FLIP ROBO

HOUSE PRICE PREDICTION

Submitted By

Shivanchal Asthana

ACKNOWLEDGMENT

It is my sensual gratification to present this report. Working on this project was an incredible experience that will have a tremendous impact on my career. I would like to express my sincere thanks to the company Flip Robo Technologies for a regular follow up and valuable suggestions provided throughout. They always been an origin of spark and direction. I also thank all the respondents who have given their valuable time, views and valid information for this project.

Shivanchal Asthana

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 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.

• Conceptual Background of the Domain Problem:

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 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?

• Review of Literature:

Based on the sample data provided, we understand that the company is looking at prospective properties to buy houses to enter the market. The data given, explains that it is a regression problem where we need to build a machine learning model in order to predict the actual prices of the prospective properties and then decide whether to invest in them or not. Also, we have other independent variables which would help us in understanding which features are important to predict the values and how do these variables describe the prices of the house.

• Motivation for the Problem Undertaken:

The main objective of doing this project is to build a machine learning model to predict the prices of the house based on the supporting features. Will be predicting the same with the help of machine learning algorithms. The data is provided to us from client database. We need to do some predictions that could help the clients in the further investments and also in improvement in selection of customers. To explore various impacts of features on prediction methods, this project will apply machine learning approaches to investigate difference between several models. This project will also validate multiple techniques in model implementation on regression and provide an optimistic result for predicting the house prices.

Analytical Problem Framing

• Mathematical/ Analytical Modeling of the Problem

We are building a machine learning model to predict the actual value of the prospective properties and to decide whether to invest in them or not. This model will help us to determine which variables are important to predict the price of variables and also how do these variables describe the price of the house. With the help of the independent variables, will be able to determine the price of the houses. Regression analysis is a form of predictive modelling technique which investigates the relationship between dependent and independent variables. This technique is used for forecasting, time series modelling, and also find the relationship between the variables. The most common form of regression analysis is linear regression, in which one finds the line that most closely fits the data according to a mathematical criterion.

• Data Sources and their formats:

Dataset provided by Flip Robo Technologies is in the CSV format. There are 2 datasets given to us. One is training data and the other is testing data.

1. Train file will be used for training the model, where the model will learn from this file. The training data contains all the independent variables and the target variable. The size of the train data is 1168 records

2. Test file contains all the independent variables, but not the target variable. Here we will be applying the model to predict the target variable. The size of the test data is 292 records.

• Data Preprocessing Done:

Data Pre-processing refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for building and training machine learning models. Whenever the data is gathered from various sources, it is collected in raw format. Data pre-processing is considered as an integral part in Machine Learning. It’s extremely important to pre-process the data before feeding it to our model. Hence, it is considered to be the first and crucial step while creating a machine learning model. Data pre-processing steps used in this project are as below:

1. Loading the dataset
2. Checked the number of rows and columns in the dataset.
3. Checked the null values in the dataset and filled them using interpolate, bfill method.
4. Checked the zero values and treat them
5. Check the outliers and remove them by using IQR Method.
6. Dropped the unrequired columns from the dataset.
7. Checked the unique values.
8. Converted all the data to numeric data type using Label Encoding technique.

• Data Inputs- Logic- Output Relationships:

Data inputs are the specifications of the house like density of house, floor area, street etc. and Output is the price of the house according to house specifications and house age.

- **State the set of assumptions (if any) related to the problem under consideration:**

  The assumption part for me was relying strictly on the data given and taking into consideration that the separate datasets are provided, training and the testing dataset which were obtained from people and survey was also done on it depending on their preferences.

- **Hardware and Software Requirements and Tools Used:**

  Hardware is used by me that is i5 intel core, 8GB RAM, 64Bit processor, and software is Jupyter Notebook for coding along with MS Word to make useful report, MS – PowerPoint to make presentation. For coding, we need to install NumPy, Pandas, Sklearn libraries.

Model/s Development and Evaluation

- **Identification of possible problem-solving approaches (methods):**

This dataset contains two csv files, first one is Train dataset and another one is Test dataset. We first concatenate both the dataset for data cleaning purpose and at the end, we separate it. There are many null values and zero values lies in these datasets. We treat zero values by replacing columns mean value and sometimes, we replace it by z=null values then null values will be treated by interpolate method and bfill method. In many features we find outliers, we treat them by using IQR method and after cleaning the dataset. We see there are many categorical

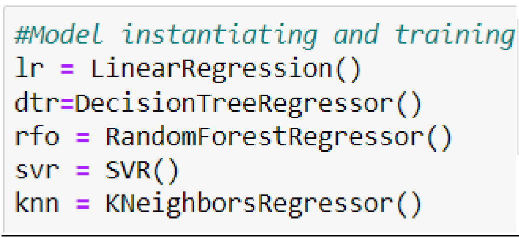
columns, we use label encoder then to convert all categorical values into numerical form.
After that, we try to remove multicollinearity problem by using Heatmap and Variance inflation method. By using standard scaling, we scale the data. Then we use almost 5-6 models to check the accuracy and select the model which is giving us best accuracy and then we perform Hyperparameter tuning to improve the accuracy and we save our best model.

- <u>**Testing of Identified Approaches (Algorithms):**</u>

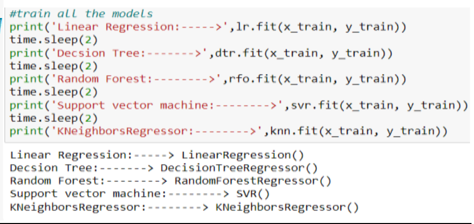
• Linear regression'
•Decision Tree regressor
•kNeighbors regressor
•Support Vector Regressor
•Random forest Regressor
•Lasso regressor
•Ridge Regressor
•AdaBoost Regressor
•XGBoost Regressor
•ExtremeGradientBoost Regressor

- <u>**Run and Evaluate selected models**</u>

We check the accuracy by using 5 models

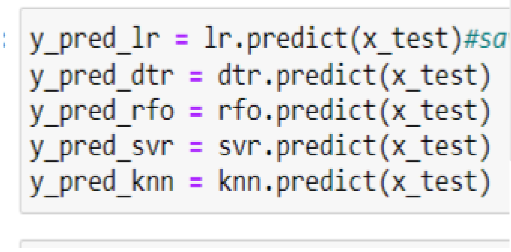


```
#Model instantiating and training
lr = LinearRegression()
dtr=DecisionTreeRegressor()
rfo = RandomForestRegressor()
svr = SVR()
knn = KNeighborsRegressor()
```

