HOUSING PRICE PREDICTION

Submitted by:

ANITA THAPA

ACKNOWLEDGMENT

I would like to express my special thanks of gratitude to my SME Swati Ma’am for giving me an opportunity to work on the Housing price prediction project. During my research on this project internet sites like towardsdatascience, medium, meanderingscience.com and Stack Overflow helped me understand the main problem statement, also to collect data and in finding their solutions.

INTRODUCTION

- ## Business Problem Framing

  Model to predict the housing price so that our client is able to invest in the right property for entering the Australian market.

- ## Conceptual Background of the Domain Problem

  Real estate and housing is one of the markets which are the major contributor in the world's economy. Real estate companies make use of data analytics to buy properties at lower rates. Our client is looking at prospective properties to buy houses to enter the market. A model is to be built to predict the house prices using the data available.

- ## Review of Literature

  The research was done on the topic Housing data in order to understand its functioning in finding the housing prices for the analysis. Then the given dataset was properly studied to understand each and every column present so better prediction

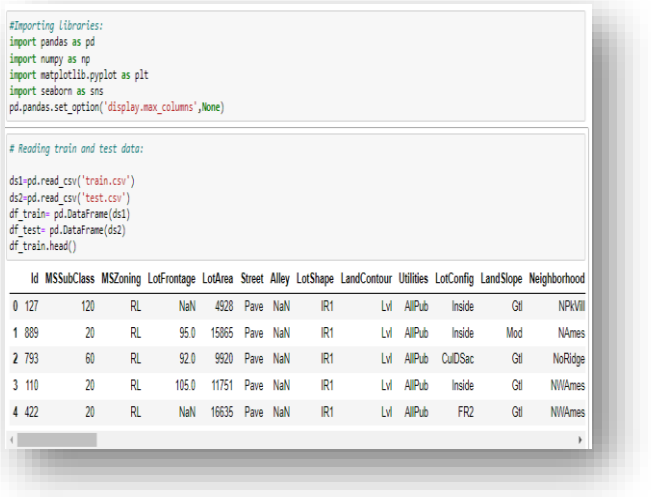
- ## Motivation for the Problem Undertaken

  The main objective was to analyse the present real estate market and to model a housing price prediction. In order to understand the Real estate industry and their market trends this project was taken up so that a House price valuation model is build to help our client to invest in the market.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

  The data was present in excel format so using pandas the excel file was read. Then the dataset details were analysed using .shape () to find the number of rows and columns, using dtypes () we found the dataset datatype like object and numeric datatype. Dataset has 1168 Rows and 81 Columns and the Target column is Sale Price. Dataset is a Regression model.



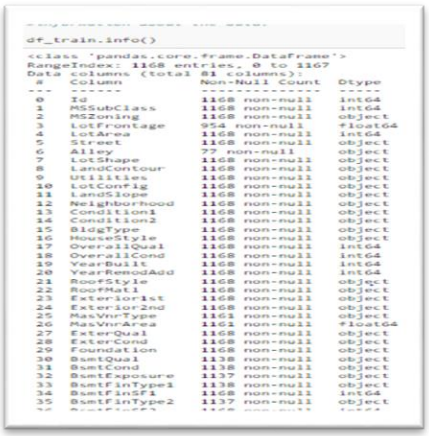
```
#Importing libraries:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
pd.pandas.set_option('display.max_columns',None)
```

```
# Reading train and test data:

ds1=pd.read_csv('train.csv')
ds2=pd.read_csv('test.csv')
df_train= pd.DataFrame(ds1)
df_test= pd.DataFrame(ds2)
df_train.head()
```

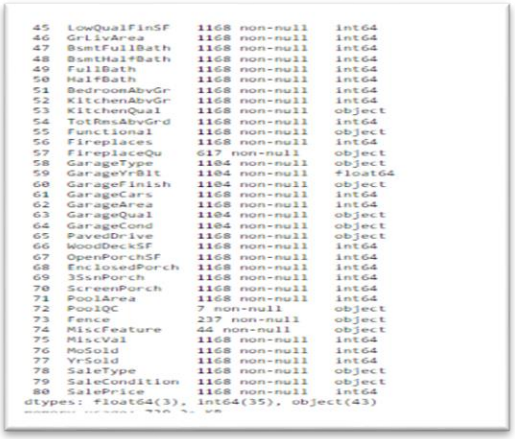
	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood
0	127	120	RL	NaN	4928	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NPkVill
1	889	20	RL	95.0	15865	Pave	NaN	IR1	Lvl	AllPub	Inside	Mod	NAmes
2	793	60	RL	92.0	9920	Pave	NaN	IR1	Lvl	AllPub	CulDSac	Gtl	NoRidge
3	110	20	RL	105.0	11751	Pave	NaN	IR1	Lvl	AllPub	Inside	Gtl	NWAmes
4	422	20	RL	NaN	16635	Pave	NaN	IR1	Lvl	AllPub	FR2	Gtl	NWAmes

