
diff= 0.7750-0.7290
diff
```

```
0.04600000000000004
```

Key Metrics for success in solving problem under consideration

The key metrics used here were r2_score, cross_val_score, MAE, MSE and RMSE. We tried to find out the best parameters and also to increase our scores by using Hyperparameter Tuning and we will be using GridSearchCV method.

1. Cross Validation:

Cross-validation helps to find out the over fitting and under fitting of the model. In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If we take 7 folds, the data will be divided into 7 pieces, where each part being 20% of full dataset. While running the Cross-validation the 1st part (17%) of the 7parts will be kept out as a holdout set for validation and everything else is used for training data. This way we will get the first estimate of the model quality of the dataset.

In the similar way further iterations are made for the second 17% of the dataset is held as a holdout set and remaining 4 parts are used for training data during process. This way we will get the second estimate of the model quality of the dataset. These steps are repeated during the cross-validation process to get the remaining estimate of the model quality.

2. R2 Score:

It is a statistical measure that represents the goodness of fit of a regression model. The ideal value for r-square is 1. The closer the value of r-square to 1, the better is the model fitted.

3. Mean Squared Error (MSE):

MSE of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors — that is, the average squared difference between the estimated values and what is estimated. MSE is a risk function, corresponding to the expected value of the squared error loss. RMSE is the Root Mean Squared Error.

4. Mean Absolute Error (MAE):

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.

5. Hyperparameter Tuning:

There is a list of different machine learning models. They all are different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named as Hyperparameters. These hyperparameters will define the architecture of the model, and the best part about these is that you get a choice to select these for your model. You must select from a specific list of hyperparameters for a given model as it varies from model to model.

We are not aware of optimal values for hyperparameters which would generate the best model output. So, what we tell the model is to explore and select the optimal model architecture automatically. This selection procedure for hyperparameter is known as Hyperparameter Tuning. We can do tuning by using GridSearchCV.

GridSearchCV is a function that comes in SK-learn model selection package. This function helps to loop through predefined hyperparameters and fit our estimator (model) on our training set. So, we can select the best parameters from the listed hyperparameters.

HyperParameter Tuning With GridSearchCV

```
from sklearn.model_selection import GridSearchCV
```

```
RandomForestRegressor()
```

```
RandomForestRegressor()
```

```
parameters={'max_features': ["auto", "sqrt", "log2"],
            'min_samples_leaf': [1,2,3],
            'criterion':["squared_error", "absolute_error", "poisson"],
            'max_depth':[3,4,5,6],
            'min_samples_split': [2,3,4,5]}
```

```
GCV=GridSearchCV(RandomForestRegressor(), parameters, cv=7, scoring='r2')
GCV.fit(xtrain,ytrain)
GCV.best_params_
```

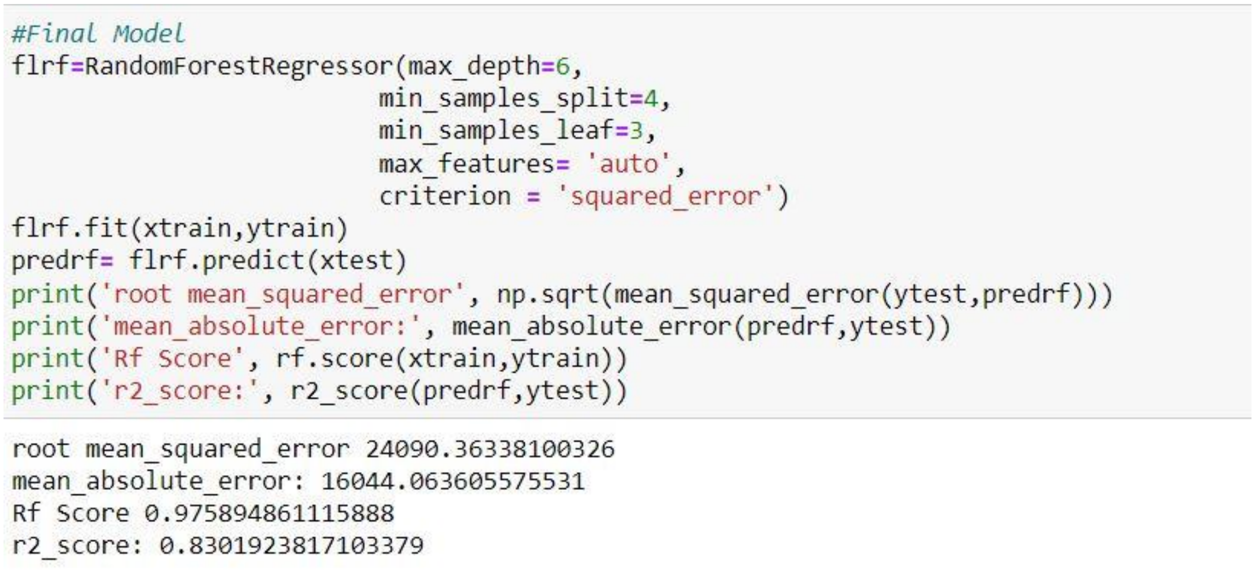
```
{'criterion': 'squared_error',
 'max_depth': 6,
 'max_features': 'auto',
 'min_samples_leaf': 3,
 'min_samples_split': 4}
```

