HOUSING: PRICE PREDICTION PROJECT

Submitted By:

Ashish Ashokrao Bhakte

ACKNOWLEDGEMENT

I am very much Thankful to FlipRobo Technologies for giving me the opportunity to work with them and to work on this project and also, I am very grateful to Data Trained Education Team for their support and help to understand each and every concept of machine learning which helped me a lot while working on this project. I thought, I am fortunate to become a part of FlipRobo Technology.

References:

Google website

Stack overflow

Analytics Vidya

Medium

Data trained notes

INTRODUCTION

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 on at a higher price. For the same purpose, the company has collected a data set from the sale of houses in Australia.

The company is looking at prospective properties to buy to enter the market. You are required to build a regression model using regularisation in order to predict the actual value of the prospective properties and decide whether to invest in them or not.

The company wants to know:

- Which variables are significant in predicting the price of a house, and
- How well those variables describe the price of a house.

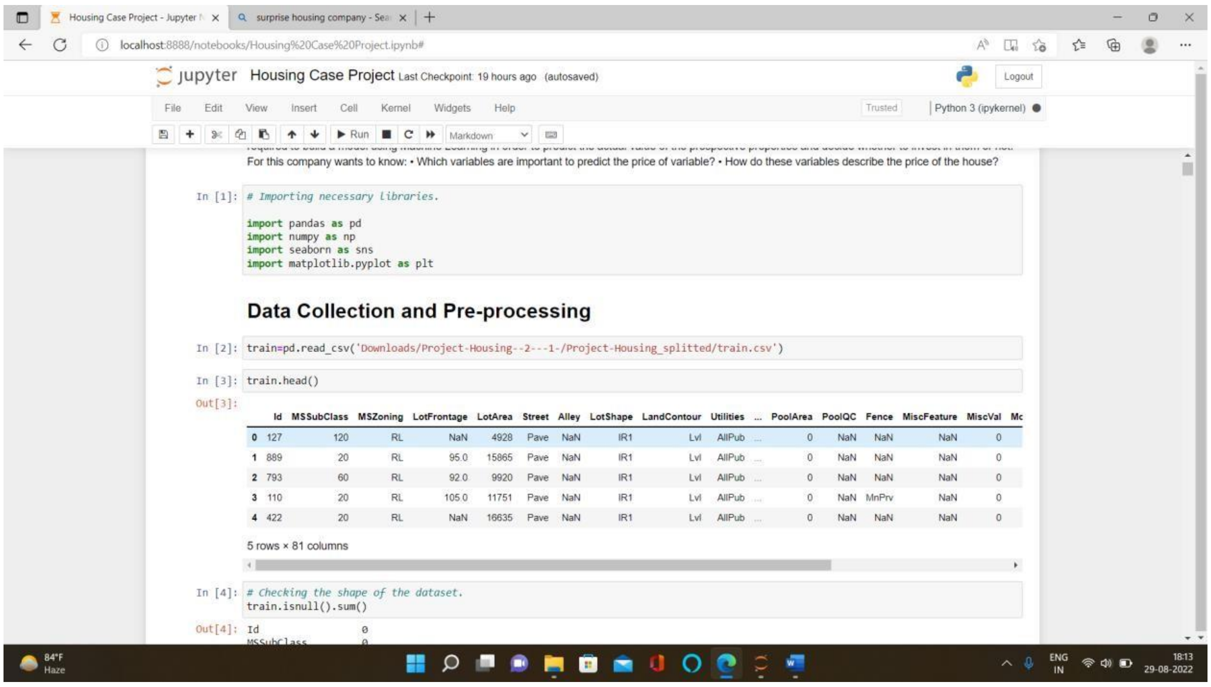
ANALYTICAL PROBLEM FRAMING

- ### Mathematical/ Analytical Modelling of the Problem

  Here I have done Data Pre-processing, Exploratory Data Analysis, then Encoding and lastly model Building and Evaluation.

- ### Data Sources and their formats

  I got the dataset in CSV format and I read the data in Jupyter Notebook using pandas data frame.



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?

```
In [1]: # Importing necessary libraries.

        import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
```

Data Collection and Pre-processing

```
In [2]: train=pd.read_csv('Downloads/Project-Housing--2---1-/Project-Housing_splitted/train.csv')

In [3]: train.head()

Out[3]:
```

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	MiscVal	Mc
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	0	
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	0	

5 rows × 81 columns

```
In [4]: # Checking the shape of the dataset.
        train.isnull().sum()

Out[4]: Id              0
        MSSubClass      0
```

- ### Data Pre-processing Done

  The dataset contains object data type columns, missing values, I have treated them with ordinal encoding, and with mean and mode imputation.

- ### Hardware and Software Requirements and Tools Used

  Here for this project, I used Jupyter notebook and tools used pandas and NumPy for mathematical operations, matplotlib and seaborn for various type of data visualizations.

- ## Identification of possible problem-solving approaches (methods)

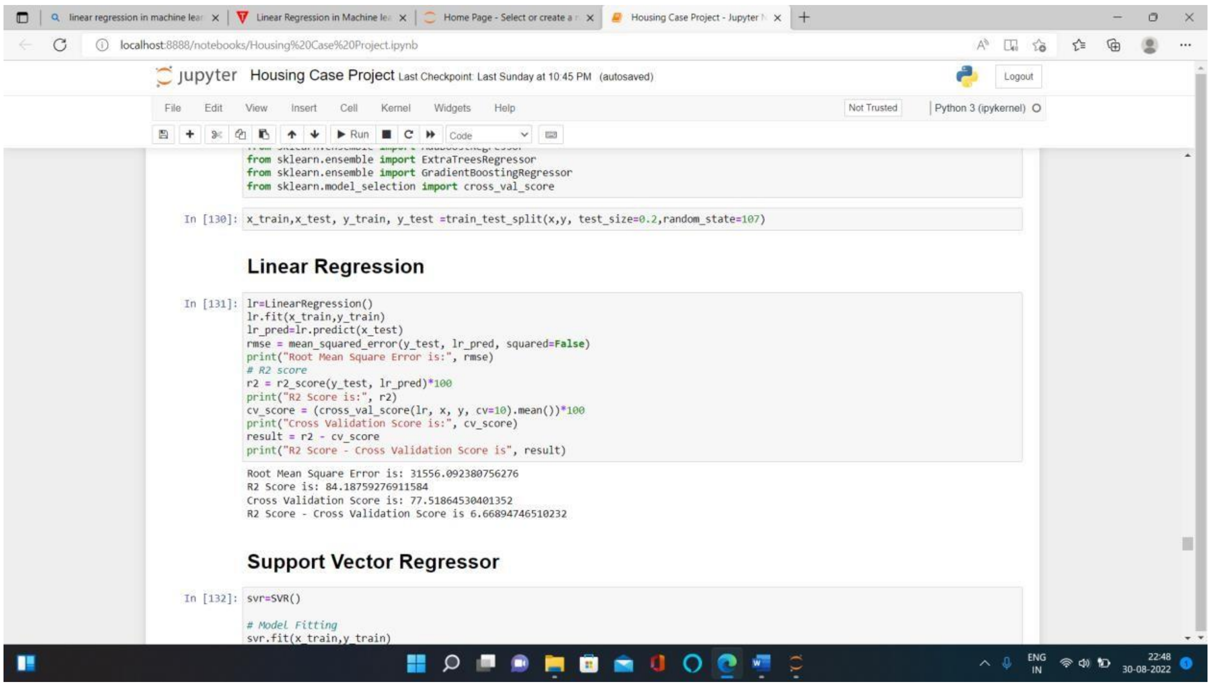
  The statistical summary shows the total count of 1168 rows then mean, min value, max value, standard deviation and quartiles shows up and down values that means the data contains outliers.

