FLIP ROBO

NAME OF THE PROJECT

Housing Price Prediction

Submitted by:Lahari S.K

ACKNOWLEDGMENT

I would like to thank everyone who helped through this project directly and indirectly.

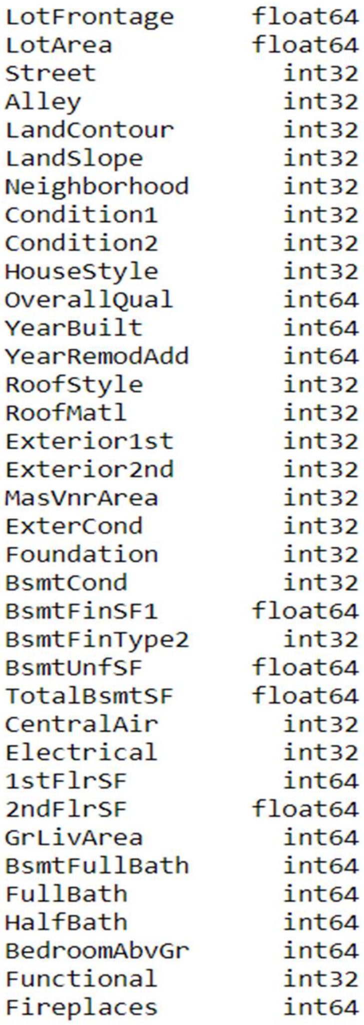
INTRODUCTION

Building the model to predict the price of houses with the available independent variables. The model will then be used by the management to understand how exactly the prices vary with the variables. Then one can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns.

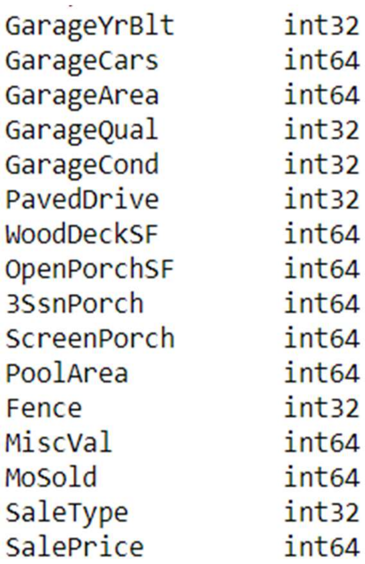
Houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy. It is a very large market and there are various companies working in the domain. Data science comes as a very important tool to solve problems in the domain to help the companies increase their overall revenue, profits, improving their marketing strategies and focusing on changing trends in house sales and purchases

Analytical Problem Framing

There are two data sets provided the train and the test data. Test data has 1168rows and 81 columns whereas the test data has 292 rows and 80 columns. The data is in csv format.



```
LotFrontage     float64
LotArea         float64
Street            int32
Alley             int32
LandContour       int32
LandSlope         int32
Neighborhood      int32
Condition1        int32
Condition2        int32
HouseStyle        int32
OverallQual       int64
YearBuilt         int64
YearRemodAdd      int64
RoofStyle         int32
RoofMatl          int32
Exterior1st       int32
Exterior2nd       int32
MasVnrArea        int32
ExterCond         int32
Foundation        int32
BsmtCond          int32
BsmtFinSF1      float64
BsmtFinType2      int32
BsmtUnfSF       float64
TotalBsmtSF     float64
CentralAir        int32
Electrical        int32
1stFlrSF          int64
2ndFlrSF        float64
GrLivArea         int64
BsmtFullBath      int64
FullBath          int64
HalfBath          int64
BedroomAbvGr      int64
Functional        int32
Fireplaces        int64
```