```
GCV.best_estimator_
```

```
RandomForestRegressor(max_depth=6, min_samples_leaf=3, min_samples_split=4)
```

```
GCV_pred=GCV.best_estimator_.predict(xtest)
r2_score(ytest,GCV_pred)
```

```
0.8684537683571734
```



```
#Final Model
flrf=RandomForestRegressor(max_depth=6,
                           min_samples_split=4,
                           min_samples_leaf=3,
                           max_features= 'auto',
                           criterion = 'squared_error')
flrf.fit(xtrain,ytrain)
predrf= flrf.predict(xtest)
print('root mean_squared_error', np.sqrt(mean_squared_error(ytest,predrf)))
print('mean_absolute_error:', mean_absolute_error(predrf,ytest))
print('Rf Score', rf.score(xtrain,ytrain))
print('r2_score:', r2_score(predrf,ytest))
```

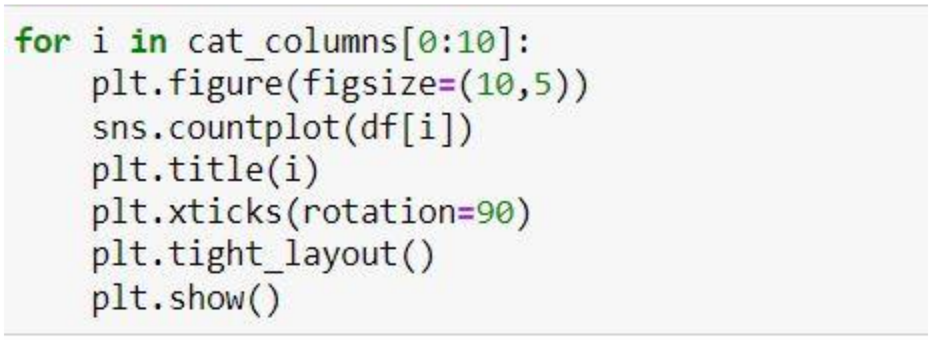
```
root mean_squared_error 24090.36338100326
mean_absolute_error: 16044.063605575531
Rf Score 0.975894861115888
r2_score: 0.8301923817103379
```