- ## Testing of Identified Approaches (Algorithms)

  1. **Linear Regression**
  2. **Random Forest Regressor**
  3. **Decision Tree Regressor**
  4. **Support Vector Regressor**
  5. **Gradient Boosting Regressor**
  6. **AdaBoost Regressor**
  7. **Extra Trees Regressor**
  8. **SGD Regressor**

- ## Run and evaluate selected models

  Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.



```
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import cross_val_score
```

```
In [130]: x_train,x_test, y_train, y_test =train_test_split(x,y, test_size=0.2,random_state=107)
```

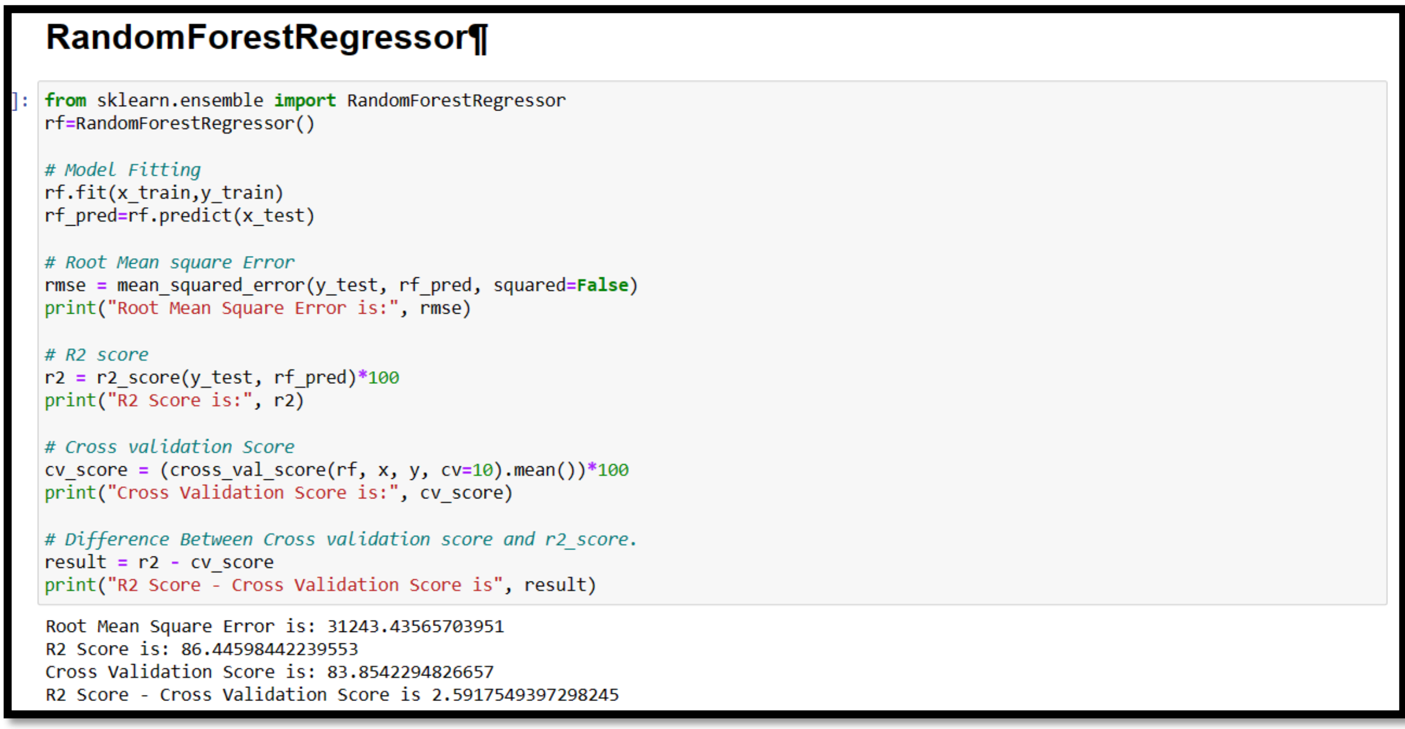
Linear Regression

```
In [131]: lr=LinearRegression()
          lr.fit(x_train,y_train)
          lr_pred=lr.predict(x_test)
          rmse = mean_squared_error(y_test, lr_pred, squared=False)
          print("Root Mean Square Error is:", rmse)
          # R2 score
          r2 = r2_score(y_test, lr_pred)*100
          print("R2 Score is:", r2)
          cv_score = (cross_val_score(lr, x, y, cv=10).mean())*100
          print("Cross Validation Score is:", cv_score)
          result = r2 - cv_score
          print("R2 Score - Cross Validation Score is", result)

          Root Mean Square Error is: 31556.092380756276
          R2 Score is: 84.18759276911584
          Cross Validation Score is: 77.51864530401352
          R2 Score - Cross Validation Score is 6.66894746510232
```

Support Vector Regressor

```
In [132]: svr=SVR()

          # Model Fitting
          svr.fit(x_train,y_train)
```

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning**,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

