NAME OF THE PROJECT

Housing Project

Submitted by:

Neha Kumari

ACKNOWLEDGMENT

i took help from my old projects,google and scikit-learn

INTRODUCTION

- Business Problem Framing

- Houses are one of the necessary need of each and every person around the globe and therefore housing and real estate market is one of the markets which is one of the major contributors in the world's economy. It is a very large market and there are various companies working in the domain.
  In this dataset , we have to predict house price help to determine the selling price of a house of a particular region and can help people to find the right time to buy a home.

- Conceptual Background of the Domain Problem

  Describe the domain related concepts that you think will be useful for better understanding of the project.

- Review of Literature

  This is a comprehensive summary of the research done on the topic. The review should enumerate, describe, summarize, evaluate and clarify the research done.

- Motivation for the Problem Undertaken

  Motivation behind to make this project we have to predict house prices by using machine learning model with python

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  Describe the mathematical, statistical and analytics modelling done during this project along with the proper justification.

  LinearRegressor

  Support victor Regressor

  AdaboostRegressor

  XgboostRegressor

  SGDRegressor

  RandomForestRegressor

  DecisionTreeRegressor

  KNeighborsRegressor

- Data Sources and their formats

  Data Sources – Primary data, official sources

  Formats – datasets in csv formate

```
In [412]: df_Train = pd.read_csv(r"C:\Users\HP\Desktop\Train.csv")
          df_Test = pd.read_csv(r"C:\MySQLite_DataBase\test.csv")

In [413]: df_Train
```

Out[413]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	PoolArea	PoolQC	Fence	MiscFeature	Mis
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	MnPrv	NaN	
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
5	1197	60	RL	58.0	14054	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
6	561	20	RL	NaN	11341	Pave	NaN	IR1	Lvl	AllPub	...	0	NaN	NaN	NaN	
7	1041	20	RL	88.0	13125	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	GdPrv	NaN	
8	503	20	RL	70.0	9170	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	GdPrv	Shed	
9	576	50	RL	80.0	8480	Pave	NaN	Reg	Lvl	AllPub	...	0	NaN	NaN	NaN	
10	449	50	RM	50.0	8600	Pave	NaN	Reg	Bnk	AllPub	...	0	NaN	NaN	NaN	

- Data Preprocessing Done

  Preprocessing process – firstly , checking null or missing values, since there are lots of null values , so we have fill all null values and

some cloumns we have drop it which is not important for further process and around 70% to 90% is null values in these columns.

label Encoder
We have apply **label Encoding technique** for handling categorical columns.

Spliting the data into training and testing formate

- Data Inputs- Logic- Output Relationships

  Functions are often described in terms of "input" and "output".

A function can be conceptualized as a black box. The input value x is put in the box and the box performs a specific set of operations on it. Once the operation is complete the output is retrieved. Once the output is retrieved, the box is ready to work on the next input.

Using this idea, function composition can be seen as a box inside of a box. The input *x* value goes into the inner box, and then the output of the inner box is used as the input of the outer box.

If the composite is fog(x), the output of "g" is the input of "f".

Example:

If f (x) = x + 2, and g (x) = 2x + 4, what is f(g(x))?

f(g(x)) = f(2x + 4) = (2x + 4) + 2 = 2x + 6.

.

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

  Here, you can describe any presumptions taken by you.

- Hardware and Software Requirements and Tools Used

  Listing down the hardware and software requirements along with the tools, libraries and packages used. Describe all the software tools used along with a detailed description of tasks done with those tools.

Hardware:-

System requirments for python installation

Software requirements:

Jupiter Notebook – it is required for run the commands

Tools : -

Scikit – **learn**

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

libraries and packages: -

Numpy - NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

Pandas- Pandas is another ML open-source data analysis library for Python. It focuses on data analysis and data manipulation. For machine learning programmers who want effortless working with structured multidimensional and time-series data, Pandas is the ML library they need.

Pandas offers numerous features for data handling, which include

- Data filtration
- Aligning data

- Handling data
- Pivoting data
- Data reshaping
- Merging of datasets
- Merging of datasets

Compared to Numpy, Pandas is fast, and it is one of the few libraries that can work with DateTime independently without getting help from external libraries. This tool works on all essential aspects of ML and data analysis.