It is possible that there are times when the default parameters perform better than the parameters list obtained from the tuning and it only indicates that there are more permutations and combinations that one needs to go through for obtaining better results.

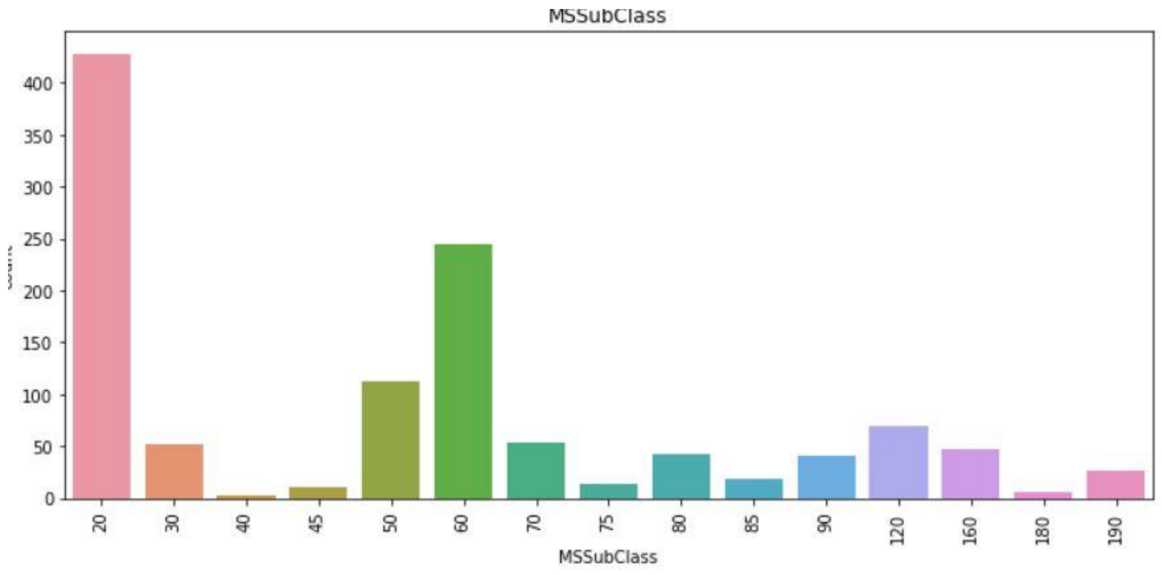
Visualizations

We used pandas profiling to get the over viewed visualization on the pre-processed data. It generates interactive reports in web format that can be presented to any person, even if they don't know programming. It also offers report generation for the dataset with lots of features and customizations for the report generated. It generates a report with all the information easily available.

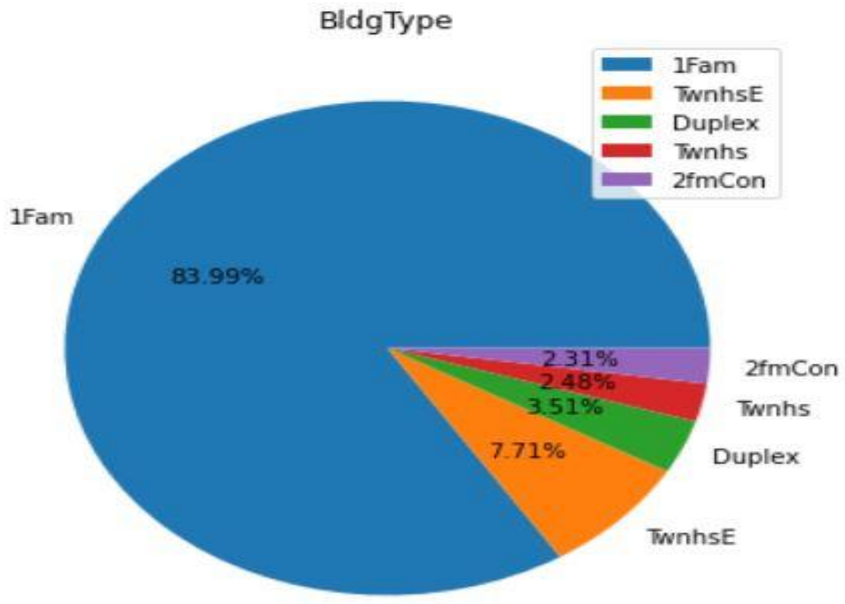
UNIVARIATE ANALYSIS



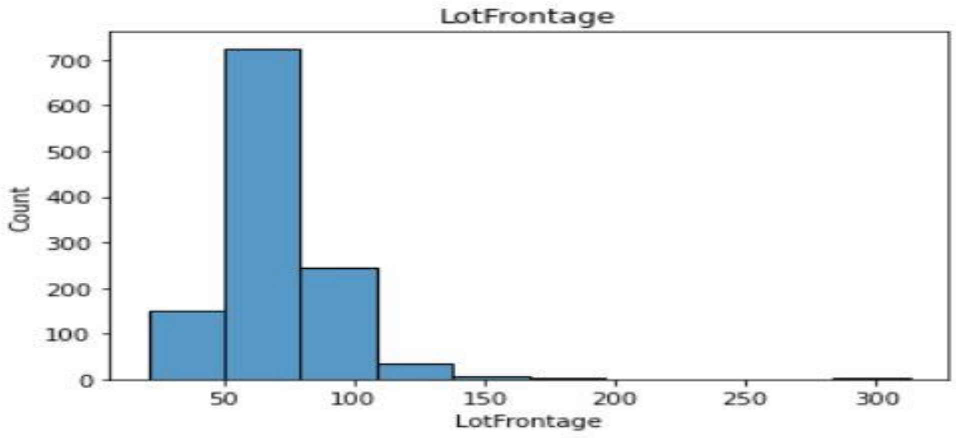
```
for i in cat_columns[0:10]:
    plt.figure(figsize=(10,5))
    sns.countplot(df[i])
    plt.title(i)
    plt.xticks(rotation=90)
    plt.tight_layout()
    plt.show()
```



