COURSE CODE : INT-254

PROJECT REPORT

ON

Boston House Pricing Prediction Project

Submitted in partial fulfilment of the requirements for the assignment

Of

B. Tech (INT-254)

In

Computer Science & Engineering (Hons.)

Submitted to:

LOVELY PROFESSIONAL UNIVERSITY

Submitted by:

1. K.S.S.Vinayak (12001772)
2. M.Prudhvi (12004601)

Under the Guidance of

Dr.Dhanpratap Singh

Student Declaration

This is to declare that this report has been written by me/us. No part of the report is copied from other sources. All information included from other sources have been duly acknowledged. I/We aver that if any part of the report is found to be copied, I/we are shall take full responsibility for it.

Signature of Students:

Vinay

Prudhvi

TABLE OF CONTENTS

BONAFIDE CERTIFICATE

Certified that this project report “Boston House Pricing Prediction” is the bonafide work of “Vinayak , Prudhvi” who carried out the project work under my supervision.

Signature of the Supervisor

Dr.Dhanpratap Singh

(25706)

INTRODUCTION

The Boston housing dataset is small, especially in today's age of big data. But there was a time where neatly collected and labeled data was extremely hard to access, so a publicly available dataset like this was very valuable to researchers. And although we now have things like Kaggle and open government initiatives which give us plenty of datasets to choose from, this one is a staple to machine learning practice as chocolate is to a break-up.

Each of the 506 rows in the dataset describes a Boston suburb or town, and it has 14 columns with information such as average number of rooms per dwelling, pupil-teacher ratio, and per capita crime rate. The last row describes the median price of owner-occupied homes (this leaves out homes that are rented out), and it's usually the row that we are trying to predict when we use it for regression tasks.

Motivation :

The Motivation behind it I just wanna know about the house prices in Boston as well as I've an idea about to do some of the useful things do in the lock down period.

About the Dataset :

Housing prices are an important reflection of the economy, and housing price ranges are of great interest for both buyers and sellers. In this project, house prices will be predicted given explanatory variables that cover many aspects of residential houses. The goal of this project is to create a regression model that is able to accurately estimate the price of the house given the features.

In this dataset made for predicting the Boston House Price Prediction. Here I just show the all of the feature for each house separately. Such as Number of Rooms, Crime rate of the House's Area and so on. We'll show in the upcoming part.

Output & Conclusion :

From the Exploratory Data Analysis, we could generate insight from the data. How each of the features relates to the target. Also, it can be seen from the evaluation of three models that Random Forest Regressor performed better than Linear Regression.

Data Overview

	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

1. **CRIM** per capital crime rate by town

2. **ZN** proportion of residential land zoned for lots over 25,000 sq.ft.

3. **INDUS** proportion of non-retail business acres per town

4. **CHAS** Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

5. **NOX** nitric oxides concentration (parts per 10 million)

6. **RM** average number of rooms per dwelling

7. **AGE** proportion of owner-occupied units built prior to 1940

8. **DIS** weighted distances to five Boston employment centers

9. **RAD** index of accessibility to radial highways

10.**TAX** full-value property-tax rate per 10,000 USD

11. **PTRATIO** pupil-teacher ratio by town

12. **Black** $1000(Bk — 0.63)^2$ where Bk is the proportion of blacks by town

13. **LSTAT** % lower status of the population

About the Algorithms used in

The major aim of in this project is to predict the house prices based on the features using some of the regression techniques and algorithms.

1. Linear Regression

2. Random Forest Regressor

Machine Learning Packages are used for in this Project

Data Collection

I got the Dataset from Kaggle. This Dataset consist several features such as Number of Rooms, Crime Rate, and Tax and so on. Let's know about how to read the dataset into the Jupyter Notebook. You can download the dataset from Kaggle in csv file format.

As well we can also able to get the dataset from the sklearn datasets. Yup! It's available into the sklearn Dataset.

