Housing Price Prediction Project

- **Submitted by Smriti Mathur**

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?

Conceptual Background:

The project will call for expertise and experience in creating graphs and plots, analysing them to determine the link between datasets, and knowing various learning models to construct and anticipate the desired result. Basic data science principles to improve the dataset's quality and knowledge of Python (a coding language) will be utilised to accomplish the Micro Credit Defaulter project. Building an appropriate model for this assignment requires an understanding of how to calculate the F2 score, accuracy, skewness, and fundamental mathematical and statistical concepts.

Review of literature:

Market value is the amount that a buyer who is eager, prepared, and pre-approved by a bank will pay for a property and that a seller will accept. The market price that is set by the transaction has an impact on the market value of subsequent transactions. Local supply and demand, the property's condition, and recent sales of comparable properties are used to calculate the price, which does not include the value component.

The market value of a property is an estimate of what it would sell for in a competitive market based on its characteristics and benefits, the situation of the wider real estate market, supply and demand, and the prices at which comparable properties in the same condition have recently sold.

The main distinction between market value and market price is that the former may, in the seller's opinion, be far higher than what a buyer is willing to pay for the asset or its actual market worth. Demand may be sparked by value, which then affects price. But value alone cannot affect price in the absence of the demand function. Price declines when supply rises and demand falls, and value has no bearing. Price will grow as supply declines and demand rises, and value will affect price. In a balanced market, market value and market price may be identical.

Therefore, value might be seen differently by buyers and sellers. A seller could consider an in-ground pool to be a plus, but a potential buyer might see it negatively and consider the home to be worth less. Or the seller may believe that the brand-new roof they installed on the home has tremendous value, but the buyer may not agree because they assume the property to have a roof in good shape. Or a builder may believe his work is of a greater calibre and ask for a higher price, but the buyer may place more emphasis on the property's lot, neighbourhood, and floor plan than on the builder's calibre.

Data Sources:

The dataset has only been made available for academic purposes, not for any commercial use, by the FlipRobo technologies. With 1460 items, the dataset describes housing-related statistics. The dataset, which includes train and test data, is in csv format. This dataset, which consists of 1460 observations and 80 characteristics for model prediction, is to be used for simple data analysis. The dataset contains both category and numerical data.

Steps Undertaken For Predicting The Housing Price:

- Importing the required packages into our python environment
- Importing the house price data and do some EDA on it
- Data Visualization on the house price data
- Feature Selection & Data Split
- Modelling the data using the algorithms
- Evaluating the built model using the evaluation metrics

Data Pre-Processing and Transformation:

The dataset was obtained from FlipRobo Technologies and contains information about housing for a US-based client company that is interested in entering the Australian market using the dataset model.

After loading the extracted CSV file, the data's structure was examined in comparison to the provided meta data. It was found that some variables were numerical rather than factors, so those were changed. The dataset's ID column was removed because the prediction doesn't require it. Because the explanation in the metadata states that where there is a missing value, it is because that feature doesn't exist for that house, missing values in the numerical variables were replaced by the mean of the column while missing values in factor variables were replaced with 'No_'. By accurately representing each level in the factor columns as it appears in the meta data, specificity was achieved. The dependent variable's standard deviation was calculated, and the results indicated that the independent variables aren't too far from the sale price's mean. It demonstrates that the dataset's values are normally distributed. Multicollinearity was used to check for the correlation between the independent variables; as checked, there is no obvious outlier to be concerned about.

Libraries and Packages required:

Pandas will be used mostly for data processing, NumPy will be used to work with arrays, Matplotlib and Seaborn will be used to visualise the data, and Scikit-Learn will be used to create and test our machine learning model.

```
In [2]: ### Importing the set of libraries here.

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')

import sklearn
from sklearn.preprocessing import power_transform
from sklearn.linear_model import LogisticRegression, LinearRegression, Lasso, Ridge, ElasticNet
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, r2_score, mean_squared_error, mean_absol

from imblearn.over_sampling import SMOTE
```

Thereafter, data is imported with the function “pd.read_csv(“”)”.

Exploratory Data Analysis:

The dataset investigation that led to the usage of the method is illustrated in this section. The following are the questions that were looked into for this project; I'll just present a few of the visuals here for the purpose of writing, but I'll offer the scripts that display the entire visualisation of all the questions looked into.

