A PROJECT REPORT ON

“BOSTON HOUSING”

TLS21A977

SKANDA G N

INDEX

ABSTRACT

We evaluate the performance and predictive power of a model that has been trained and tested on data collected from homes in suburbs of Boston, Massachusetts. A model trained on this data that is seen as a good fit could then be used to make certain predictions about a home — in particular, its monetary value. This model would prove to be invaluable for someone like a real estate agent who could make use of such information on a daily basis. The dataset for this project originates from the UCI Machine Learning Repository. The Boston housing data was collected in 1978 and each of the 506 entries represent aggregated data about 14 features for homes from various suburbs in Boston, Massachusetts.

INTRODUCTION

In this project, we will develop and evaluate the performance and the predictive power of a model trained and tested on data collected from houses in Boston's suburbs. Once we get a good fit, we will use this model to predict the monetary value of a house located at the Boston's area. A model like this would be very valuable for a real estate agent who could make use of the information provided in a daily basis. Housing prices are an important reflection of the economy, and housing price ranges are of great interest for both buyers and sellers. Ask a home buyer to describe their dream house, and they probably won't begin with the height of the basement ceiling or the proximity to an east-west railroad. But this playground competition's data-set proves that much more influences price negotiations than the number of bedrooms or a white-picket fence. Data Science is the process of making some assumptions and hypothesis on the data, and testing them by performing some tasks.

TASKS

- DATA ACQUISITION AND CLEANING
- DATA VISUALIZATION
- DATA MODELLING
- TESTING
- COMPARISON AND MEASUREMENT

TASK 1: DATA ACQUISITION AND CLEANING

1.1 DATA COLLECTION

The dataset used in this project comes from the UCI Machine Learning Repository. This data was collected in 1978 and each of the 506 entries represents aggregate information about 14 features of homes from various suburbs located in Boston.

We have downloaded the dataset from Kaggle.com the link to the dataset is:

https://www.kaggle.com/altavish/boston-housing-dataset

1.2 DATA OVERVIEW

This data has 506 entries and 14 columns. The input of this data contains various attributes like crime rate, nitric oxide concentration and many other. An insight of the original dataset is given below:

	ID	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0	1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
3	5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2
4	7	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9

Now with the help of pandas library we will print the datset:

CODE:

```
import pandas as pd
data=pd.read_csv("C:/Users/Skanda/Desktop/boston_housing.csv")
print(data)
```

OUTPUT:

```
        crim    zn  indus  chas    nox  ...    tax  ptratio   black  lstat  medv
0    0.00632  18.0   2.31     0  0.538  ...  296.0     15.3  396.90   4.98  24.0
1    0.02731   0.0   7.07     0  0.469  ...  242.0     17.8  396.90   9.14  21.6
2    0.02729   0.0   7.07     0  0.469  ...  242.0     17.8  392.83   4.03  34.7
3    0.03237   0.0   2.18     0  0.458  ...  222.0     18.7  394.63   2.94  33.4
4    0.06905   0.0   2.18     0  0.458  ...  222.0     18.7  396.90   5.33  36.2
..       ...   ...    ...   ...    ...  ...    ...      ...     ...    ...   ...
501  0.06263   0.0  11.93     0  0.573  ...  273.0     21.0  391.99   9.67  22.4
502  0.04527   0.0  11.93     0  0.573  ...  273.0     21.0  396.90   9.08  20.6
503  0.06076   0.0  11.93     0  0.573  ...  273.0     21.0  396.90   5.64  23.9
504  0.10959   0.0  11.93     0  0.573  ...  273.0     21.0  393.45   6.48  22.0
505  0.04741   0.0  11.93     0  0.573  ...  273.0     21.0  396.90   7.88  11.9

[506 rows x 14 columns]
>>>
```

The attributes can be summarized as follows:

1. **CRIM:** This is the per capita crime rate by town
2. **ZN:** This is the proportion of residential land zoned for lots larger than 25,000 sq.ft.
3. **INDUS:** This is the proportion of non-retail business acres per town.
4. **CHAS:** This is the Charles River dummy variable (this is equal to 1 if tract bounds river; 0 otherwise)
5. **NOX:** This is the nitric oxides concentration (parts per 10 million)