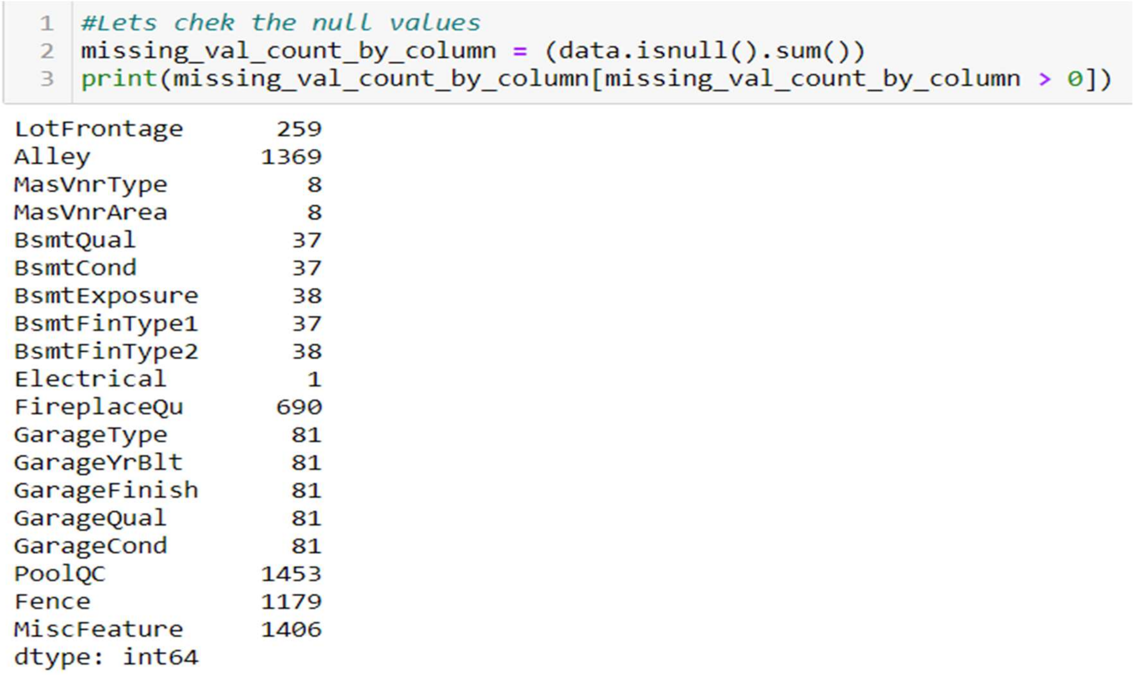
```
GarageYrBlt       int32
GarageCars        int64
GarageArea        int64
GarageQual        int32
GarageCond        int32
PavedDrive        int32
WoodDeckSF        int64
OpenPorchSF       int64
3SsnPorch         int64
ScreenPorch       int64
PoolArea          int64
Fence             int32
MiscVal           int64
MoSold            int64
SaleType          int32
SalePrice         int64
```

Data Pre-processing Done

The pre-processing steps taken under consideration before building the model are as follows

1.Handling missing values:

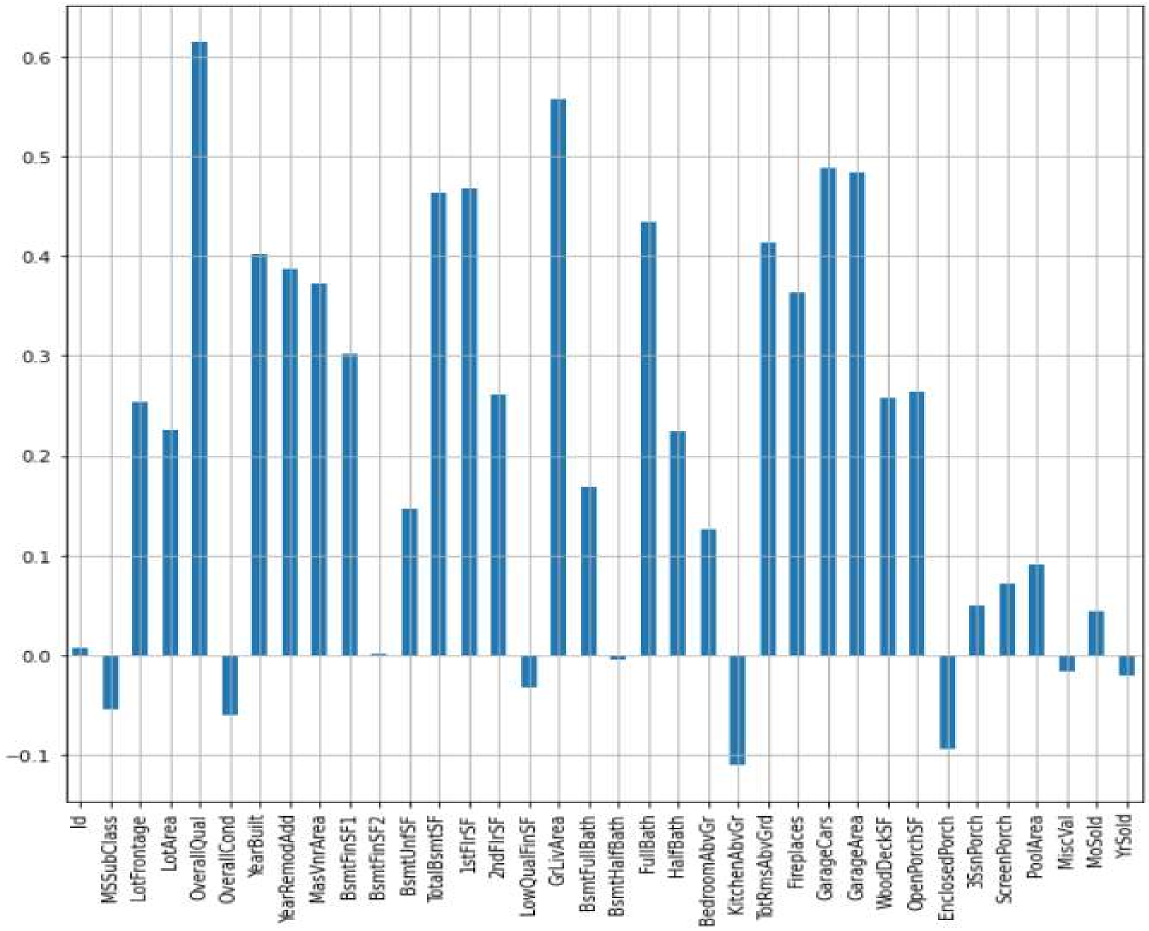
Below snapshot is the count of the missing values. Missing values are handled by using fillna. Categorical missing values are handled by filling "Not Available" whereas the numerical missing values is handled by using mean. Alley,FirePlaceQu,PoolQC,Fence and MiscFeature are dropped because they have many missing values.



```
#Lets chek the null values
missing_val_count_by_column = (data.isnull().sum())
print(missing_val_count_by_column[missing_val_count_by_column > 0])
```

```
LotFrontage      259
Alley           1369
MasVnrType         8
MasVnrArea         8
BsmtQual          37
BsmtCond          37
BsmtExposure      38
BsmtFinType1      37
BsmtFinType2      38
Electrical         1
FireplaceQu      690
GarageType        81
GarageYrBlt       81
GarageFinish      81
GarageQual        81
GarageCond        81
PoolQC          1453
Fence           1179
MiscFeature     1406
dtype: int64
```