Importing necessary libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

Importing data modelling libraries

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

from sklearn import metrics

from sklearn import svm
```

Assigning the name of the column in the form of 'list'

```
column_names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']
```

Importing the dataset in the variable 'df' and assigning the column names as specified above

```
df = pd.read_csv('C:/Users/vinay/OneDrive/Desktop/Machine Learning/housing.csv', header=None, delimiter=r"\s+", names=column_names)
```

Viewing the top 5 values of the dataset

```
df.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

viewing the bottom 5 values of the dataset

```
df.tail()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90	7.88	11.9

Shape of the dataset

```
df.shape
```

```
(506, 14)
```

Checking if they are any null values present in the dataset

```
df.isnull().sum()
```

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

There are no null values in the dataset!

Function to identify numeric features:

```
def numeric_features(dataset):
    numeric_col = dataset.select_dtypes(include=np.number).columns.tolist()
    return dataset[numeric_col].head()

numeric_columns = numeric_features(df)
print("Numerical Features:")
print(numeric_columns)

print("====="*20)
```

```
Numerical Features:
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD    TAX  \
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296.0
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242.0
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242.0
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222.0
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222.0

   PTRATIO       B  LSTAT  MEDV
0     15.3  396.90   4.98  24.0
1     17.8  396.90   9.14  21.6
2     17.8  392.83   4.03  34.7
3     18.7  394.63   2.94  33.4
4     18.7  396.90   5.33  36.2
================================================================================
```

Function to identify categorical features:

```python
def categorical_features(dataset):
    categorical_col = dataset.select_dtypes(exclude=np.number).columns.tolist()
    return dataset[categorical_col].head()

categorical_columns = categorical_features(df)
print("Categorical Features:")
print(categorical_columns)

print("====="*20)
```

```
Categorical Features:
Empty DataFrame
Columns: []
Index: [0, 1, 2, 3, 4]
================================================================================
```

Function to check the datatypes of all the columns:

```python
def check_datatypes(dataset):
    return dataset.dtypes

print("Datatypes of all the columns:")
check_datatypes(df)
```

```
Datatypes of all the columns:
CRIM       float64
ZN         float64
INDUS      float64
CHAS         int64
NOX        float64
RM         float64
AGE        float64
DIS        float64
RAD          int64
TAX        float64
PTRATIO    float64
B          float64
LSTAT      float64
MEDV       float64
dtype: object
```

Detecting the outliers in the continuous columns

```
def detect_outliers(df):
    cols = list(df)
    outliers = pd.DataFrame(columns = ['Feature', 'Number of Outliers'])
    for column in cols:
        if column in df.select_dtypes(include=np.number).columns:
            q1 = df[column].quantile(0.25)
            q3 = df[column].quantile(0.75)
            iqr = q3 - q1
            fence_low = q1 - (1.5*iqr)
            fence_high = q3 + (1.5*iqr)
            outliers = outliers.append({'Feature':column, 'Number of Outliers':df.lo
    return outliers

