HOUSE PRICE PREDICTION

INTRODUCTION

BUSINESS PROBLEM FRAMING

Houses are one of the necessary needs of each and every person around the globe and therefore housing and real estate market is one of the markets which are 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 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 OF THE DOMAIN PROBLEM

While buying or selling a house, there are many factors that play a role in determining the price like type of the house, land, basement, Internal and external conditions, sq. feet, extra amenities, area, etc.

ANALYTICAL PROBLEM FRAMING

➢ **Mathematical/ Analytical Modelling Of The Problem**

The target column is price which is continuous variable. So we use Regression model to predict the target column

Different algorithms have been implemented

MODEL	R2 SCORE	CV SCORE
Linear Regression	0.60	0.72
SVR	0.03	0.06
Random Forest Regressor	0.81	0.82
Gradient Boosting Regressor	0.84	0.82

➢ **Data Sources and their formats**

Data provided by Flip Robo Technologies

Format: CSV two csv files provided,

1 -> Train data: 1168 rows & 81 columns

2-> Test data: 292 rows & 80 columns

➢ **Data Pre-processing Done**

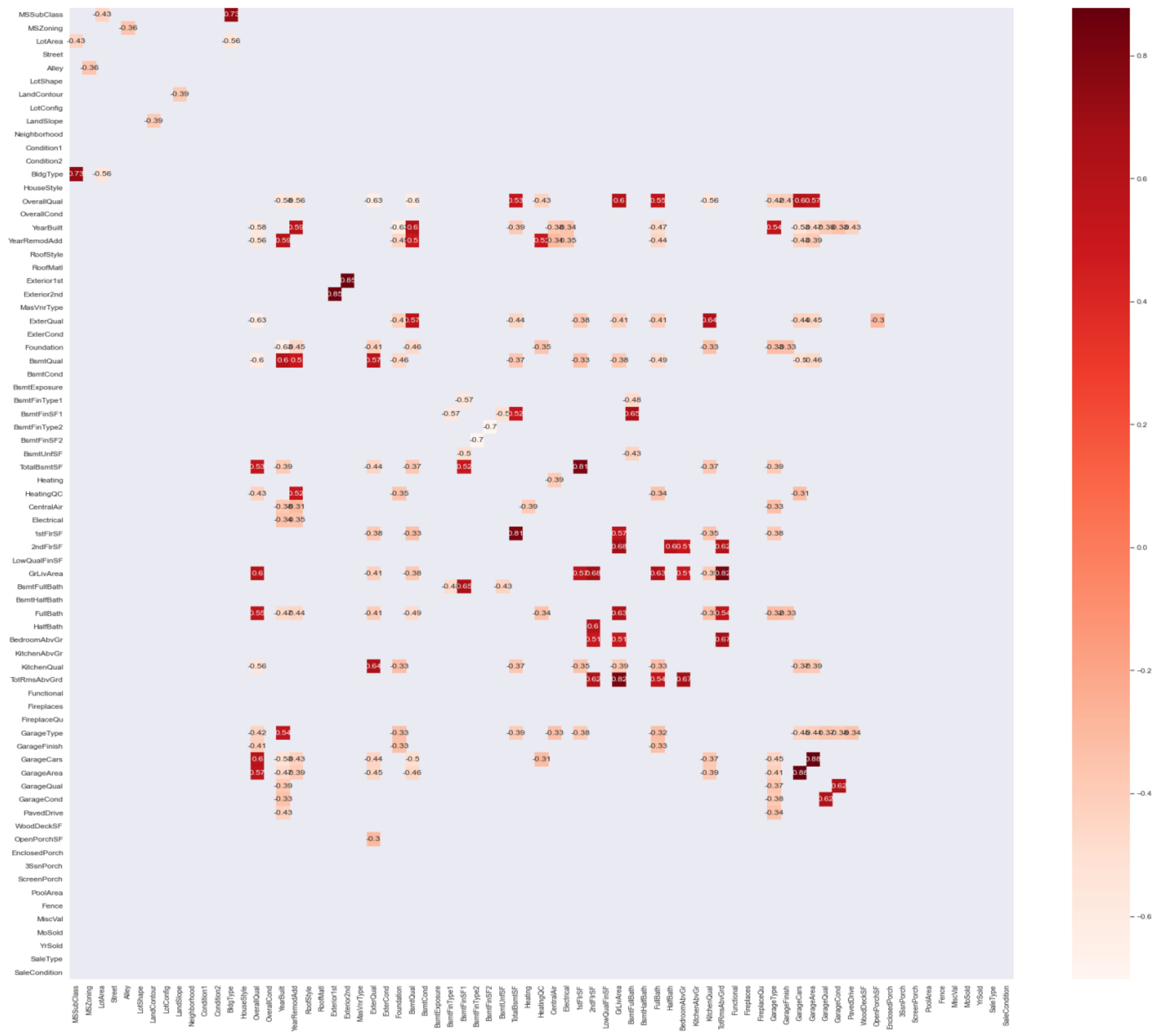
- ✓ Checked for the missing values
- ✓ Dropped the columns with more number of missing data
- ✓ Filled the missed values with appropriate values
- ✓ Encoded the categorical data

