Boston Housing Prediction

In this project we will predict the price of the houses in Boston,Massachusetts.We will use our housing.csv data to train our model to successfully predict the prices in future.

Importing required libraries

Our first step is to import all the required libraries.

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Importing Housing Data

Import the data.

In [4]:
```python
df=pd.read_csv('housing.csv')
```

Displaying the top 5 rows of the data.

In [5]:
```python
df.head()
```

Out[5]:

	RM	LSTAT	PTRATIO	MEDV
0	6.575	4.98	15.3	504000.0
1	6.421	9.14	17.8	453600.0
2	7.185	4.03	17.8	728700.0
3	6.998	2.94	18.7	701400.0
4	7.147	5.33	18.7	760200.0

Features

RM: average number of rooms LSTAT: percentage of population considered lower status PTRATIO: pupil-teacher ratio by town MEDV: median value of owner-occupied homes

Our target value is MEDV.

Finding Correlation

We know that correlation coefficient close to 1 represents a large positive relation, -1 represents large negative relation and 0 represents no relation at all.Therefore, let's find the correlation between different features and the target variable.

In [6]:
```python
df.corr()
```

Out[6]:

	RM	LSTAT	PTRATIO	MEDV
RM	1.000000	-0.612033	-0.304559	0.697209
LSTAT	-0.612033	1.000000	0.360445	-0.760670
PTRATIO	-0.304559	0.360445	1.000000	-0.519034
MEDV	0.697209	-0.760670	-0.519034	1.000000

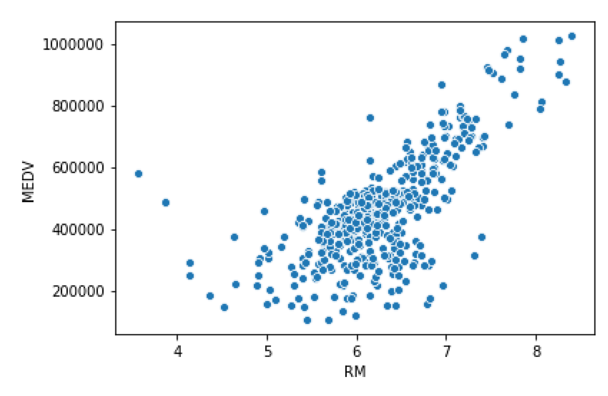
After analysing the correlation coefficients we find that RM is correlated to MEDV but LSTAT and PTRATIO has negative coefficients.SO, we will plot them and observe the relation.

Plotting the features with target variable

Plotting RM and MEDV.

