HOUSING PROJECT

Submitted by: **ASHISH YADAV**

ACKNOWLEDGMENT

We would like to express our deep and sincere gratitude to FlipnWork for giving us the opportunity to do this project. As a great bridge between academic and industry, this program educated us how to perform theoretical methodology in real life. We would like to express our sincere thankfulness to my mentor Sapna Verma for the continuous support , for their patience, enthusiasm, motivation and immense knowledge. As our academic mentor, Sapna Verma supported and helped in this project. Additionally, we would also like to thank all our friends who offered us some help, such as Divyanshu who helped me during this project.

INTRODUCTION

Problem Statement:

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. 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.

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 data is provided in the CSV file below.

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?

Business Goal:

We are required to model the price of houses with the available independent variables. This model will then be used by the management to understand how exactly the prices vary with the variables. They can accordingly manipulate the strategy of the firm and concentrate on areas that will yield high returns. Further, the model will be a good way for the management to understand the pricing dynamics of a new market.

Domain Understanding :

- The real estate sector is an important industry with many stakeholders ranging from regulatory bodies to private companies and investors. Among these stakeholders, there is a high demand for a better understanding of the industry operational mechanism and driving factors. Today there is a large amount of data available on relevant statistics as well as on additional contextual factors, and it is natural to try to make use of these in order to improve our understanding of the industry.

Literature :

- The main steps in our research were the following.
  • **Exploratory Data Analysis (EDA):** By conducting explanatory data analysis, we obtain a better understanding of our data. This yields insights that can be helpful later when building a model, as well as insights that are independently interesting.
  • **Feature Selection:** In order to avoid overfitting issues, we select 25(according to PCA) variables out of the original 81 by using methods Ridge , feature selection
  • **Modeling:** We apply Decision Tree , Random Forest , Linear Regression , K-Neighbours models for prediction of the sale price

• Motivation for the Problem Undertaken:

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 data is provided in the CSV file below.
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

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the 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 data is provided in the CSV file.

DATASET :

```
df=pd.read_csv(r'C:\ProgramData\train.csv')
df.head()
```

Out[94]:

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

5 rows × 81 columns

In [72]:
```
df.shape
```

Out[72]: (1168, 81)

EXPLORATORY DATA ANALYSIS :

Data exploration is the first step in data analysis and typically involves summarizing the main characteristics of a data set, including its size, accuracy, initial patterns in the data and other attributes. It is commonly conducted by data analysts using visual analytics tools, but it can also be done in more advanced statistical software, Python. Before it can conduct analysis on data collected by multiple data sources and stored in data warehouses, an organization must know how many cases are in a data set, what variables are included, how many missing values there are and what general hypotheses the data is likely to support. An initial exploration of the data set can help answer these questions by familiarizing analysts with the data with which they are working.

```
df.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
```

Statistical Summary

In [6]:
```
# Statistical summary
df.describe()
```

Out[6]:

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	...	WoodDeck
count	1168.000000	1168.000000	954.00000	1168.000000	1168.000000	1168.000000	1168.000000	1168.000000	1161.000000	1168.000000	...	1168.000
mean	724.136130	56.767979	70.98847	10484.749144	6.104452	5.595890	1970.930651	1984.758562	102.310078	444.726027	...	96.206
std	416.159877	41.940650	24.82875	8957.442311	1.390153	1.124343	30.145255	20.785185	182.595606	462.664785	...	126.158
min	1.000000	20.000000	21.00000	1300.000000	1.000000	1.000000	1875.000000	1950.000000	0.000000	0.000000	...	0.000
25%	360.500000	20.000000	60.00000	7621.500000	5.000000	5.000000	1954.000000	1966.000000	0.000000	0.000000	...	0.000
50%	714.500000	50.000000	70.00000	9522.500000	6.000000	5.000000	1972.000000	1993.000000	0.000000	385.500000	...	0.000
75%	1079.500000	70.000000	80.00000	11515.500000	7.000000	6.000000	2000.000000	2004.000000	160.000000	714.500000	...	171.000
max	1460.000000	190.000000	313.00000	164660.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	...	857.000