- Data Sources and their formats

  Data was taken from various websites and saved as excel file. Then data was imported in python using pandas . Information of dataset is found using datasetname.info(). As per the information each column has count of 1168 but 18 columns have nan values and the datatypes are int64, float and object type as shown in below figure.



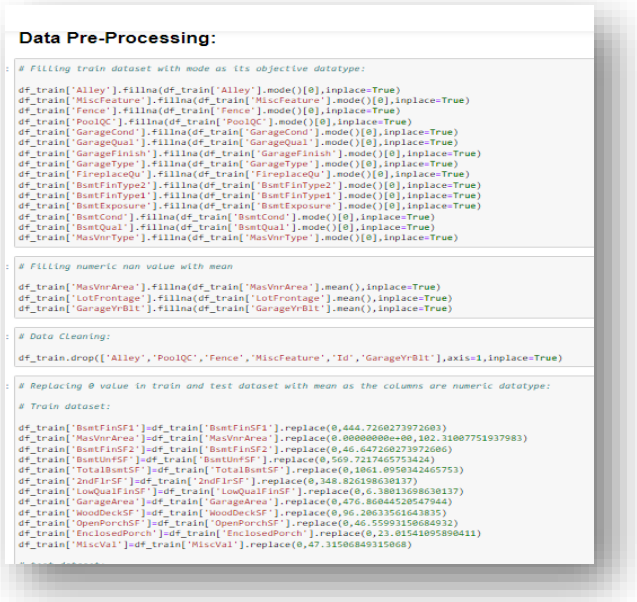
```
df_train.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1168 entries, 0 to 1167
Data columns (total 81 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             1168 non-null   int64
 1   MSSubClass     1168 non-null   int64
 2   MSZoning       1168 non-null   object
 3   LotFrontage    954 non-null    float64
 4   LotArea        1168 non-null   int64
 5   Street         1168 non-null   object
 6   Alley          77 non-null     object
 7   LotShape       1168 non-null   object
 8   LandContour    1168 non-null   object
 9   Utilities      1168 non-null   object
 10  LotConfig      1168 non-null   object
 11  LandSlope      1168 non-null   object
 12  Neighborhood   1168 non-null   object
 13  Condition1     1168 non-null   object
 14  Condition2     1168 non-null   object
 15  BldgType       1168 non-null   object
 16  HouseStyle     1168 non-null   object
 17  OverallQual    1168 non-null   int64
 18  OverallCond    1168 non-null   int64
 19  YearBuilt      1168 non-null   int64
 20  YearRemodAdd   1168 non-null   int64
 21  RoofStyle      1168 non-null   object
 22  RoofMatl       1168 non-null   object
 23  Exterior1st    1168 non-null   object
 24  Exterior2nd    1168 non-null   object
 25  MasVnrType     1161 non-null   object
 26  MasVnrArea     1161 non-null   float64
 27  ExterQual      1168 non-null   object
 28  ExterCond      1168 non-null   object
 29  Foundation     1168 non-null   object
 30  BsmtQual       1138 non-null   object
 31  BsmtCond       1138 non-null   object
 32  BsmtExposure   1137 non-null   object
 33  BsmtFinType1   1138 non-null   object
 34  BsmtFinSF1     1168 non-null   int64
 35  BsmtFinType2   1137 non-null   object
```



```
 45  LowQualFinSF   1168 non-null   int64
 46  GrLivArea      1168 non-null   int64
 47  BsmtFullBath   1168 non-null   int64
 48  BsmtHalfBath   1168 non-null   int64
 49  FullBath       1168 non-null   int64
 50  HalfBath       1168 non-null   int64
 51  BedroomAbvGr   1168 non-null   int64
 52  KitchenAbvGr   1168 non-null   int64
 53  KitchenQual    1168 non-null   object
 54  TotRmsAbvGrd   1168 non-null   int64
 55  Functional     1168 non-null   object
 56  Fireplaces     1168 non-null   int64
 57  FireplaceQu    617 non-null    object
 58  GarageType     1104 non-null   object
 59  GarageYrBlt    1104 non-null   float64
 60  GarageFinish   1104 non-null   object
 61  GarageCars     1168 non-null   int64
 62  GarageArea     1168 non-null   int64
 63  GarageQual     1104 non-null   object
 64  GarageCond     1104 non-null   object
 65  PavedDrive     1168 non-null   object
 66  WoodDeckSF     1168 non-null   int64
 67  OpenPorchSF    1168 non-null   int64
 68  EnclosedPorch  1168 non-null   int64
 69  3SsnPorch      1168 non-null   int64
 70  ScreenPorch    1168 non-null   int64
 71  PoolArea       1168 non-null   int64
 72  PoolQC         7 non-null      object
 73  Fence          237 non-null    object
 74  MiscFeature    44 non-null     object
 75  MiscVal        1168 non-null   int64
 76  MoSold         1168 non-null   int64
 77  YrSold         1168 non-null   int64
 78  SaleType       1168 non-null   object
 79  SaleCondition  1168 non-null   object
 80  SalePrice      1168 non-null   int64
dtypes: float64(3), int64(35), object(43)
```