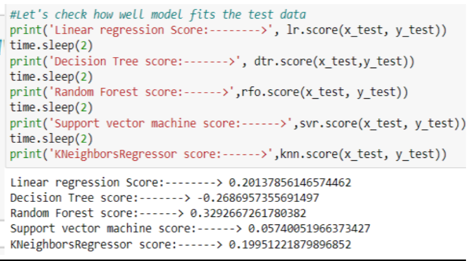


```
#train all the models
print('Linear Regression:----->',lr.fit(x_train, y_train))
time.sleep(2)
print('Decsion Tree:------->',dtr.fit(x_train, y_train))
time.sleep(2)
print('Random Forest:-------->',rfo.fit(x_train, y_train))
time.sleep(2)
print('Support vector machine:-------->',svr.fit(x_train, y_train))
time.sleep(2)
print('KNeighborsRegressor:-------->',knn.fit(x_train, y_train))
```

```
Linear Regression:-----> LinearRegression()
Decsion Tree:-------> DecisionTreeRegressor()
Random Forest:--------> RandomForestRegressor()
Support vector machine:--------> SVR()
KNeighborsRegressor:--------> KNeighborsRegressor()
```



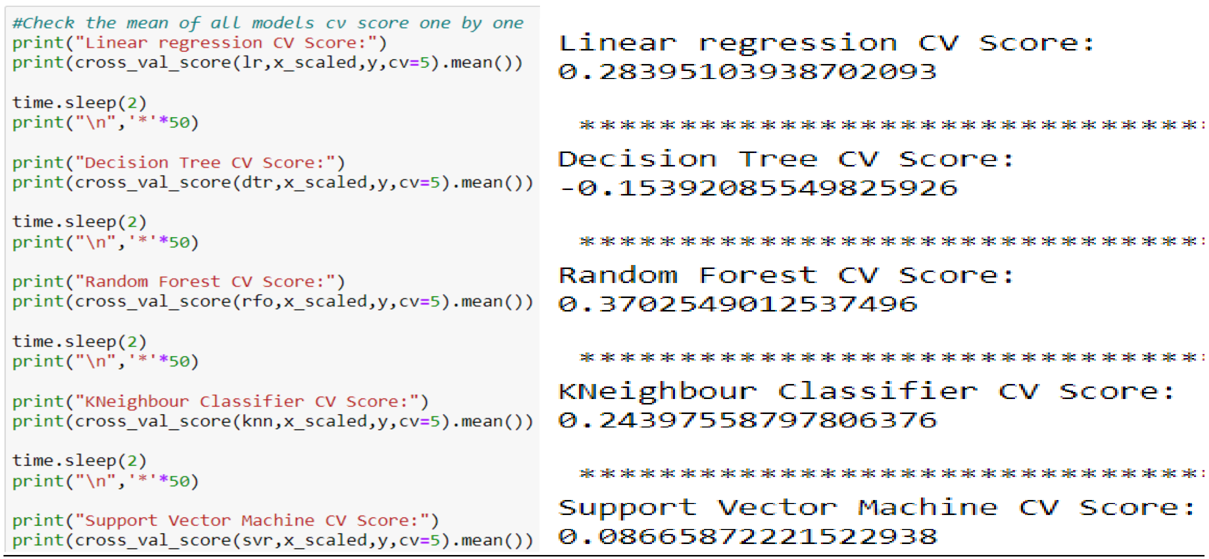
```
y_pred_lr = lr.predict(x_test)#sa
y_pred_dtr = dtr.predict(x_test)
y_pred_rfo = rfo.predict(x_test)
y_pred_svr = svr.predict(x_test)
y_pred_knn = knn.predict(x_test)
```



```
#Let's check how well model fits the test data
print('Linear regression Score:-------->', lr.score(x_test, y_test))
time.sleep(2)
print('Decision Tree score:------->', dtr.score(x_test,y_test))
time.sleep(2)
print('Random Forest score:------>',rfo.score(x_test, y_test))
time.sleep(2)
print('Support vector machine score:------>',svr.score(x_test, y_test))
time.sleep(2)
print('KNeighborsRegressor score:------>',knn.score(x_test, y_test))
```

```
Linear regression Score:--------> 0.20137856146574462
Decision Tree score:-------> -0.2686957355691497
Random Forest score:------> 0.3292667261780382
Support vector machine score:------> 0.05740051966373427
KNeighborsRegressor score:------> 0.19951221879896852
```

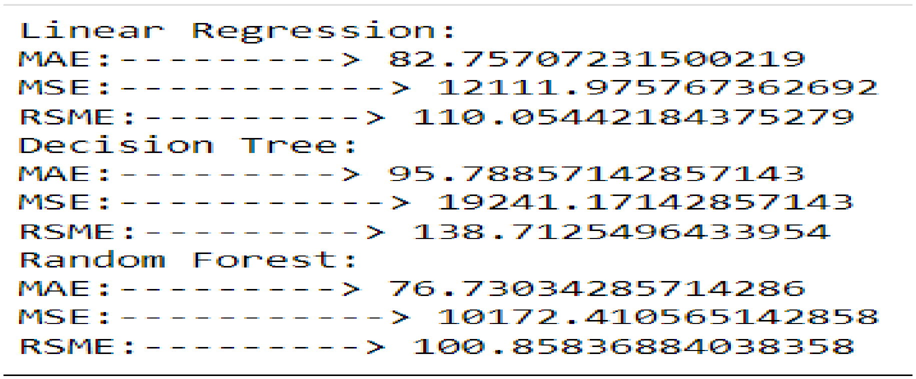
CV Score:



```
#Check the mean of all models cv score one by one
print("Linear regression CV Score:")
print(cross_val_score(lr,x_scaled,y,cv=5).mean())

time.sleep(2)
print("\n",'*'*50)

print("Decision Tree CV Score:")
print(cross_val_score(dtr,x_scaled,y,cv=5).mean())

time.sleep(2)
print("\n",'*'*50)

print("Random Forest CV Score:")
print(cross_val_score(rfo,x_scaled,y,cv=5).mean())

time.sleep(2)
print("\n",'*'*50)

print("KNeighbour Classifier CV Score:")
print(cross_val_score(knn,x_scaled,y,cv=5).mean())

time.sleep(2)
print("\n",'*'*50)

print("Support Vector Machine CV Score:")
print(cross_val_score(svr,x_scaled,y,cv=5).mean())
```