8 rows × 38 columns

Checking null values in dataset

In [74]:
```
# checking for null values in dataset

df.isnull().sum()
```

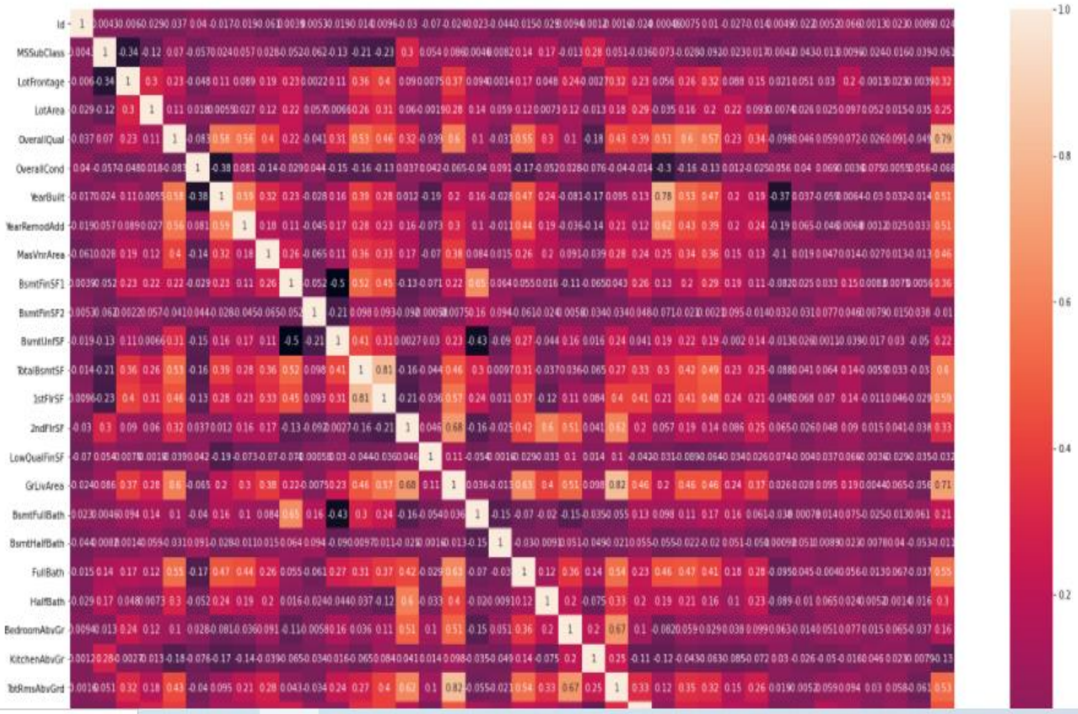
Out[74]:
```
Id                 0
MSSubClass         0
MSZoning           0
LotFrontage      214
LotArea            0
                ...
MoSold             0
YrSold             0
SaleType           0
SaleCondition      0
SalePrice          0
Length: 81, dtype: int64
```

Filling null values in dataset

```
In [95]: # Filling null values for categorical features with mode and numerical features with mean

         df['MasVnrType'].fillna(df['MasVnrType'].mode()[0], inplace=True)
         df['LotFrontage'].fillna(df['LotFrontage'].mean(), inplace=True)
         df['MasVnrArea'].fillna(df['MasVnrArea'].mode()[0], inplace=True)
         df['BsmtQual'].fillna(df['BsmtQual'].mode()[0], inplace=True)

         df['BsmtCond'].fillna(df['BsmtCond'].mode()[0], inplace=True)

         df['BsmtExposure'].fillna(df['BsmtExposure'].mode()[0], inplace=True)

         df['BsmtFinType1'].fillna(df['BsmtFinType1'].mode()[0], inplace=True)

         df['BsmtFinType2'].fillna(df['BsmtFinType2'].mode()[0], inplace=True)

         df['FireplaceQu'].fillna(df['FireplaceQu'].mode()[0], inplace=True)

         df['GarageType'].fillna(df['GarageType'].mode()[0], inplace=True)
         df['GarageYrBlt'].fillna(df['GarageYrBlt'].mean(), inplace=True)
         df['GarageFinish'].fillna(df['GarageFinish'].mode()[0], inplace=True)

         df['GarageQual'].fillna(df['GarageQual'].mode()[0], inplace=True)
         df['GarageCond'].fillna(df['GarageCond'].mode()[0], inplace=True)

         df['Fence'].fillna(df['Fence'].mode()[0], inplace=True)
```

CORRELATION

```
In [89]: # checking correlation of independent variables with 'SalePrice' variable

         plt.figure(figsize=(25,20))
         corr_matrix=df.corr()
         sns.heatmap(corr_matrix,annot=True)
         plt.show()
```



