Importing the Essential Libraries, Metrics

In [1]:

```
import pandas as pd
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
```

Loading the Data

In [2]:

```
dataset=pd.read_csv(r"C:\Users\admin\Desktop\Machine learning\All datasets\HousePrice.csv
dataset
```

Out[2]:

	POSTED_BY	UNDER_CONSTRUCTION	RERA	BHK_NO.	BHK_OR_RK	SQUARE_FT	R
0	Owner	0	0	2	BHK	1300.236407	
1	Dealer	0	0	2	BHK	1275.000000	
2	Owner	0	0	2	BHK	933.159722	
3	Owner	0	1	2	BHK	929.921143	
4	Dealer	1	0	2	BHK	999.009247	
...	...	...	...	...	...	...	
29446	Owner	0	0	3	BHK	2500.000000	
29447	Owner	0	0	2	BHK	769.230769	
29448	Dealer	0	0	2	BHK	1022.641509	
29449	Owner	0	0	2	BHK	927.079009	
29450	Dealer	0	1	2	BHK	896.774194	

29451 rows × 12 columns

Exploratory Data Analysis

Taking a look at the first 5 rows of the dataset

In [3]:

```
dataset.head()
```

Out[3]:

	POSTED_BY	UNDER_CONSTRUCTION	RERA	BHK_NO.	BHK_OR_RK	SQUARE_FT	READ
0	Owner	0	0	2	BHK	1300.236407	
1	Dealer	0	0	2	BHK	1275.000000	
2	Owner	0	0	2	BHK	933.159722	
3	Owner	0	1	2	BHK	929.921143	
4	Dealer	1	0	2	BHK	999.009247	

Checking the shape—i.e. size—of the data

In [4]:

```
dataset.shape
```

Out[4]:

```
(29451, 12)
```

Learning the dtypes of columns' and how many non-null values are there in those columns

In [5]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29451 entries, 0 to 29450
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   POSTED_BY              29451 non-null  object 
 1   UNDER_CONSTRUCTION     29451 non-null  int64  
 2   RERA                   29451 non-null  int64  
 3   BHK_NO.                29451 non-null  int64  
 4   BHK_OR_RK              29451 non-null  object 
 5   SQUARE_FT              29451 non-null  float64
 6   READY_TO_MOVE          29451 non-null  int64  
 7   RESALE                 29451 non-null  int64  
 8   ADDRESS                29451 non-null  object 
 9   LONGITUDE              29451 non-null  float64
 10  LATITUDE               29451 non-null  float64
 11  TARGET(PRICE_IN_LACS)  29451 non-null  float64
dtypes: float64(4), int64(5), object(3)
memory usage: 2.7+ MB
```

Delete the unrealeted columan in the dataset

In [6]:

```
del dataset["BHK_OR_RK"]
del dataset["ADDRESS"]
```

Encoding the categorical features in dataset by using Label Encoding method

In [7]:

```
a=LabelEncoder()
dataset["POSTED_BY"]=a.fit_transform(dataset["POSTED_BY"])
```

In [8]:

```
dataset.head()
```

Out[8]:

	POSTED_BY	UNDER_CONSTRUCTION	RERA	BHK_NO.	SQUARE_FT	READY_TO_MOVE
0	2	0	0	2	1300.236407	1
1	1	0	0	2	1275.000000	1
2	2	0	0	2	933.159722	1
3	2	0	1	2	929.921143	1
4	1	1	0	2	999.009247	0

Getting the statistical summary of dataset

In [9]:

```
dataset.describe()
```

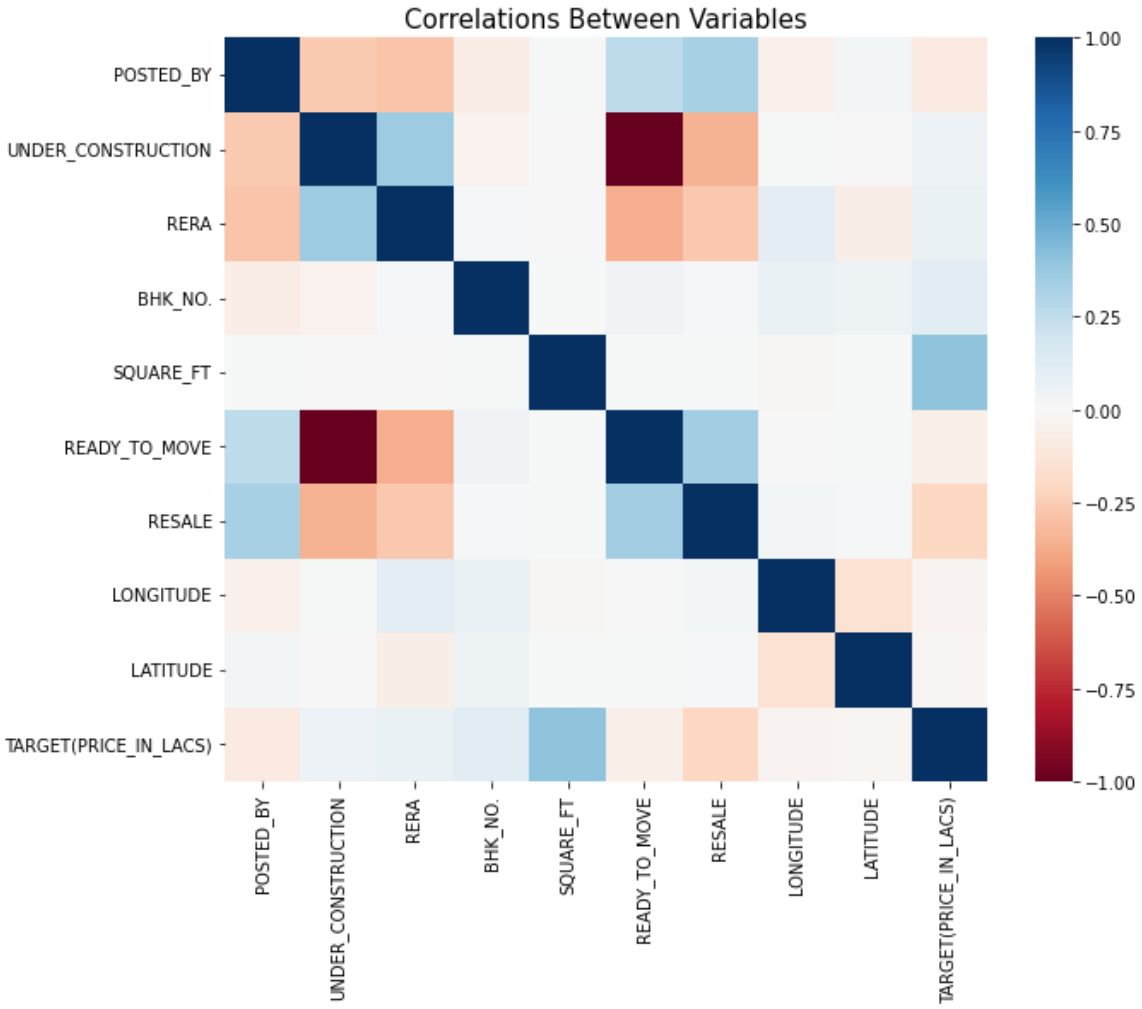
Out[9]:

	POSTED_BY	UNDER_CONSTRUCTION	RERA	BHK_NO.	SQUARE_FT	RE/
count	29451.000000	29451.000000	29451.000000	29451.000000	2.945100e+04	
mean	1.336695	0.179756	0.317918	2.392279	1.980217e+04	
std	0.515345	0.383991	0.465675	0.879091	1.901335e+06	
min	0.000000	0.000000	0.000000	1.000000	3.000000e+00	
25%	1.000000	0.000000	0.000000	2.000000	9.000211e+02	
50%	1.000000	0.000000	0.000000	2.000000	1.175057e+03	
75%	2.000000	0.000000	1.000000	3.000000	1.550688e+03	
max	2.000000	1.000000	1.000000	20.000000	2.545455e+08	

Visualizing the correlations between numerical variables

In [10]:

```
plt.figure(figsize=(10,8))
sns.heatmap(dataset.corr(), cmap="RdBu")
plt.title("Correlations Between Variables", size=15)
plt.show()
```



Feature Selection

In [11]:

```
x=dataset.iloc[:,:-1].values
y=dataset.iloc[:,-1].values
```

Standardizing the Data

Standardizing the numerical columns in X dataset.

StandardScaler() adjusts the mean of the features as 0 and standard deviation of features as 1.

Formula that StandardScaler() uses is as follows:

In [12]:

```
sc=StandardScaler()
x=sc.fit_transform(x)
x=pd.DataFrame(x)
```

As you can see, standardization is done successfully

In [13]:

```
x.head()
```

Out[13]:

	0	1	2	3	4	5	6	7	
0	1.287132	-0.468134	-0.682715	-0.44624	-0.009731	0.468134	0.27524	-1.342478	0.07201
1	-0.653350	-0.468134	-0.682715	-0.44624	-0.009744	0.468134	0.27524	-1.454541	-0.01828
2	1.287132	-0.468134	-0.682715	-0.44624	-0.009924	0.468134	0.27524	-1.373400	0.07525
3	1.287132	-0.468134	1.464741	-0.44624	-0.009926	0.468134	0.27524	1.183208	0.04800
4	-0.653350	2.136139	-0.682715	-0.44624	-0.009890	-2.136139	0.27524	0.208204	1.10321

Train-Test Split

Splitting the data into Train and Test chunks for better evaluation

In [14]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=20)
```

Defining several evaluation functions for convenience

In [15]:

```
def rmse_cv(model):
    rmse = np.sqrt(-cross_val_score(model, x, y, scoring="neg_mean_squared_error", cv=5))
    return rmse



def evaluation(y, predictions):
    mae = mean_absolute_error(y, predictions)
    mse = mean_squared_error(y, predictions)
    rmse = np.sqrt(mean_squared_error(y, predictions))
    r_squared = r2_score(y, predictions)
    return mae, mse, rmse, r_squared
```

Machine Learning Models

In [16]:

```
models= pd.DataFrame(columns=["Model","MAE","MSE","RMSE","R2 Score","RMSE (Cross-Validati
```

Linear Regression

In [17]:

```
regressor=LinearRegression()
regressor.fit(x_train,y_train)
```

Out[17]:

```
LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [18]:

```
predictions=regressor.predict(x_test)
```

In [19]:

```
mae, mse, rmse, r_squared = evaluation(y_test, predictions)
print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R2 Score:", r_squared)
print("-"*30)
rmse_cross_val = rmse_cv(regressor)
print("RMSE Cross-Validation:", rmse_cross_val)

new_row = {"Model": "LinearRegression","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score":
models = models.append(new_row, ignore_index=True)
```

```
MAE: 144.4814040612618
MSE: 417777.34174899437
RMSE: 646.3569770250758
R2 Score: 0.1937304007197146
------------------------------
RMSE Cross-Validation: 652.3291941271401

C:\Users\admin\AppData\Local\Temp\ipykernel_13008\868320567.py:11: FutureW
arning: The frame.append method is deprecated and will be removed from pan
das in a future version. Use pandas.concat instead.
  models = models.append(new_row, ignore_index=True)
```

Ridge Regression

In [20]:

```
ridge = Ridge()
ridge.fit(x_train, y_train)
predictions = ridge.predict(x_test)

mae, mse, rmse, r_squared = evaluation(y_test, predictions)
print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R2 Score:", r_squared)
print("-"*30)
rmse_cross_val = rmse_cv(ridge)
print("RMSE Cross-Validation:", rmse_cross_val)


new_row = {"Model": "Ridge","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r_squared,
models = models.append(new_row, ignore_index=True)
```

```
MAE: 144.47922818613455
MSE: 417779.54506594234
RMSE: 646.3586814346523
R2 Score: 0.19372614853249626
------------------------------
RMSE Cross-Validation: 652.2166246651173