Matplotlib- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

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

Sklearn.preprocessing - package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators.

grid search - The grid search provided by `GridSearchCV` exhaustively generates candidates from a grid of parameter values specified with the `param_grid` parameter.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

We are using python library and machine learning algorithms for problem solving.

Checking null or missing values
Checking duplictates values
Using data visualization, using scatter plot, hist plot
Using regulazition method
Spliting the data into training and testing formate
Using around 8 machine learning model
Find best model
Predicting the Sales price

- Testing of Identified Approaches (Algorithms)

  Listing down all the algorithms used for the training and testing.

  LinearRegression

  KNeighborsRegressor

  DecisionTreeRegressor

  RandomForestRegressor

  AdaBoostRegressor

  GradientBoostingRegressor

  XGBRegressor

  GridSearchCV

- Run and Evaluate selected models

  Describe all the algorithms used along with the snapshot of their code and what were the results observed over different evaluation metrics.

```
316]: from sklearn.linear_model import LinearRegression
      from sklearn.model_selection import train_test_split

      from sklearn.metrics import mean_squared_error,mean_absolute_error
```

```
317]: lm = LinearRegression()
```

```
318]: lm.fit(x_train,y_train)
```

```
318]: ▾ LinearRegression
       LinearRegression()
```

```
319]: lm.coef_
```

```
319]: array([-7.13625586e+01,  2.49572117e+03, -5.90598793e+01,  1.64958258e-01,
              7.04511297e+03,  1.09836839e+02,  2.20560832e+03,  5.10293785e+02,
             -3.75053199e+03, -7.29964821e+01,  1.74526897e+03,  4.90276198e+03,
             -1.61459596e+03,  9.08303511e+02,  8.53178594e+03,  3.16751904e+03,
              1.97714254e+00,  5.38407065e+01, -1.53460579e+03,  1.00045181e+04,
             -9.68996329e+02,  1.68208161e+02,  4.48519605e+03,  3.52197918e+01,
             -7.31387410e+03,  2.71132125e+03,  2.66794614e+02, -5.04532942e+03,
              8.40532656e+02, -3.19538009e+02, -2.01804785e+03, -1.21572100e+00,
             -4.31990388e+02,  1.84153169e+00,  2.89629922e+00,  3.52210981e+00,
             -3.42898963e+03, -1.53576710e+03,  3.00186077e+03, -4.18811756e+02,
              2.46482232e+01,  1.77930609e+01, -1.83765901e+01,  2.40646940e+01,
              5.07703445e+03, -5.57603614e+02,  1.18104432e+03,  1.27058055e+03,
             -2.21500262e+02, -4.46748639e+02, -6.25702441e+03, -1.86728727e+03,
              2.90524689e+03,  4.01067594e+03, -1.19537837e+03, -1.72047002e+01,
             -9.67434581e-01, -1.45749281e+03,  2.24901905e+03,  1.95592959e+01,
              5.87550250e+02, -5.28784813e+02,  7.44400111e+02,  2.06853632e+01,
              2.62690138e+01,  1.30482622e+01,  3.94939382e+01,  2.72647728e+01,
             -3.29572654e+02, -6.03073948e+04,  1.18905020e+00,  2.64837217e+02,
             -2.24327469e+03, -7.87608783e+02,  3.94387821e+03])
```

```
320]: lm.score(x_train,y_train)
```

```
320]: 0.6794854160441473
```