- Does the sale price and age of the building have a significant relationship? It was utilised to test for this, and the results show a correlation between the building's age and the price it sold for.
- What is the typical sale price taking into account the general state of the home, the year it was constructed, condition1 - closeness to social amenities, and sale condition? We have levels 1 through 9, with 1 being the lowest and 9 being the highest, and the

average of each level is displayed. The output is contained in the code given for the others.

- What is the distribution of the sale price dependent on the house's overall quality?
- Which kind of homes (according to the year they were built) sell for the most money?
- Here, it was clear that homes sold for more than $700,000 in sales price compared to the month they were purchased. From this, we learned that the winter sees the largest rises in home prices, followed by the spring and summer.
- What kind of sale has the most expensive sale price? There are nine main sales type categories that were compared to the selling price.
- Seasonal price dispersion and percentage of persons buying homes at less than $200,000 are depicted in bar charts.
- How much will it cost for consumers to buy more even with a garage attached? The density of the garage type peaks at a claim size of roughly 160k$. It reveals that as long as a garage is linked to the home, buyers are likely to purchase it at that time, regardless of the asking price. In comparison to autumn and winter, the chances of people buying homes are higher in the summer and spring.

Missing null values handled:

In [15]:
```
### Dropping the variables that have majority of the null values here.
df.drop(['MiscFeature','PoolQC','Fence','FireplaceQu','Alley','Id'],inplace=True,axis=1)
```

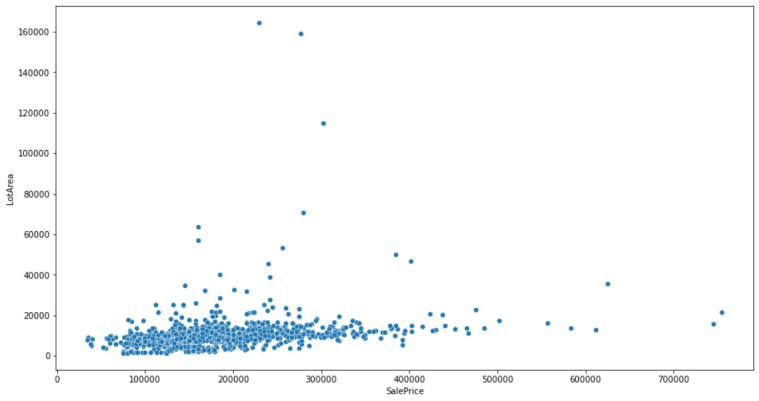
In [16]:
```
### Now replacing and filling in the null values with mode, mean of their respective distribution.
df['BsmtQual'].fillna(df['BsmtQual'].mode()[0].strip(),inplace=True)
df['BsmtCond'].fillna(df['BsmtCond'].mode()[0].strip(),inplace=True)
df['BsmtExposure'].fillna(df['BsmtExposure'].mode()[0].strip(),inplace=True)
df['BsmtFinType1'].fillna(df['BsmtFinType1'].mode()[0].strip(),inplace=True)
df['BsmtFinType2'].fillna(df['BsmtFinType2'].mode()[0].strip(),inplace=True)
df['GarageType'].fillna(df['GarageType'].mode()[0].strip(),inplace=True)
df['GarageFinish'].fillna(df['GarageFinish'].mode()[0].strip(),inplace=True)
df['GarageQual'].fillna(df['GarageQual'].mode()[0].strip(),inplace=True)
df['GarageCond'].fillna(df['GarageCond'].mode()[0].strip(),inplace=True)
df['LotFrontage'].fillna(df['LotFrontage'].median(),inplace=True)
df['GarageYrBlt'].fillna(df['GarageYrBlt'].median(),inplace=True)
df['MasVnrArea'].fillna(df['MasVnrArea'].median(),inplace=True)
print("All null values are now taken care of, either filled or dropped as per the significance it has with target variable.")
```

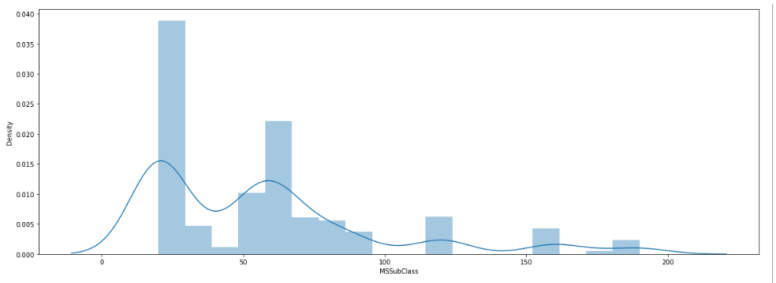
```
All null values are now taken care of, either filled or dropped as per the significance it has with target variable.
```