2.Dropping negatively correlated variables



Negatively correlated variables are dropped.

3.Outliers removal

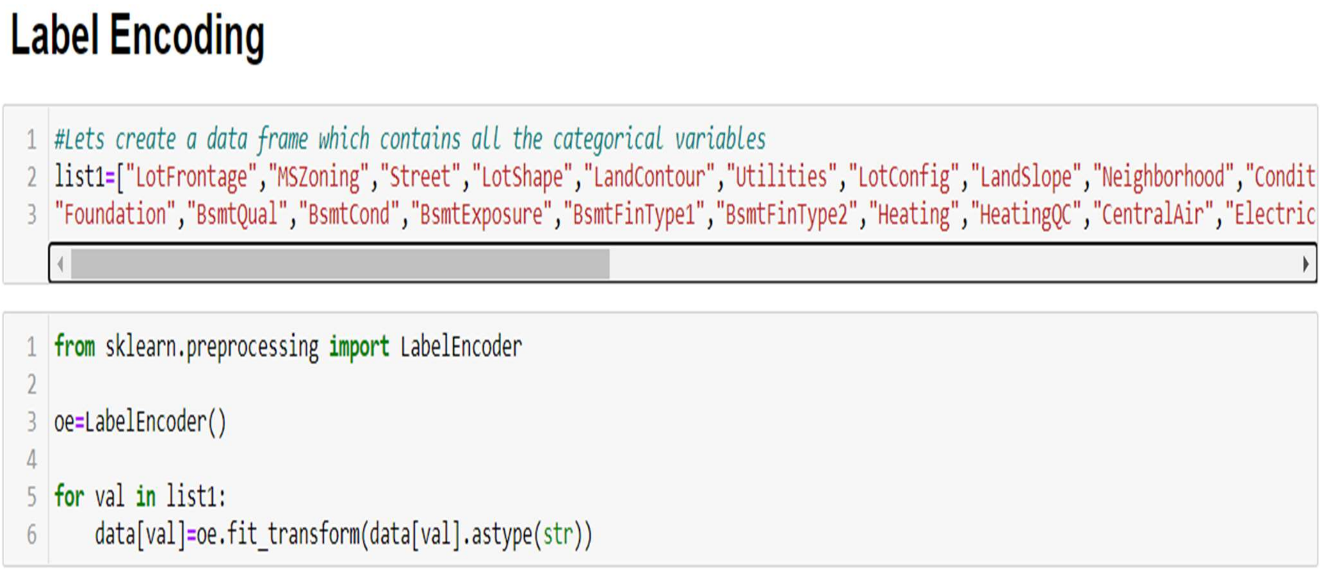
The presence of outliers is big disadvantage in obtaining good results because some algorithms are very much sensitive to the outliers. Modelling along with the outliers present would result in very bad results. Hence outliers had to be removed

Outliers are removed using quantiles.0.1 and 0.9 quantiles are calculated ,values lesser than 0.1 and greater than 0.9 data points are removed respectively.