- ✓ Found the correlation
- ✓ Dropped the columns with multicollinearity
- ✓ Removed outliers
- ✓ Split features and Target column
- ✓ Scaled the data of features

- **Data Inputs- Logic- Output Relationships**

Now the data is split into train and test. Then different algorithms were applied to predict the price.

Correlation: Plotting the corr, which with threshold > 0.3



- State the set of assumptions (if any) related to the problem under consideration
  - Assumed missing values as Nan ->i.e. no such thing.
  - Reduced the number of features to avoid curse of dimensionality using pca (Principal component analysis)

- Hardware and Software Requirements and Tools Used
  - Hardware used:
  - RAM: 8GB
  - Processor: Intel I5
  - Software's: Jupyter Notebook (Anaconda Framework)
  - Python (coding language)
  - Libraries / Packages used – pandas, Numpy, Sklearn, Seaborn, Selenium (Web scrapping)

- Model/S Development and Evaluation
  - Identification of possible problem-solving approaches (methods)

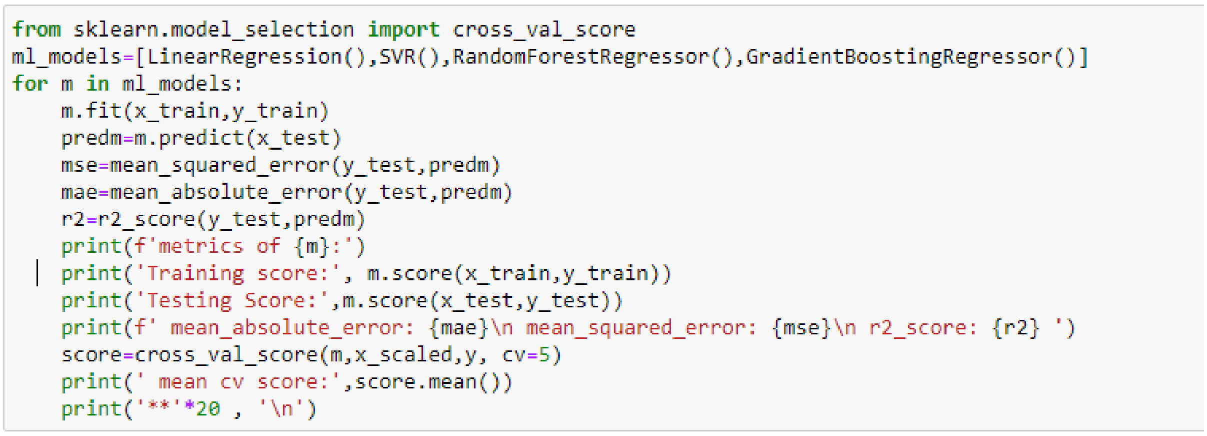
    1. Clean the dataset from unwanted details.
    2. Rename values with meaningful information. Fill missing values.
    3. Encoding the categorical data to get numerical input data.
    4. Compare different models and identify the suitable model.
    5. R2 score is used as the primary evaluation metric.
    6. MSE and RMSE are used as secondary metrics.
    7. Cross Validation Score was used to ensure there is no overfitting our underfitting model.