C:\Users\admin\AppData\Local\Temp\ipykernel_13008\4070779089.py:16: Future
Warning: The frame.append method is deprecated and will be removed from pa
ndas in a future version. Use pandas.concat instead.
  models = models.append(new_row, ignore_index=True)
```

Lasso Regression

In [21]:

```
lasso = Lasso()
lasso.fit(x_train, y_train)
predictions = lasso.predict(x_test)

mae, mse, rmse, r_squared = evaluation(y_test, predictions)
print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R2 Score:", r_squared)
print("-"*30)
rmse_cross_val = rmse_cv(lasso)
print("RMSE Cross-Validation:", rmse_cross_val)


new_row = {"Model": "Lasso","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r_squared,
models = models.append(new_row, ignore_index=True)
```

```
MAE: 143.76633610706787
MSE: 418026.05439100927
RMSE: 646.5493441269655
R2 Score: 0.19325040953260564
------------------------------
RMSE Cross-Validation: 648.0527929667207

C:\Users\admin\AppData\Local\Temp\ipykernel_13008\1908560978.py:16: Future
Warning: The frame.append method is deprecated and will be removed from pa
ndas in a future version. Use pandas.concat instead.
  models = models.append(new_row, ignore_index=True)
```

Random Forest Regressor

In [22]:

```
random_forest = RandomForestRegressor(n_estimators=100)
random_forest.fit(x_train, y_train)
predictions = random_forest.predict(x_test)

mae, mse, rmse, r_squared = evaluation(y_test, predictions)
print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R2 Score:", r_squared)
print("-"*30)
rmse_cross_val = rmse_cv(random_forest)
print("RMSE Cross-Validation:", rmse_cross_val)


new_row = {"Model": "RandomForestRegressor","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Sco
models = models.append(new_row, ignore_index=True)
```

```
MAE: 36.289863941832905
MSE: 120194.68584913852
RMSE: 346.69105245036036
R2 Score: 0.7680359571691924
------------------------------
RMSE Cross-Validation: 200.65188580140145

C:\Users\admin\AppData\Local\Temp\ipykernel_13008\2222580869.py:16: Future
Warning: The frame.append method is deprecated and will be removed from pa
ndas in a future version. Use pandas.concat instead.
  models = models.append(new_row, ignore_index=True)
```

XGBoost Regressor

In [23]:

```
xgb = XGBRegressor(n_estimators=1000, learning_rate=0.01)
xgb.fit(x_train, y_train)
predictions = xgb.predict(x_test)

mae, mse, rmse, r_squared = evaluation(y_test, predictions)
print("MAE:", mae)
print("MSE:", mse)
print("RMSE:", rmse)
print("R2 Score:", r_squared)
print("-"*30)
rmse_cross_val = rmse_cv(xgb)
print("RMSE Cross-Validation:", rmse_cross_val)

new_row = {"Model": "XGBRegressor","MAE": mae, "MSE": mse, "RMSE": rmse, "R2 Score": r_sq
models = models.append(new_row, ignore_index=True)
```

```
MAE: 38.89211450601874
MSE: 116533.03347031109
RMSE: 341.3693505139427
R2 Score: 0.7751025898013533
------------------------------
RMSE Cross-Validation: 174.76303034140454