- Data Preprocessing Done

  Data had nan values in approximately 18 columns so they were filled using mean and mode as the datatype were numeric and object respectively. Numeric values having 0 were also replaced by the mean of the column.

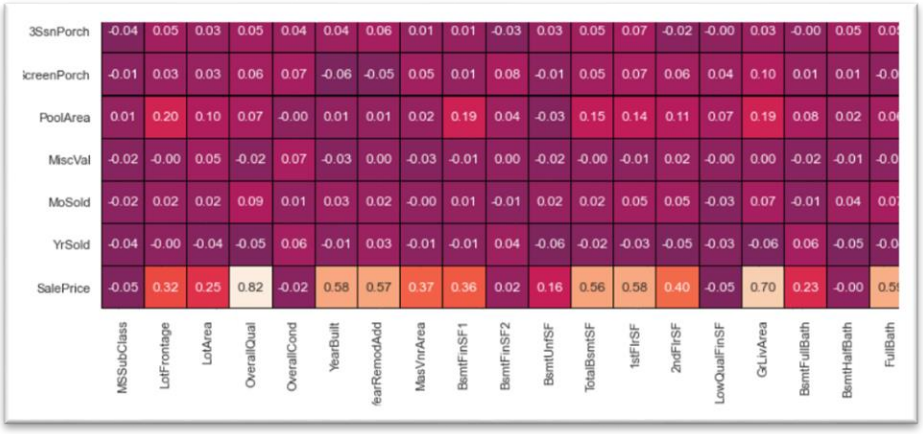


Data Pre-Processing:

```
# Filling train dataset with mode as its objective datatype:

df_train['Alley'].fillna(df_train['Alley'].mode()[0],inplace=True)
df_train['MiscFeature'].fillna(df_train['MiscFeature'].mode()[0],inplace=True)
df_train['Fence'].fillna(df_train['Fence'].mode()[0],inplace=True)
df_train['PoolQC'].fillna(df_train['PoolQC'].mode()[0],inplace=True)
df_train['GarageCond'].fillna(df_train['GarageCond'].mode()[0],inplace=True)
df_train['GarageQual'].fillna(df_train['GarageQual'].mode()[0],inplace=True)
df_train['GarageFinish'].fillna(df_train['GarageFinish'].mode()[0],inplace=True)
df_train['GarageType'].fillna(df_train['GarageType'].mode()[0],inplace=True)
df_train['FireplaceQu'].fillna(df_train['FireplaceQu'].mode()[0],inplace=True)
df_train['BsmtFinType2'].fillna(df_train['BsmtFinType2'].mode()[0],inplace=True)
df_train['BsmtFinType1'].fillna(df_train['BsmtFinType1'].mode()[0],inplace=True)
df_train['BsmtExposure'].fillna(df_train['BsmtExposure'].mode()[0],inplace=True)
df_train['BsmtCond'].fillna(df_train['BsmtCond'].mode()[0],inplace=True)
df_train['BsmtQual'].fillna(df_train['BsmtQual'].mode()[0],inplace=True)
df_train['MasVnrType'].fillna(df_train['MasVnrType'].mode()[0],inplace=True)

# Filling numeric non value with mean

df_train['MasVnrArea'].fillna(df_train['MasVnrArea'].mean(),inplace=True)
df_train['LotFrontage'].fillna(df_train['LotFrontage'].mean(),inplace=True)
df_train['GarageYrBlt'].fillna(df_train['GarageYrBlt'].mean(),inplace=True)

# Data Cleaning:

df_train.drop(['Alley','PoolQC','Fence','MiscFeature','Id','GarageYrBlt'],axis=1,inplace=True)

# Replacing 0 value in train and test dataset with mean as the columns are numeric datatype:

# Train dataset:

df_train['BsmtFinSF1']=df_train['BsmtFinSF1'].replace(0,444.72602739726603)
df_train['MasVnrArea']=df_train['MasVnrArea'].replace(0.00000000e+00,102.31007751937983)
df_train['BsmtFinSF2']=df_train['BsmtFinSF2'].replace(0,46.647260273972606)
df_train['BsmtUnfSF']=df_train['BsmtUnfSF'].replace(0,569.7217465753424)
df_train['TotalBsmtSF']=df_train['TotalBsmtSF'].replace(0,1061.0950342465753)
df_train['2ndFlrSF']=df_train['2ndFlrSF'].replace(0,348.82619863013717)
df_train['LowQualFinSF']=df_train['LowQualFinSF'].replace(0,6.380136986301371)
df_train['GarageArea']=df_train['GarageArea'].replace(0,476.86044520547944)
df_train['WoodDeckSF']=df_train['WoodDeckSF'].replace(0,96.20633561643835)
df_train['OpenPorchSF']=df_train['OpenPorchSF'].replace(0,46.559931506849321)
df_train['EnclosedPorch']=df_train['EnclosedPorch'].replace(0,23.015410958904111)
df_train['MiscVal']=df_train['MiscVal'].replace(0,47.315068493150685)
```

- Data Inputs- Logic- Output Relationships