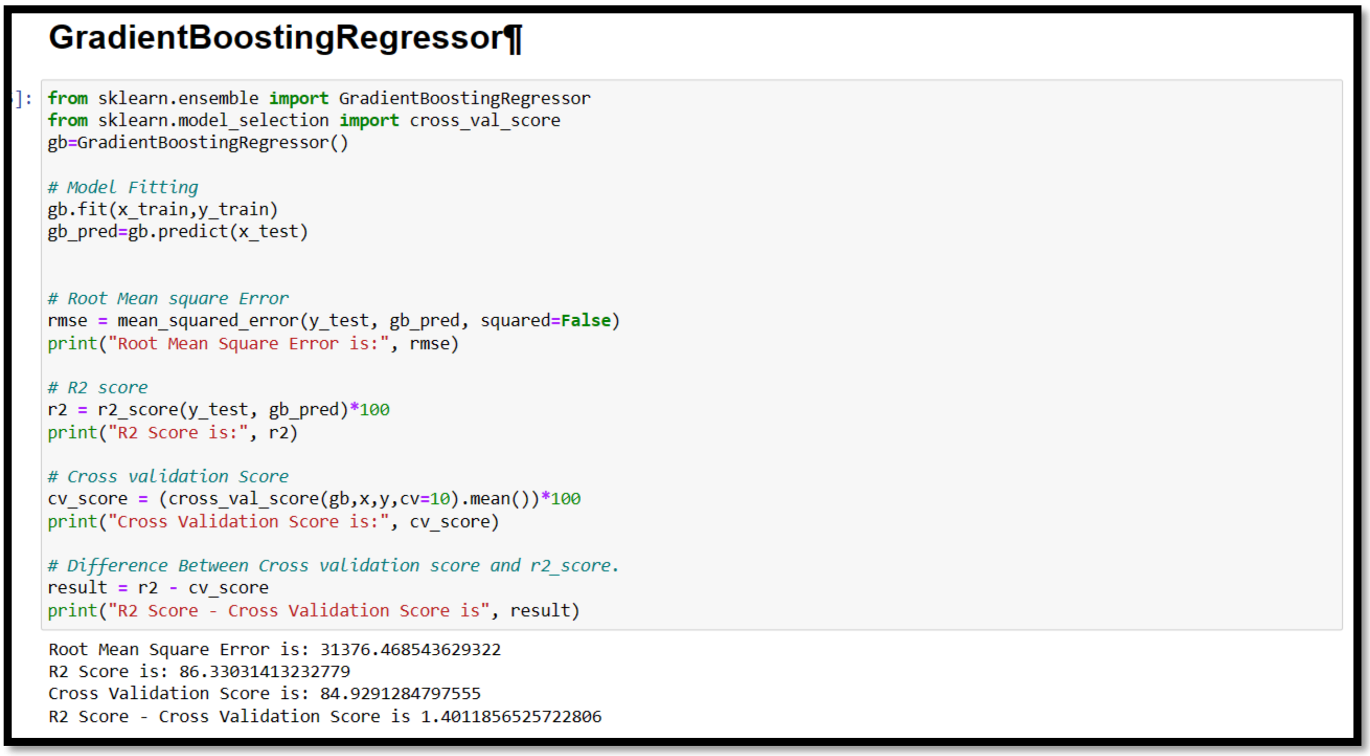


RandomForestRegressor¶

```
]: from sklearn.ensemble import RandomForestRegressor
   rf=RandomForestRegressor()

   # Model Fitting
   rf.fit(x_train,y_train)
   rf_pred=rf.predict(x_test)

   # Root Mean square Error
   rmse = mean_squared_error(y_test, rf_pred, squared=False)
   print("Root Mean Square Error is:", rmse)

   # R2 score
   r2 = r2_score(y_test, rf_pred)*100
   print("R2 Score is:", r2)

   # Cross validation Score
   cv_score = (cross_val_score(rf, x, y, cv=10).mean())*100
   print("Cross Validation Score is:", cv_score)

   # Difference Between Cross validation score and r2_score.
   result = r2 - cv_score
   print("R2 Score - Cross Validation Score is", result)
```

```
Root Mean Square Error is: 31243.43565703951
R2 Score is: 86.44598442239553
Cross Validation Score is: 83.8542294826657
R2 Score - Cross Validation Score is 2.5917549397298245
```

Gradient Boosting Regression is an analytical technique that is designed to explore the relationship between two or more variables (X, and Y)**.**

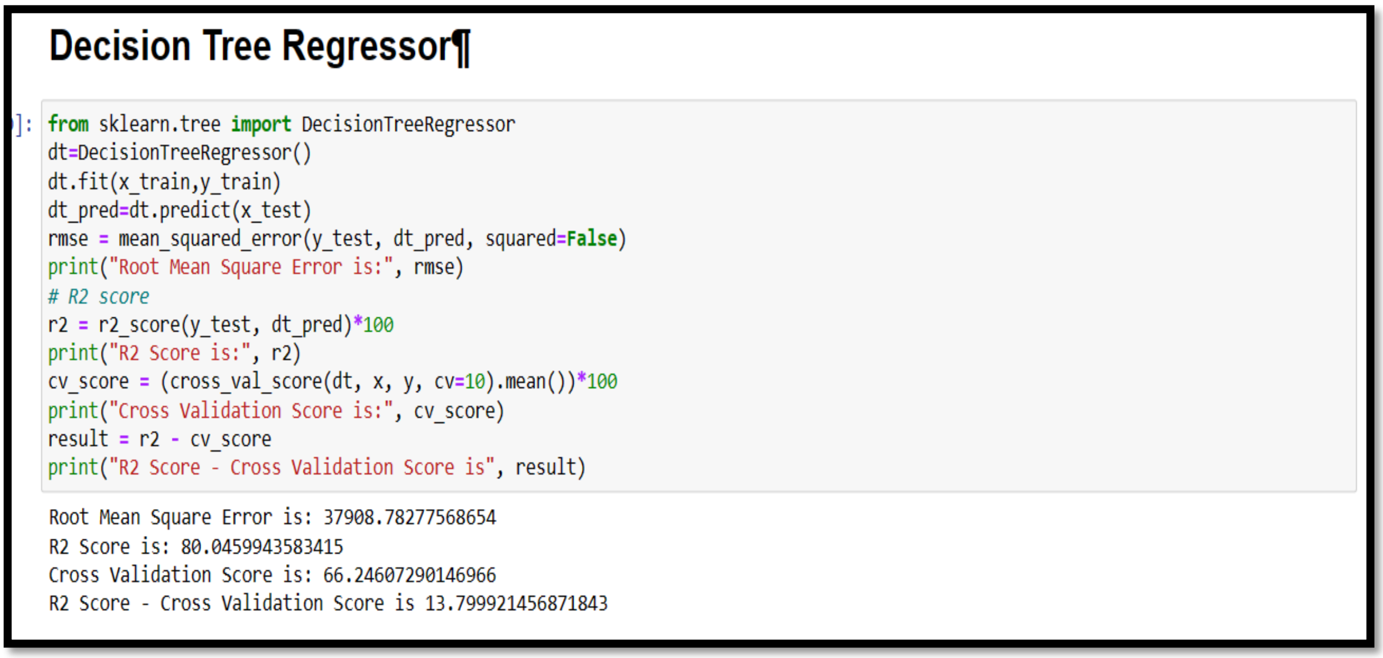


GradientBoostingRegressor¶

```
]: from sklearn.ensemble import GradientBoostingRegressor
   from sklearn.model_selection import cross_val_score
   gb=GradientBoostingRegressor()

   # Model Fitting
   gb.fit(x_train,y_train)
   gb_pred=gb.predict(x_test)


   # Root Mean square Error
   rmse = mean_squared_error(y_test, gb_pred, squared=False)
   print("Root Mean Square Error is:", rmse)

   # R2 score
   r2 = r2_score(y_test, gb_pred)*100
   print("R2 Score is:", r2)

   # Cross validation Score
   cv_score = (cross_val_score(gb,x,y,cv=10).mean())*100
   print("Cross Validation Score is:", cv_score)

   # Difference Between Cross validation score and r2_score.
   result = r2 - cv_score
   print("R2 Score - Cross Validation Score is", result)
```

```
Root Mean Square Error is: 31376.468543629322
R2 Score is: 86.33031413232779
Cross Validation Score is: 84.9291284797555
R2 Score - Cross Validation Score is 1.4011856525722806
```

A decision tree can be used for **classification or regression**. It operates by dividing the data into smaller and smaller subgroups in a tree-like arrangement. When estimating the output value of a set of characteristics, it will do so based on the subset into which the set of features falls.