4.Label Encoding

Label Encoding is used to convert the categorical features into the numeric ones. One of the alternative for the label encoding is get dummies. The decision for choosing label encoder in place of get dummies is that when get dummies is implemented there will be increase in the dimensions. The size is already bit large. It might consume much larger space. Due to space constraint Label

encoder was apt.



Label Encoding

```
#Lets create a data frame which contains all the categorical variables
list1=["LotFrontage","MSZoning","Street","LotShape","LandContour","Utilities","LotConfig","LandSlope","Neighborhood","Condit
"Foundation","BsmtQual","BsmtCond","BsmtExposure","BsmtFinType1","BsmtFinType2","Heating","HeatingQC","CentralAir","Electric
```

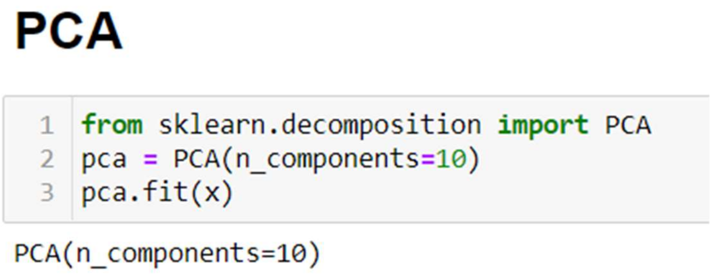
```
from sklearn.preprocessing import LabelEncoder

oe=LabelEncoder()

for val in list1:
    data[val]=oe.fit_transform(data[val].astype(str))
```

5.PCA

The most important use of PCA is to represent a multivariate data table as smaller set of variables (summary indices) in order to observe trends, jump clusters and outliers. This overview may uncover the relationships between observations and variables, and among the variables.



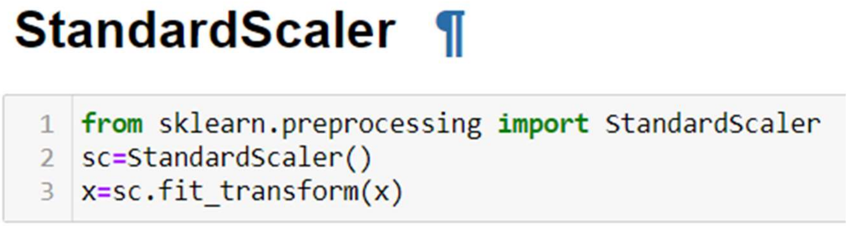
PCA

```python
from sklearn.decomposition import PCA
pca = PCA(n_components=10)
pca.fit(x)
```

```
PCA(n_components=10)
```

6. StandardScaler

Standard scaler is used to scale all values in the same range so that the the graph becomes normalized, mean somewhere lies **nearly 0.**



StandardScaler ¶

```python
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x=sc.fit_transform(x)
```

Software Requirements

1.JUPYTER NOTEBOOK: The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning.

Libraries used in jupyter notebook

1.pandas: Here pandas is used to read the csv file.

2.numpy: It is used to compute the zscores.

3.matplotlib.pyplot: Creates a figure, creates a plotting area in a figure,

plots some lines in a plotting area, decorates the plot with labels.

4. seaborn: Python data visualization library based on matplotlib. It

provides a high-level interface for drawing attractive and informative statistical graphics.
Ex: heatmaps,pie charts,scatterplot,pairplot,countplot,violinplot and Distort

5. sklearn: Provides supervised algorithms and pre-processing.

a.sklearn.preprocessing import LabelEncoder:
Label encoder is used to encode categorical values into numerical values.

b. sklearn.preprocessing import StandardScaler:

Standard scaler is used to scale all values in the same range so that the the graph becomes normalized, mean somewhere lies nearly 0.

c. from sklearn.decomposition import PCA :
Principal Component Analysis (PCA) is an unsupervised, non-parametric statistical technique primarily used for dimensionality reduction in machine learning.