  Output Column is SalePrice column and rest other columns are Input Columns. Using Correlation we can find out the relationship between Input and output columns. OverallQual column is positively correlated with output column.



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

  No assumption was set.

- Hardware and Software Requirements and Tools Used

  Listing Libraries used were:

  1. Numpy : It is a Numerical Python library used for numerical computation like arrays in Python.
  2. Panda: It was used for reading and converting excel/csv file into dataframe.
  3. matplotlib: this library was used for plotting the graph in the EDA.
  4. Seaborn: This library is also used for visualization as it has helped to plot different types of plots like countplot, displot, boxplot and heatmap in the notebook.
  5. Label encoder: It helped to encode the object datatype into numeric datatype as the Machine learning algorithm works on numeric datatype only.
  6. sklearn.preprocessing and power transform: It was used to remove the skewness from the dataset.
  7. Standard Scaler : It was used to scale the feature column of the dataset before model selection.
  8. sklearn: This library was used to import train_test_split, metrics like accuracy score, classification report and also importing Algorithms likes Linear Regression, Decision Tree Regressor, SVR, Random Forest Regressor and K-Neighbors Regressor..
  9. pickle: This library was used to save the final model.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  The Standard Scaled data was used in Model building. As the model is Regressor so Linear Regressor library along with r2 score, cross validation score, means square error metrics were imported. Then 4 Algorithms were used which were Decision Tree Regressor, SVR, Random Forest Regressor and K-Neighbors Regressor. Then after selecting best model, then the final model was deployed and the accuracy of model was 88.5% as it predicted values approximately equal to actual values.

```
# Model selection:

from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
from sklearn.metrics import mean_squared_error,mean_absolute_error
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

import warnings
warnings.filterwarnings('ignore')
```

```
for i in range(0,100):
    train_x,test_x,train_y,test_y=train_test_split(x,y,test_size=0.2,random_state=i)
    lr.fit(train_x,train_y)
    pred_train=lr.predict(train_x)
    pred_test=lr.predict(test_x)
    print(f"At random state {i},the training accuracy is:- {r2_score(train_y,pred_train)}")
    print(f"At random state {i},the testing accuracy is:- {r2_score(test_y,pred_test)}")
    print("\n")
```

- Testing of Identified Approaches (Algorithms)

  Algorithms used for the training and testing are:

  1. Linear Regression
  2. Decision Tree Regressor
  3. SVR
  4. Random Forest Regressor
  5. K-Neighbors Regressor

- Run and Evaluate selected models

  Out of the entire 4 algorithm used the best algorithm is Random Forest Regression as the r2 score and cross validation score have lesser difference.