detect_outliers(df)
```

	Feature	Number of Outliers
0	CRIM	66
1	ZN	68
2	INDUS	0
3	CHAS	35
4	NOX	0
5	RM	30
6	AGE	0
7	DIS	5
8	RAD	0
9	TAX	0
10	PTRATIO	15
11	B	77
12	LSTAT	7
13	MEDV	40

Function to plot histograms

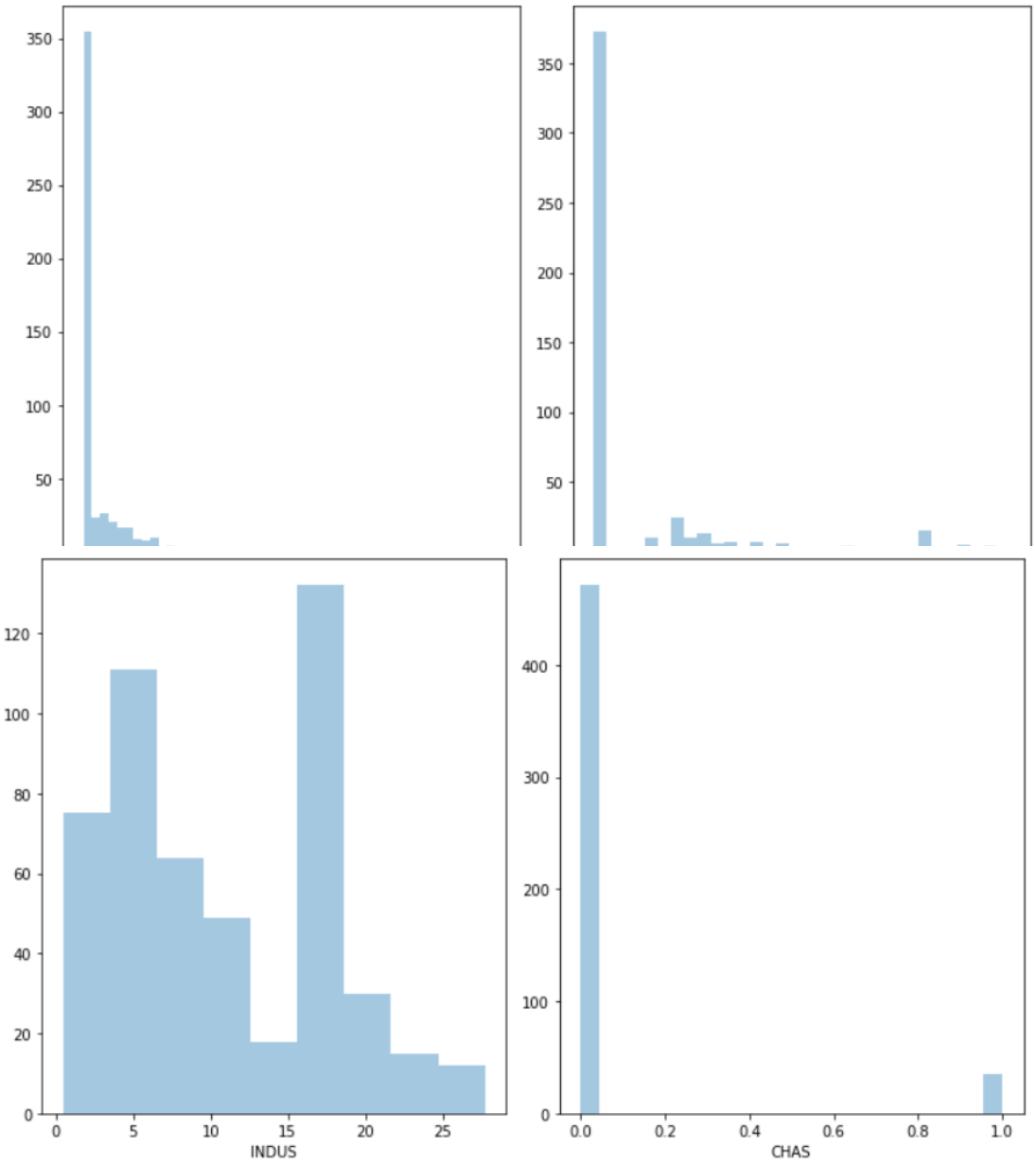
```
def plot_continuous_columns(dataframe):
    numeric_columns = dataframe.select_dtypes(include=['number']).columns.tolist()
    dataframe = dataframe[numeric_columns]

    for i in range(0,len(numeric_columns),2):
        if len(numeric_columns) > i+1:
            plt.figure(figsize=(10,6))
            plt.subplot(121)
            sns.distplot(dataframe[numeric_columns[i]], kde=False)
            plt.subplot(122)
            sns.distplot(dataframe[numeric_columns[i+1]], kde=False)
            plt.tight_layout()
            plt.show()

        else:
            sns.distplot(dataframe[numeric_columns[i]], kde=False)
```

Function to plot boxplots

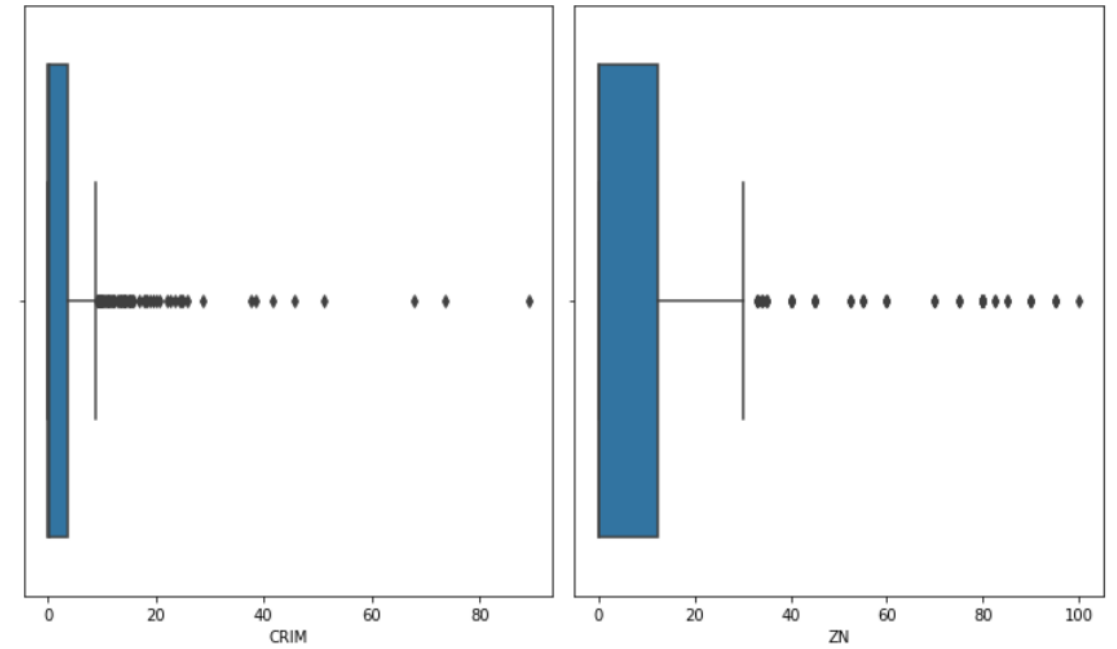
```
def plot_box_plots(dataframe):
    numeric_columns = dataframe.select_dtypes(include=['number']).columns.tolist()
    dataframe = dataframe[numeric_columns]