6. **RM:** This is the average number of rooms per dwelling
7. **AGE:** This is the proportion of owner-occupied units built prior to 1940
8. **DIS:** This is the weighted distances to five Boston employment centers
9. **RAD:** This is the index of accessibility to radial highways
10. **TAX:** This is the full-value property-tax rate per $10,000
11. **PTRATIO:** This is the pupil-teacher ratio by town
12. **B:** This is calculated as $1000(Bk — 0.63)^2$, where Bk is the proportion of people of African American descent by town
13. **LSTAT:** This is the percentage lower status of the population
14. **MEDV:** This is the median value of owner-occupied homes in $1000s

1.3 DATA PREPROCESSING

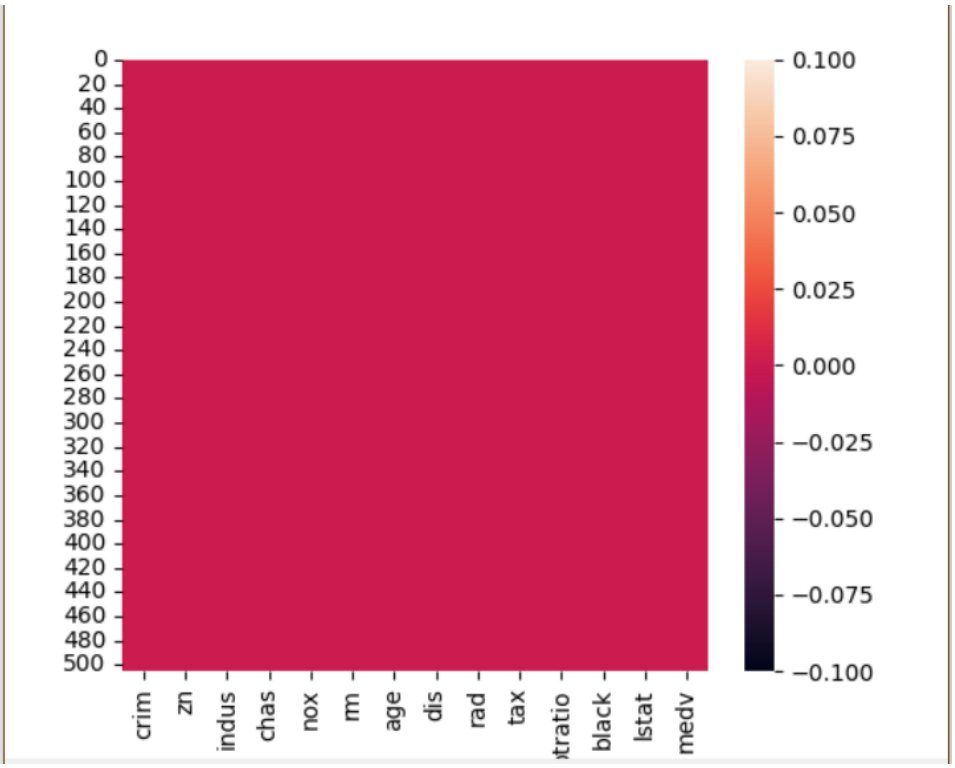
For cleaning of the dataset we will check for any null values or for any missing values.

CODE:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
data=pd.read_csv("C:/Users/Skanda/Desktop/boston_housing.csv")
print(data.isnull().sum())
sns.heatmap(data.isnull())
plt.show()
```

OUTPUT:

```
crim       0
zn         0
indus      0
chas       0
nox        0
rm         0
age        0
dis        0
rad        0
tax        0
ptratio    0
black      0
lstat      0
medv       0
dtype: int64
```



In this dataset there is no need for cleaning, as the dataset is already cleaned and there are no missing values present in the dataset, this can also be observed from the above graph.

1.4 DATA EXPLORATION

In this section we will make an exploratory analysis of the dataset and provide some observations.

- Information about the dataset feature:

```
print(data.info())
```

OUTPUT:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   crim     506 non-null    float64
 1   zn       506 non-null    float64
 2   indus    506 non-null    float64
 3   chas     506 non-null    int64
 4   nox      506 non-null    float64
 5   rm       506 non-null    float64
 6   age      506 non-null    float64
 7   dis      506 non-null    float64
 8   rad      506 non-null    int64
 9   tax      506 non-null    float64
 10  ptratio  506 non-null    float64
 11  black    506 non-null    float64
 12  lstat    506 non-null    float64
 13  medv     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
None
```

- Description of the dataset:

```
data=pd.read_csv("C:/Users/Skanda/Desktop
print(data.describe())
```

OUTPUT:

```
             crim          zn       indus  ...       black       lstat        medv
count  506.000000  506.000000  506.000000  ...  506.000000  506.000000  506.000000
mean     3.613524   11.363636   11.136779  ...  356.674032   12.653063   22.532806
std      8.601545   23.322453    6.860353  ...   91.294864    7.141062    9.197104
min      0.006320    0.000000    0.460000  ...    0.320000    1.730000    5.000000
25%      0.082045    0.000000    5.190000  ...  375.377500    6.950000   17.025000
50%      0.256510    0.000000    9.690000  ...  391.440000   11.360000   21.200000
75%      3.677083   12.500000   18.100000  ...  396.225000   16.955000   25.000000
max     88.976200  100.000000   27.740000  ...  396.900000   37.970000   50.000000

[8 rows x 14 columns]
```

TASK 2: DATA VISUALIZATION

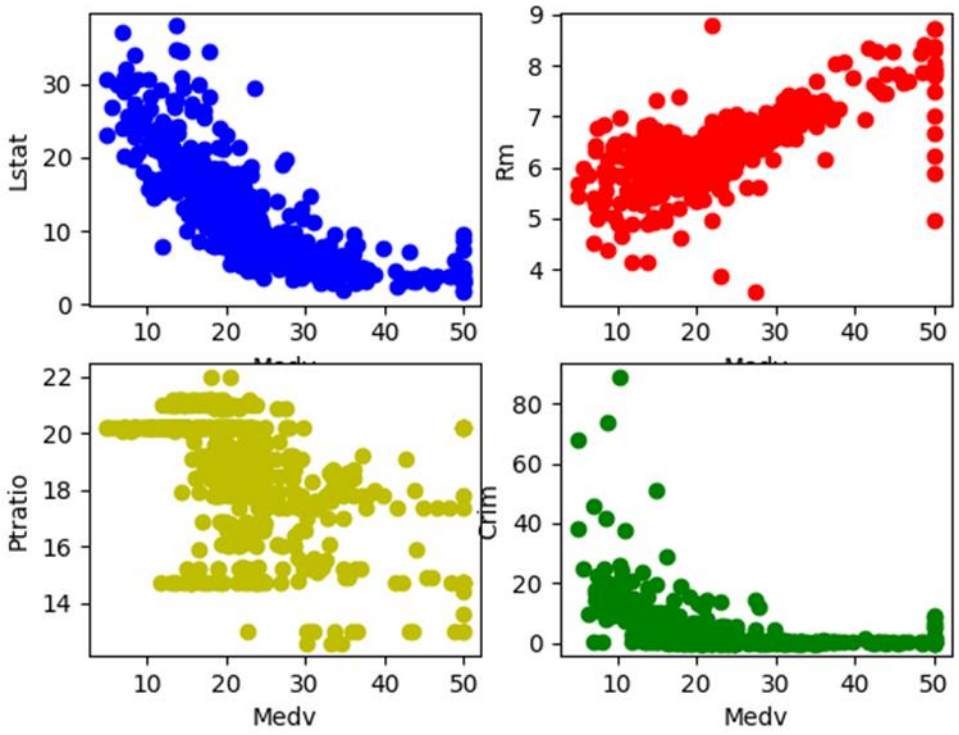
In this section of the project, we will now make a visual analysis of the dataset and provide some observations. As our goal is to develop a model that has the capacity of predicting the value of houses, we will split the dataset into features and the target variable. And store them in features.

- The features 'RM', 'LSTAT' and 'PTRATIO', give us quantitative information about each datapoint. We will store them in *features*.
- The target variable, 'MEDV', will be the variable we seek to predict.