LinearRegression has given the 0.67 score

```
error
Mean absolute error: 28561.41268528727
Mean squared error: 2343825087.037773
```

```
: from sklearn.neighbors import KNeighborsRegressor
```

```
: knr = KNeighborsRegressor()
  knr.fit(x_train,y_train)

  knr.score(x_train,y_train)
  predknr = knr.predict(x_test)
  print(predknr)

  knr.score(x_test,y_test)
```

```
[160895.40119863 106940.         139495.40119863 123975.40119863
  87680.         254866.20359589 191115.40119863 222490.80239726
 265995.00119863 193780.         185647.         174886.20359589
 156090.80239726 188495.40119863 199133.60239726 242995.40119863
 123690.80239726 135475.40119863 140430.4        140520.
 210903.40239726 129590.80239726 105634.20119863 128745.40119863
 123080.         166090.80239726 132270.80239726 184255.
 179335.40119863 208250.80239726 177510.80239726 126095.40119863
 206335.40119863 183790.80239726 143995.40119863 147486.20359589
 192086.20359589 199835.40119863 177719.80119863 252174.6
 181477.00599315 135190.80239726 243886.20239726 182190.80239726
 118175.40119863 235800.         124090.80239726 281687.60119863
 145770.80239726 157190.80239726 152690.80239726 149595.40119863
 158580.         188095.40119863 281687.60119863 152290.80239726
 192400.         186770.80239726 154590.80239726 144695.40119863
 107755.40119863 171975.40119863 288183.8        288150.
 157380.         193340.         182841.40119863 244095.40119863
 274015.40119863 162895.40119863 166480.         150075.40119863
 123200.         183886.20359589 172675.40119863 122900.
 139885.40119863 196866.20359589 133990.80239726 197180.
 165670.80239726 153100.         172780.         156900.
 100195.40119863 206520.         178095.40119863 126775.40119863
 202195.         167790.80239726 165196.80239726 215490.80239726
 177495.40119863 163232.         168880.         171972.40119863
 200135.40119863 213995.40119863 233286.20359589 215695.40119863
 246886.20359589 165286.20359589 169395.40119863 187550.
 164876.20359589 110650.         207475.40119863 188590.80239726
 115400.         197706.80239726 158600.         161555.40119863
 330062.00119863 153535.40119863 156498.80239726 182841.40119863
 188901.80359589 152490.80239726 196600.         221466.60239726
 187495.40119863 147790.80239726 145966.20359589 117175.40119863
 118995.40119863 194455.40119863 181477.00599315 286600.
 258595.40119863 172390.80239726 100195.40119863 186190.80239726
 322595.40119863 144995.40119863 177000.         184995.40119863
```

KneighborsRegressor has given 0.5015358905363494 score
mean_absolute_error - 26909.494517410625

```
n [347]: rfr = RandomForestRegressor()
         rfr.fit(x_train,y_train)
         rfrpred = rfr.predict(x_test)
         print(dtrpred)

         rfr.score(x_test,y_test)
```

RandomforestRegrssor has given the 0.7038287158660581 score
mean_absolute_error - 29559.368284867054

```
52]: Gbr = GradientBoostingRegressor()
     Gbr.fit(x_train,y_train)
     Gbrpred = Gbr.predict(x_test)
     print(Gbrpred)

     Gbr.score(x_test,y_test)
```

GradientBoosRegressor has given the 0.7389630326730408

```
mean_absolute_error - 25594.674903718635
```

```
from xgboost import XGBRegressor
```

```
xgboost = XGBRegressor()
xgboost.fit(x_train,y_train)
xgboostpred = xgboost.predict(x_test)
print(xgboostpred)

xgboost.score(x_test,y_test)
```

```
XGBRegressor has given the 0.6401074576496931
mean_absolute_error - 30201.14465183137
```

```
383]: from sklearn.linear_model import SGDRegressor

       SGD = SGDRegressor()
       SGD.fit(x_train, y_train)
       SGD_pred = SGD.predict(x_test)
```

```
384]: print(mean_absolute_error(y_test, SGD_pred))
```

```
3.0360692746089744e+16
```

```
mean_absolute_error - 3.0360692746089744e+16
```

Gradient Boosting Regressor RandomForestRegressor has given the best score

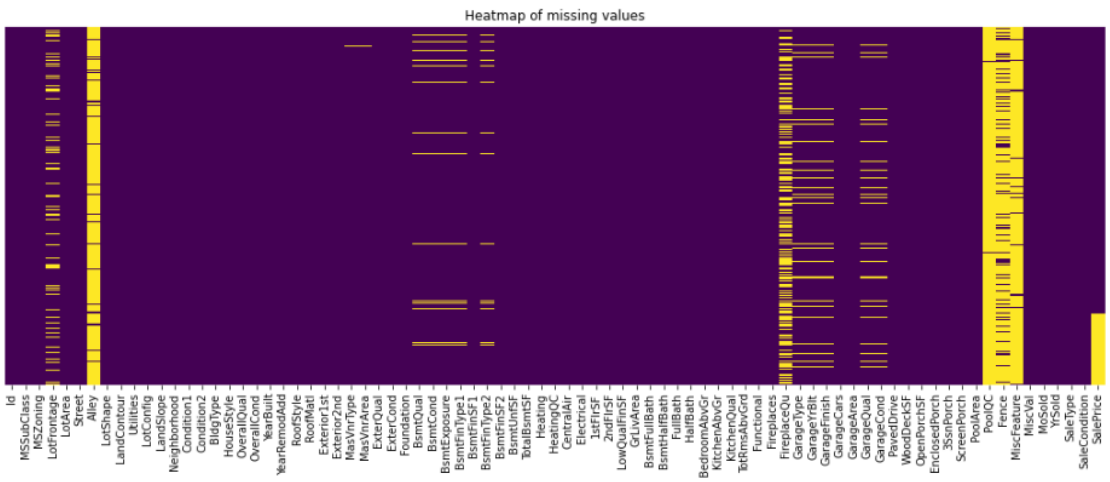
- Key Metrics for success in solving problem under consideration