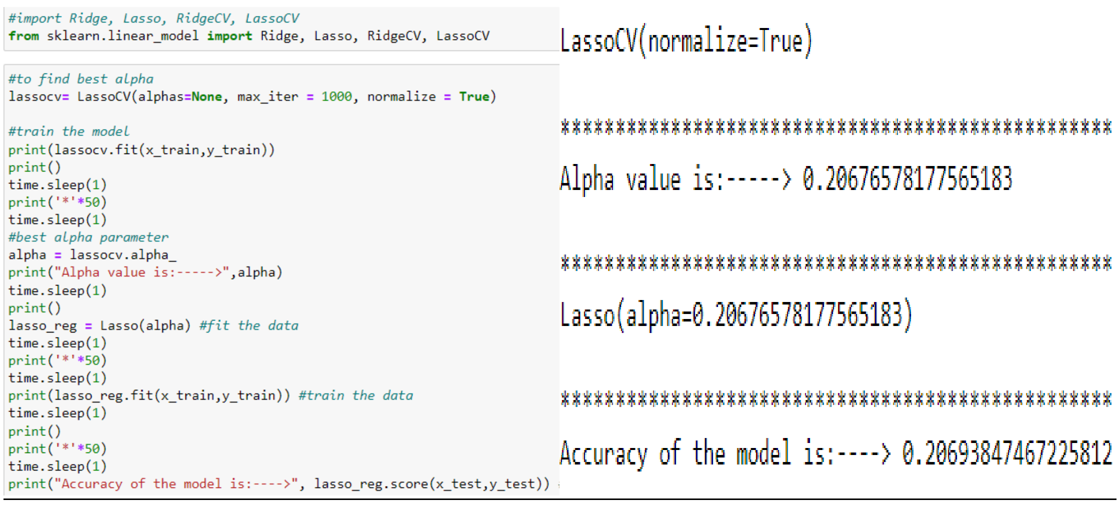
```
Linear regression CV Score:
0.28395103938702093

 ********************************

Decision Tree CV Score:
-0.15392085549825926

 ********************************

Random Forest CV Score:
0.3702549012537496

 ********************************

KNeighbour Classifier CV Score:
0.24397558797806376

 ********************************

Support Vector Machine CV Score:
0.08665872221522938
```

MAE/MSE/RSME:



```
Linear Regression:
MAE:----------> 82.75707231500219
MSE:------------> 12111.975767362692
RSME:----------> 110.05442184375279
Decision Tree:
MAE:----------> 95.78857142857143
MSE:------------> 19241.17142857143
RSME:----------> 138.7125496433954
Random Forest:
MAE:----------> 76.73034285714286
MSE:------------> 10172.410565142858
RSME:----------> 100.85836884038358
```

LASSO Regression:

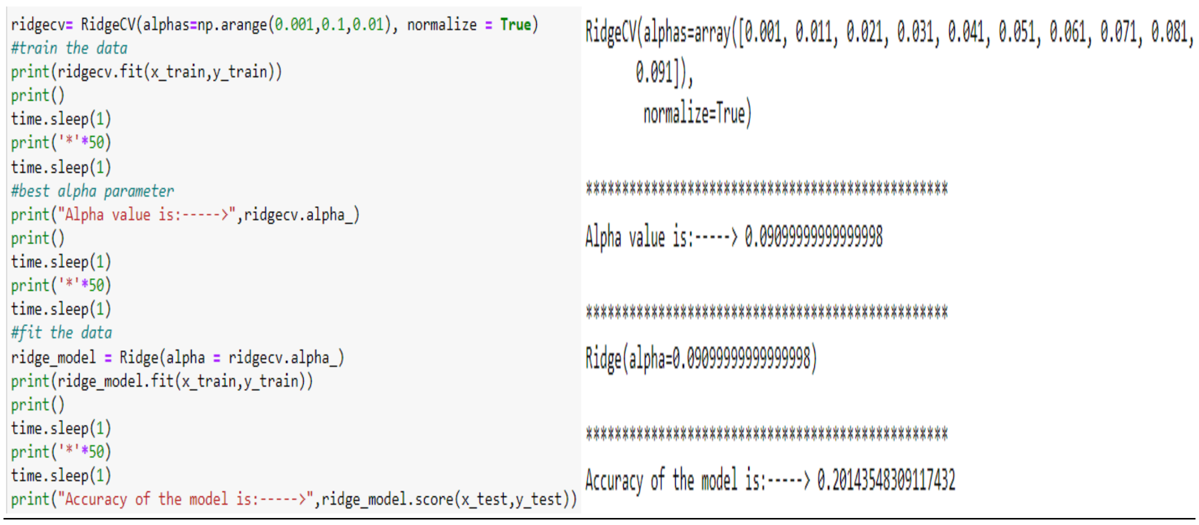


```
#import Ridge, Lasso, RidgeCV, LassoCV
from sklearn.linear_model import Ridge, Lasso, RidgeCV, LassoCV
```

```
#to find best alpha
lassocv= LassoCV(alphas=None, max_iter = 1000, normalize = True)

#train the model
print(lassocv.fit(x_train,y_train))
print()
time.sleep(1)
print('*'*50)
time.sleep(1)
#best alpha parameter
alpha = lassocv.alpha_
print("Alpha value is:----->",alpha)
time.sleep(1)
print()
lasso_reg = Lasso(alpha) #fit the data
time.sleep(1)
print('*'*50)
time.sleep(1)
print(lasso_reg.fit(x_train,y_train)) #train the data
time.sleep(1)
print()
print('*'*50)
time.sleep(1)
print("Accuracy of the model is:---->", lasso_reg.score(x_test,y_test))
```

```
LassoCV(normalize=True)

**************************************************

Alpha value is:-----> 0.20676578177565183

**************************************************

Lasso(alpha=0.20676578177565183)

**************************************************

Accuracy of the model is:----> 0.20693847467225812
```