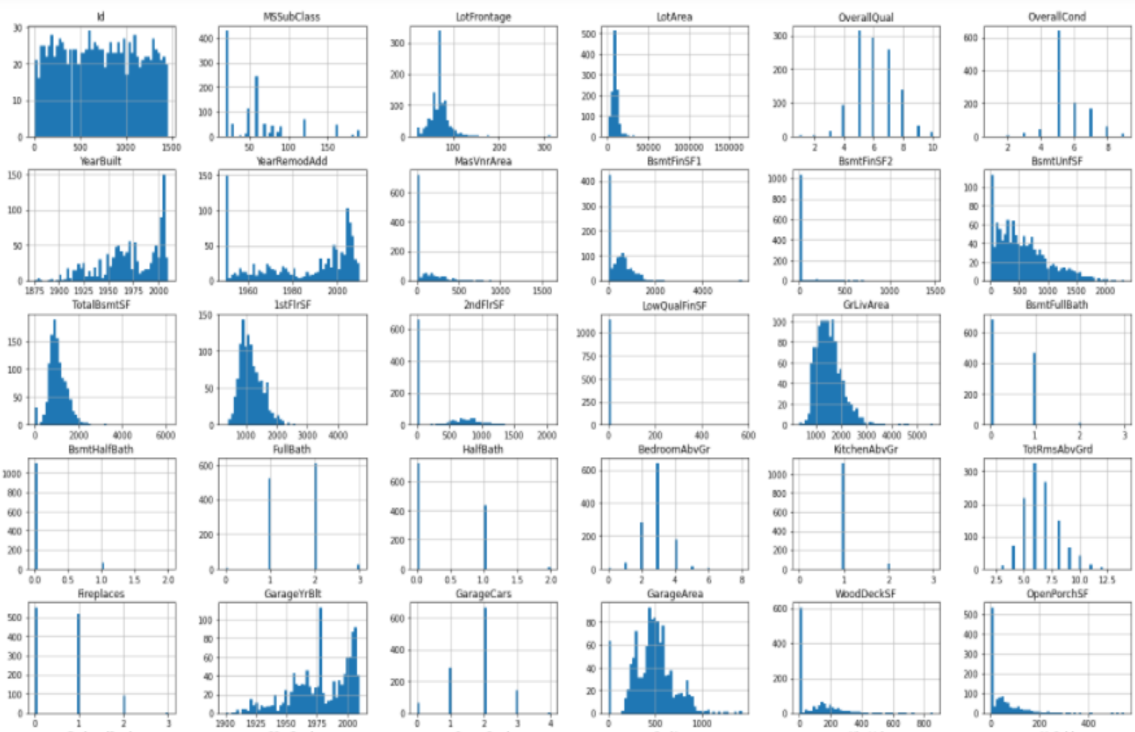
CORRELATION OF INDEPENDENT VARIABLES WITH ‘SALE PRICE’ :

```
In [90]: corr_matrix['SalePrice'].sort_values(ascending=False)

Out[90]: SalePrice        1.000000
         OverallQual      0.789185
         GrLivArea        0.707300
         GarageCars       0.628329
         GarageArea       0.619000
         TotalBsmtSF      0.595042
         1stFlrSF         0.587642
         FullBath         0.554988
         TotRmsAbvGrd     0.528363
         YearBuilt        0.514408
         YearRemodAdd     0.507831
         MasVnrArea       0.460535
         Fireplaces       0.459611
         GarageYrBlt      0.458007
         BsmtFinSF1       0.362874
         OpenPorchSF      0.339500
         2ndFlrSF         0.330386
         LotFrontage      0.323779
         WoodDeckSF       0.315444
         HalfBath         0.295592
         LotArea          0.249499
         BsmtUnfSF        0.215724
         BsmtFullBath     0.212924
         BedroomAbvGr     0.158281
         PoolArea         0.103280
         ScreenPorch      0.100284
         MoSold           0.072764
         3SsnPorch        0.060119
         BsmtFinSF2      -0.010151
         BsmtHalfBath    -0.011109
         MiscVal         -0.013071
```

DATA VISUALIZATION

- Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. In the world of Big Data, data visualization tools and technologies are essential to analyse massive amounts of information and make data-driven decisions.

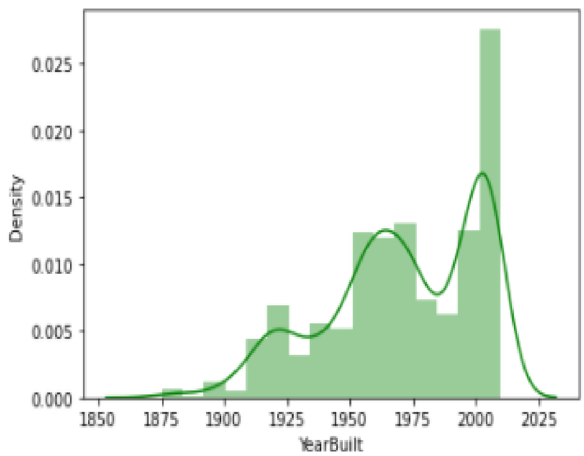
Distribution Plots



Distribution plot of YearBuilt Feature

In [86]:
```
# checking distibution of YearBuilt feature

sns.distplot(df['YearBuilt'], color = 'green')
```

Out[86]: <AxesSubplot:xlabel='YearBuilt', ylabel='Density'>

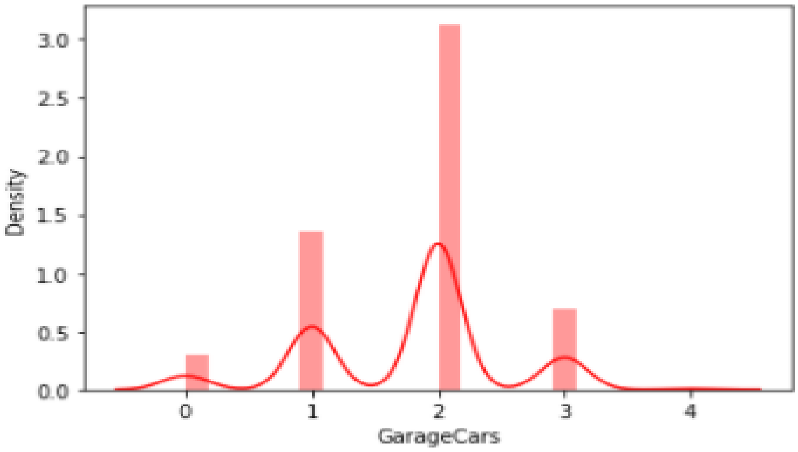


In [ ]:
```
# Clearly majority of houses are built in between 1950-2010 period and it is rightly skewed
```

Distribution plot of GarageCars Feature

In [91]:
```
# checking distibution of GarageCars feature

sns.distplot(df['GarageCars'], color = 'red')
```