  What were the key metrics used along with justification for using it? You may also include statistical metrics used if any.

- Visualizations

```
[426]: plt.figure(figsize=(18,6))
       plt.title('Heatmap of missing values')
       sns.heatmap(df_combined.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

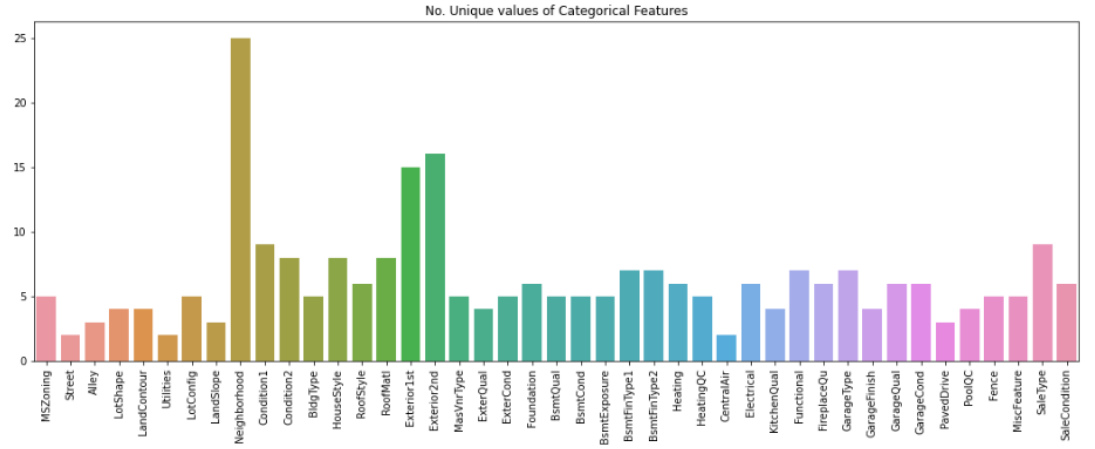
```
t[426]: <AxesSubplot:title={'center':'Heatmap of missing values'}>
```



Using heat map for checking null values. Since there are lots of null values

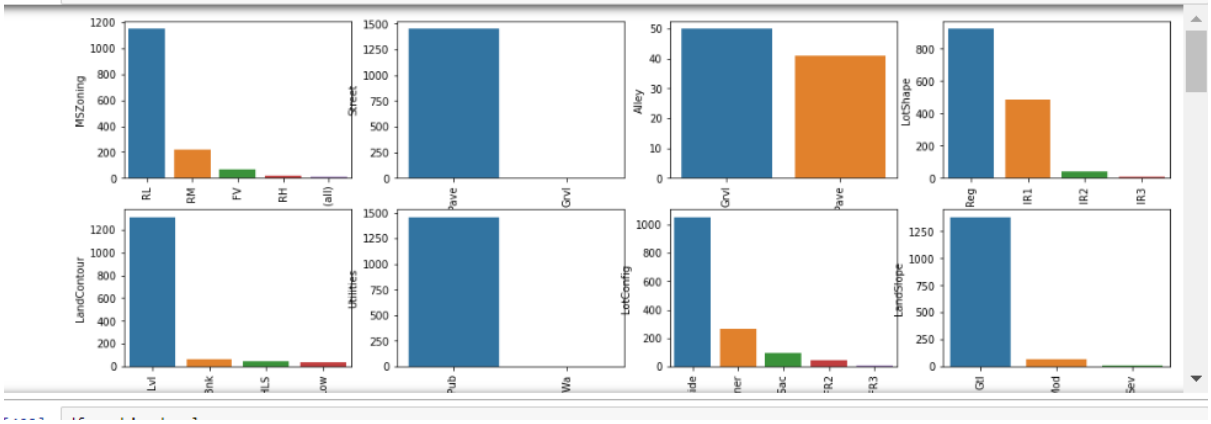
```
In [427]: unique_values = []
for col in object_cols:
  unique_values.append(df_combined[col].unique().size)
plt.figure(figsize=(18,6))
plt.title('No. Unique values of Categorical Features')
plt.xticks(rotation=90)
sns.barplot(x=object_cols,y=unique_values)
```

