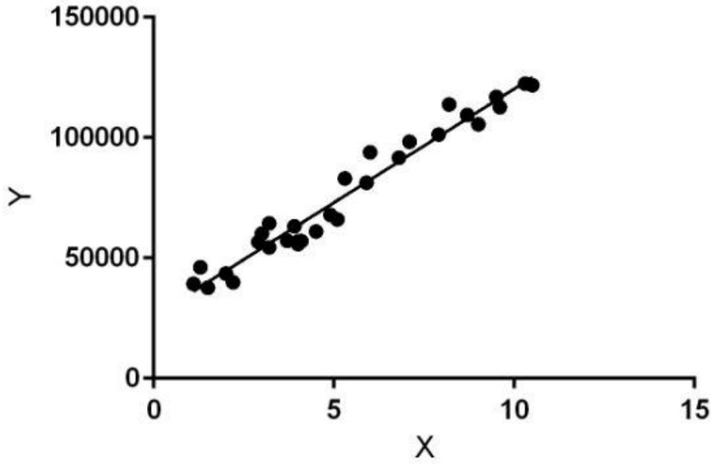
| |
|---|
| Experiment No. 1 |
| Analyze the Boston Housing dataset and apply appropriate Regression Technique |
| Date of Performance: 17-07-2023 |
| Date of Submission: 11-08-2023 |

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

**Objective:** Ablility 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.

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 a 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(Bk - 0.63)^2 where Bk is the proportion of blacks by town

LSTAT - % lower status of the population

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

**Code:**

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

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

```
dataset = pd.read_csv("boston_train.csv")
dataset
dataset.head()
```

| | ID | crim | zn | indus | chas | nox | rm | age | dis | rad | tax | ptratio | black | lstat |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 |
| **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 |
| **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 |
| **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 |
| **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 |

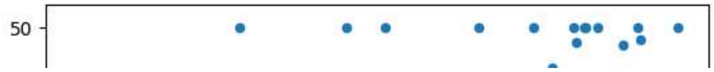
```
dataset.info()
# dataset.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 333 entries, 0 to 332
Data columns (total 15 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   ID       333 non-null    int64  
 1   crim     333 non-null    float64
 2   zn       333 non-null    float64
 3   indus    333 non-null    float64
 4   chas     333 non-null    int64  
 5   nox      333 non-null    float64
 6   rm       333 non-null    float64
 7   age      333 non-null    float64
 8   dis      333 non-null    float64
 9   rad      333 non-null    int64  
 10  tax      333 non-null    int64  
 11  ptratio  333 non-null    float64
 12  black    333 non-null    float64
 13  lstat    333 non-null    float64
 14  medv     333 non-null    float64
dtypes: float64(11), int64(4)
memory usage: 39.1 KB
```

```
dataset = dataset.drop('ID',axis=1)
```

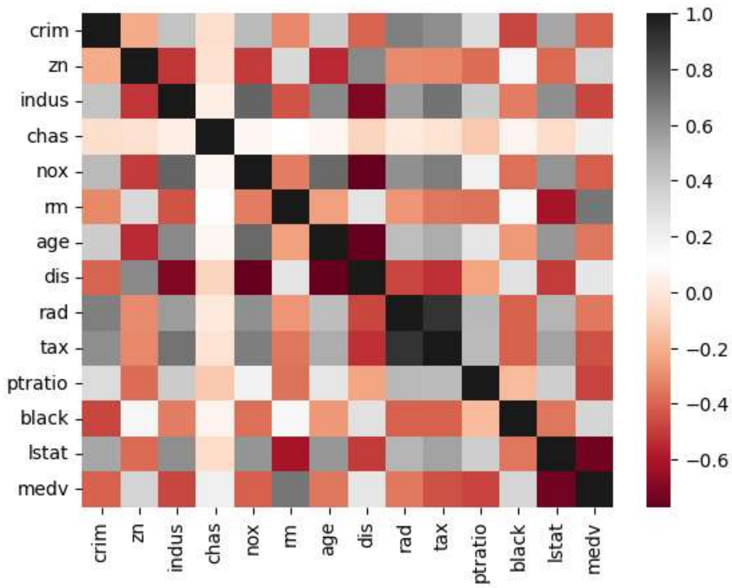
```
dataset.plot.scatter('rm', 'medv')
# dataset.plot.scatter('dis', 'medv')
```

```
<Axes: xlabel='rm', ylabel='medv'>
```




```
sns.heatmap(dataset.corr(), cmap = 'RdGy')
```

```
<Axes: >
```




```
#split x and y
x=dataset[['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax',
       'ptratio', 'black', 'lstat']]
y = dataset['medv']
```

```
# split train and test set
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