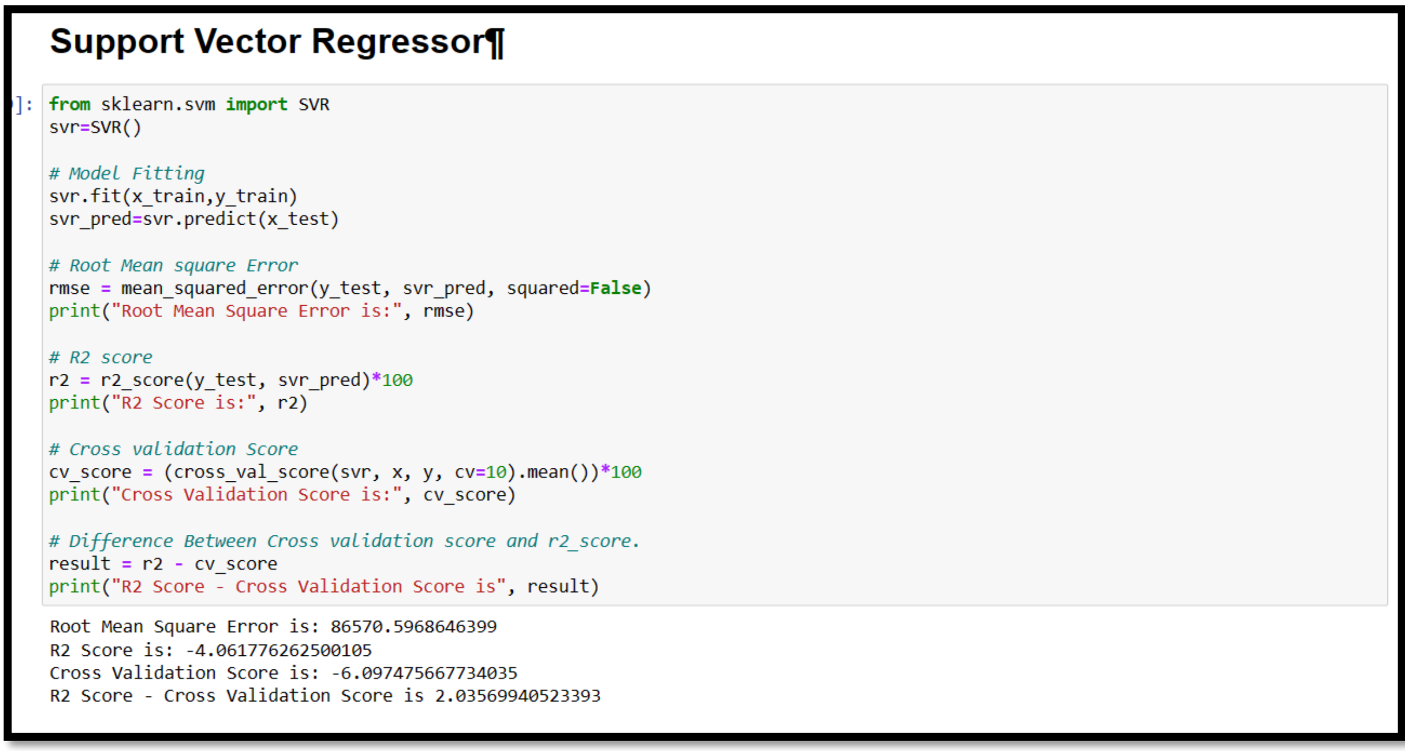


Decision Tree Regressor¶

```
]: from sklearn.tree import DecisionTreeRegressor
dt=DecisionTreeRegressor()
dt.fit(x_train,y_train)
dt_pred=dt.predict(x_test)
rmse = mean_squared_error(y_test, dt_pred, squared=False)
print("Root Mean Square Error is:", rmse)
# R2 score
r2 = r2_score(y_test, dt_pred)*100
print("R2 Score is:", r2)
cv_score = (cross_val_score(dt, x, y, cv=10).mean())*100
print("Cross Validation Score is:", cv_score)
result = r2 - cv_score
print("R2 Score - Cross Validation Score is", result)
```

```
Root Mean Square Error is: 37908.78277568654
R2 Score is: 80.0459943583415
Cross Validation Score is: 66.24607290146966
R2 Score - Cross Validation Score is 13.799921456871843
```

Support Vector Regression as the name suggests is a regression algorithm that supports both linear and non-linear regressions.

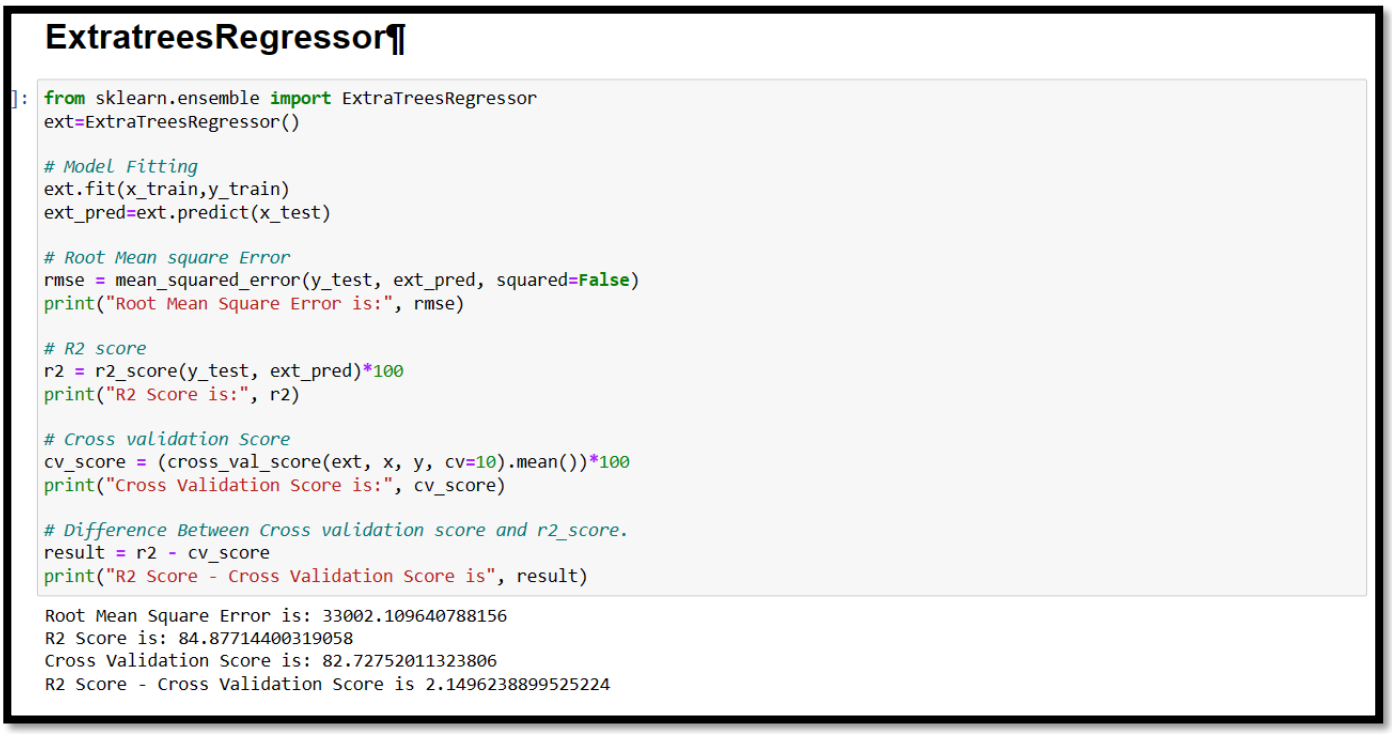


Support Vector Regressor¶

```
]: from sklearn.svm import SVR
   svr=SVR()

   # Model Fitting
   svr.fit(x_train,y_train)
   svr_pred=svr.predict(x_test)

   # Root Mean square Error
   rmse = mean_squared_error(y_test, svr_pred, squared=False)
   print("Root Mean Square Error is:", rmse)

   # R2 score
   r2 = r2_score(y_test, svr_pred)*100
   print("R2 Score is:", r2)

   # Cross validation Score
   cv_score = (cross_val_score(svr, x, y, cv=10).mean())*100
   print("Cross Validation Score is:", cv_score)

   # Difference Between Cross validation score and r2_score.
   result = r2 - cv_score
   print("R2 Score - Cross Validation Score is", result)
```

```
Root Mean Square Error is: 86570.5968646399
R2 Score is: -4.061776262500105
Cross Validation Score is: -6.097475667734035
R2 Score - Cross Validation Score is 2.03569940523393
```

An extra-trees regressor. This class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

