



Vidyavardhini's College of Engineering & Technology Department of Computer Engineering

Experiment No. 1
Analyze the Boston Housing dataset and apply appropriate Regression Technique
Date of Performance:
Date of Submission:



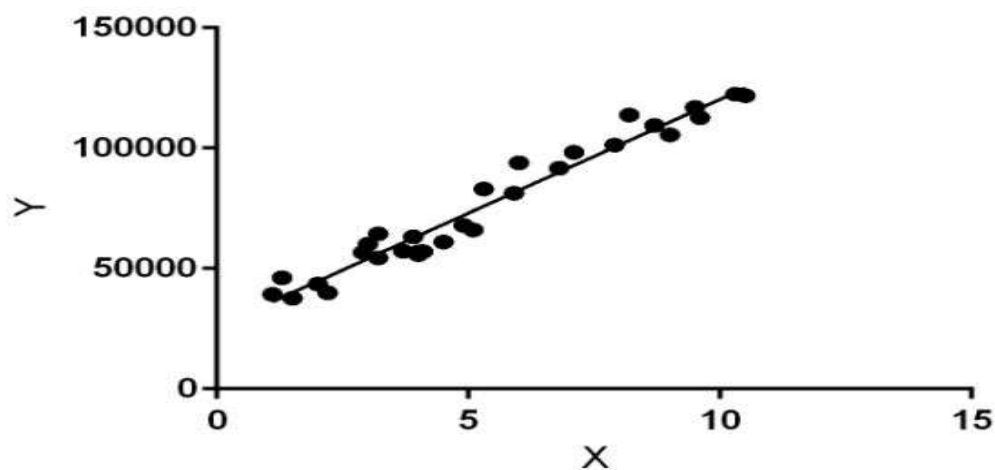
Vidyavardhini's College of Engineering & Technology Department of Computer Engineering

Aim: Analyze the Boston Housing dataset and apply appropriate Regression Technique.

Objective: Ability to perform various feature engineering tasks, apply linear regression on the given dataset and minimise the error.

Theory:

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used.



Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.



Vidyavardhini's College of Engineering & Technology Department of Computer Engineering

In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

Dataset:

The Boston Housing Dataset

The Boston Housing Dataset is derived from information collected by the U.S. Census Service concerning housing in the area of Boston MA. The following describes the dataset columns:

CRIM - per capita crime rate by town

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

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

CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)

NOX - nitric oxides concentration (parts per 10 million)

RM - average number of rooms per dwelling

AGE - proportion of owner-occupied units built prior to 1940

DIS - weighted distances to five Boston employment centres

RAD - index of accessibility to radial highways

TAX - full-value property-tax rate per \$10,000

PTRATIO - pupil-teacher ratio by town

B - $1000(B_k - 0.63)^2$ where B_k is the proportion of blacks by town

LSTAT - % lower status of the population

MEDV - Median value of owner-occupied homes in \$1000's



Vidyavardhini's College of Engineering & Technology Department of Computer Engineering