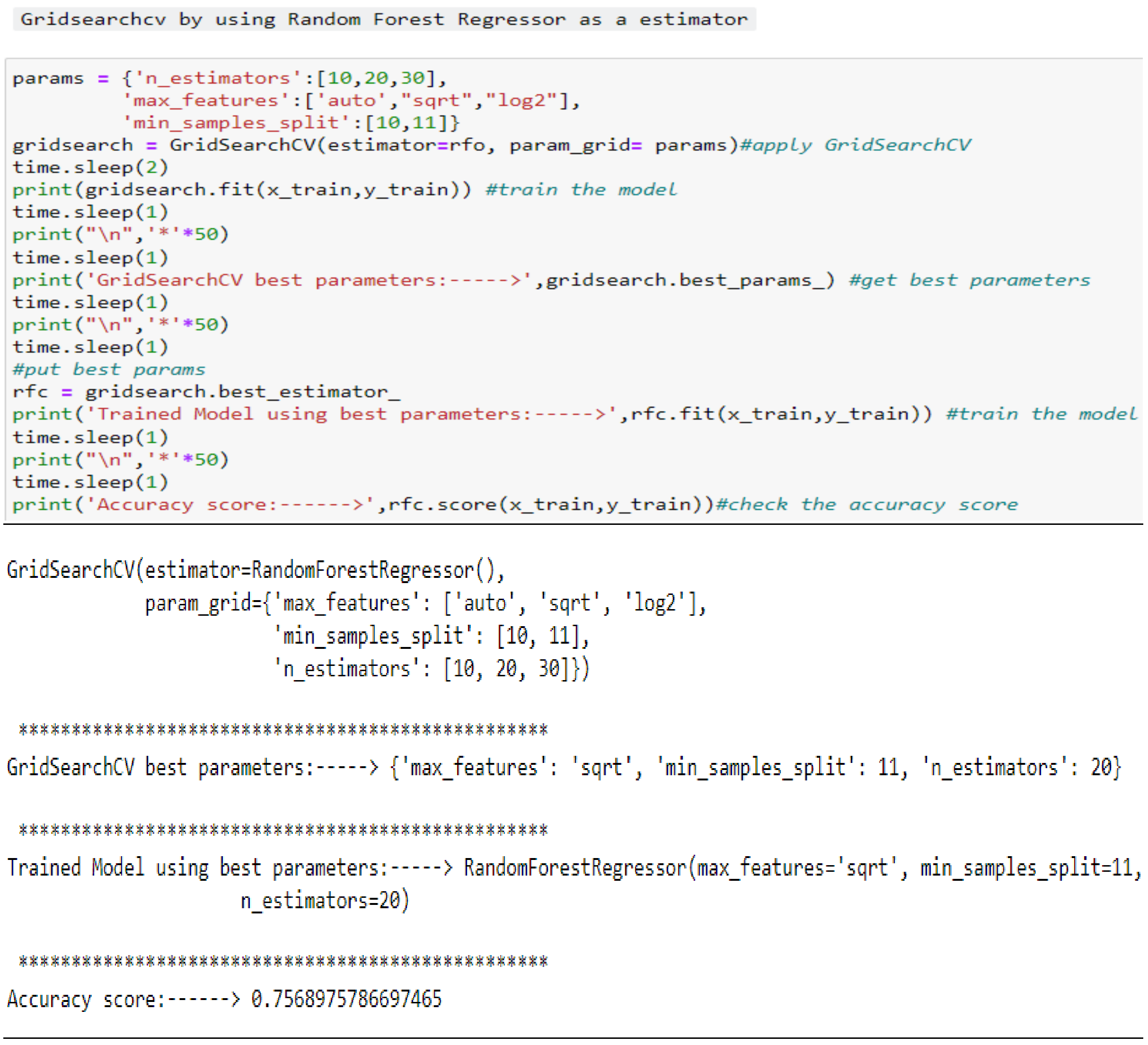
Ridge Regression:



```
ridgecv= RidgeCV(alphas=np.arange(0.001,0.1,0.01), normalize = True)
#train the data
print(ridgecv.fit(x_train,y_train))
print()
time.sleep(1)
print('*'*50)
time.sleep(1)
#best alpha parameter
print("Alpha value is:----->",ridgecv.alpha_)
print()
time.sleep(1)
print('*'*50)
time.sleep(1)
#fit the data
ridge_model = Ridge(alpha = ridgecv.alpha_)
print(ridge_model.fit(x_train,y_train))
print()
time.sleep(1)
print('*'*50)
time.sleep(1)
print("Accuracy of the model is:----->",ridge_model.score(x_test,y_test))
```

```
RidgeCV(alphas=array([0.001, 0.011, 0.021, 0.031, 0.041, 0.051, 0.061, 0.071, 0.081,
       0.091]),
        normalize=True)

**************************************************

Alpha value is:-----> 0.09099999999999998

**************************************************

Ridge(alpha=0.09099999999999998)

**************************************************

Accuracy of the model is:-----> 0.20143548309117432
```

GridSearchCV – RandomForestRegressor as an Estimator



Gridsearchcv by using Random Forest Regressor as a estimator

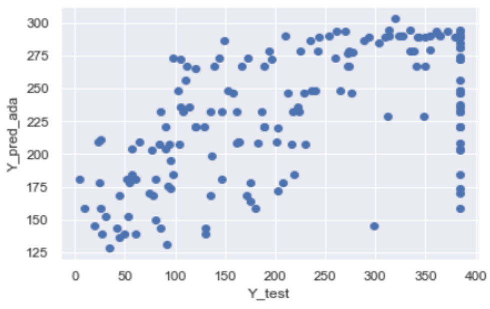
```
params = {'n_estimators':[10,20,30],
          'max_features':['auto',"sqrt","log2"],
          'min_samples_split':[10,11]}
gridsearch = GridSearchCV(estimator=rfo, param_grid= params)#apply GridSearchCV
time.sleep(2)
print(gridsearch.fit(x_train,y_train)) #train the model
time.sleep(1)
print("\n",'*'*50)
time.sleep(1)
print('GridSearchCV best parameters:----->',gridsearch.best_params_) #get best parameters
time.sleep(1)
print("\n",'*'*50)
time.sleep(1)
#put best params
rfc = gridsearch.best_estimator_
print('Trained Model using best parameters:----->',rfc.fit(x_train,y_train)) #train the model
time.sleep(1)
print("\n",'*'*50)
time.sleep(1)
print('Accuracy score:------>',rfc.score(x_train,y_train))#check the accuracy score
```

```
GridSearchCV(estimator=RandomForestRegressor(),
             param_grid={'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_split': [10, 11],
                         'n_estimators': [10, 20, 30]})

 **************************************************
GridSearchCV best parameters:-----> {'max_features': 'sqrt', 'min_samples_split': 11, 'n_estimators': 20}

 **************************************************
Trained Model using best parameters:-----> RandomForestRegressor(max_features='sqrt', min_samples_split=11,
                      n_estimators=20)

 **************************************************
Accuracy score:------> 0.7568975786697465
```

AdaBoostRegressor:

```python
ada = AdaBoostRegressor()
time.sleep(2)
print('Adaboost Regressor:-------->',ada.fit(x_train,y_train))
print()
time.sleep(1)
print('*'*50)
time.sleep(1)
print('Adaboost Regressor Training Score:-------->',ada.score(x_train,y_train))
print()
time.sleep(1)
print('*'*50)
time.sleep(1)
print('Adaboost Regressor Testing Score:-------->',ada.score(x_test,y_test))
time.sleep(2)
y_pred_ada = ada.predict(x_test)

time.sleep(2)
print("\n",'*'*50)

print("Ada Boost CV Score:")
print(cross_val_score(ada,x_scaled,y,cv=5))

time.sleep(2)
print("\n",'*'*50)

print("Ada Boost CV Score:")
print(cross_val_score(ada,x_scaled,y,cv=5).mean())

time.sleep(2)
print("\n",'*'*50)
print("Ada Booster:")
plt.scatter(y_test,y_pred_ada)
plt.xlabel('Y_test')
plt.ylabel('Y_pred_ada')
plt.show()
```

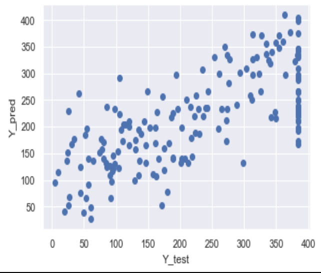
```
Adaboost Regressor:--------> AdaBoostRegressor()

**************************************************
Adaboost Regressor Training Score:--------> 0.5698564045920229

**************************************************
Adaboost Regressor Testing Score:--------> 0.3220811507881942

 **************************************************
Ada Boost CV Score:
[0.3605392  0.29636807 0.2974772  0.40193098 0.33081137]

 **************************************************
Ada Boost CV Score:
0.3281360970210375

 **************************************************
Ada Booster:
```