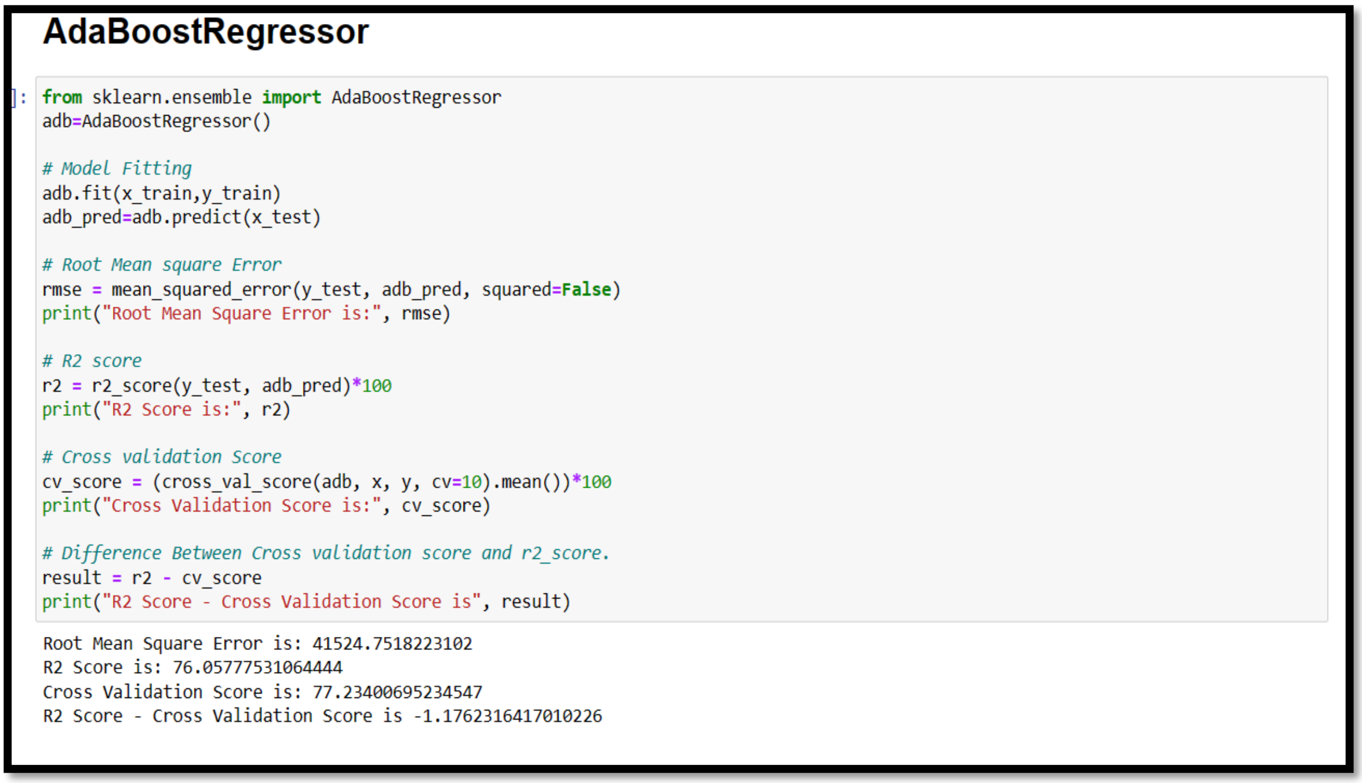


ExtratreesRegressor¶

```
]: from sklearn.ensemble import ExtraTreesRegressor
ext=ExtraTreesRegressor()

# Model Fitting
ext.fit(x_train,y_train)
ext_pred=ext.predict(x_test)

# Root Mean square Error
rmse = mean_squared_error(y_test, ext_pred, squared=False)
print("Root Mean Square Error is:", rmse)

# R2 score
r2 = r2_score(y_test, ext_pred)*100
print("R2 Score is:", r2)

# Cross validation Score
cv_score = (cross_val_score(ext, x, y, cv=10).mean())*100
print("Cross Validation Score is:", cv_score)

# Difference Between Cross validation score and r2_score.
result = r2 - cv_score
print("R2 Score - Cross Validation Score is", result)
```

```
Root Mean Square Error is: 33002.109640788156
R2 Score is: 84.87714400319058
Cross Validation Score is: 82.72752011323806
R2 Score - Cross Validation Score is 2.1496238899525224
```

An AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction.

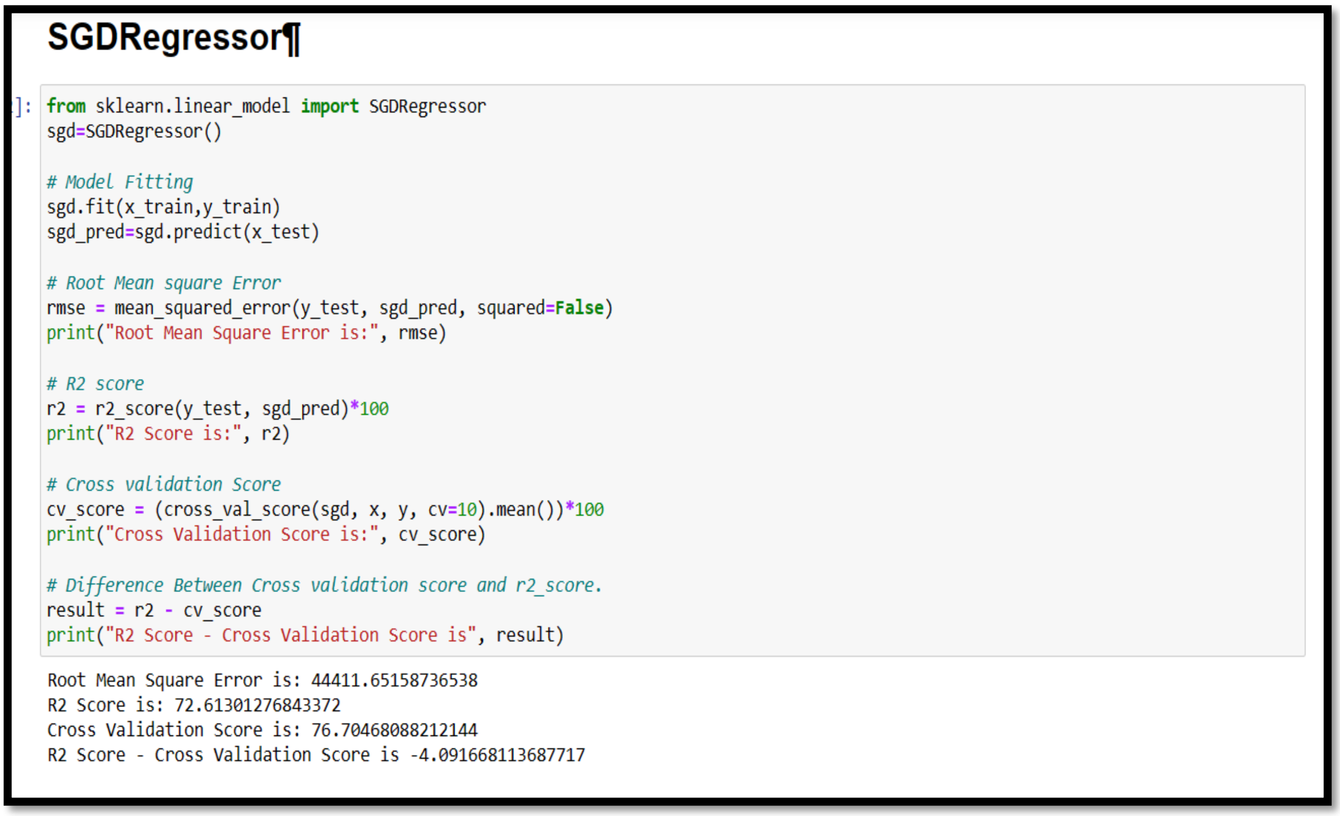


AdaBoostRegressor

```
from sklearn.ensemble import AdaBoostRegressor
adb=AdaBoostRegressor()

# Model Fitting
adb.fit(x_train,y_train)
adb_pred=adb.predict(x_test)

# Root Mean square Error
rmse = mean_squared_error(y_test, adb_pred, squared=False)
print("Root Mean Square Error is:", rmse)

# R2 score
r2 = r2_score(y_test, adb_pred)*100
print("R2 Score is:", r2)

# Cross validation Score
cv_score = (cross_val_score(adb, x, y, cv=10).mean())*100
print("Cross Validation Score is:", cv_score)

# Difference Between Cross validation score and r2_score.
result = r2 - cv_score
print("R2 Score - Cross Validation Score is", result)
```

```
Root Mean Square Error is: 41524.7518223102
R2 Score is: 76.05777531064444
Cross Validation Score is: 77.23400695234547
R2 Score - Cross Validation Score is -1.1762316417010226
```

SGD stands for Stochastic Gradient Descent: the gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength schedule (aka learning rate).