In [7]:
```python
sns.scatterplot(df['RM'],df['MEDV'])
```

Out[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b5fb516448>
```

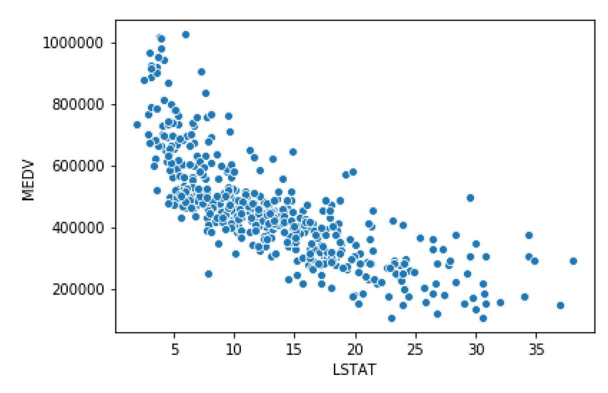


We can conclude that RM,average number of rooms, is increasing linearly.

Plotting LSTAT and MEDV.

In [8]:
```python
sns.scatterplot(df['LSTAT'],df['MEDV'])
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b5fbc8b8c8>
```

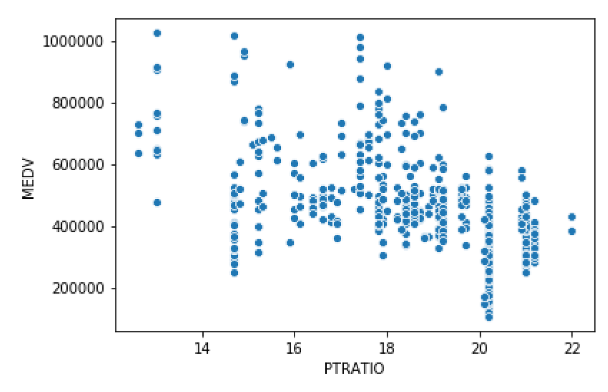


We can see that LSTAT,percentage of population considered lower status, is decreasing linearly.

Lastly, we will plot PTRATIO and MEDV.

In [9]:
```python
sns.scatterplot(df['PTRATIO'],df['MEDV'])
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b5fbd15688>
```



We can see that PTRATIO,pupil-teacher ratio by town,does not have any linear relation with MEDV.

Extracting features and target variable

As we saw that RM and LSTAT are linearly related with MEDV but PTRATIO is not.Therefore, our features will include only RM,average number of rooms, and LSTAT,percentage of population considered lower status.

In [10]:
```python
features=df.drop(['PTRATIO','MEDV'],axis=1)
target_variable=df['MEDV']
```

Printing features and target_variable.

In [11]:
```python
features
```

Out[11]:

	RM	LSTAT
0	6.575	4.98
1	6.421	9.14
2	7.185	4.03
3	6.998	2.94
4	7.147	5.33
...	...	...
484	6.593	9.67
485	6.120	9.08
486	6.976	5.64
487	6.794	6.48
488	6.030	7.88

489 rows × 2 columns

In [12]:
```python
target_variable
```

Out[12]:
```
0      504000.0
1      453600.0
2      728700.0
3      701400.0
4      760200.0
         ...
484    470400.0
485    432600.0
486    501900.0
487    462000.0
488    249900.0
Name: MEDV, Length: 489, dtype: float64
```

Building our model

Firstly, we will divide our data into training and testing sets.

Importing train_test_split.

In [13]:
```python
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(features,target_variable,test_size=0.2)
```

Training our model using Logistic Regression

Importing Logistic Regression.

In [14]:
```python
from sklearn.linear_model import LogisticRegression
```

Defining our classifier.

In [15]:
```python
classifier=LogisticRegression()
```

In [16]:
```python
classifier.fit(x_train,y_train)
```

```
C:\Users\JASSAR\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:940: Convergenc
eWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```

Out[16]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Prediction from training set

In [17]:
```python
training_predictions=classifier.predict(x_train)
```

Model Evaluation

Evaluation based on training set predictions.

Evaluating mean_absolute_error.

In [18]:
```python
from sklearn.metrics import mean_absolute_error
train_mean_absolute_error=mean_absolute_error(y_train,training_predictions)
train_mean_absolute_error
```

Out[18]:
```
87716.62404092071
```

Evaluating mean_squared_error.

In [19]:
```python
from sklearn.metrics import mean_squared_error
train_mean_squared_error=mean_squared_error(y_train,training_predictions)
train_mean_squared_error
```

Out[19]:
```
13673052736.57289
```

Evaluating r2 score.

In [20]:
```python
from sklearn.metrics import r2_score
train_r2_score=r2_score(y_train,training_predictions)
train_r2_score
```

Out[20]:
```
0.5019739734731801
```

Predictions from testing set

In [21]:
```python
testing_predictions=classifier.predict(x_test)
```

Model Evaluation

Evaluation based on testing set predictions.

Evaluating mean_absolute_error.

In [22]:
```python
test_mean_absolute_error=mean_absolute_error(y_test,testing_predictions)
test_mean_absolute_error
```

Out[22]:
```
88285.71428571429
```

Evaluating mean_squared_error.

In [23]:
```python
test_mean_squared_error=mean_squared_error(y_test,testing_predictions)
test_mean_squared_error
```

Out[23]:
```
14567580000.0
```

Evaluating r2 score.

In [24]:
```python
test_r2_score=r2_score(y_test,testing_predictions)
test_r2_score
```

Out[24]:
```
0.4519343825999951
```

Accuracy

Let's find out the accuracy.

Importing library.

In [25]:
```python
from sklearn.metrics import accuracy_score
```

Calculating accuracy of training predictions.

In [28]:
```python
training_set_accuracy=accuracy_score(y_train,training_predictions)
training_set_accuracy
```

Out[28]:
```
0.05115089514066496
```

Calculating accuracy of testing predictions.

In [29]:
```python
testing_set_accuracy=accuracy_score(y_test,testing_predictions)
testing_set_accuracy
```

Out[29]:
```
0.030612244897959183
```

Conclusion

Accuracy score of training set is 0.05 and R2 score is 0.50 and that of testing set is 0.03 and 0.45. R2 score should be close to 1.So, we conclude that we can use this model to predict prices.