Out[91]: <AxesSubplot:xlabel='GarageCars', ylabel='Density'>

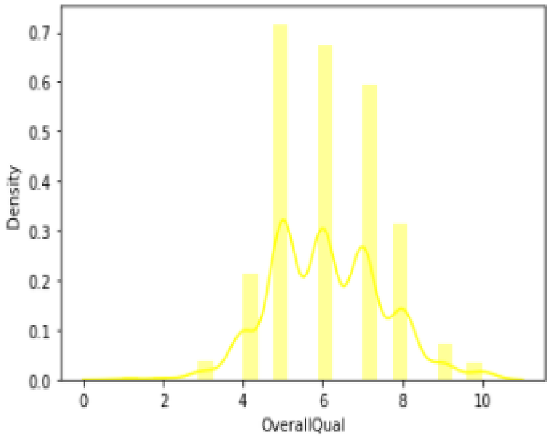


In [ ]:
```
# Clearly maximum houses with cars capacity in garage are 2
```

Distribution plot of OverallQual Feature

In [92]:
```
# checking distibution of OverallQual feature

sns.distplot(df['OverallQual'], color = 'yellow')
```

Out[92]: <AxesSubplot:xlabel='OverallQual', ylabel='Density'>

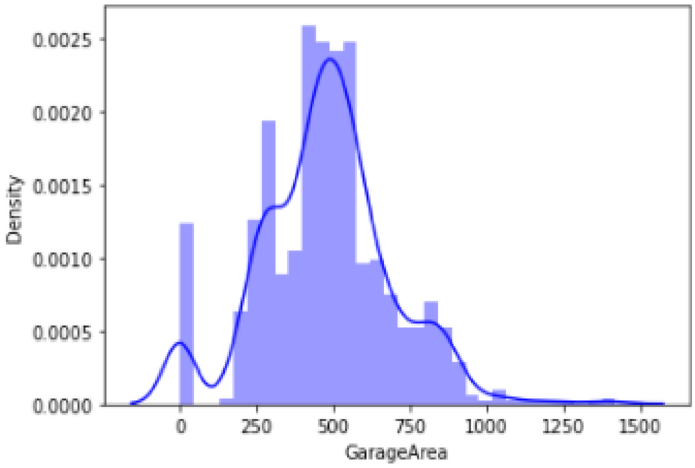


In [ ]:
```
# Clearly majority of the houses belong to rating between 4 to 8 in overall quality category
```

Distribution plot of GarageArea Feature

In [88]:
```
# checking distibution of 'GarageArea feature

sns.distplot(df['GarageArea'], color = 'blue')
```

Out[88]: <AxesSubplot:xlabel='GarageArea', ylabel='Density'>



LABEL ENCODING

In [13]:
```
# converting categorical features into ordinal

df_cat = df.select_dtypes(include=['object'])

le= LabelEncoder()
for i in df_cat:
    df[i] = le.fit_transform(df[i])
```

In [14]:
```
df.head()
```

Out[14]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	LotShape	LandContour	Utilities	LotConfig	...	3SsnPorch	ScreenPorch	PoolArea	Fence	Mis
0	127	120	3	70.98847	4928	1	0	3	0	4	...	0	0	0	2	
1	889	20	3	95.00000	15865	1	0	3	0	4	...	0	224	0	2	
2	793	60	3	92.00000	9920	1	0	3	0	1	...	0	0	0	2	
3	110	20	3	105.00000	11751	1	0	3	0	4	...	0	0	0	2	
4	422	20	3	70.98847	16635	1	0	3	0	2	...	0	0	0	2	

5 rows × 78 columns

In [15]:
```
# Clearly all the features have been converted to numeric type
```

Splitting Dataset into X , Y

In [16]:
```
# Splitting dataset into X and Y

X=df.drop('SalePrice',axis=1)
y=df.SalePrice
```

Scaling of Dataset

In [17]:
```

# Scaling the dataset and normalizing feature variables

from sklearn.preprocessing import StandardScaler

scale = StandardScaler()
X_features=X
X= scale.fit_transform(X)
```

PCA

In [18]:
```
# Applying PCA for dimensionality reduction

from sklearn.decomposition import PCA

pca = PCA(n_components=25)
X = pd.DataFrame(pca.fit_transform(X))
```

Skewness

In [21]:
```
# Checking skewness in features
X.skew().sort_values()
```

Out[21]:
```
3     -0.072931
13    -0.031502
14     0.038063
19     0.070232
9      0.117535
10     0.185770
21     0.208692
15     0.299205
8      0.345126
23     0.399811
5      0.433153
0      0.466926
12     0.584978
2      0.604561
17     0.664396
1      0.704573
18     0.763981
22     0.771283
24     0.898323
6      0.921789
4      1.064561
11     1.122206
7      1.935207
20     2.138213
16     2.264157
dtype: float64
```

Removing skewness

In [22]:
```
# Removing skewness

from sklearn.preprocessing import power_transform
z = power_transform(X[0:])
data_new= pd.DataFrame(z,columns=X.columns)
X = data_new
```

In [23]:
```
# Checking skewness in features

X.skew().sort_values()
```

Out[23]:
```
11   -0.311276
7    -0.249197
24   -0.159901
20   -0.155492
18   -0.114567
22   -0.102067
21   -0.074959
8    -0.071144
16   -0.066046
14   -0.040894
1    -0.006777
3     0.009046
23    0.015258
9     0.027642
5     0.027878
17    0.028487
13    0.031039
6     0.033264
15    0.035356
4     0.036745
10    0.058615
```