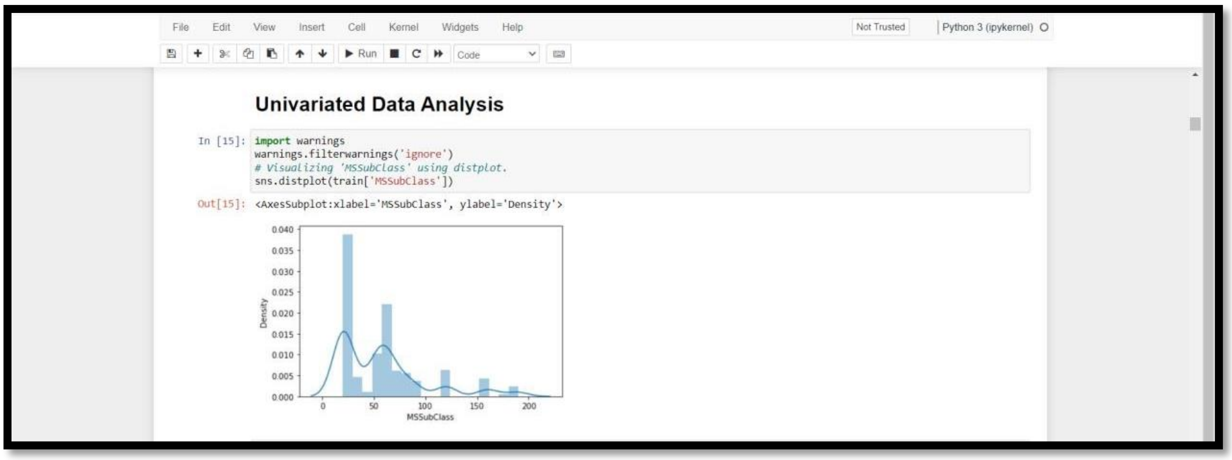


SGDRegressor¶

```
]: from sklearn.linear_model import SGDRegressor
sgd=SGDRegressor()

# Model Fitting
sgd.fit(x_train,y_train)
sgd_pred=sgd.predict(x_test)

# Root Mean square Error
rmse = mean_squared_error(y_test, sgd_pred, squared=False)
print("Root Mean Square Error is:", rmse)

# R2 score
r2 = r2_score(y_test, sgd_pred)*100
print("R2 Score is:", r2)

# Cross validation Score
cv_score = (cross_val_score(sgd, x, y, cv=10).mean())*100
print("Cross Validation Score is:", cv_score)

# Difference Between Cross validation score and r2_score.
result = r2 - cv_score
print("R2 Score - Cross Validation Score is", result)
```

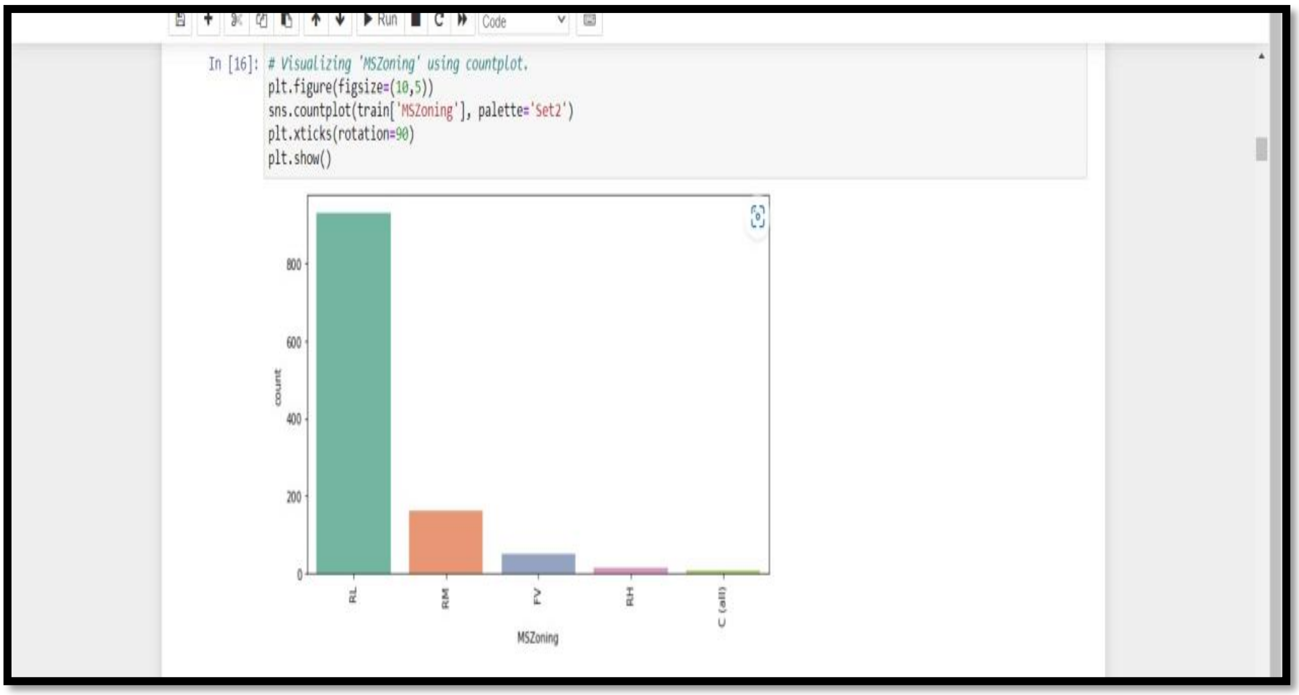
```
Root Mean Square Error is: 44411.65158736538
R2 Score is: 72.61301276843372
Cross Validation Score is: 76.70468088212144
R2 Score - Cross Validation Score is -4.091668113687717
```

- **Key Metrics for success in solving problem under consideration**
  A company's units can use these dashboards to create milestones and monitor their progress by tracking all the most relevant metrics in one location.
- **Visualizations**