XGBoostRegressor:

```python
xgbr = xgb.XGBRegressor()
time.sleep(1)
print("XGBoost Training score:------>",xgbr.fit(x_train,y_train))
time.sleep(1)
print('*'*50)
time.sleep(1)
y_pred = xgbr.predict(x_test)
print("XGBoost Testing Score:------->", r2_score(y_test, y_pred))

time.sleep(2)
print("\n",'*'*50)

print("XGBoost CV Score:")
print(cross_val_score(xgbr,x_scaled,y,cv=5))

time.sleep(2)
print("\n",'*'*50)

print("XGBoost CV Score:")
print(cross_val_score(xgbr,x_scaled,y,cv=5).mean())

time.sleep(2)
print("\n",'*'*50)
print("XGBooster:")
plt.scatter(y_test,y_pred)
plt.xlabel('Y_test')
plt.ylabel('Y_pred')
plt.show()
```

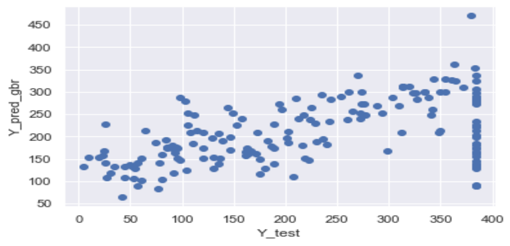
```
XGBoost Training score:------> XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.300000012,
             max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=8,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
**************************************************
XGBoost Testing Score:-------> 0.5051066443034442

 **************************************************
XGBoost CV Score:
[0.58330763 0.39464063 0.46033469 0.60613183 0.47412683]

 **************************************************
XGBoost CV Score:
0.5037083216046441

 **************************************************
XGBooster:
```



GradientBoostingRegressor:

```
gbr = GradientBoostingRegressor(max_depth=3, n_estimators=6, learning_rate=.4)
time.sleep(1)
print("GradientBoostingRegressor Training Score:----->", gbr.fit(x_train,y_train))
time.sleep(1)
print("*"*50)
time.sleep(1)
y_pred_gbr = gbr.predict(x_test)
print("GradientBoostingRegressor Testing score:------>", r2_score(y_test,y_pred_gbr))
time.sleep(2)
print("\n",'*'*50)

print("GBR CV Score:")
print(cross_val_score(gbr,x_scaled,y,cv=5))

time.sleep(2)
print("\n",'*'*50)

print("GBR CV Score:")
print(cross_val_score(gbr,x_scaled,y,cv=5).mean())

time.sleep(2)
print("\n",'*'*50)
print("GBR:")
plt.scatter(y_test,y_pred_gbr)
plt.xlabel('Y_test')
plt.ylabel('Y_pred_gbr')
plt.show()
```

```
GradientBoostingRegressor Training Score:-----> GradientBoostingRegressor(learning_rate=0.4, n_estimators=6)
**************************************************
GradientBoostingRegressor Testing score:------> 0.23891969389791579

 **************************************************
GBR CV Score:
[0.49807001 0.36411379 0.38212903 0.54282093 0.35631056]

 **************************************************
GBR CV Score:
0.432543864995196

 **************************************************
GBR:
```



• Key Metrics for success in solving problem under consideration:

Will go with Random Forest Regressor Reasons

1. Random Forest reduces overfitting in decision tree and helps to improve accuracy.

2. It is flexible for both classification and regression tasks.

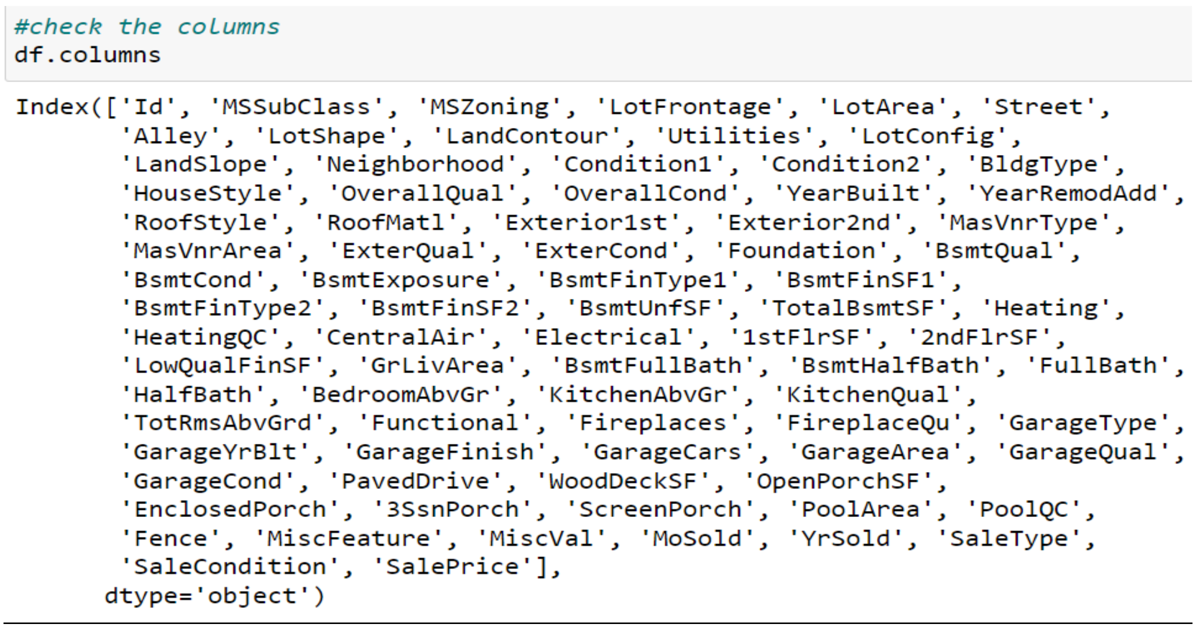
3. It also works well with both categorical and continuous values.

4. It is a rule-based approach.

5. It automates missing values present in the data The key metrics used here were R2_Score, MSE, MAE, RMSE and Cross Validation Score. We also tried to find out the best parameters by using Hyper Parameter Tuning and Random Grid Search to increase the accuracy score.

- ## <u>Visualizations:</u>

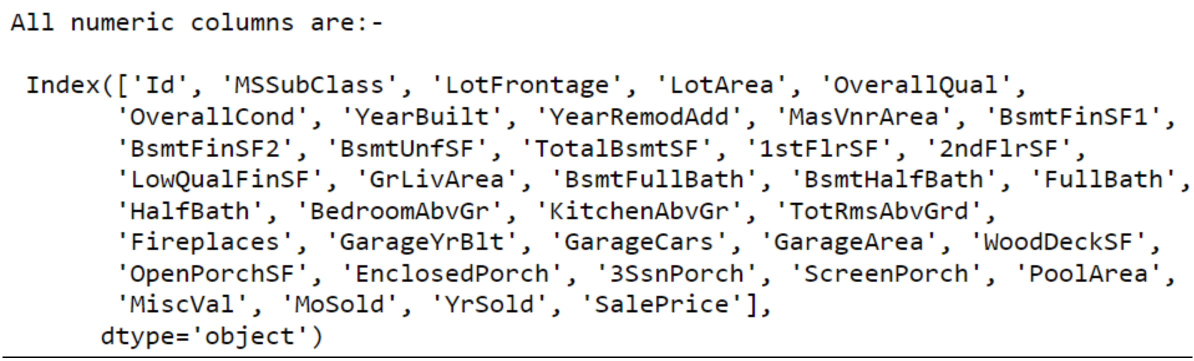
<u>All Columns:</u>



```
#check the columns
df.columns
```