Code:

```
import numpy as np
import pandas as pd
import os
print(os.listdir("../input"))
from pandas import read_csv
column_names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS',
                'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']
data=read_csv('../input/housing.csv',header=None,delimiter=r"\s+",names=column_names)
print(data.head(5))
```

```
['housing.csv']
   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

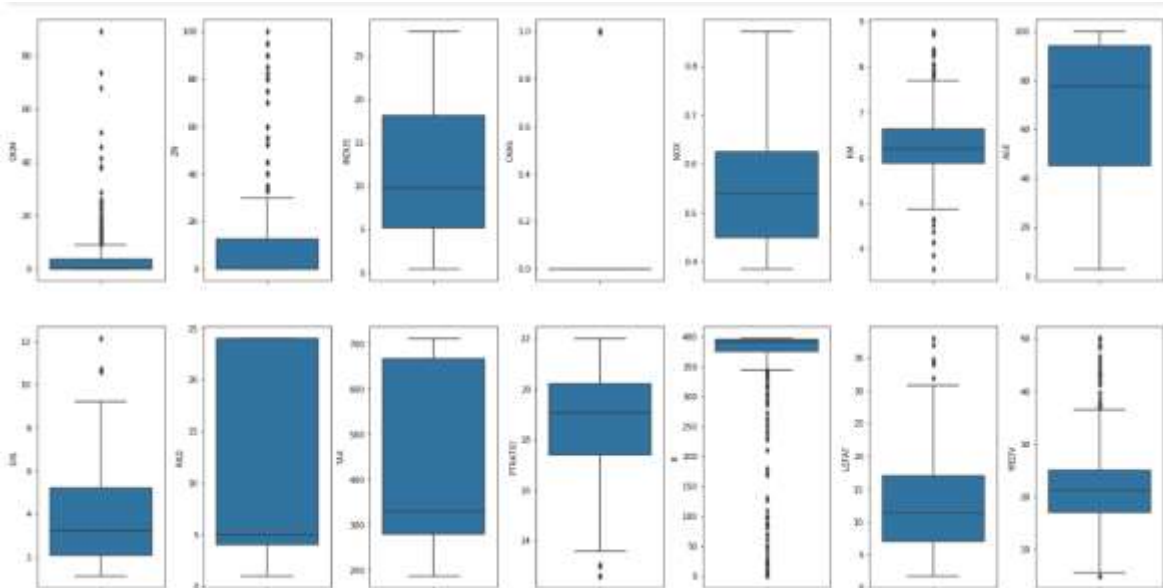
   PTRATIO    B  LSTAT  MEDV
0    15.3  396.90   4.98  24.0
1    17.8  396.90   9.14  21.6
2    17.8  392.83   4.03  34.7
3    18.7  394.63   2.94  33.4
4    18.7  396.90   5.33  36.2
```

```
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
fig, axs = plt.subplots(ncols=7, nrows=2, figsize=(20, 10))
index = 0
```



Vidyavardhini's College of Engineering & Technology Department of Computer Engineering

```
axs = axs.flatten()
for k,v in data.items():
    sns.boxplot(y=k, data=data, ax=axs[index])
    index += 1
plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=5.0)
```



```
for k, v in data.items():
    q1 = v.quantile(0.25)
    q3 = v.quantile(0.75)
    irq = q3 - q1
    v_col = v[(v <= q1 - 1.5 * irq) | (v >= q3 + 1.5 * irq)]
    perc = np.shape(v_col)[0] * 100.0 / np.shape(data)[0]
    print("Column %s outliers = %.2f%%" % (k, perc))
```



Vidyavardhini's College of Engineering & Technology Department of Computer Engineering

```
data = data[~(data['MEDV'] >= 50.0)]  
print(np.shape(data))
```

```
fig, axs = plt.subplots(ncols=7, nrows=2, figsize=(20, 10))
```

```
index = 0
```