4. Random forest regressor

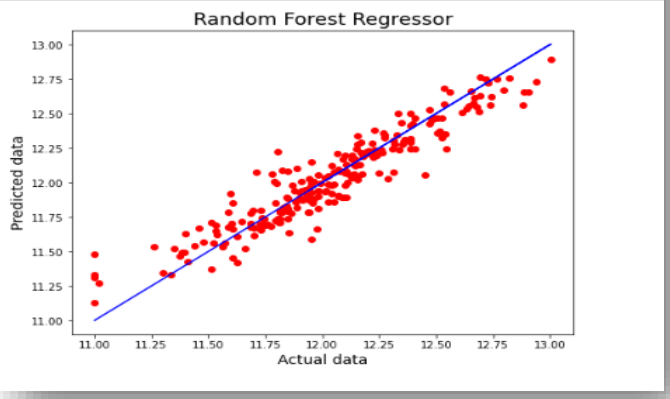
```
# Importing Libraries and Hyper parameter tuning:
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor

parameters = {'criterion':['mse', 'mae'],'bootstrap': [True, False], 'max_depth': [5, 10, None], 'max_features': ['auto', 'log2']
rf =RandomForestRegressor()
clf = GridSearchCV(rf,parameters,cv = 9, n_jobs = -1, verbose = 2)
clf.fit(train_x,train_y)

print(clf.best_params_)
```

```
Fitting 9 folds for each of 240 candidates, totalling 2160 fits
{'bootstrap': True, 'criterion': 'mse', 'max_depth': None, 'max_features': 'auto', 'n_estimators': 13}
```

```
rf= RandomForestRegressor(criterion="mse",max_features="auto",max_depth=None,bootstrap=True,n_estimators=13)
rf.fit(train_x,train_y)
rf.score(train_x,train_y)
predrf = rf.predict(test_x)
print('rf score',rf.score(train_x,train_y))
rfs = r2_score(test_y,predrf)
print('R2 Score:',rfs*100)

rfscore = cross_val_score(rf,x,y,cv=9)
rfc = rfscore.mean()
print('Cross Val Score:',rfc*100)
print("Mean squared error:",mean_squared_error(test_y,predrf))
print("Mean absolute error:",mean_absolute_error(test_y,predrf))
```

```
rf score 0.9754355977056027
R2 Score: 88.50760703856633
Cross Val Score: 85.60935987955799
Mean squared error: 0.017750123858741758
Mean absolute error: 0.10063611344539777
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(8,6))
plt.scatter(x=test_y, y=predrf, color='r')
plt.plot(test_y,test_y, color='b')
plt.xlabel('Actual data',fontsize=14)
plt.ylabel('Predicted data',fontsize=14)
plt.title('Random Forest Regressor',fontsize=18)
plt.show()
```



.

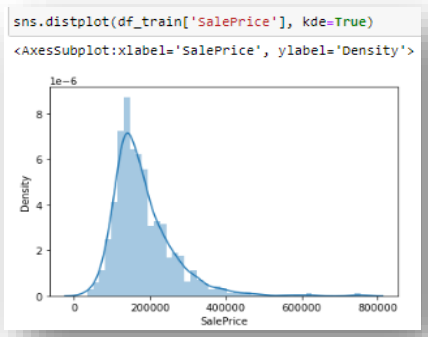
- Key Metrics for success in solving problem under consideration

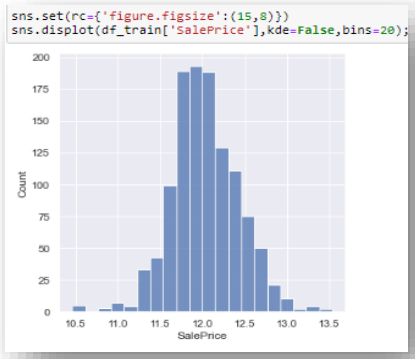
  The main metric which helped to determine the best model is comparing r2 score with the cross validation score of the algorithm. For model selection the algorithm must have small difference between cross validation score and r2 score then that algorithm is considered to be the best model.