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

<u>All Numerical columns:</u>



```
All numeric columns are:-

 Index(['Id', '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'],
      dtype='object')
```

<u>All Categorical Columns:</u>

```
['Street', 'LotConfig', 'HouseStyle', 'GarageCond', 'GarageType', 'Condition2', 'Fence', 'Alley', 'LotShape', 'LandContour', 'BsmtFinType2', 'Utilities', 'Functional', 'BsmtQual', 'MSZoning', 'ExterQual', 'FireplaceQu', 'MiscFeature', 'BsmtCond', 'CentralAir', 'MasVnrType', 'BldgType', 'Electrical', 'BsmtExposure', 'HeatingQC', 'GarageQual', 'RoofMatl', 'Exterior2nd', 'LandSlope', 'GarageFinish', 'Neighborhood', 'PoolQC', 'BsmtFinType1', 'ExterCond', 'SaleType', 'Exterior1st', 'PavedDrive', 'Heating', 'Condition1', 'SaleCondition', 'Foundation', 'KitchenQual', 'RoofStyle']
```

```python
#check the duplicate values
df.duplicated().sum()
```

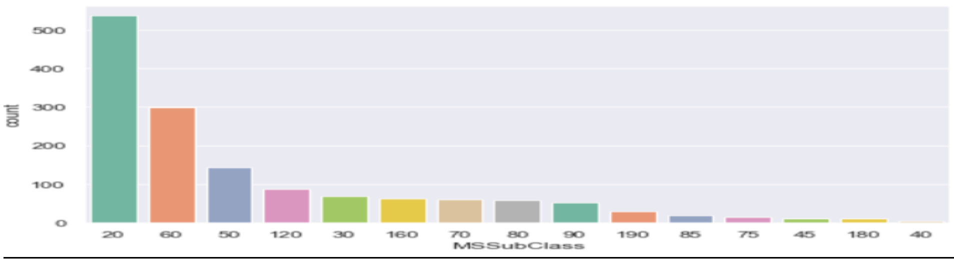
```
0
```

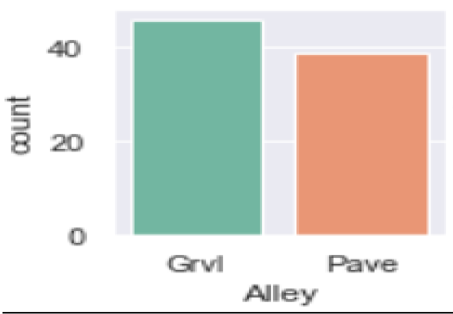
```python
#check the zero values
df.all().tail(20)
```

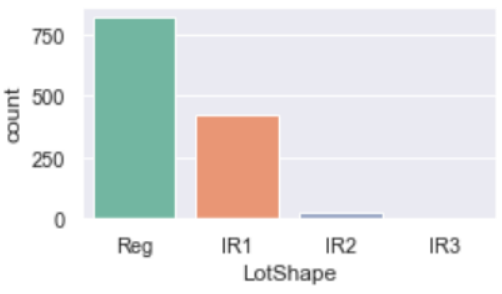
```
GarageCars       False
GarageArea       False
GarageQual        True
GarageCond        True
PavedDrive        True
WoodDeckSF       False
OpenPorchSF      False
EnclosedPorch    False
3SsnPorch        False
ScreenPorch      False
PoolArea         False
PoolQC            True
Fence             True
MiscFeature       True
MiscVal          False
MoSold            True
YrSold            True
SaleType          True
SaleCondition     True
SalePrice         True
dtype: bool
```

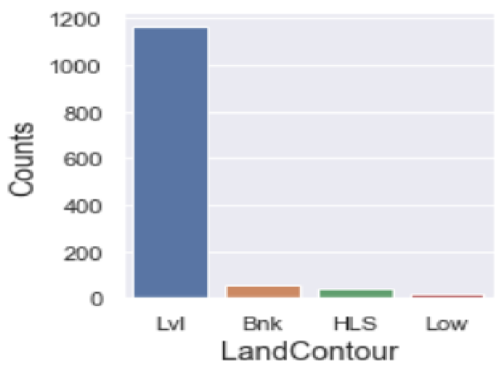
```python
#check the skewness
df.skew()
```

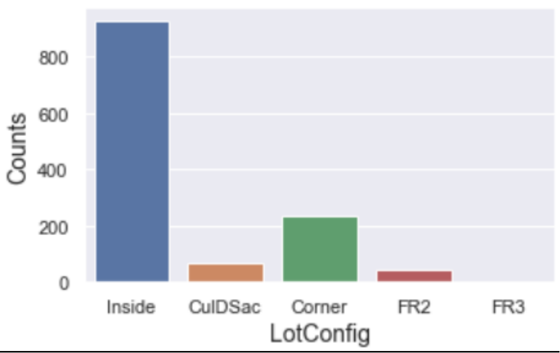
```
Id                0.000000
MSSubClass        1.407657
LotFrontage       2.163569
LotArea          12.207688
OverallQual       0.216944
OverallCond       0.693067
YearBuilt        -0.613461
YearRemodAdd     -0.503562
MasVnrArea        2.669084
BsmtFinSF1        1.685503
BsmtFinSF2        4.255261
BsmtUnfSF         0.920268
TotalBsmtSF       1.524255
1stFlrSF          1.376757
2ndFlrSF          0.813030
LowQualFinSF      9.011341
GrLivArea         1.366560
BsmtFullBath      0.596067
BsmtHalfBath      4.103403
FullBath          0.036562
HalfBath          0.675897
BedroomAbvGr      0.211790
KitchenAbvGr      4.488397
TotRmsAbvGrd      0.676341
Fireplaces        0.649565
GarageYrBlt      -0.649415
GarageCars       -0.342549
GarageArea        0.179981
WoodDeckSF        1.541376
OpenPorchSF       2.364342
EnclosedPorch     3.089872
3SsnPorch        10.304342
ScreenPorch       4.122214
PoolArea         14.828374
MiscVal          24.476794
MoSold            0.212053
YrSold            0.096269
SalePrice         1.953878
dtype: float64
```