EVALUATION OF MODELS 1.) LINEAR REGRESSOR

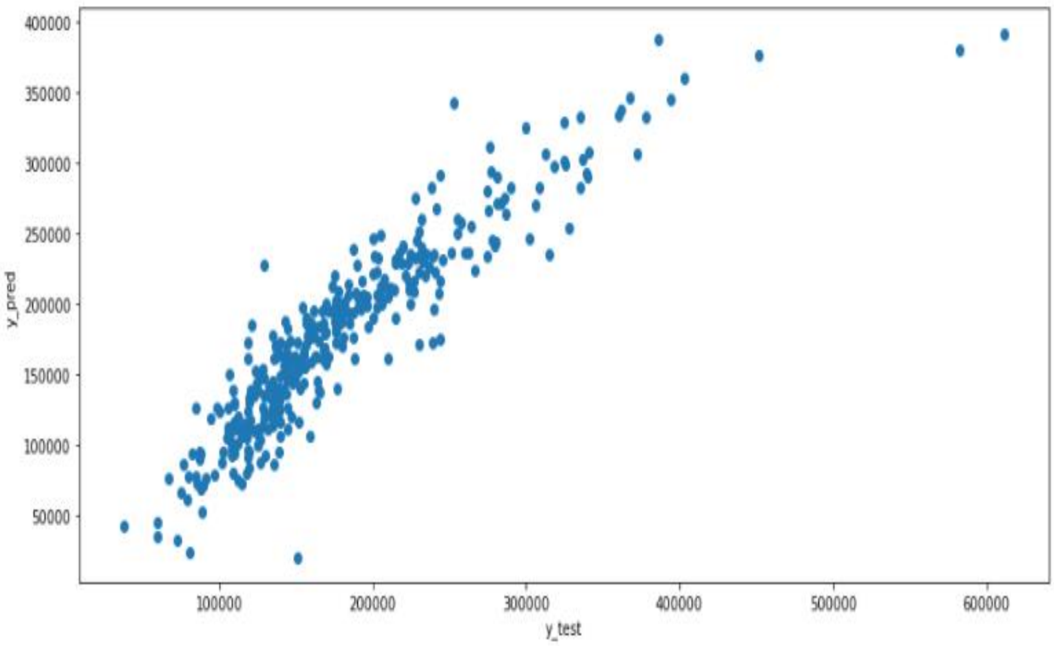
```
#Training model with LinearRegression and finding the best state,r2_score

from sklearn.metrics import mean_squared_error,r2_score
from sklearn.model_selection import train_test_split

model_lr = LinearRegression()

score_s=0
state=0
for i in range(0,25):
    X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state =i)
    model_lr.fit(X_train, y_train)
    y_pred_lr = model_lr.predict(X_test)
    score=r2_score(y_test,y_pred_lr)
    if score>score_s:
        score_s=score
        state=i

print('best random_state : ',state)
print('best r2 score : ',score_s)
```

```
best random_state :  24
best r2 score :  0.8356931962190143
```

In [25]:
```
# finding mean_squared_error,rmse  for LinearRegression

mse=mean_squared_error(y_test,y_pred_lr)
rmse=np.sqrt(mse)

rmse
```

Out[25]: 30812.38577862211

In [26]:
```
# # plotting original training data wth predicted values for LinearRegression model

plt.figure(figsize=(14,6))
plt.scatter(x=y_test,y=y_pred_lr)
plt.xlabel('y_test')
plt.ylabel('y_pred')
plt.show()
```



2.)RANDOMFOREST REGRESSOR

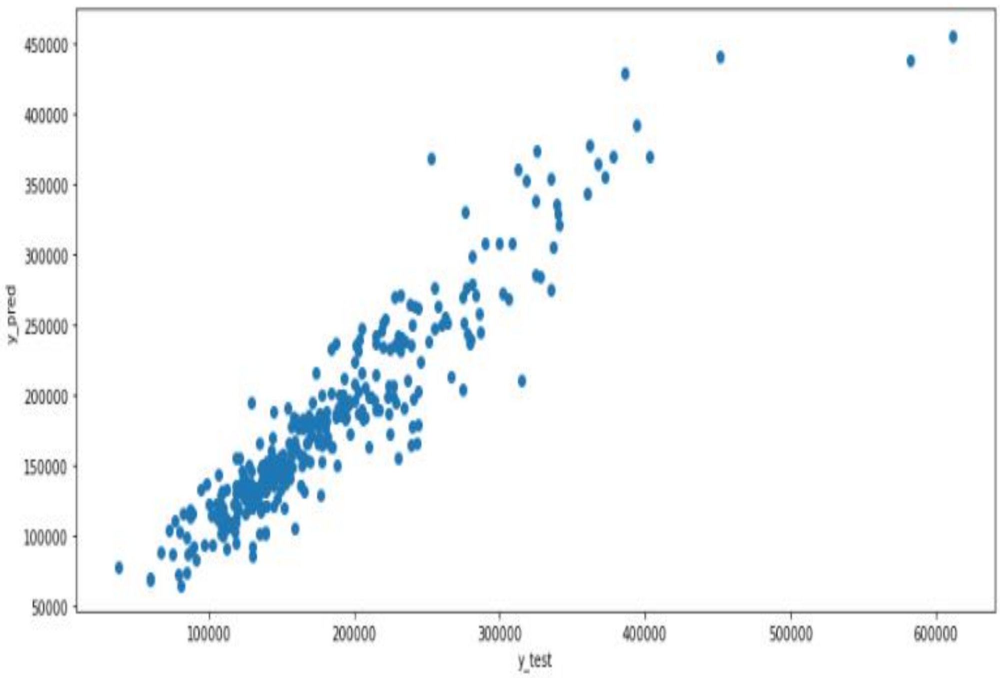
```
#Training model with RandomForestRegressor and finding the best state,r2_score

from sklearn.metrics import mean_squared_error,r2_score
from sklearn.model_selection import train_test_split

model_rfr = RandomForestRegressor()

score_s=0
state=0
for i in range(0,25):
    X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state =i)
    model_rfr.fit(X_train, y_train)
    y_pred_rfr = model_rfr.predict(X_test)
    score=r2_score(y_test,y_pred_rfr)
    if score>score_s:
        score_s=score
        state=i

print('best random_state : ',state)
print('best r2 score : ',score_s)
```

```
best random_state :  24
best r2 score :  0.8783658218810384
```