```
Out[427]: <AxesSubplot:title={'center':'No. Unique values of Categorical Features'}>
```



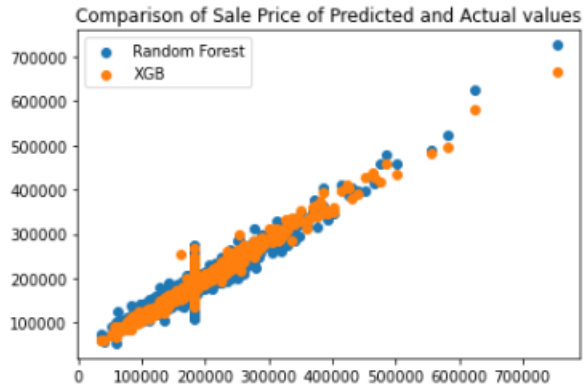
We are using bar plot for categoricals variables

```
[428]: plt.figure(figsize=(18,36))
plt.title('Categorical Features: Distribution')
plt.xticks(rotation=90)
index = 1
for col in object_cols:
  y = df_combined[col].value_counts()
  plt.subplot(11,4,index)
  plt.xticks(rotation=90)
  sns.barplot(x=list(y.index), y=y)
  index +=1
```



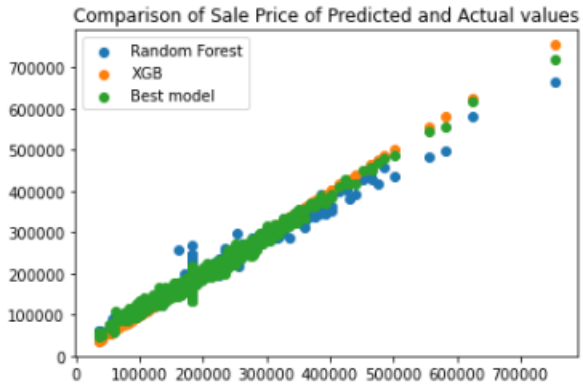
We have use bar plot for checking Distribution.

```
In [388]: plt.figure()
          plt.title('Comparison of Sale Price of Predicted and Actual values')
          plt.scatter(y_train,Gbr.predict(x_train),label='Random Forest')
          plt.scatter(y_train,rfr.predict(x_train),label='XGB')
          plt.legend()
Out[388]: <matplotlib.legend.Legend at 0x1604e84f9a0>
```



Using scatter plot for Comparison of sale price of prediction and actual price

```
n [399]: plt.figure()
         plt.title('Comparison of Sale Price of Predicted and Actual values')
         plt.scatter(y_train,rfr.predict(x_train),label='Random Forest')
         plt.scatter(y_train,xgboost.predict(x_train),label='XGB')
         plt.scatter(y_train,regressor.predict(x_train),label='Best model')
         plt.legend()
ut[399]: <matplotlib.legend.Legend at 0x16050389340>
```



Using scatter plot for Comparison of sale price of prediction and actual price after hyperparameter_grid

- ## Interpretation of the Results

  Give a summary of what results were interpreted from the visualizations, preprocessing and modelling.

CONCLUSION

- ## Key Findings and Conclusions of the Study

- The data has been separated into training and test sets. The goal is to predict the final Sale Price of the houses in the test data set.

- Advanced feature engineering is used to infer missing values in the data set. The categorical features are transformed into numerical features using one hot encoding. The performances of various regression techniques such as Random Forest, Stochastic Gradient Descent, Gradient Boosting, and Xtremme Gradient Boosting are compared. Using the best fit model,

  Gradient Boosting and random regressor has given the best score

- ## Learning Outcomes of the Study in respect of Data Science

  Gradient Boosting and random regressor has given the best score

- ## Limitations of this work and Scope for Future Work

  What are the limitations of this solution provided, the future scope? What all steps/techniques can be followed to further extend this study and improve the results.