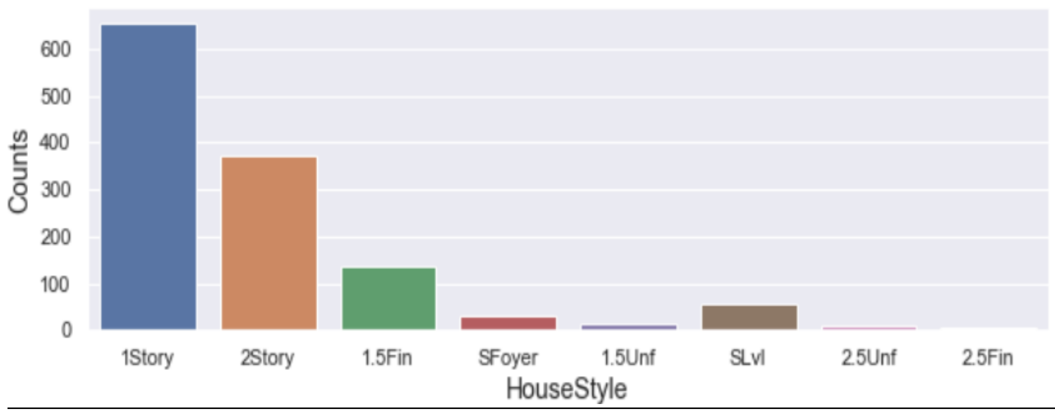


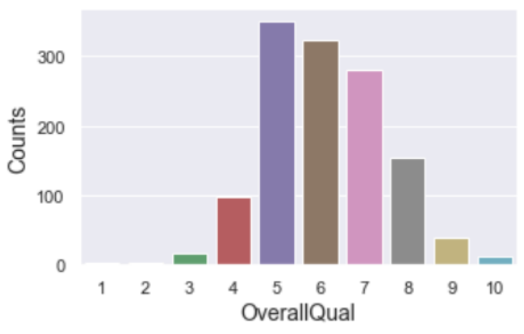

count
40
20
0
Grvl
Pave
Alley

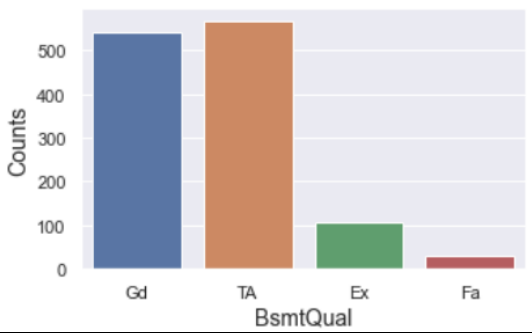

count
750
500
250
0
Reg
IR1
IR2
IR3
LotShape

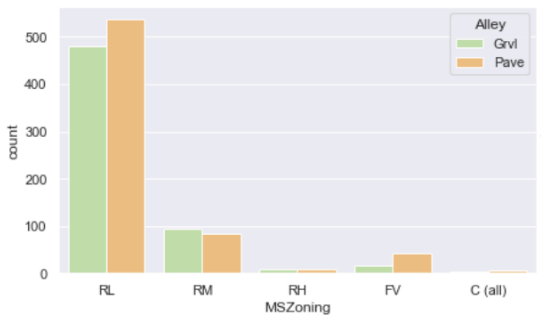

Counts
1200
1000
800
600
400
200
0
Lvl
Bnk
HLS
Low
LandContour

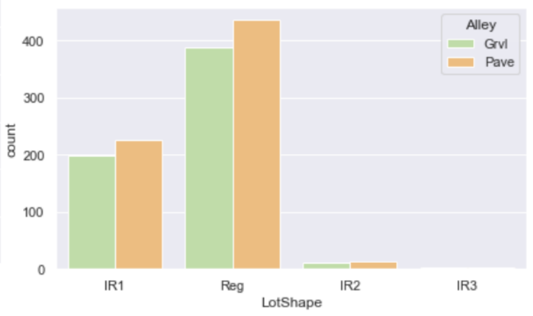

Counts
800
600
400
200
0
Inside
CulDSac
Corner
FR2
FR3
LotConfig

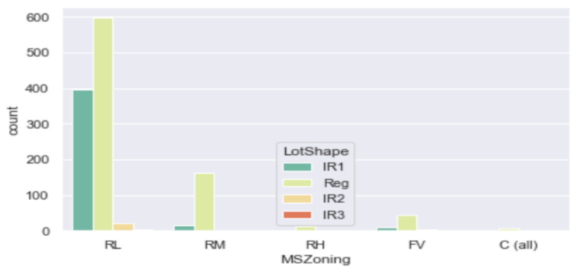

Counts
600
500
400
300
200
100
0
1Story
2Story
1.5Fin
SFoyer
1.5Unf
SLvl
2.5Unf
2.5Fin
HouseStyle