    for i in range(0,len(numeric_columns),2):
        if len(numeric_columns) > i+1:
            plt.figure(figsize=(10,6))
            plt.subplot(121)
            sns.boxplot(dataframe[numeric_columns[i]])
            plt.subplot(122)
            sns.boxplot(dataframe[numeric_columns[i+1]])
            plt.tight_layout()
            plt.show()

        else:
            sns.boxplot(dataframe[numeric_columns[i]])
```

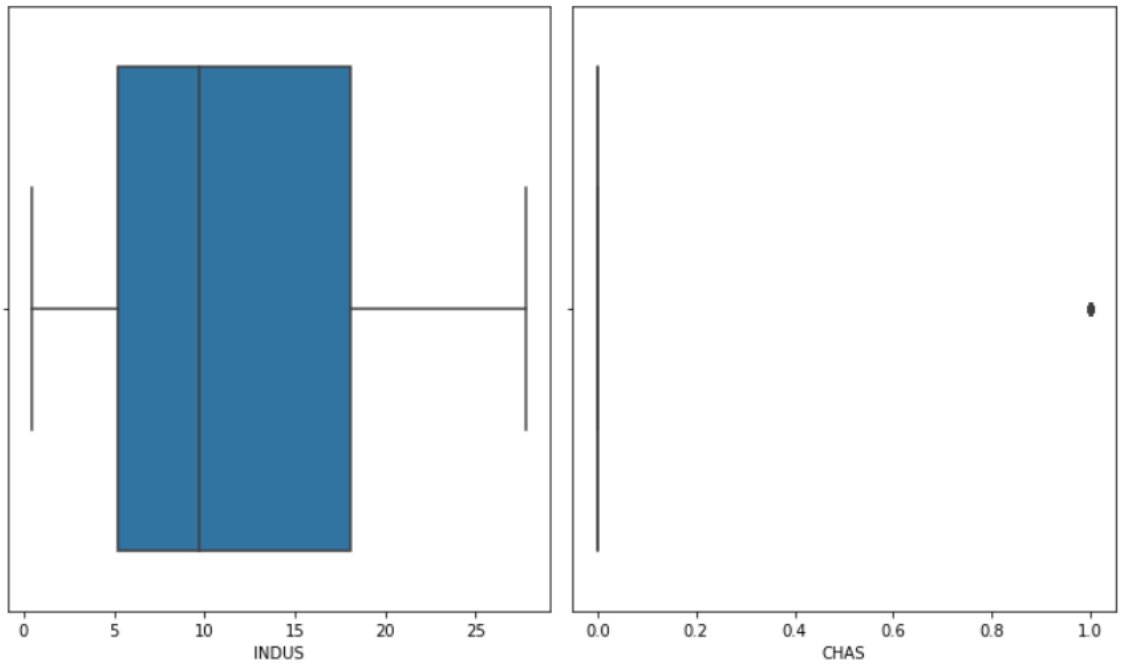
```
print("Histograms\n")
plot_continuous_columns(df)

print("====="*30)
print('\nBox Plots\n')
plot_box_plots(df)
```

Histograms



Box Plots





Observations:

- The columns CRIM,ZN,B and MEDV are heavily skewed. This is due to the presence of the Outliers present in our dataset. We will deal with outliers in the upcoming steps.
- We can see that the values in the column CHAS are almost 0. This means that Charles River dummy variables are all 0, which in turn means that tract does not bound rivers.
- Since the features CHAS consist majorly only of a single value, it's variance is quite less and hence we can drop it since technically will be of no help in prediction.

Treating outliers in the continuous columns