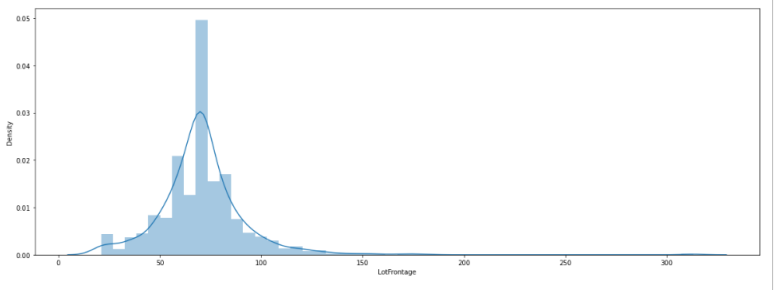
Data Visualisation:

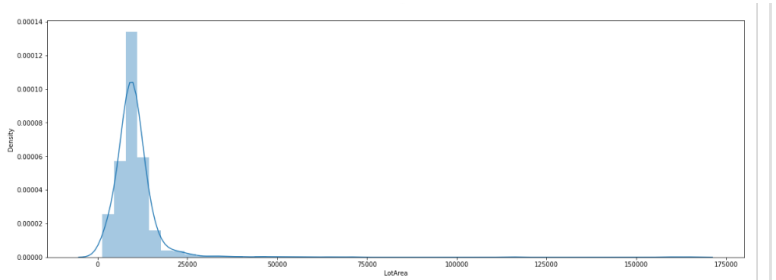
In [26]:
```
### Looking at the individual target variable's distribution now.
plt.figure(figsize=[15,8])
sns.scatterplot(x='SalePrice',y='LotArea',data=df)
```

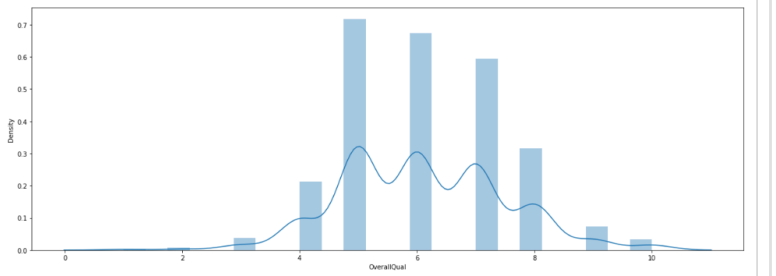
Out[26]: <AxesSubplot:xlabel='SalePrice', ylabel='LotArea'>





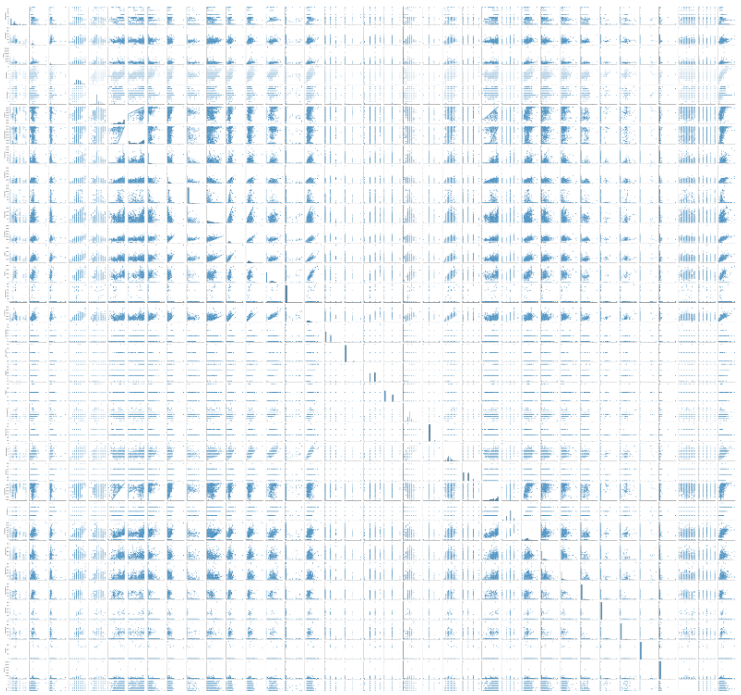






Data Distribution:

```
In [24]: sns.pairplot(df)
Out[24]: <seaborn.axisgrid.PairGrid at 0x161559d8d30>
```



Model Development and Evaluation:

- Identification of potential solutions to an issue (methods)
- It is necessary to identify the variables that may affect the cost of housing.
- Analysing the numerous variables and the respondent's preferred shop can help with this. This will be accomplished by examining the effects of each component on respondents' decision-making.