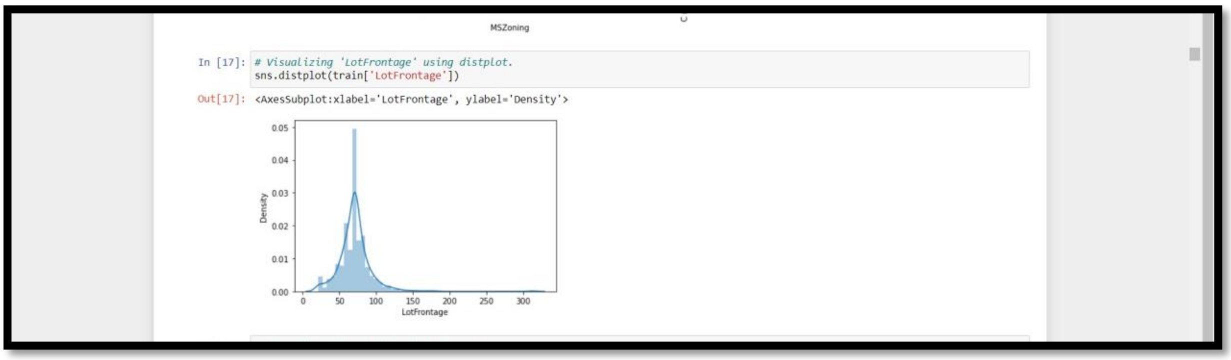


Univariated Data Analysis

```
In [15]: import warnings
         warnings.filterwarnings('ignore')
         # Visualizing 'MSSubClass' using distplot.
         sns.distplot(train['MSSubClass'])
Out[15]: <AxesSubplot:xlabel='MSSubClass', ylabel='Density'>
```

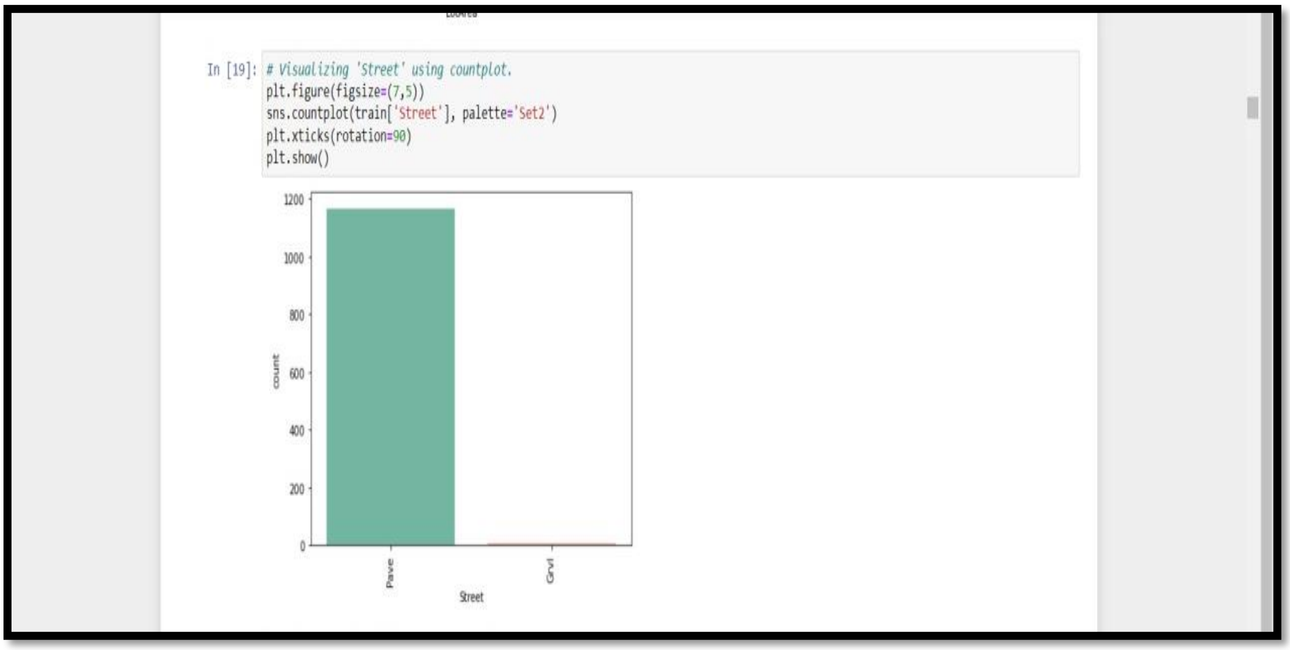


```
In [16]: # Visualizing 'MSZoning' using countplot.
         plt.figure(figsize=(10,5))
         sns.countplot(train['MSZoning'], palette='Set2')
         plt.xticks(rotation=90)
         plt.show()
```