```
X_train.head()
```

| | crim | zn | indus | chas | nox | rm | age | dis | rad | tax | ptratio | black | lstat |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **39** | 0.14932 | 25.0 | 5.13 | 0 | 0.453 | 5.741 | 66.2 | 7.2254 | 8 | 284 | 19.7 | 395.11 | 13.15 |
| **147** | 0.14052 | 0.0 | 10.59 | 0 | 0.489 | 6.375 | 32.3 | 3.9454 | 4 | 277 | 18.6 | 385.81 | 9.38 |
| **275** | 37.66190 | 0.0 | 18.10 | 0 | 0.679 | 6.202 | 78.7 | 1.8629 | 24 | 666 | 20.2 | 18.82 | 14.52 |
| **277** | 9.33889 | 0.0 | 18.10 | 0 | 0.679 | 6.380 | 95.6 | 1.9682 | 24 | 666 | 20.2 | 60.72 | 24.08 |
| **259** | 13.35980 | 0.0 | 18.10 | 0 | 0.693 | 5.887 | 94.7 | 1.7821 | 24 | 666 | 20.2 | 396.90 | 16.35 |

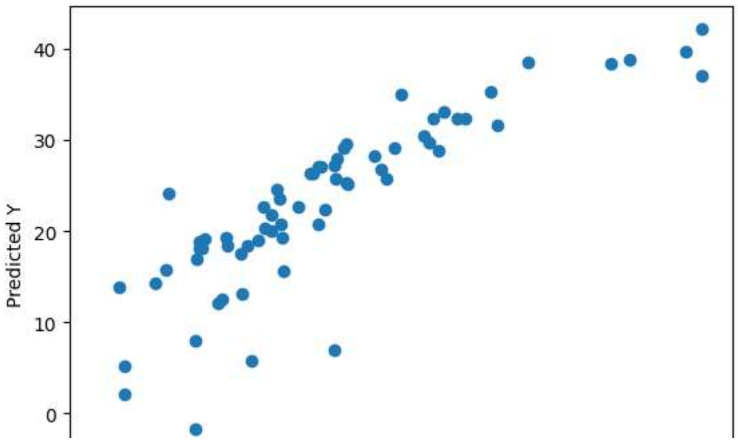
```
lr = LinearRegression()
lr.fit(X_train,y_train)
# print(lr)
```

```
▾ LinearRegression
LinearRegression()
```

```
predictions = lr.predict(X_test)
```

```
plt.scatter(y_test,predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

```
Text(0, 0.5, 'Predicted Y')
```




```
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 3.9970055491851104
MSE: 29.761537920840542
RMSE: 5.455413634257309
```

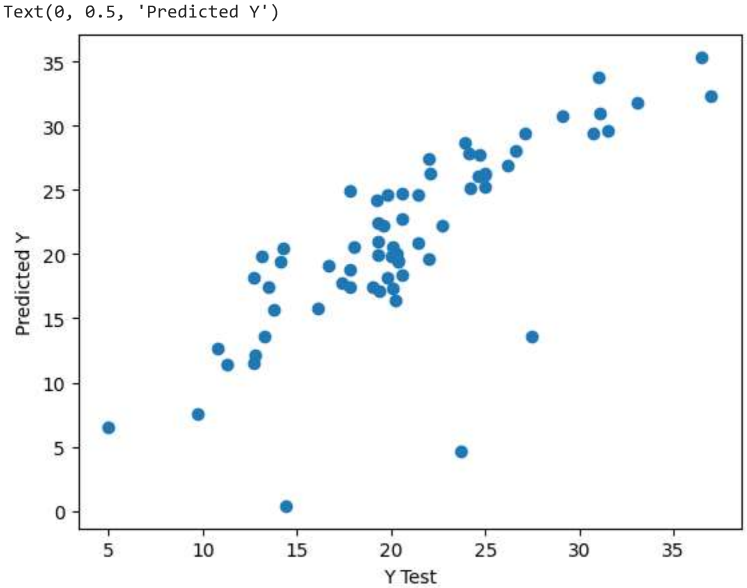
```
x=dataset[['crim', 'indus', 'rm', 'age',  'tax','ptratio', 'lstat']]
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2)

lr = LinearRegression()
lr.fit(X_train,y_train)

predictions = lr.predict(X_test)
```

```
plt.scatter(y_test,predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

```
Text(0, 0.5, 'Predicted Y')
```




```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 2.8576136730269948
MSE: 19.133725606075703
RMSE: 4.37421142676891
```

**Conclusion:**

1. Features used:

Housing Price Dataset:

We initially trained a model using the features ['crim', 'zn', 'indus', 'chas', 'nox', 'rm', 'age', 'dis', 'rad', 'tax', 'ptratio', 'black', 'lstat'] and obtained the following results:

Mean Absolute Error (MAE): 3.997

Mean Squared Error (MSE): 29.762

Root Mean Squared Error (RMSE): 5.455

Feature Reduction and Improvement:

You then performed feature reduction using a heatmap analysis, selecting the features ['crim', 'indus', 'rm', 'age', 'tax', 'ptratio', 'lstat'], which helped in reducing the errors. The results after feature reduction were as follows:

MAE: 2.858

MSE: 19.134

RMSE: 4.374

2:MSA: Mean Squared Error represents the average of the squared difference between the original and predicted values in the data set. It measures the variance of the residuals.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$