In [31]:
```
# finding mean_squared_error,rmse  for RandomForestRegressor

mse=mean_squared_error(y_test,y_pred_rfr)
rmse=np.sqrt(mse)

rmse
```

Out[31]: 26510.954565659984

In [32]:
```
# # plotting original training data wth predicted values for RandomForestRegressor model

plt.figure(figsize=(14,6))
plt.scatter(x=y_test,y=y_pred_rfr)
plt.xlabel('y_test')
plt.ylabel('y_pred')
plt.show()
```



3.) DECISION TREE REGRESSOR

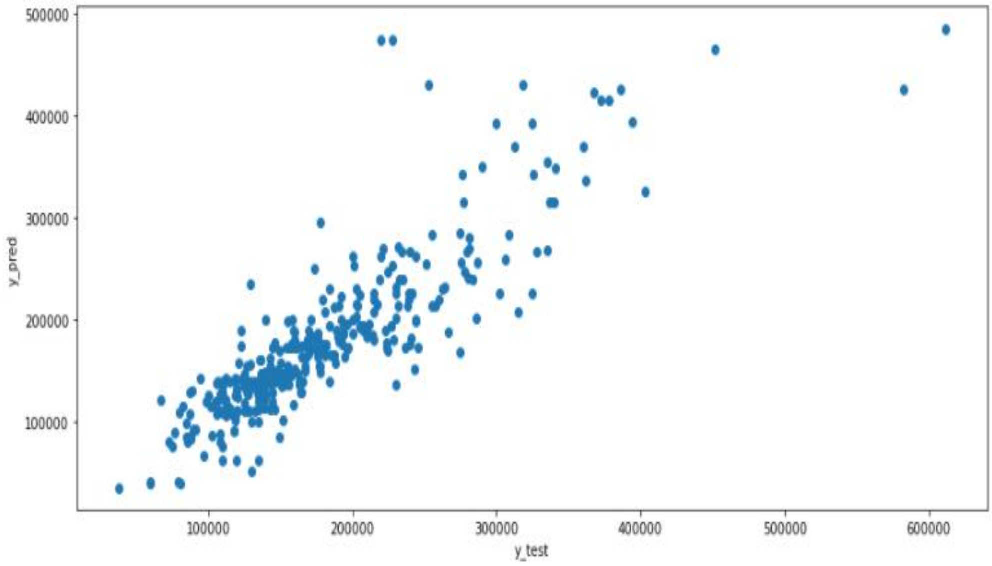
```
#Training model with DecisionTreeRegressor and finding the best state,r2_score

from sklearn.metrics import mean_squared_error,r2_score
from sklearn.model_selection import train_test_split

model_dt = DecisionTreeRegressor()

score_s=0
state=0
for i in range(0,25):
    X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state =i)
    model_dt.fit(X_train, y_train)
    y_pred_dt = model_dt.predict(X_test)
    score=r2_score(y_test,y_pred_dt)
    if score>score_s:
        score_s=score
        state=i

print('best random_state : ',state)
print('best r2 score : ',score_s)
```

```
best random_state :  8
best r2 score :  0.7764959771479828
```

In [37]:
```
# finding mean_squared_error,rmse  for DecisionTreeRegressor

mse=mean_squared_error(y_test,y_pred_dt)
rmse=np.sqrt(mse)

rmse
```

Out[37]: 39917.093547420314

In [38]:
```
# # plotting original training data wth predicted values for DecisionTreeRegressor model

plt.figure(figsize=(14,6))
plt.scatter(x=y_test,y=y_pred_dt)
plt.xlabel('y_test')
plt.ylabel('y_pred')
plt.show()
```



4.) RIDGE

In [27]:
```
# Evaluation of models

#Training model with Ridge (L REGULARIZATION )and finding the best state,r2_score



from sklearn.linear_model import Ridge

model_r=Ridge(alpha=1)
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state =12)
model_r.fit(X_train, y_train)
y_pred_r = model_lr.predict(X_test)
score=r2_score(y_test,y_pred_r)
```