CODE:

```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
import seaborn as sns
data=pd.read_csv('bostonhousing.csv')

data.info()

x1=data.medv
y1=data.lstat
y2=data.rm
y3=data.ptratio
y4=data.crim


plt.subplot(221)
plt.scatter(x1,y1,c='b')
plt.xlabel('Medv')
plt.ylabel('Lstat')

plt.subplot(222)
plt.scatter(x1,y2,c='r')
plt.xlabel('Medv')
plt.ylabel('Rm')

plt.subplot(223)
plt.scatter(x1,y3,c='y')
plt.xlabel('Medv')
plt.ylabel('Ptratio')

plt.subplot(224)
plt.scatter(x1,y4,c='g')
plt.xlabel('Medv')
plt.ylabel('Crim')
plt.show()


plt.figure(figsize=(12,6))
sns.heatmap(data.corr(),annot=True)
plt.show()
```

REPRESENTATION:



OBSERVATION:

Houses with more rooms (higher ‘RM’ value) will worth more. Usually houses with more rooms are bigger and can fit more people, so it is reasonable that they cost more money. They are directly proportional variables.

- Neighbourhoods with more lower-class workers (higher ‘LSTAT’ value) will worth less. If the percentage of lower working-class people is higher, it is likely that they have low purchasing power and therefore, they houses will cost less. They are inversely proportional variables.
- Neighbourhoods with more students to teacher’s ratio (higher ‘PTRATIO’ value) will be worth less. If the percentage of students to teacher’s ratio people is higher, it is likely that in the neighbourhood there are less schools, this could be because there is less tax income which could be because in that neighbourhood people earn less money. If people earn less money, it is likely that their houses are worth less. They are inversely proportional variables.

We can spot a linear relationship between ‘RM’ and House prices ‘MEDV’.

TASK 3: DATA MODELLING

CREATING THE ML MODEL:

We first split the dataset into 'X' and 'Y' values, in this Project we have stored all the columns of dataset other than the 'medv' column as 'X' value and the 'medv' column as 'Y' value and later apply the chosen LinearRegression to our model.

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

The libraries which need to be imported in order to create the LinearRegression ML and find the accuracy and mean squared error.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

We use scikit-learn's LinearRegression to train our model on both the training and test sets.

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
```

regressor is a variable name which is assigned to **LinearRegression ().**

SPLITTING THE DATA INTO TRAINING AND TESTING SETS:

Next, we split the data into training and testing sets. We train the model with **80%** of the samples and test with the remaining **20%**. To split the data, we use '**test_train_split**' function provided by '**scikit-learn**' library. We finally print the sizes of our training and test set to verify if the splitting has occurred properly.

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size = 0.20 ,random_state = 3)
```

Now we fit the data ('xtrain', 'ytrain') into the **LinearRegression ()** function. And predict the 'Y'

Value using the 'xtest' and store it in 'ypred'.

```
regressor.fit(xtrain,ytrain)
ypred=regressor.predict(xtest)
```

Mean squared error and accuracy:

We find mean squared error through **mean_squared_error ()** with 'ytest' and 'ypred' as the arguments. We find the accuracy though **regressor.score ()** which is a function of **LinearRegression** with 'xtest' and 'ytest' as the parameters.

```
from sklearn.metrics import mean_squared_error

print(" \n Mean Squared Error:")
print(mean_squared_error(ytest,ypred))

print("\n Accuracy:")
print(regressor.score(xtest,ytest))
```

```
 Mean Squared Error:
16.943073013833818

 Accuracy:
0.7952617563243853
```

The above image is the screenshot of output window which was obtained from all the above codes we can see in the image that mean squared error of the dataset found using the LinearRegression ML is 16.943 and the accuracy was found to be 79%.

The accuracy can be increased by increasing the train data or by changing the random state which was 3 in the above case.

TASK 4: TESTING

After applying LinearRegression and sckit-learn we can test our data.