```
from scipy.stats.mstats import winsorize
```

Function to treat outliers

```
def treat_outliers(dataframe):
    cols = list(dataframe)
    for col in cols:
        if col in dataframe.select_dtypes(include=np.number).columns:
            dataframe[col] = winsorize(dataframe[col], limits=[0.05, 0.1],inclusive=(True, True))

    return dataframe


df = treat_outliers(df)

# Checking for outliers after applying winsorization
# We see this using a fuction called 'detect_outliers', defined above.

detect_outliers(df)
```

We can see that the outliers are removed. The outliers, shown above, in columns CRIM,ZN and B are actually not outliers. They are the majority values present in out dataset.

Prediction of house Price

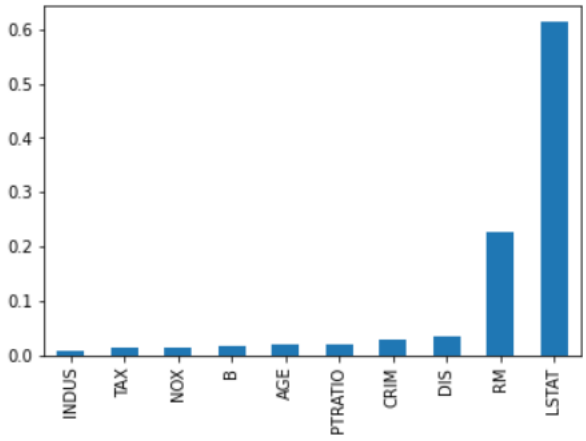
```
# Predictors

x = df.iloc[:,:-1]

# This means that we are using all the columns, except 'MEDV', to predict the house price

# Target

y = df.iloc[:,-1]

# This is because MEDV is the 'Median value of owner-occupied homes in $1000s'.
# This shows that this is what we need to predict. So we call it the target variable.
```

Feature Selection using Random Forest

Random Forests are often used for feature selection in a data science workflow. This is because the tree based strategies that random forests use, rank the features based on how well they improve the purity of the node. The nodes having a very low impurity get split at the start of the tree while the nodes having a very high impurity get split towards the end of the tree. Hence by

pruning the tree after desired amount of splits, we can create a subset of the most important features.

```
def rfc_feature_selection(dataset,target):
    X_train, X_test, y_train, y_test = train_test_split(dataset, target, test_size=0.3, random_state=42)
    rfc = RandomForestRegressor(random_state=42)
    rfc.fit(X_train, y_train)
    y_pred = rfc.predict(X_test)
    rfc_importances = pd.Series(rfc.feature_importances_, index=dataset.columns).sort_values().tail(10)
    rfc_importances.plot(kind='bar')
    plt.show()

rfc_feature_selection(x,y)
```



Observation:

- We can see that the Important features are sorted in ascending order, along with their importance in the form of bar graph.
- We can clearly observe that LSTAT, RM, DIS and CRIM are the most important features that can be used for prediction.

```
x.head(2)
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.02763	18.0	2.31	0	0.538	6.575	65.2	4.0900	2	296.0	15.3	396.9	4.98
1	0.02763	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.9	9.14

Modifying the Predictors to improve the effeciency of the model

```
x= x[['CRIM','DIS','RM','LSTAT']]
x.head(2)
```

]:

	CRIM	DIS	RM	LSTAT
0	0.02763	4.0900	6.575	4.98
1	0.02763	4.9671	6.421	9.14

Scaling the feature variables using MinMaxScaler :

```
mms= MinMaxScaler()
x = pd.DataFrame(mms.fit_transform(x), columns=x.columns)
```

```
x.head()
```

	CRIM	DIS	RM	LSTAT
0	0.000000	0.490733	0.686656	0.066013
1	0.000000	0.654441	0.603458	0.280557
2	0.000000	0.654441	1.000000	0.017019
3	0.000439	0.858839	0.915181	0.000000
4	0.003833	0.858839	0.995678	0.084064