In [28]:
```
print("R2 score for Ridge = " , score)
```

```
R2 score for Ridge =  0.7600311579400052
```

In [29]:
```
# finding mean_squared_error,rmse  for Ridge (L2 regularization)

mse=mean_squared_error(y_test,y_pred_r)
rmse=np.sqrt(mse)

rmse
```

Out[29]:
```
42816.28765342523
```

5.)K-NEIGHBOURS REGRESSOR

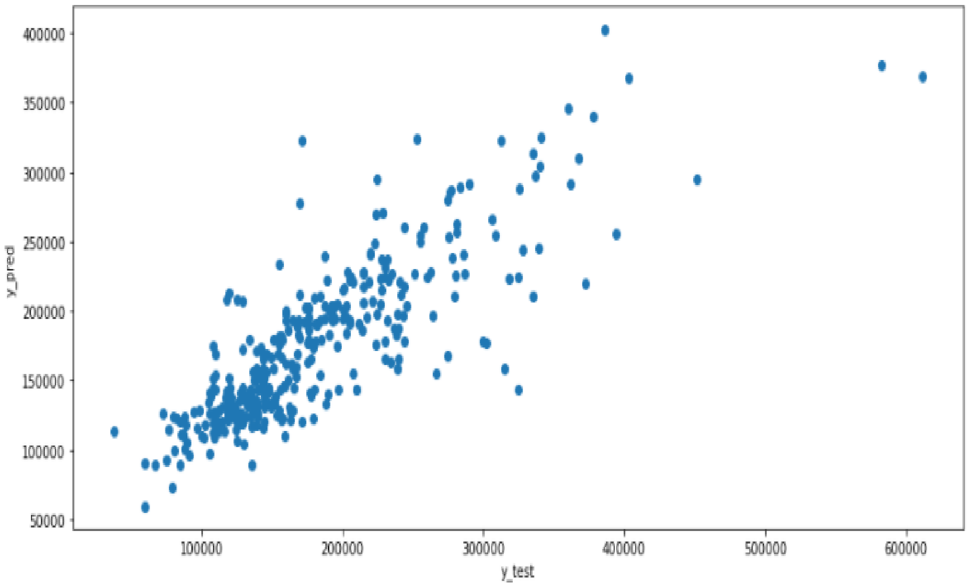
```
#Training model with KNeighborsRegressor and finding the best state,r2_score

from sklearn.metrics import mean_squared_error,r2_score
from sklearn.model_selection import train_test_split

model_knr = KNeighborsRegressor()

score_s=0
state=0
for i in range(0,25):
    X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3, random_state =i)
    model_knr.fit(X_train, y_train)
    y_pred_knr = model_knr.predict(X_test)
    score=r2_score(y_test,y_pred_knr)
    if score>score_s:
        score_s=score
        state=i

print('best random_state : ',state)
print('best r2 score : ',score_s)
```

```
best random_state :  6
best r2 score :  0.6904656665648351
```

In [34]:
```
# finding mean_squared_error,rmse  for KNeighborsRegressor

mse=mean_squared_error(y_test,y_pred_knr)
rmse=np.sqrt(mse)

rmse
```

Out[34]: `43066.88369178601`

In [35]:
```
# # plotting original training data wth predicted values for KNeighborsRegressor model

plt.figure(figsize=(14,6))
plt.scatter(x=y_test,y=y_pred_knr)
plt.xlabel('y_test')
plt.ylabel('y_pred')
plt.show()
```



HYPERPARAMETER TUNING OF

RANDOM FOREST REGRESSOR

In [40]:
```
# HyperParameterTuning with  RandomForestRegressor

from sklearn.model_selection import GridSearchCV
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25, random_state =24)

estimator = RandomForestRegressor()
param_grid = {
            "n_estimators"      : [10,20,30],
            "max_features"      : ["auto", "sqrt", "log2"],
            "min_samples_split" : [2,4,8],
            "bootstrap": [True, False],
            }
```

In [41]:
```
grid = GridSearchCV(estimator, param_grid, n_jobs=-1, cv=5 ,verbose=2)

grid.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 54 candidates, totalling 270 fits
```

Out[41]:
```
GridSearchCV(cv=5, estimator=RandomForestRegressor(), n_jobs=-1,
             param_grid={'bootstrap': [True, False],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_split': [2, 4, 8],
                         'n_estimators': [10, 20, 30]},
             verbose=2)
```

In [42]:
```
grid.best_score_
```

Out[42]:
```
0.8100614196731823
```

HYPERPARAMETER TUNING OF K-NEIHBOURS REGRESSOR

In [48]:
```
# HyperParameterTuning with  KNeighborsRegressor

from sklearn.model_selection import  GridSearchCV, KFold

param_grid = {'n_neighbors': np.arange(1, 12, 2),
              'weights': ['uniform', 'distance']}


knn = KNeighborsRegressor(metric='euclidean')
gscv = GridSearchCV(knn, param_grid, cv=KFold(n_splits=3,
                                              shuffle=True, random_state=0))
gscv.fit(X_train,y_train)
```

Out[48]:
```
GridSearchCV(cv=KFold(n_splits=3, random_state=0, shuffle=True),
             estimator=KNeighborsRegressor(metric='euclidean'),
             param_grid={'n_neighbors': array([ 1,  3,  5,  7,  9, 11]),
                         'weights': ['uniform', 'distance']})
```