Testing Of Identified Approaches (Algorithms)

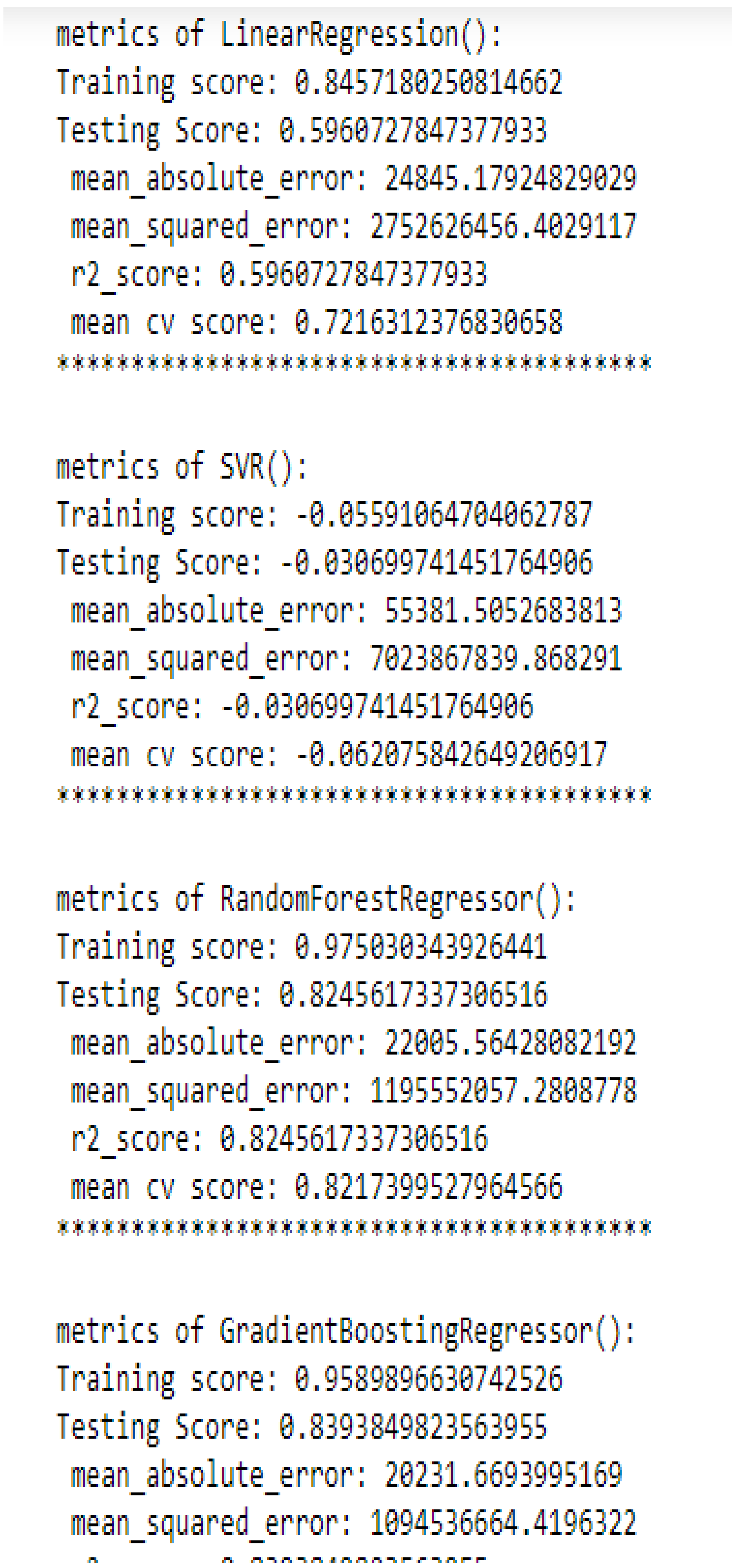
Since the target variable is continuous, we used regression model. Different regression models we used to predict are