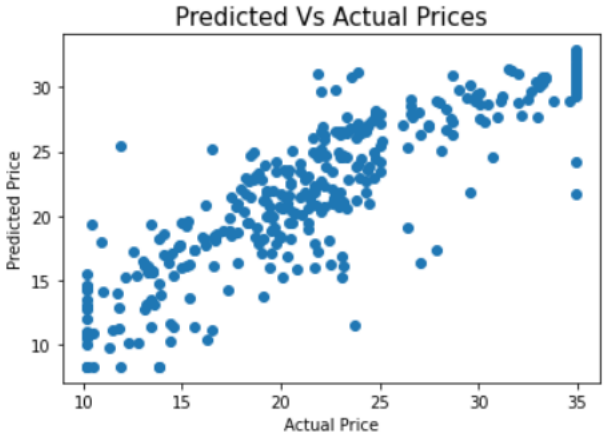
1. Linear Regression

```
lr=LinearRegression()

lr.fit(xtrain, ytrain)

coefficients=pd.DataFrame([xtrain.columns, lr.coef_]).T
coefficients=coefficients.rename(columns={0:'Attributes',1:'Coefficients'})
```

```
coefficients
```

	Attributes	Coefficients
0	CRIM	-3.877522
1	DIS	-2.6488
2	RM	7.640161
3	LSTAT	-13.810714

```
# visualizing the difference between the actual and predicted price

plt.scatter(ytrain, y_pred)
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Predicted Vs Actual Prices", fontsize=15)
plt.show()
```



2. Random Forest

```
rfr= RandomForestRegressor()

rfr.fit(xtrain, ytrain)
```

```
print("R^2: ",metrics.r2_score(ytrain, y_pred))
print("Adusted R^2: ", 1-(1-metrics.r2_score(ytrain, y_pred))*(len(ytrain)-1)/(len(ytrain)-xtrain.shape[1]-1))
print("MAE: ", metrics.mean_absolute_error(ytrain, y_pred))
print("MSE: ", metrics.mean_squared_error(ytrain, y_pred))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(ytrain, y_pred)))

print("\nMaximum Error: ",metrics.max_error(ytrain, y_pred))
```

```
R^2:  0.9744040124204983
Adusted R^2:  0.9741106486660055
MAE:  0.8105683705026889
MSE:  1.318620107198445
RMSE:  1.1483118510223802

Maximum Error:  5.916000000000011
```

3. Support Vector Machine (SVM)

```
svm_reg=svm.SVR()
svm_reg.fit(xtrain, ytrain)
```

```
print("R^2: ",metrics.r2_score(ytrain, y_pred))
print("Adusted R^2: ", 1-(1-metrics.r2_score(ytrain, y_pred))*(len(ytrain)-1)/(len(ytrain)-xtrain.shape[1]-1))
print("MAE: ", metrics.mean_absolute_error(ytrain, y_pred))
print("MSE: ", metrics.mean_squared_error(ytrain, y_pred))
print("RMSE: ",np.sqrt(metrics.mean_squared_error(ytrain, y_pred)))

print("\nMaximum Error: ",metrics.max_error(ytrain, y_pred))
```

Output :

R^2: 0.8309969913608581
Adusted R^2: 0.8290599941271717
MAE: 2.1579841631330376
MSE: 8.706472632729088
RMSE: 2.950673250756357

Maximum Error: 12.779908114045892

```
models=pd.DataFrame({
    'Model':['Linear Regression', 'Random Forest', 'Support Vector Machine'],
    'R_squared Score':[lin_acc*100, rfr_acc*100,svm_acc*100]
})
models.sort_values(by='R_squared Score', ascending=False)
```

Output:

	Model	R_squared Score
1	Random Forest	80.166575
2	Support Vector Machine	78.995238
0	Linear Regression	69.975824

Observations:

- We can see that thr R_squared value for Linera regression is the lowest and the Random Forest is the highest.
- It means that Linear Regression gives us better results on test data, when compared to the other 2 models.

Work Division

Prudhvi

- Project code
- Project Implementation
- Helping in Report

Vinay

- Project code
- Collecting Data set
- Report

Technologies and FrameWork used

*Linear Regression

*Random Forest

*Matplotlib

*Sklearn

Thank You!