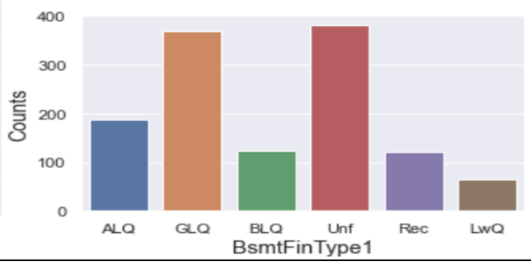

Counts
300
200
100
0
1
2
3
4
5
6
7
8
9
10
OverallQual

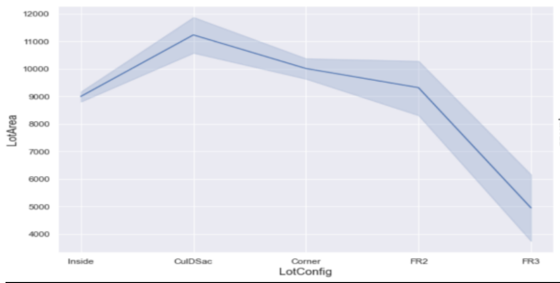

Counts
500
400
300
200
100
0
Gd
TA
Ex
Fa
BsmtQual

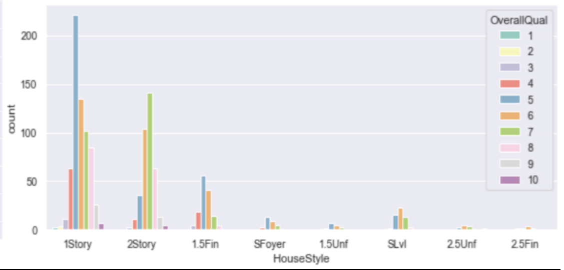

Alley
Grvl
Pave
count
0
100
200
300
400
500
RL
RM
RH
FV
C (all)
MSZoning

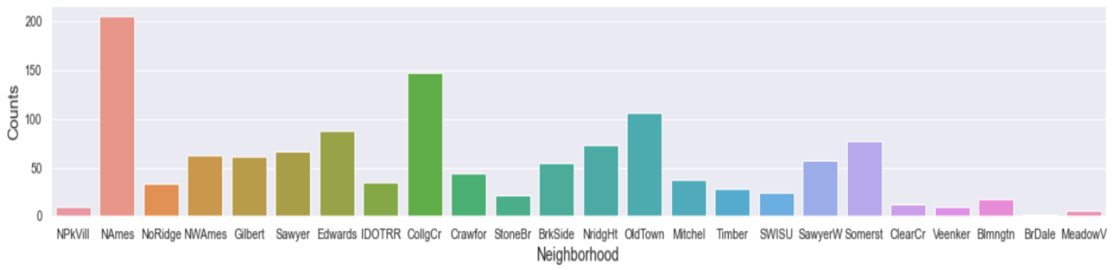

Alley
Grvl
Pave
count
0
100
200
300
400
IR1
Reg
IR2
IR3
LotShape

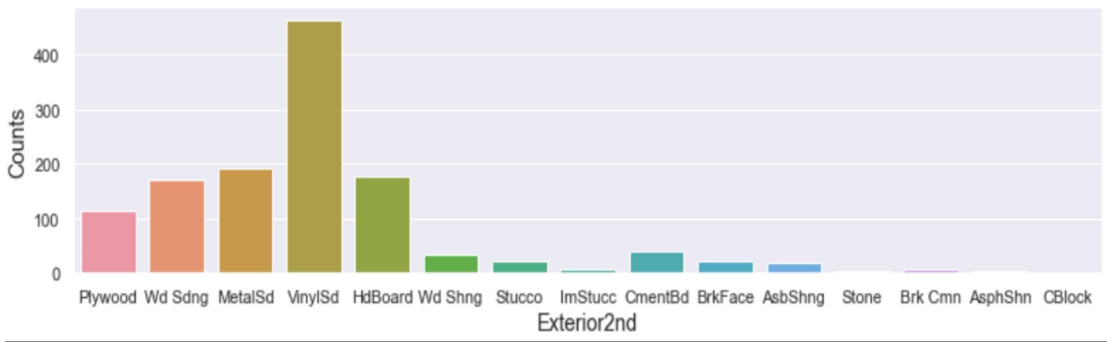

LotShape
IR1
Reg
IR2
IR3
count
0
100
200
300
400
500
600
RL
RM
RH
FV
C (all)
MSZoning

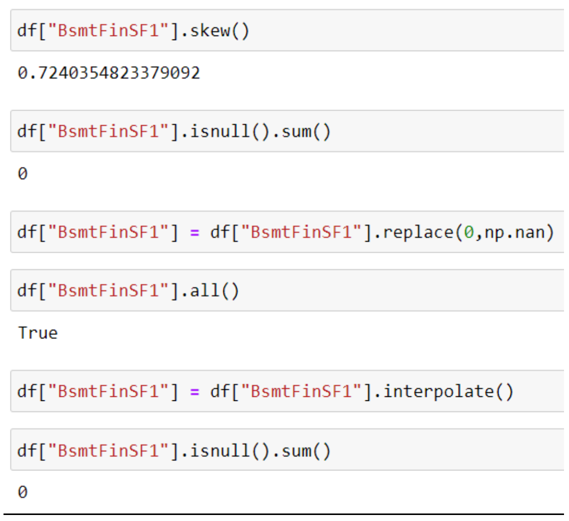

Counts
0
100
200
300
400
ALQ
GLQ
BLQ
Unf
Rec
LwQ
BsmtFinType1


LotArea
4000
5000
6000
7000
8000
9000
10000
11000
12000
Inside
CulDSac
Corner
FR2
FR3
LotConfig


OverallQual
1
2
3
4
5
6
7
8
9
10
count
0
50
100
150
200
1Story
2Story
1.5Fin
SFoyer
1.5Unf
SLvl
2.5Unf
2.5Fin
HouseStyle


Counts
0
50
100
150
200
NPkVill
NAmes
NoRidge
NWAmes
Gilbert
Sawyer
Edwards
IDOTRR
CollgCr
Crawfor
StoneBr
BrkSide
NridgHt
OldTown
Mitchel
Timber
SWISU
SawyerW
Somerst
ClearCr
Veenker
Blmngtn
BrDale
MeadowV
Neighborhood


Counts
0
100
200
300
400
Plywood
Wd Sdng
MetalSd
VinylSd
HdBoard
Wd Shng
Stucco
ImStucc
CmentBd
BrkFace
AsbShng
Stone
Brk Cmn
AsphShn
CBlock
Exterior2nd



```
df["BsmtFinSF1"].skew()
```

```
0.7240354823379092
```

```
df["BsmtFinSF1"].isnull().sum()
```

```
0
```

```
df["BsmtFinSF1"] = df["BsmtFinSF1"].replace(0,np.nan)
```

```
df["BsmtFinSF1"].all()
```

```
True
```

```
df["BsmtFinSF1"] = df["BsmtFinSF1"].interpolate()
```

```
df["BsmtFinSF1"].isnull().sum()
```

```
0
```

INTERPRETATION

The purpose of this article was twofold: to understand the pattern of Australian real estate market and make predictive model, which is able to effectively predict the price of houses in Australia.
We use many algorithms but We see, RandomForestRegressor is giving us better accuracy than others. We use MAE, MSE and RSME also to check the error values. And this algorithm is giving us better error results than other algorithms.
We use GridSearchCV where RandomForestRegressor as an estimator and it is giving us approximately 75% of the accuracy.

CONCLUSION

- **Key Findings and Conclusions of the Study**

1. 20- 1-STORY 1946 & NEWER ALL STYLES has more counts than other, 40 1-STORY W/FINISHED ATTIC ALL AGES has less number of counts.
2. In commercial, 40% is 60 linear feet of street connected to property to residential low density
3. Approximately 9% of 60 feet is connected to RL
4. we see, the variance of 40-1-STORY W/FINISHED ATTIC ALL AGES is high.

5. Lot Area of 160-2-STORY PUD - 1946 & NEWER is very less.
6. Lot area of low residential density lies between 8000-10000
7. Approximately, the range of 50 - 100 linear feet of street connected to property related to 6000 - 10000 lot area.
8. There is no Agriculture, Industrial and Residential Low density park
9. Mostly houses are residential low density, approximately 78%
10. Mostly Low Residential areas are 1-STORY 1946 & NEWER ALL STYLES.
11. In Commercial, there are no 1-1/2 STORY PUD - ALL AGES, 2-STORY PUD - 1946 & NEWER, PUD - MULTILEVEL - INCL SPLIT LEV/FOYER
12. Same for Floating Village Residential, they are also less.
13. By observing this count plot, we can see, CVount of RL is higher in both the cases.
14. more than 60% property shapes are regular.
15. LotArea of Regular shape property is less than others.
16. But LotArea of IR3 is higher than others.
17. Regular properties are more in RL
18. Inside lot is more in count than others
19. Inside lot is more in low residential area and regular properties
20. Lotarea of FR3 is very low and CulDSac is high than others
21. Mostly, flat level is in low residential area.

- <u>**Learning Outcomes of the Study in respect of Data Science**</u>

This study helps everyone to predict the real or actual price of any house. This study also helps anyone to know the specifications regarding house and factors that depends on the price of the house. Machine Learning or Data Science is a very important field to measure or analyse these types of studies.

- **<u>Limitations of this work and Scope for Future Work</u>**:

The main issue in this study, that the price of houses is variable and we found many null values and zero values in our dataset. But if we got a dataset which has no null values and no zero values then we can analyse the price of houses very accurately but still the price of the houses are variable. We need to update our data time to time.