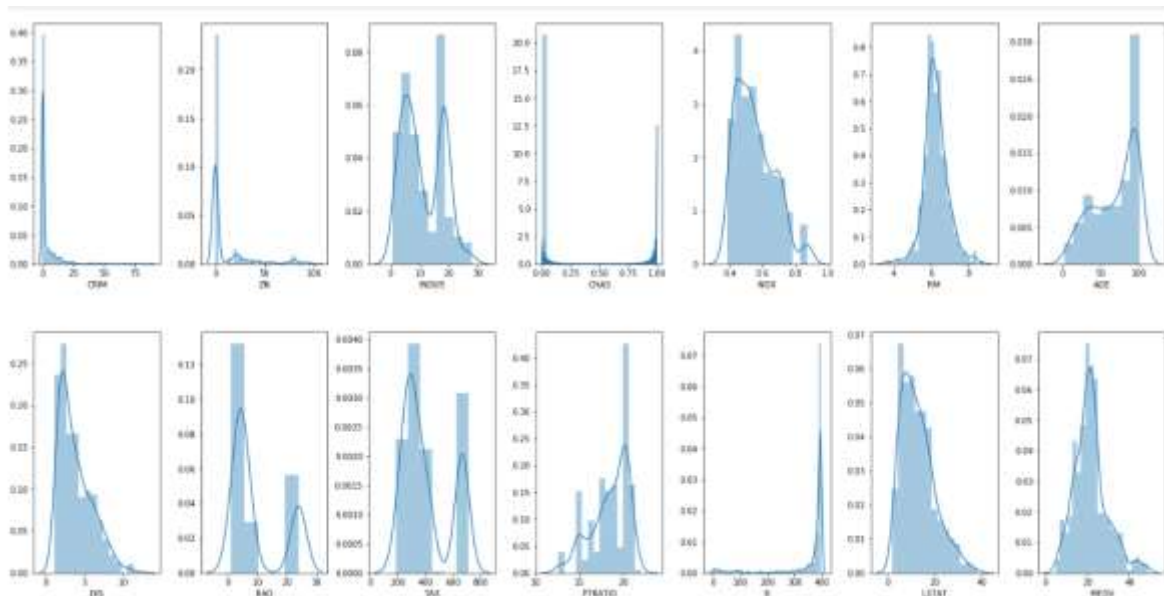
```
axs = axs.flatten()
```

```
for k,v in data.items():
```

```
    sns.distplot(v, ax=axs[index])
```

```
    index += 1
```

```
plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=5.0)
```



```
plt.figure(figsize=(20, 10))
```

```
sns.heatmap(data.corr().abs(), annot=True)
```



Vidyavardhini's College of Engineering & Technology Department of Computer Engineering





Vidyavardhini's College of Engineering & Technology Department of Computer Engineering

```
from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler()
column_sels = ['LSTAT', 'INDUS', 'NOX', 'PTRATIO', 'RM', 'TAX', 'DIS',
               'AGE']
x = data.loc[:,column_sels]
y = data['MEDV']
x=pd.DataFrame(data=min_max_scaler.fit_transform(x),
               columns=column_sels)
fig, axs = plt.subplots(ncols=4, nrows=2, figsize=(20, 10))
index = 0
axs = axs.flatten()
for i, k in enumerate(column_sels):
    sns.regplot(y=y, x=x[k], ax=axs[i])
plt.tight_layout(pad=0.4, w_pad=0.5, h_pad=5.0)
y = np.log1p(y)
for col in x.columns:
    if np.abs(x[col].skew()) > 0.3:
        x[col] = np.log1p(x[col])
from sklearn import datasets, linear_model
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
import numpy as np
l_regression = linear_model.LinearRegression()
kf = KFold(n_splits=10)
min_max_scaler = preprocessing.MinMaxScaler()
```




Vidyavardhini's College of Engineering & Technology Department of Computer Engineering

```
x_scaled = min_max_scaler.fit_transform(x)
scores=cross_val_score(l_regression,x_scaled,y,cv=kf,scoring='neg_mean_squared_error')
print("MSE: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std()))
MSE: -0.04 (+/- 0.04)
```

Conclusion:

Features have been chosen to develop the model:

1. CRIM - Per capita crime rate by town
2. CHAS - Charles River dummy variable (1 if tract bounds river; else 0)
3. NOX - Nitric oxides concentration (parts per 10 million)
4. RM - Average number of rooms per dwelling
5. DIS - weighted distances to five Boston employment centres
6. RAD - Index of accessibility to radial highways
7. TAX - Full-value property-tax rate per \$10,000
8. PTRATIO - Pupil-teacher ratio by town
9. LSTAT - Lower status of the population

Mean Squared Error calculated:

- Calculated Mean Squared Error: 0.04 (+/- 0.04)
- The Mean Squared Error measures how close a regression line is to a set of data points.
- Lesser the Mean Squared Error refers to Smaller is the error and Better the estimator.