In [1]:

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

In [2]:

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

In [3]:

```
data = pd.read_csv('Melbourne_housing_FULL.csv')
data.head()
```

Out[3]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	La
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016	2.5	3067.0	...	1.0	1.0	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	3067.0	...	1.0	1.0	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	3067.0	...	1.0	0.0	
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016	2.5	3067.0	...	2.0	1.0	
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	3067.0	...	2.0	0.0	

5 rows × 21 columns

In [4]:

```
# checking for unique values in the data
data.nunique()
```

Out[4]:

```
Suburb             351
Address          34009
Rooms               12
Type                 3
Price             2871
Method               9
SellerG            388
Date                78
Distance           215
Postcode           211
Bedroom2            15
Bathroom            11
Car                 15
Landsize          1684
BuildingArea       740
YearBuilt          160
CouncilArea         33
Lattitude        13402
Longtitude       14524
Regionname           8
Propertycount      342
dtype: int64
```

In [5]:

```
data.shape
```

Out[5]:

```
(34857, 21)
```

In [8]:

```
working_df = data[["Suburb","Rooms","Type","Method","SellerG","Regionname","Propertycount
","Distance","CouncilArea","Bedroom2",
                   "Bathroom","Car","Landsize","BuildingArea","Price"]]
working_df.head()
```

Out[8]:

	Suburb	Rooms	Type	Method	SellerG	Regionname	Propertycount	Distance	CouncilArea	Bedroom2	Bathroom	Ca
0	Abbotsford	2	h	SS	Jellis	Northern Metropolitan	4019.0	2.5	Yarra City Council	2.0	1.0	1.
1	Abbotsford	2	h	S	Biggin	Northern Metropolitan	4019.0	2.5	Yarra City Council	2.0	1.0	1.
2	Abbotsford	2	h	S	Biggin	Northern Metropolitan	4019.0	2.5	Yarra City Council	2.0	1.0	0.
3	Abbotsford	3	u	VB	Rounds	Northern Metropolitan	4019.0	2.5	Yarra City Council	3.0	2.0	1.
4	Abbotsford	3	h	SP	Biggin	Northern Metropolitan	4019.0	2.5	Yarra City Council	3.0	2.0	0.

In [9]:

```
working_df.shape
```

Out[9]:

```
(34857, 15)
```

In [11]:

```
# check for missing values
working_df.isna().sum()
```

Out[11]:

```
Suburb               0
Rooms                0
Type                 0
Method               0
SellerG              0
Regionname           3
Propertycount        3
Distance             1
CouncilArea          3
Bedroom2          8217
Bathroom          8226
Car               8728
Landsize         11810
BuildingArea     21115
Price             7610
dtype: int64
```

In [13]:

```
# replacing NaN value with 0 for the following columns
col_fill_zero = ['Propertycount','Distance','Bedroom2','Bathroom','Car']
working_df[col_fill_zero] = working_df[col_fill_zero].fillna(0)
working_df.isna().sum()
```

Out[13]:

```
Suburb               0
Rooms                0
Type                 0
Method               0
SellerG              0
Regionname           3
Propertycount        0
Distance             0
CouncilArea          3
Bedroom2             0
Bathroom             0
Car                  0
Landsize         11810
BuildingArea     21115
Price             7610
dtype: int64
```

In [16]:

```
# replacing missing values in landsize and building area by their mean
working_df['Landsize'] = working_df['Landsize'].fillna(working_df['Landsize'].mean())
working_df['BuildingArea'] = working_df['Landsize'].fillna(working_df['Landsize'].mean())
working_df.isna().sum()
```

Out[16]:

```
Suburb              0
Rooms               0
Type                0
Method              0
SellerG             0
Regionname          3
Propertycount       0
Distance            0
CouncilArea         3
Bedroom2            0
Bathroom            0
Car                 0
Landsize            0
BuildingArea        0
Price            7610
dtype: int64
```

In [18]:

```
# we will drop the NaN values from Regionname and CouncilArea
working_df.dropna(inplace = True)
working_df.isna().sum()

# now our dataset is free of missing values
```

Out[18]:

```
Suburb           0
Rooms            0
Type             0
Method           0
SellerG          0
Regionname       0
Propertycount    0
Distance         0
CouncilArea      0
Bedroom2         0
Bathroom         0
Car              0
Landsize         0
BuildingArea     0
Price            0
dtype: int64
```

In [27]:

```
# now we encode the text columns to numerical values using get_dummies
working_df = pd.get_dummies(working_df, drop_first = True)
```

In [28]:

```
x = working_df.drop('Price', axis = 'columns')
```

In [29]:

```
y = working_df.Price
```

In [30]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,random_state = 2)
```

In [32]:

```
from sklearn.linear_model import LinearRegression
lin_reg_model = LinearRegression()
lin_reg_model.fit(x_train, y_train)
lin_reg_model.score(x_test, y_test)
# we see the score is very less on the test data
```

Out[32]:

```
-442607.11915660405
```

In [33]:

```
# now lets see the score on training data
lin_reg_model.score(x_train, y_train)
# for training sample it gives a decent accuracy but very poor accuracy on test data
# this shows that the model is over fitting
```

Out[33]:

```
0.678947915038123
```

In [36]:

```
# we will use lasso regression that is L1 regularised to cure this problem of overfitting
from sklearn.linear_model import Lasso
lasso_reg_model = Lasso(alpha = 50, max_iter = 100, tol = 0.1)
lasso_reg_model.fit(x_train,y_train)
```

Out[36]:

```
Lasso(alpha=50, copy_X=True, fit_intercept=True, max_iter=100, normalize=False,
      positive=False, precompute=False, random_state=None, selection='cyclic',
      tol=0.1, warm_start=False)
```

In [37]:

```
lasso_reg_model.score(x_test, y_test)
# geting a decent score on testing data
```

Out[37]:

```
0.6778951491503731
```

In [38]:

```
# now lets test the score on training data
lasso_reg_model.score(x_train, y_train)
# the score is similar to what we get on testing data
# the overfitting problem is resolved
```

Out[38]:

```
0.6748011426788028
```

In [40]:

In [40]:

```
# using Ridge regression that is L2 regularized to solve the problem of overfitting
from sklearn.linear_model import Ridge
ridge_reg_model = Ridge(alpha = 50, max_iter = 100, tol = 0.1)
ridge_reg_model.fit(x_train,y_train)
```

Out[40]:

```
Ridge(alpha=50, copy_X=True, fit_intercept=True, max_iter=100, normalize=False,
      random_state=None, solver='auto', tol=0.1)
```

In [41]:

```
ridge_reg_model.score(x_test,y_test)
```

Out[41]:

```
0.6712051031791796
```

In [42]:

```
ridge_reg_model.score(x_train,y_train)
```

Out[42]:

```
0.6631988324137152
```