- Linear Regression
- SVR
- DecisionTreeRegressor
- RandomForestRegressor
- GradientBoostingRegressor

Run and Evaluate Selected Models



```
from sklearn.model_selection import cross_val_score
ml_models=[LinearRegression(),SVR(),RandomForestRegressor(),GradientBoostingRegressor()]
for m in ml_models:
    m.fit(x_train,y_train)
    predm=m.predict(x_test)
    mse=mean_squared_error(y_test,predm)
    mae=mean_absolute_error(y_test,predm)
    r2=r2_score(y_test,predm)
    print(f'metrics of {m}:')
    print('Training score:', m.score(x_train,y_train))
    print('Testing Score:',m.score(x_test,y_test))
    print(f' mean_absolute_error: {mae}\n mean_squared_error: {mse}\n r2_score: {r2} ')
    score=cross_val_score(m,x_scaled,y, cv=5)
    print(' mean cv score:',score.mean())
    print('**'*20 , '\n')
```



```
metrics of LinearRegression():
Training score: 0.8457180250814662
Testing Score: 0.5960727847377933
 mean_absolute_error: 24845.17924829029
 mean_squared_error: 2752626456.4029117
 r2_score: 0.5960727847377933
 mean cv score: 0.7216312376830658
****************************************

metrics of SVR():
Training score: -0.05591064704062787
Testing Score: -0.030699741451764906
 mean_absolute_error: 55381.5052683813
 mean_squared_error: 7023867839.868291
 r2_score: -0.030699741451764906
 mean cv score: -0.062075842649206917
****************************************

metrics of RandomForestRegressor():
Training score: 0.975030343926441
Testing Score: 0.8245617337306516
 mean_absolute_error: 22005.56428082192
 mean_squared_error: 1195552057.2808778
 r2_score: 0.8245617337306516
 mean cv score: 0.8217399527964566
****************************************

metrics of GradientBoostingRegressor():
Training score: 0.9589896630742526
Testing Score: 0.8393849823563955
 mean_absolute_error: 20231.6693995169
 mean_squared_error: 1094536664.4196322
```

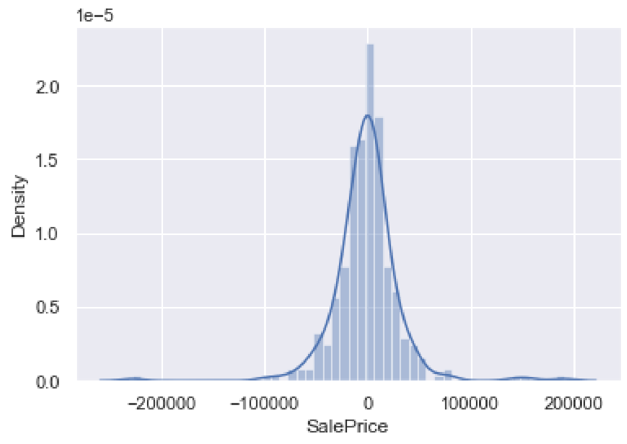
Gradient boosting has the best accuracy, less cv difference with r2 score. We selected it as best model.

```
#Makikng the model
gbr=GradientBoostingRegressor()
gbr.fit(x_train,y_train)
pred=gbr.predict(x_test)
mse=mean_squared_error(y_test,pred)
mae=mean_absolute_error(y_test,pred)
r2=r2_score(y_test,pred)
print(f' mean_absolute_error: {mae}\n mean_squared_error: {mse}\n r2_score: {r2} ')
```

```
 mean_absolute_error: 20070.149191706863
 mean_squared_error: 1059283553.831475
 r2_score: 0.8445581110081535
```

```
sns.distplot(y_test-pred)
```

```
<AxesSubplot:xlabel='SalePrice', ylabel='Density'>
```



The graph of difference between the original and predicted value is normally distributed.