C:\Users\admin\AppData\Local\Temp\ipykernel_13008\936592420.py:15: FutureW
arning: The frame.append method is deprecated and will be removed from pan
das in a future version. Use pandas.concat instead.
  models = models.append(new_row, ignore_index=True)
```

Model Comparison

The less the Root Mean Squared Error (RMSE), The better the model is.

In [24]:

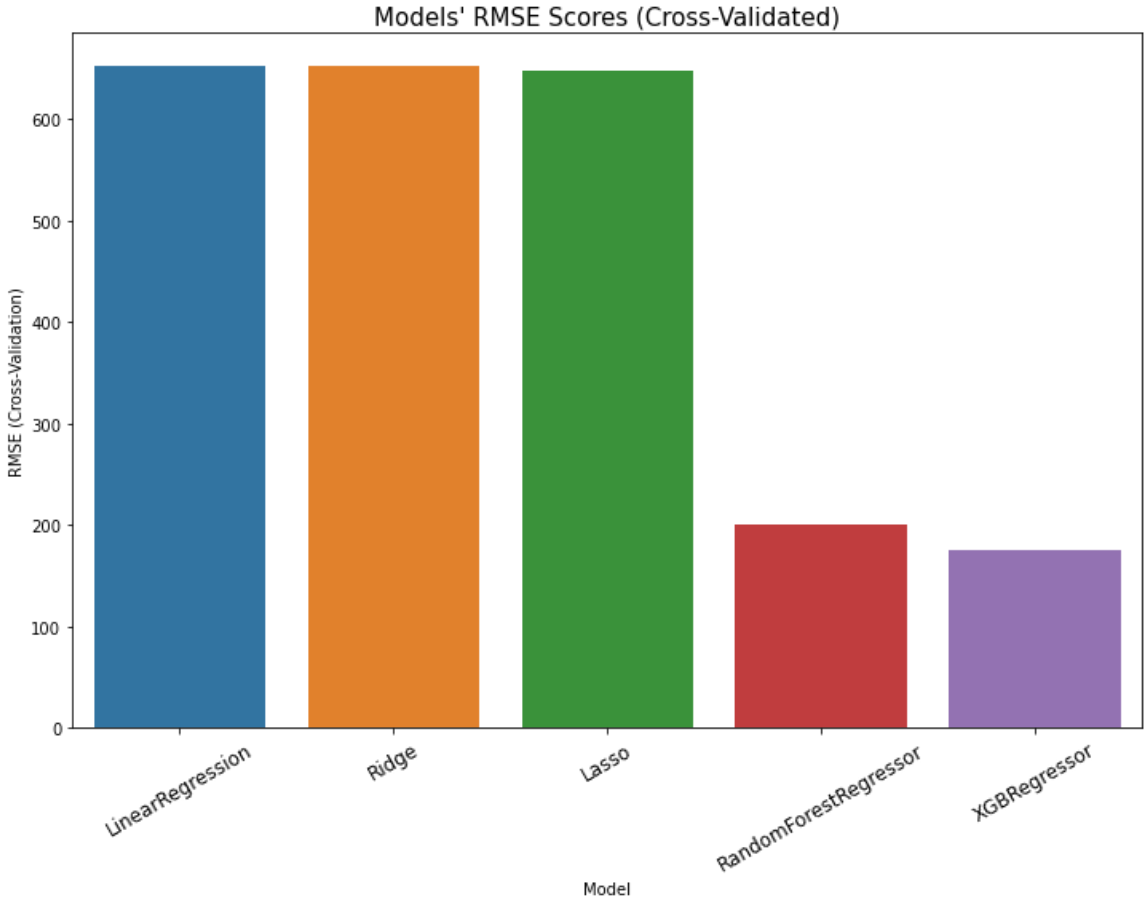
```
models.sort_values(by="RMSE (Cross-Validation)")
```

Out[24]:

	Model	MAE	MSE	RMSE	R2 Score	RMSE (Cross-Validation)
4	XGBRegressor	38.892115	116533.03347	341.369351	0.775103	174.76303
3	RandomForestRegressor	36.289864	120194.685849	346.691052	0.768036	200.651886
2	Lasso	143.766336	418026.054391	646.549344	0.19325	648.052793
1	Ridge	144.479228	417779.545066	646.358681	0.193726	652.216625
0	LinearRegression	144.481404	417777.341749	646.356977	0.19373	652.329194

In [25]:

```
plt.figure(figsize=(12,8))
sns.barplot(x=models["Model"], y=models["RMSE (Cross-Validation)"])
plt.title("Models' RMSE Scores (Cross-Validated)", size=15)
plt.xticks(rotation=30, size=12)
plt.show()
```



In [ ]:

```
```