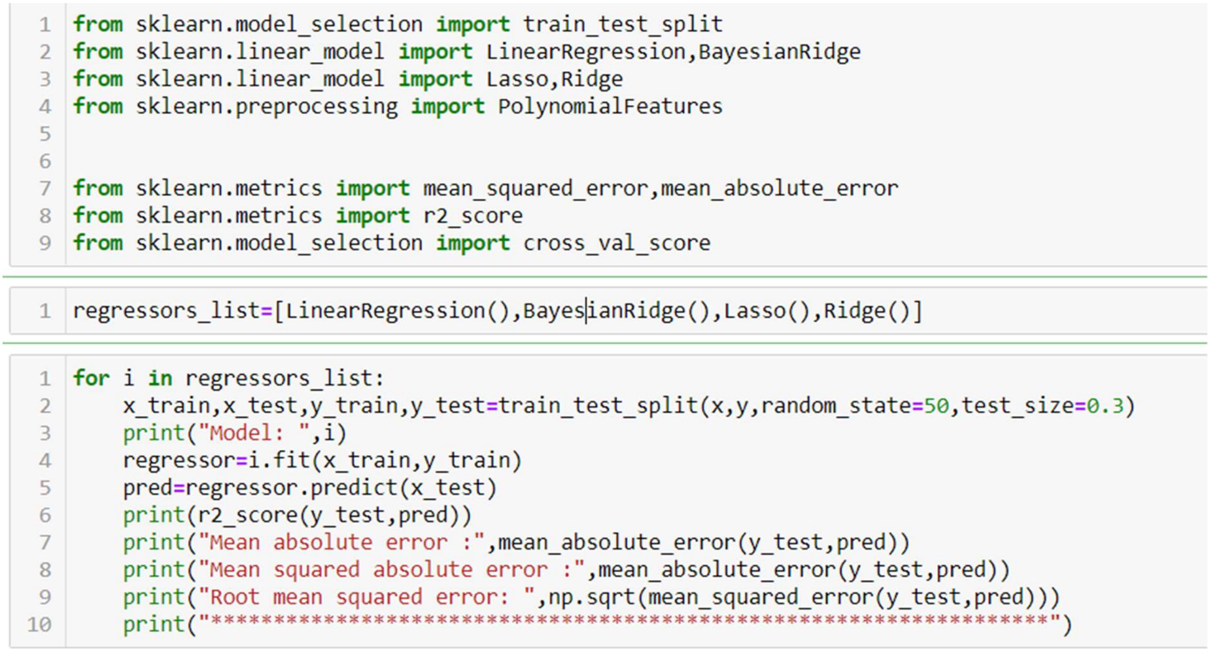
Model/s Development and Evaluation

- ## Testing of Identified Approaches (Algorithms)

  The list of algorithms used are follows

1. LinearRegression
2. BayesianRidge
3. Lasso
4. Ridge
5. RandomForestRegressor
6. ExtraTreesRegressor
7. DecisionTreeRegressor

Code snippet:



```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression,BayesianRidge
from sklearn.linear_model import Lasso,Ridge
from sklearn.preprocessing import PolynomialFeatures


from sklearn.metrics import mean_squared_error,mean_absolute_error
from sklearn.metrics import r2_score
from sklearn.model_selection import cross_val_score
```

```
regressors_list=[LinearRegression(),BayesianRidge(),Lasso(),Ridge()]
```

```
for i in regressors_list:
    x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=50,test_size=0.3)
    print("Model: ",i)
    regressor=i.fit(x_train,y_train)
    pred=regressor.predict(x_test)
    print(r2_score(y_test,pred))
    print("Mean absolute error :",mean_absolute_error(y_test,pred))
    print("Mean squared absolute error :",mean_absolute_error(y_test,pred))
    print("Root mean squared error: ",np.sqrt(mean_squared_error(y_test,pred)))
    print("*********************************************************************")
```

1.Linear Regression:

Linear regression is a linear model, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).

2.Bayesian Ridge:

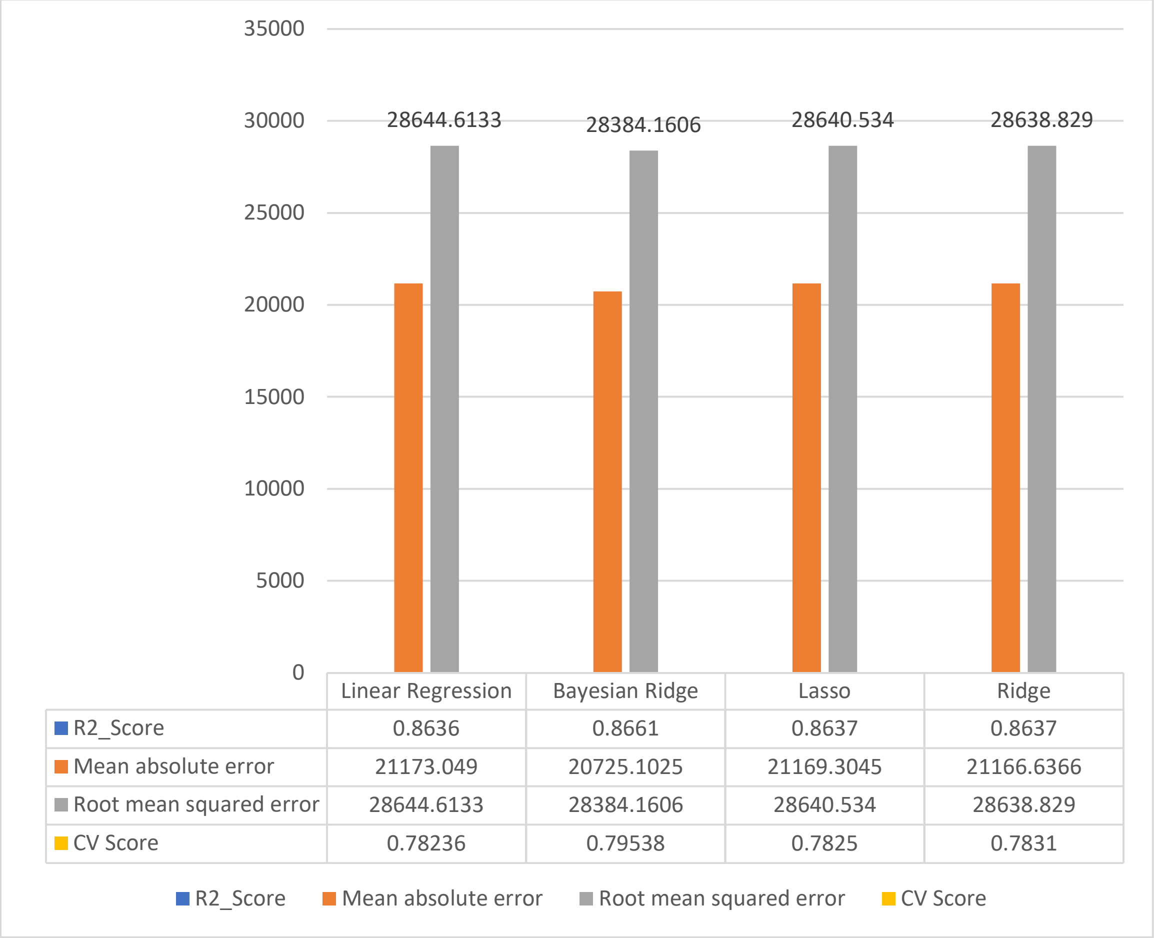
Bayesian linear regression is an approach to linear regression in which the statistical analysis is undertaken within the context of Bayesian inference. When the regression model has errors that have a normal distribution, and if a particular form of prior distribution is assumed, explicit results are available for the posterior probability distributions of the model's parameters.

3.Lasso:

Lasso regression is a type of linear regression that uses shrinkage. Shrinkage is where data values are shrunk towards a central point, like the mean.

4.Ridge:

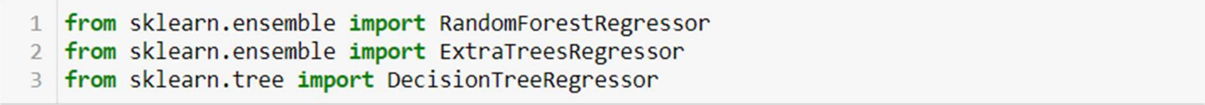
Ridge Regression is a technique for analyzing multiple regression data that suffer from multicollinearity. When multicollinearity occurs, least squares estimates are unbiased, but their variances are large so they may be far from the true value.



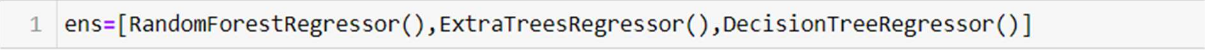
	Linear Regression	Bayesian Ridge	Lasso	Ridge
🟦 R2_Score	0.8636	0.8661	0.8637	0.8637
🟧 Mean absolute error	21173.049	20725.1025	21169.3045	21166.6366
⬜ Root mean squared error	28644.6133	28384.1606	28640.534	28638.829
🟨 CV Score	0.78236	0.79538	0.7825	0.7831