```
for i in cat_columns[10:20]:
    plt.figure(figsize=(10,5))
    plt.pie(df[i].value_counts(), labels=df[i].value_counts().index, autopct='%1.2f%%')
    plt.legend(prop={'size':10})
    plt.title(i)
    plt.tight_layout()
    plt.show()
```

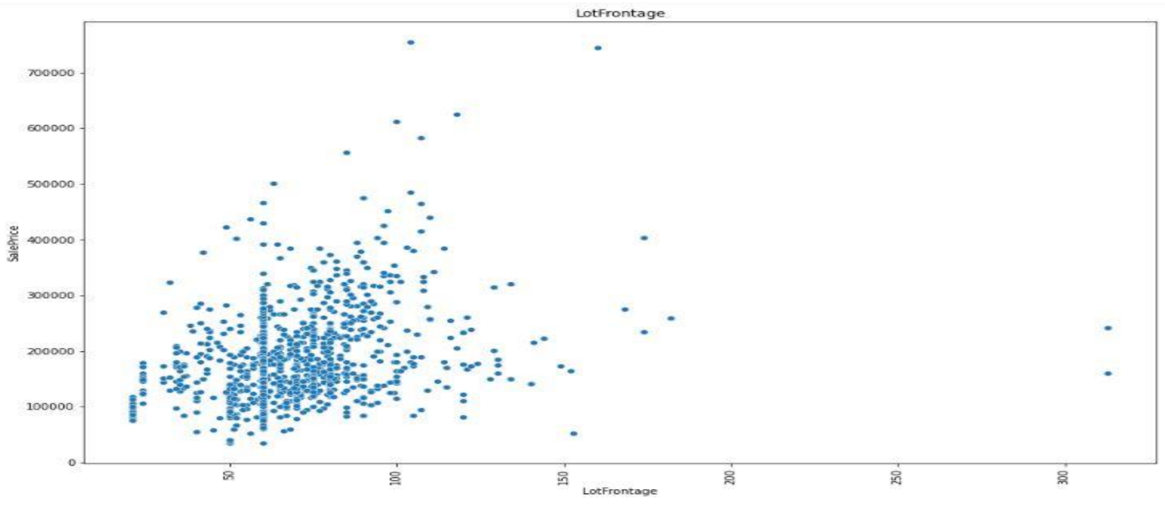


```
for i in num_columns[0:10]:
    sns.histplot(df[i], bins=10)
    plt.title(i)
    plt.show()
```



BIVARIATE ANALYSIS

```
for i in col1:
    plt.figure(figsize=(15,10))
    sns.scatterplot(x=df[i], y= df['SalePrice'])
    plt.xticks(rotation=90)
    plt.title(i)
    plt.show()
```



Interpretation of the Results

With Visualization, It is easy to understand the correlation between independent and dependent features. Detected outliers with the help of boxplot. We got to know the count of a particular category for each feature by using count plot. Before building the model the dataset should be cleaned and scaled by performing scaling.

After performing the train test split, we have xtrain, xtest, ytrain and ytest, which are required to build Machine learning models. Then we built multiple regression models to get the best R2 score, MSE, RMSE & MAE out of all the models.

Learning Outcomes of the Study in respect of Data Science

There is a huge pressure on real estate industry to unlock the potential of data science. Individual appraisers and values bring their own experience, metrics and skills to a job. Consistency is difficult, with UK and Australian-based studies suggesting valuations between two professionals can differ by up to 40%.

Perhaps a well-trained machine could perform this task in place of a human, with greater consistency and accuracy. Thanks to machine learning and data analytics, real estate professionals and investors are now able to make more accurate property assessments than ever before.

The decision to purchase real estate is undeniably very essential in the life of most adults. Thus, the appraisal and prediction of real estate can provide useful information to help facilitate real estate transactions. Real estate prices vary due to a wide variety of attributes. The machine learning-based model is a substantial and feasible way to forecast real estate prices, and can provide relatively competitive and satisfactory results.

Limitations of this work and Scope for Future Work

- Many features have NaN values more than 50%, and imputation of them can decrease the effectiveness. And dropping them had the loss of data.

- The biggest limitation we observed was that not all categories of a particular feature were available in the training data. So, if there were new category in the test data the model would not be able to identify that.

- The high skewness of data reduces the effectivity.

- we can increase the efficiency of a model by selecting a better method to remove outliers and skewness also how to make the search of perfect model in a way that if we want to change some parameters in model then we don't have to run all the model again.