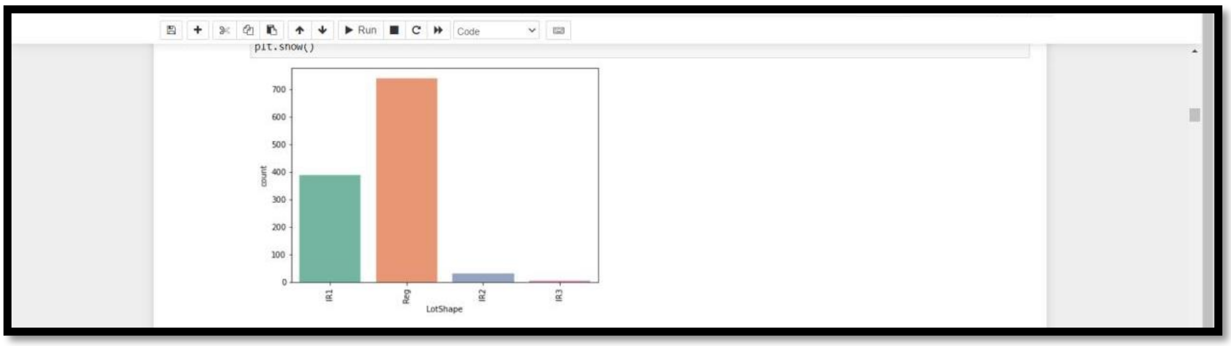


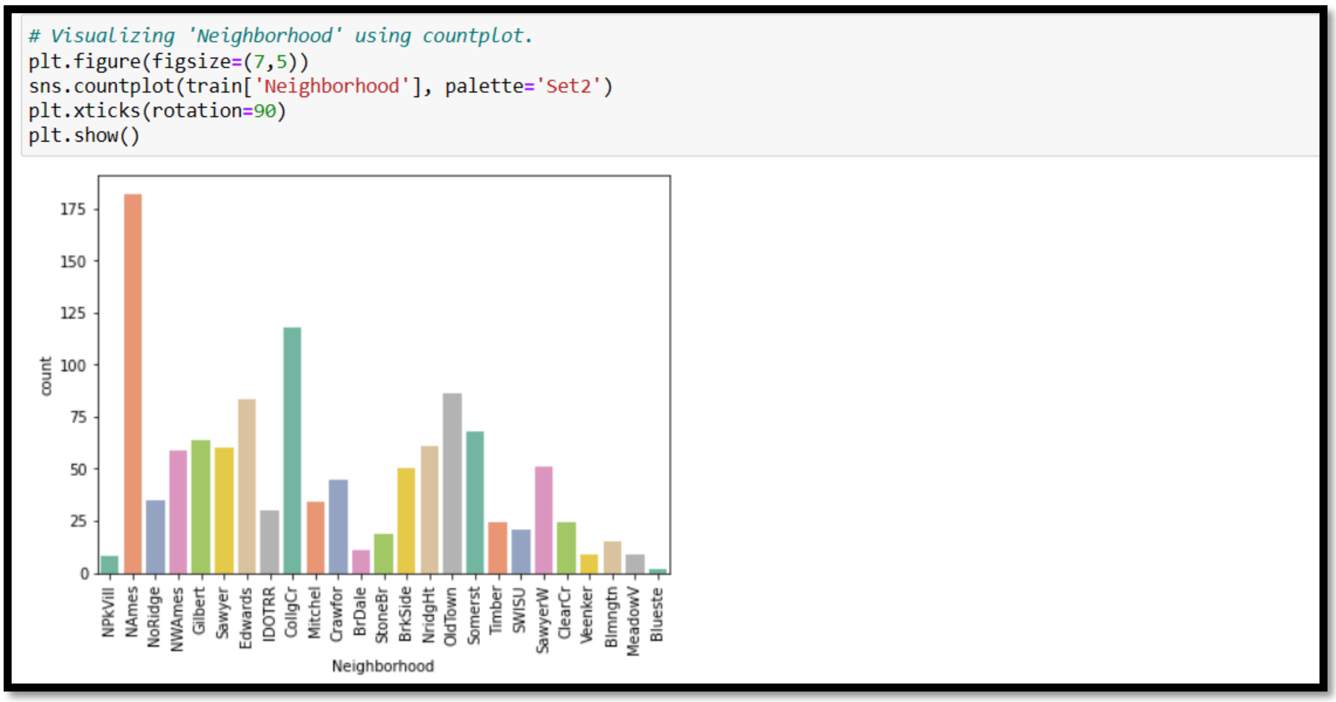
MSZoning

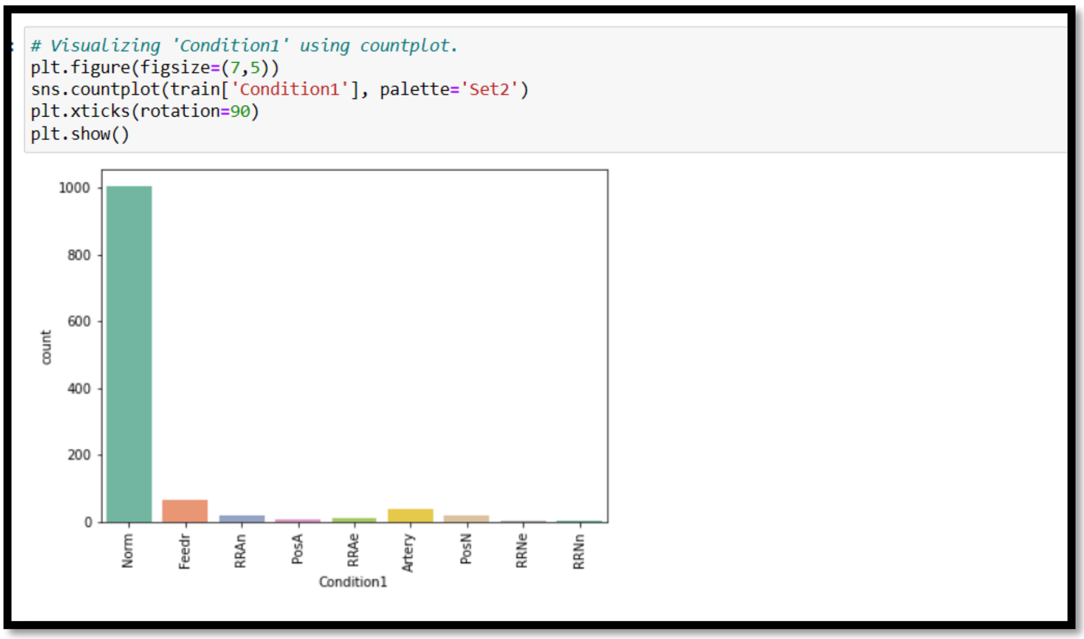
```
In [17]: # Visualizing 'LotFrontage' using distplot.
         sns.distplot(train['LotFrontage'])

Out[17]: <AxesSubplot:xlabel='LotFrontage', ylabel='Density'>
```



```
In [19]: # Visualizing 'Street' using countplot.
         plt.figure(figsize=(7,5))
         sns.countplot(train['Street'], palette='Set2')
         plt.xticks(rotation=90)
         plt.show()
```


```
plt.show()
```


```
# Visualizing 'Neighborhood' using countplot.
plt.figure(figsize=(7,5))
sns.countplot(train['Neighborhood'], palette='Set2')
plt.xticks(rotation=90)
plt.show()
```



```
# Visualizing 'Condition1' using countplot.
plt.figure(figsize=(7,5))
sns.countplot(train['Condition1'], palette='Set2')
plt.xticks(rotation=90)
plt.show()
```

Observations:

1.lotFrontage: Almost all houses have LotFrontage between 20 to 150

2.lotArea: Around 580 house have lot Area between (0-10000) sqft. Very few houses have lot area around 120000sqft & around 160000sqft

3.OverallQual: Rates the overall material and finish of the house, around 300 houses sold were in average condition. Only 10-15 houses were in excellent condition.

4.YearBuilt: Original construction date, a greater number of people have brought the houses build after 1990.

5.MasVnrArea: Masonry veneer area in square feet, 50% of houses have Masonry veneer area as '0-50' and out of rest 50% houses most houses have Masonry veneer area 50-1200

6.BsmtFinSF1: Type 1 finished square feet, most houses have Type 1 finished square feet area of basement between 0 and 1500

7.BsmtFinSF2: Type 2 finished square feet, around 1000 houses have Type 2 finished square feet area of 0

8.BsmtUnfSF: Unfinished square feet of basement area, around 130 houses have unfinished basement of area around 100-500 sqft.

9.1stFlrSF: First Floor square feet, around 280 houses have 1st floor square feet area between 800-1200sqft.

10. GrLivArea: Above grade (ground) living area square feet, most houses have above ground living sq. ft area in between 800 to 3000

11. BsmtFullBath: Basement full bathrooms,50% houses have no full bathrooms in basement and in remaining houses most have 1 full bathroom in basement and very few has 2 full bathrooms.

12. FullBath: Full bathrooms above grade,25% houses have 1 full bathroom above ground and 50% have 2 full bathrooms located above ground and very less have 3.

13. HalfBath: Half baths above grade, around 700 houses have no half bathrooms very few has 1 half bathroom.

14. Bedroom: Bedrooms above grade (does NOT include basement bedrooms), Most houses have 3 bedrooms above ground followed by 2 and 4.

15. Kitchen: Kitchens above grade, Maximum houses have 1 Kitchen. very few have 2.

16. TotRmsAbvGrd: Total rooms above grade (does not include bathrooms), Around 300 houses have 6 rooms, around 200 have 5, &250 have 7. Very few have 12 & 14 rooms.

17. Fireplaces: Number of fireplaces, most houses have 0 fireplaces followed by 1.

18. GarageCars: Size of garage in car capacity, most houses have garage with 2 car capacity.

19. GarageArea: Size of garage in square feet, most houses have Garage area in between 200 to 800.

20. woodDeckSF: Wood deck area in square feet, more than 50% of houses have 0 Wood Deck sqft area and rest have in between 0 to 400

21. OpenPorchSF: Open porch area in square feet, 25% of houses have 0 open porch sqft area and rest have in between 0 to 300

22. EnclosedPorch: Enclosed porch area in square feet, almost all houses have 0 enclosed porch sqft area

23. ScreenPorch: Screen porch area in square feet, almost all houses have 0 screen porch area sqft

24. Sale Price: Around 500 houses have sale price in between 100000 to 200000. Very few houses have sale price of 600000 & 700000

- **Interpretation of the Results**

  Here after pre-processing we get the data encoded for framing the model and after visualization, we observe that the data contains skewness and we removed it using power transformation (yeo-john method), After checking the VIF factor, we observe that the dataset contains multicollinearity and we removed it by removing one column from dataset which has highest multicollinearity, After data pre-processing and EDA we build 8 different algorithms for dataset and from them Gradient Boosting Regressor algorithm perform very well , Lastly we perform hyper parameter tunning to enhance the r2 score. Here we finalise the gradient boosting algorithm as our best fit model.