Ensemble Techniques

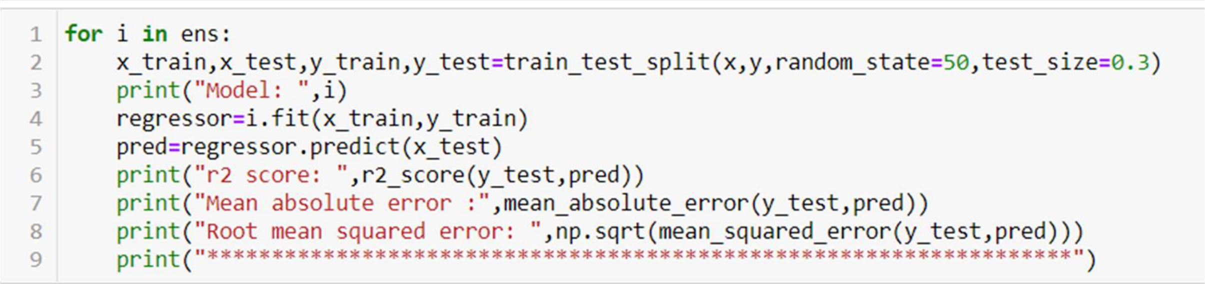
Code Snippet:



```
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.tree import DecisionTreeRegressor
```



```
ens=[RandomForestRegressor(),ExtraTreesRegressor(),DecisionTreeRegressor()]
```



```
for i in ens:
    x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=50,test_size=0.3)
    print("Model: ",i)
    regressor=i.fit(x_train,y_train)
    pred=regressor.predict(x_test)
    print("r2 score: ",r2_score(y_test,pred))
    print("Mean absolute error :",mean_absolute_error(y_test,pred))
    print("Root mean squared error: ",np.sqrt(mean_squared_error(y_test,pred)))
    print("*****************************************************************")
```

1.RandomForest Regressor:

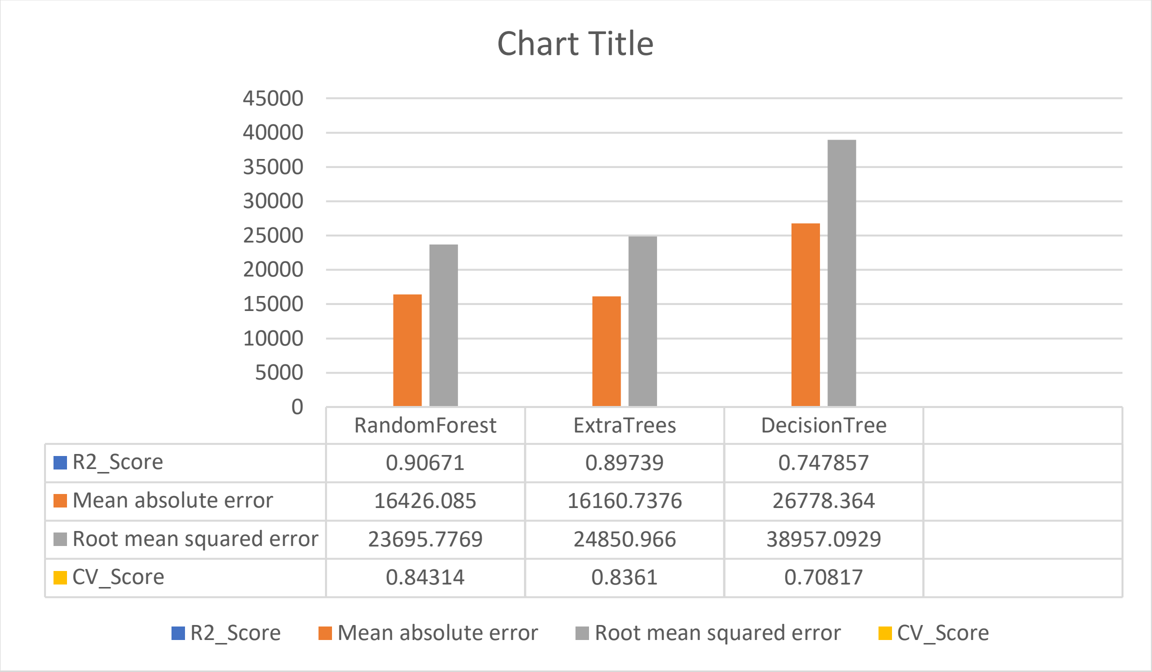
Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. ... A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees.

2.ExtraTreesRegressor:

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. ... The maximum depth of the tree.