➢ Interpretation Of The Results

Gradient boosting is selected as best model and done tuning.
The accuracy increased from 0.84 to 0.86 after tuning.

```
#Makikng the model
gbr=GradientBoostingRegressor()
gbr.fit(x_train,y_train)
pred=gbr.predict(x_test)
mse=mean_squared_error(y_test,pred)
mae=mean_absolute_error(y_test,pred)
r2=r2_score(y_test,pred)
print(f' mean_absolute_error: {mae}\n mean_squared_error: {mse}\n r2_score: {r2} ')
```

```
 mean_absolute_error: 20070.149191706863
 mean_squared_error: 1059283553.831475
 r2_score: 0.8445581110081535
```

The parameters that played role in increasing the accuracy:

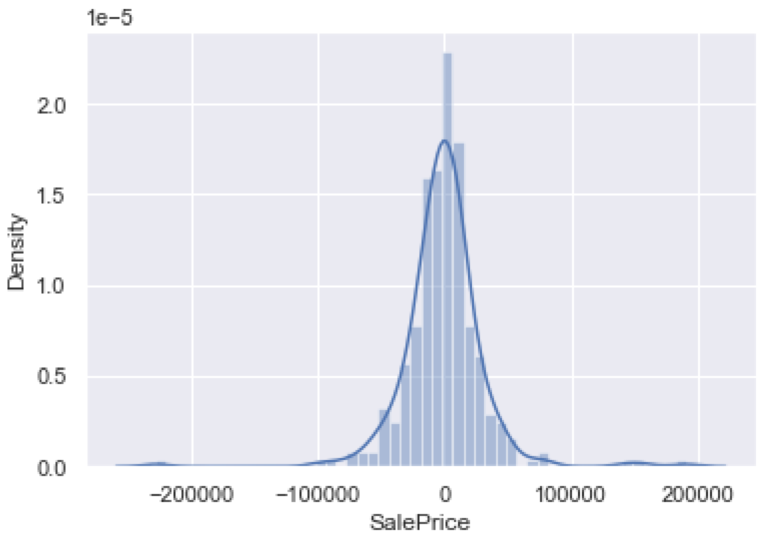
Min_sample_leaf = 2

N_estimators = 700

Min_sample_split=2

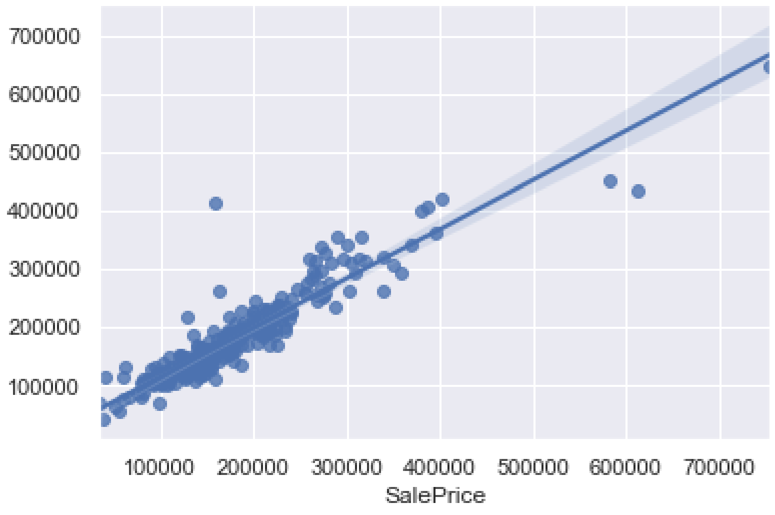
```
sns.distplot(y_test-pred)
```

```
<AxesSubplot:xlabel='SalePrice', ylabel='Density'>
```



```
: sns.regplot(y_test,pred)
```

```
: <AxesSubplot:xlabel='SalePrice'>
```


SalePrice