Average medv value: The average of the medv coloumn we found from the dataset was 20.021

Test 1: In The test 1 we use the data of the first row of the data set. And the test result is, medv=30.082.

```
print("Test case 1: 1st row of dataset")
#1st row of dataset
ypred1=regressor.predict([[0.0063,18,2.31,0,0.538,6.575,65.2,4.09,1,296,15.3,396.9,4.98]])
print(ypred1)
```

```
Test case 1: 1st row of dataset
[30.08291307]
```

Test 2: In the test 2 we use the data of the last row of the data set. And the result is, medv=23.619.

```
print("Test case 2: Last row of dataset")
#Last row
ypred2=regressor.predict([[0.04741,0,11.93,0,0.573,6.03,80.8,2.505,1,273,21,396.9,7.88]])
print(ypred2)
```

```
Test case 2: Last row of dataset
[23.61910355]
```

Test 3: In test 3 we take the average of all column of the data set. And the result is, medv=22.172 and the average value of medv=20.021 through which we can infer the test case resulted in a higher value.

```
print("Test case 3: Average of every columns of dataset")
#Average
ypred3=regressor.predict([[3.613,11.36,11.136,0.0691,0.554,6.284,68.574,3.795,9.549,408.237,18.455,356.674,12.653]])
print(ypred3)
```

```
Test case 3: Average of every columns of dataset
[22.17205611]
```

Test 4: In the test 4 we are using the highest value of Lstat column and average values of other columns of the dataset. And the result is, medv value=0.6433 which is much lower than the average medv value when Lstat is higher which states that Lstat is inversely proportional to medv value.

```
print("Test case 4: Highest Lstat all others are avarage of dataset")
#highest Lstat all others are avarage
ypred4=regressor.predict([[3.613,11.36,11.136,0.0691,0.554,6.284,68.574,3.795,9.549,408.237,18.455,356.674,37.97]])
print(ypred4)
```

```
Test case 4: Highest Lstat all others are avarage of dataset
[0.64336259]
```

Test 5: In the test 6 we are using the highest value of ptratio and the average value of other column of data set. And the result is, medv=18.609 which is lower than the average of medv value found in dataset.

```
print("Test case 6: Highest ptratio all other are avarage of dataset")
#highest ptratio all other are avarage
ypred6=regressor.predict([[3.613,11.36,11.136,0.0691,0.554,6.284,68.574,3.795,9.549,408.237,22,356.674,12.653]])
print(ypred6)
```

```
Test case 6: Highest ptratio all other are avarage of dataset
[18.60933488]
```

Test 6: In the test 9 we are using the highest value of dis and the average value of other column of data set. And the result is 9.886.

```
print("Test case 9: Highest dis all other are avarage of dataset")
#highest dis all other are avarage
ypred9=regressor.predict([[3.613,11.36,11.136,0.0691,0.554,6.284,68.574,12.1265,9.549,408.237,18.455,356.674,12.653]])
print(ypred9)
```

```
Test case 9: Highest dis all other are avarage of dataset
[9.88611987]
```

Test 7: In the test 10 we are using highest value of crime and the average value of other column of data set. And the result is 34.811 which is higher than the average medv value found in the dataset.

```
print("Test case 10: Highest crime all other are avarage of dataset")
#highest crime all other are avarage
ypred10=regressor.predict([[88.9762,11.36,11.136,0.0691,0.554,6.284,68.574,3.795,9.549,408.237,18.455,356.674,12.653]])
print(ypred10)
```

```
Test case 10: Highest crime all other are avarage of dataset
[34.81107733]
>>>
```

INFERENCE:

From the previous test case analysis, we found that

- When average values of columns were given predicted medv value resulted in a higher value than the average of medv value found from dataset.
- Lstat, tax, ptratio, dis and medv values were in a inversely proportional relationship during test cases.

CONCLUSION

TASK 1 concluded that in this Boston Dataset we need not to clean the data, the dataset is already cleaned and there are no missing values or unrequired/trash values present in this dataset. **TASK 2** declared and displayed the relationships between the data in the dataset and visualized them. In **TASK 3** we created a machine learning algorithm which would be able to predict the values of 'medv' while using all the other values as a base we also found out the accuracy and mean squared error of our model and mentioned the methods to make the algorithm more efficient. In **TASK 4** we provided our model with ten different yet meaningful test cases and found out a lot more about the dataset and the model we were testing on. Throughout this project we made a machine learning regression project from a dataset and we learned and obtained several insights about regression models and how Machine learning algorithms works.