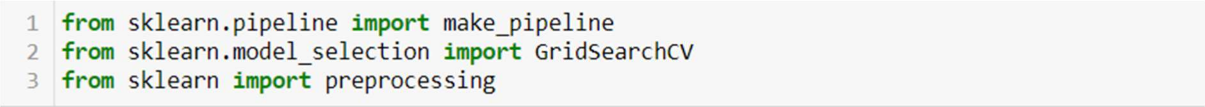
3.DecisionTreeRegressor:

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

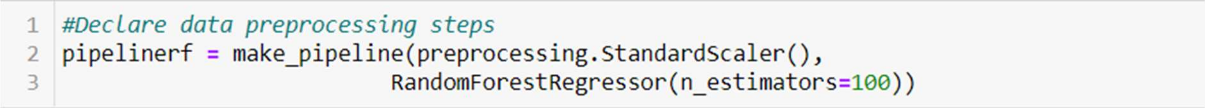


	RandomForest	ExtraTrees	DecisionTree	
🟦 R2_Score	0.90671	0.89739	0.747857	
🟧 Mean absolute error	16426.085	16160.7376	26778.364	
⬜ Root mean squared error	23695.7769	24850.966	38957.0929	
🟨 CV_Score	0.84314	0.8361	0.70817	

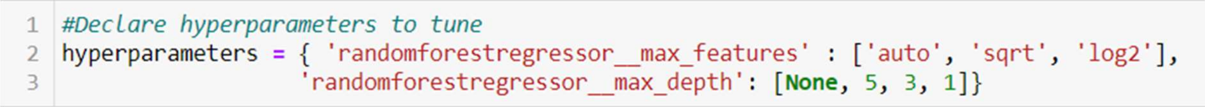
Key Metrics for success in solving problem under consideration



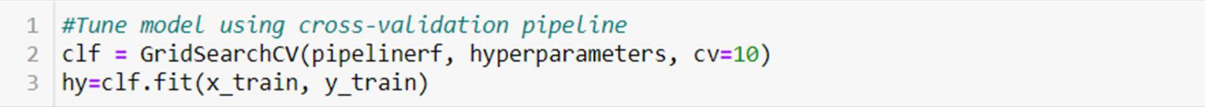
```python
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn import preprocessing
```



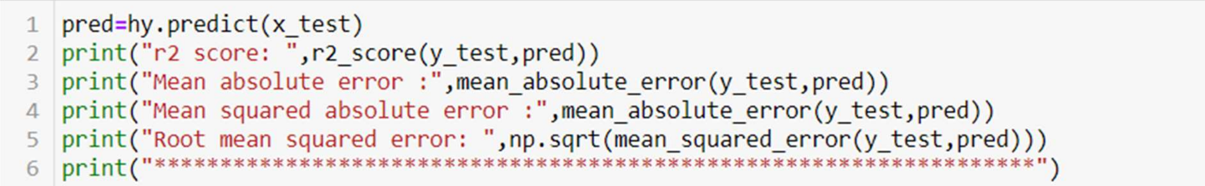
```python
#Declare data preprocessing steps
pipelinerf = make_pipeline(preprocessing.StandardScaler(),
                          RandomForestRegressor(n_estimators=100))
```



```python
#Declare hyperparameters to tune
hyperparameters = { 'randomforestregressor__max_features' : ['auto', 'sqrt', 'log2'],
                  'randomforestregressor__max_depth': [None, 5, 3, 1]}
```



```python
#Tune model using cross-validation pipeline
clf = GridSearchCV(pipelinerf, hyperparameters, cv=10)
hy=clf.fit(x_train, y_train)
```



```python
pred=hy.predict(x_test)
print("r2 score: ",r2_score(y_test,pred))
print("Mean absolute error :",mean_absolute_error(y_test,pred))
print("Mean squared absolute error :",mean_absolute_error(y_test,pred))
print("Root mean squared error: ",np.sqrt(mean_squared_error(y_test,pred)))
print("*******************************************************************")
```

1.R2_Score:

R-squared is a goodness-of-fit measure for linear regression models. This statistic indicates the percentage of the variance in the dependent variable that the independent variables explain collectively. R-squared measures the strength of the relationship between your model and the dependent variable on a convenient 0 – 100% scale.

2. Mean absolute error:

Mean Absolute Error, also known as MAE, is one of the many metrics for summarizing and assessing the quality of a machine learning model. ... Given any test data-set, Mean Absolute Error of your model refers to the mean of the absolute values of each prediction error on all instances of the test data-set.

3. Root mean squared error:

RMSE is calculated as the square root of the mean of the squared differences between actual outcomes and predictions.Squaring each error forces the values to be positive, and the square root of the mean squared error returns the error metric back to the original units for comparison.

R2 score is the desirable metric to be considered because rmse can be opted when we are comparing 2 models.R2 score tells us the dependency between dependent and independent variables.

CONCLUSION

Preference for choosing properties

Residential low density zone, Paved alley access, Regular lot size, near flat or level land contour, all public utilities, inside lot, gentle slope property, single family detached lots, 1 story houses, gable roofstyles,roofs built from Standard Composite shingle, Exterior covering on the house built using vinyl sliding, cider block foundation, gas forced warmair furnace for heating, central air conditioning, standard circuits breakers and romex, attached garages, paved driveway and minimum privacy fence.