Using Label Encoding method and Interquartile method and applying MinMax scaling to remove the skewness and outliers.

```
In [33]:
### Label  Encoding the variables now.

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for col in categorical_columns:
    df[col] =  le.fit_transform(df[col])
```

```
In [35]:
### Removing the outliers from the dataset.

for i in continous_columns:
    IQR = df[i].quantile(0.75)-df[i].quantile(0.25)
    bmax =  df[i].quantile(0.75) + 1.5*IQR
    bmin =  df[i].quantile(0.25) - 1.5*IQR
    df.loc[df[i]>bmax,i] = bmax
    df.loc[df[i]<bmin,i] = bmin
```

```
In [36]:
### Performing the MinMax Scaling
from sklearn.preprocessing import MinMaxScaler

final  = df.drop(categorical_columns,axis=1)
final = final.drop('SalePrice',axis=1)
scale = MinMaxScaler()
minmax = scale.fit_transform(final)
final = pd.DataFrame(minmax, columns = final.columns)
```

```
In [38]:
### Bringing the target variable in the scaled dataframe.
final_new = power_transform(final)
final_new = pd.DataFrame(final_new,columns=final.columns)
final_new= final_new.join(df['SalePrice'])
for i in categorical_columns:
    final_new=final_new.join(df[i])
```

Splitting the dataset now:

Splitting the dataset:

In [39]:
```
x = final_new.drop('SalePrice',axis=1)
y = final_new['SalePrice']
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.30,random_state=33)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(817, 74)
(351, 74)
(817,)
(351,)
```

Building the model instances now:

1. Linear Regression

In [40]:
```
lr = LinearRegression()
lr.fit(x_train, y_train)
pred = lr.predict(x_test)
print('R2 score',r2_score(y_test, pred))
print('MAE:', mean_absolute_error(y_test, pred))
print('MSE:', mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, pred)))
```

```
R2 score 0.9122522295050699
MAE: 14889.955766381772
MSE: 371736028.123931
RMSE: 19280.45715547043
```

2. Ridge Regression

In [41]:
```
rr = Ridge(alpha=0.01)
rr.fit(x_train, y_train)
pred = rr.predict(x_test)
print('R2 score',r2_score(y_test, pred))
print('MAE:', mean_absolute_error(y_test, pred))
print('MSE:', mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, pred)))
```

```
R2 score 0.911823459114965
MAE: 14934.890606094272
MSE: 373552477.7145672
RMSE: 19327.505729259716
```

3. Lasso Regression

In [42]:
```
lasso = Lasso(alpha=0.01)
lasso.fit(x_train, y_train)
pred = lasso.predict(x_test)
print('R2 score',r2_score(y_test, pred))
print('MAE:', mean_absolute_error(y_test, pred))
print('MSE:', mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, pred)))
```

```
R2 score 0.9118234658159894
MAE: 14935.104867786691
MSE: 373552449.3262428
RMSE: 19327.504994857532
```

4. ElasticNet Regression

In [43]:
```
enet = ElasticNet(alpha = 0.01)
enet.fit(x_train, y_train)
pred = enet.predict(x_test)
print('R2 score',r2_score(y_test, pred))
print('MAE:', mean_absolute_error(y_test, pred))
print('MSE:', mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, pred)))
```

```
R2 score 0.9119966779517421
MAE: 14928.021981156287
MSE: 372818650.72367525
RMSE: 19308.512390230255
```

5. Decision Tree Regressor and Random Forest Regressor

In [44]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [45]:
```
dtr=DecisionTreeRegressor()
dtr.fit(x_train,y_train)
pred = dtr.predict(x_test)
print('R2 score', r2_score(y_test, pred))
print('MAE:', mean_absolute_error(y_test, pred))
print('MSE:', mean_squared_error(y_test, pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, pred)))
```

```
R2 score 0.7717072567318615
MAE: 23041.418803418805
MSE: 967142950.2236468
RMSE: 31098.922010636426
```