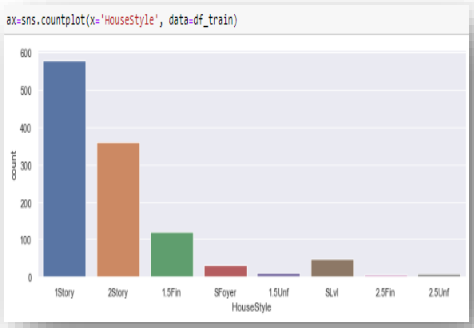
  .

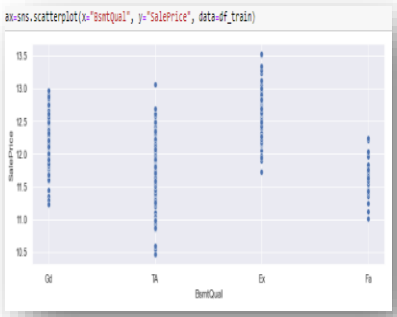
- Visualizations

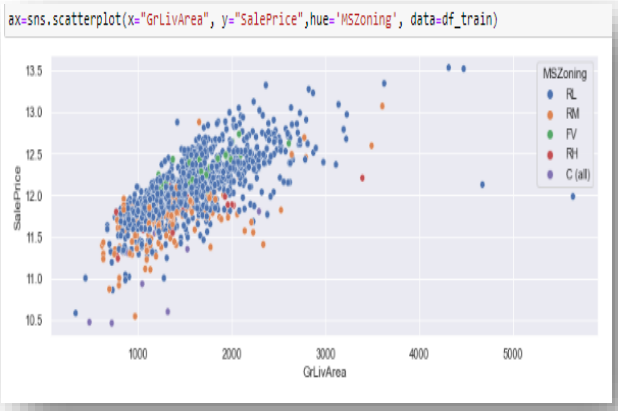
  Different plots were used in the problem like displot, countplot, histplot, heatmap, boxplot and pyplot using matplotlib and seaborn library as shown in the images below.

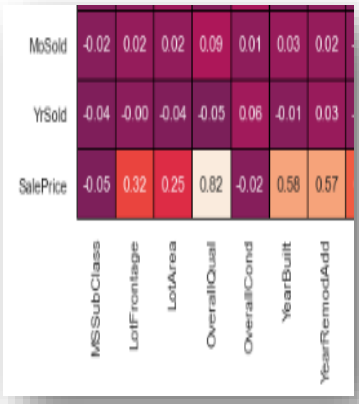


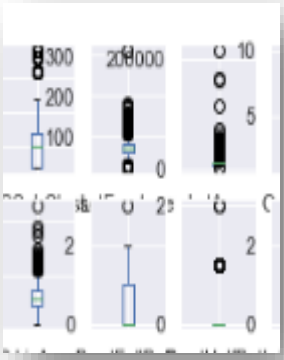












- Interpretation of the Results

  From the visualizations, preprocessing and modeling of the data it was interpreted that sale price of houses majorly depends upon the overall quality of the house. Maximum houses were priced high which were 2 stories and have ta garage area. Houses were built in 2000. As per the model the data can be predicted with 88.5% accuracy.

CONCLUSION

- Key Findings and Conclusions of the Study

  Describe the key findings, inferences, observations from the whole problem. From the project it is inferred that sale price majorly depends upon overall quality of house and the house with good and excellent Overall Quality are priced higher.

- Learning Outcomes of the Study in respect of Data Science

  Using Visualization libraries like matplotlib and seaborn it was easy to find the correlation between the dataset columns and the plotting of each column along the target column helped to analyse the problem pretty well. As visualization helped to get better insight of the data like skewness present in the data. While checking the metrics like mse and mae for various algorithm helped a lot in determining the error in the dataset and helped to correct the error for accurate prediction.

- Limitations of this work and Scope for Future Work

  As the model accuracy is 88.5% so it can approximately predict the correct value as sometimes it might predict wrong values but chances are less. Only incase few more features or columns add up in the dataset then the model won't predict accurately so need to update the model accordingly. Rest the model is working well now.