In [49]:
```
gscv.best_params_
```

Out[49]:
```
{'n_neighbors': 9, 'weights': 'distance'}
```

In [50]:
```

print("Best score", gscv.best_score_)
```

```
Best score 0.5938074083872137
```

In [51]:
```
gscv.score(X_test,y_test)
```

Out[51]:
```
0.7074471231779311
```

IMPORTING OF MODEL

In [53]:
```
# Clearly after hyperparameter tuning, RandomForestRegressor performs better than KNeighborsRegressor
```

In [54]:
```
# Exporting the model through pickle

import pickle
filename='HousingSalePricePred.pkl'
pickle.dump(grid,open(filename,'wb'))
```

IMPORTING TEST DATA

In [56]:
```
# Appying RandomForestRegressor grid model after hyperparameter tuning on test data

# Importing test data
```

In [57]:
```
df_test=pd.read_csv(r'C:\ProgramData\test.csv')
df_test.head()
```

Out[57]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	...	ScreenPorch	PoolArea	PoolQC	Fence	MiscFeatur
0	337	20	RL	86.0	14157	Pave	NaN	IR1	HLS	AllPub	...	0	0	NaN	NaN	Na
1	1018	120	RL	NaN	5814	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	Na
2	929	20	RL	NaN	11838	Pave	NaN	Reg	Lvl	AllPub	...	0	0	NaN	NaN	Na
3	1148	70	RL	75.0	12000	Pave	NaN	Reg	Bnk	AllPub	...	0	0	NaN	NaN	Na
4	1227	60	RL	86.0	14598	Pave	NaN	IR1	Lvl	AllPub	...	0	0	NaN	NaN	Na

5 rows × 80 columns

In [69]:
```python
# Applying RandomForestRegressor model after Hyperparameter tuning on test data to get Sale price

y_pred_rfr_test = grid.predict(X_test)
```

In [70]:
```python
print("Final sale price of test data : ",y_pred_rfr_test)
```

```
Final sale price of test data :  [518054.3  266368.05 469493.1  107170.   402482.7   90997.7  114262.5
 498317.2  456740.55 273833.35  73270.   113190.    77940.   551323.5
 513783.2  100803.75  94115.6  111995.   262630.1  212489.75  97287.15
 138423.   108917.5   61399.65  97258.8  100135.4  232592.4  117432.05
 171920.    98346.65  96570.2  397023.35 456883.7  175291.85 104795.
 203533.5  293603.85  91467.7  101102.15  99832.7  102525.   394220.
 448508.3  399218.4  131555.55 100029.15 105012.5   78105.   479160.05
 516651.9  110504.15 546048.65  73655.55  96900.   411393.2   88990.
 100435.4  406350.6   64460.2  477498.25  87103.25 255109.75 109290.
 205923.4  510036.15  90592.5  188977.1  385228.85 113484.6  135186.95
 405085.85 131642.5  102034.15 119545.9  124300.   391714.45 412449.5
 374328.6  457636.9  135150.   489623.85 105990.   173245.   163127.5
 248366.25 563399.95 102171.3  460203.85 114372.9  269340.   452377.8
  94032.5   94154.15  96797.7  257646.75 236743.95 466230.25 219360.15
 428051.3   94125.25 440984.25  81553.25  88401.1  206651.15 305788.85
  97300.8  355242.   194695.   470075.2  243127.5  295728.85 162530.
 108956.65 419464.6  106080.    98574.15  93789.15 331743.7  150239.25
 107140.    90592.5  487496.45 293800.   108838.1  104595.   159877.5
 107220.   230574.35  97339.35  64780.75 120261.25 383784.25  98002.9
 183577.5   87710.95 485803.45 524713.7   85143.05 431144.75  99895.
  96280.55 397794.75  92419.75 471160.25 206801.85 395547.45  93994.7
  98765.    95142.9  393487.1  279752.4  104737.5  389742.45  96637.9
  94705.   190160.   322164.55 260940.    91743.45 240640.9  534468.45
  98222.9  482277.65 106770.    72370.55 531992.65 286650.   400407.2
```

CONCLUSION

Clearly we can see that the actual value and predicted values are very close to each other,Hence RandomForestRegressor is a good choice for predicting Housing Sale price

- Data Sources and their formats

  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 data is provided in the CSV file.

- Data Preprocessing Done

  What were the steps followed for the cleaning of the data? What were the assumptions done and what were the next actions steps over that?

- Data Inputs- Logic- Output Relationships

  Describe the relationship behind the data input, its format, the logic in between and the output. Describe how the input affects the output.

- 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.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  Describe the approaches you followed, both statistical and analytical, for solving of this problem.

- Testing of Identified Approaches (Algorithms)

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

- 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.
- 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

  Mention all the plots made along with their pictures and what were the inferences and observations obtained from those. Describe them in detail.

  If different platforms were used, mention that as well.

- 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

  Describe the key findings, inferences, observations from the whole problem.

- Learning Outcomes of the Study in respect of Data Science

List down your learnings obtained about the power of visualization, data cleaning and various algorithms used. You can describe which algorithm works best in which situation and what challenges you faced while working on this project and how did you overcome that.

- 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.