In [46]:
```
from sklearn.ensemble import RandomForestRegressor
rdr = RandomForestRegressor()
rdr.fit(x_train,y_train)
pred1=rdr.predict(x_test)
print('R2 score',r2_score(y_test, pred1))
print('MAE:', mean_absolute_error(y_test, pred1))
print('MSE:', mean_squared_error(y_test, pred1))
print('RMSE:', np.sqrt(mean_squared_error(y_test,pred1)))
```

```
R2 score 0.8903397061885381
MAE: 15678.91819088319
MSE: 464566584.8197414
RMSE: 21553.80673616012
```

Performing the cross-validation on the aforementioned models:

Performing the cross validation score on the basis of R2 Score models got:

In [47]:
```
print("Cross validation score of Linear Regression:",cross_val_score(lr,x,y,cv=5).mean())
print("Cross validation score of Ridge Regressor:",cross_val_score(rr,x,y,cv=5).mean())
print("Cross validation score of Lasso Regressor:",cross_val_score(lasso,x,y,cv=5).mean())
print("Cross validation score of ElasticNet Regressor:",cross_val_score(enet,x,y,cv=5).mean())
print("Cross validation score fo Decision Tree Regressor:",cross_val_score(dtr,x,y,cv=5).mean())
print("Cross validation score of Random Forest Regressor:",cross_val_score(rdr,x,y,cv=5).mean())
```

```
Cross validation score of Linear Regression: 0.8588006444728107
Cross validation score of Ridge Regressor: 0.8596712295672658
Cross validation score of Lasso Regressor: 0.8595843302432258
Cross validation score of ElasticNet Regressor: 0.8639115938544559
Cross validation score fo Decision Tree Regressor: 0.7265022690708568
Cross validation score of Random Forest Regressor: 0.8828788141266127


Random Forest Regressor is giving the highest cross score in comparison to other model instances. We will be hyper-param
eter tuning the variables now and check the model again.
```

Hyper Parameter Tuning:

Performing hyper parameter tuning:

In [48]:
```
from sklearn.model_selection import GridSearchCV
parameter = {
 'max_depth': [100, 400, 800,1200],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5],
 'n_estimators': [100,200, 400, 800]
}
GCV = GridSearchCV(RandomForestRegressor(),parameter,cv=5)
```

In [49]:
```
GCV.fit(x_train,y_train)
```

Out[49]:
```
GridSearchCV(cv=5, estimator=RandomForestRegressor(),
             param_grid={'max_depth': [100, 400, 800, 1200],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5],
                         'n_estimators': [100, 200, 400, 800]})
```

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 [50]:
```
GCV.best_params_
```

Out[50]:
```
{'max_depth': 800,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'n_estimators': 200}
```

In [52]:
```
rdr = RandomForestRegressor(max_depth= 400,
min_samples_leaf= 1,
min_samples_split= 2,
n_estimators= 100)
rdr.fit(x_train,y_train)
pred1=rdr.predict(x_test)
print('R2 score',r2_score(y_test, pred1))
print('MAE:', mean_absolute_error(y_test, pred1))
print('MSE:', mean_squared_error(y_test, pred1))
print('RMSE:', np.sqrt(mean_squared_error(y_test,pred1)))
```

```
R2 score 0.8861992420212079
MAE: 16127.335327635328
MSE: 482107312.01396364
RMSE: 21956.94222823305
```

Interpretations:

In this study, two experiments were run. The first experiment used all of the variables that were present in the dataset after pre-processing, whereas the second experiment only used the most significant variables with the aim of enhancing the model's performance with fewer variables.

- To ensure model correctness, it is necessary to develop several models and train and evaluate them.
- The best fit model is determined by a variety of matrices, including R-squared, RMSE value, etc.
- Database helps create the ideal model and will aid in comprehension of the Australian market.

Conclusion

After perusing the aforementioned studies, the first thing I realised about the dataset is that there has been a lot of study done in the past on the US housing market; as a result, I obtained data on homes in other countries, including Australian homes. Additionally, a variety of algorithms have been used to predict housing prices, but regression is the most effective one. In my research, I will use a variety of regression algorithms on this dataset and present the results to determine which algorithm provides the best accuracy using the R-squared and RMSE value.

The dataset was particularly intriguing to work with because it had both category and numerical data. Sale Price versus Month demonstrates that winter is when property prices rise the most, as opposed to autumn, spring, and summer. Seasonal sales. In comparison to fall and winter, the chances of individuals buying homes are higher in the summer and spring. As we can see, the roads or streets seem to play an essential influence in house/property choosing. Geographic location or condition. Neighborhood homes, condominiums, villas, and other properties sell for a lot of money. Customers choose overall developed areas over emerging ones.