11

Preksha A. Patel
60004210126
C2 - 1
comps

Machine Learning

19

Exp 2 :- Model Building using regression

Aim :- To perform linear regression and find the error associated with the model

Description of Experiment :-

Linear regression is one of the easiest and most popular supervised machine learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous / real or numeric values such as sales, salary, age, product, price, etc. Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (x) variables, hence called as linear regression. Since linear regression shows the linear relationship which means it finds how the value of the dependent variable is changing according to the value of the independent variable. The linear regression model provides a sloped straight line representing the relationship between the variables. cleaning data in python we will now separate the numerical column from the categorical column.

Mathematically, we can represent a linear regression as :-

$$y = b_0 + b_1 x + c$$

Here, y = Dependent variable

x = Independent variable

b_0 = intercept of the line

b_1 = Linear regression coeffecient

c = random error

P.T.O.

The different values for weights or coeffecient of lines (b_0, b_1) gives the different lines of regression & the cost function is used to estimate the values of the coeffecient for the best fit line. It cost function optimize the regression coeffecient or weights. It measures how a linear regression model is performing. we can use the cost function to find the accuracy of mapping function, which maps the input variable to the output variable. This mapping function is also known as hypothesis function. for linear regression, we use the MSE Cost function which is the average of square error occurs between the predicted value & actual values. It can be written as :

$$MSE = \frac{1}{N} \sum_{i=1}^{n} (y_i - (b_1 x_i + b_0))^2$$

where N = Total no. of observations

y_i = actual value

$(b_1 x_i + b_0)$ = predicted values.

Linear regression using least square method we have linear regression equation as :

$$y = b_0 + b_1 x$$

using LSM,

$$b_1 = \frac{\Sigma (x_i - \bar{x})(y_i - \bar{y})}{\Sigma (x_i - \bar{x})^2}$$

$$b_0 = \bar{y} - b_1 \bar{x}$$

Page No.
Date / /

Conclusion :-

In the "Model-building using regression" experiment, we delved into linear regression, a foundational supervised Machine Learning algorithm. By understanding the linear relationship between dependent (y) and independent (x) variables, we constructed a regression model. The least square method enabled us to find optimal coeffecients (b_0, b_1) for the best fit line, minimizing the mean squared Error (MSE). This approach ~~et~~ enhances our ability to predict & analyze real-word numeric variables, contributing to effective predictive modeling in data analysis.

NAME:-PREKSHA PATEL
SAPID:-60004210126
BRANCH:-COMPUTER ENGINEERING

MACHINE LEARNING

EXPERIMENT-02

CODE AND OUTPUT:-

Case 1]:-

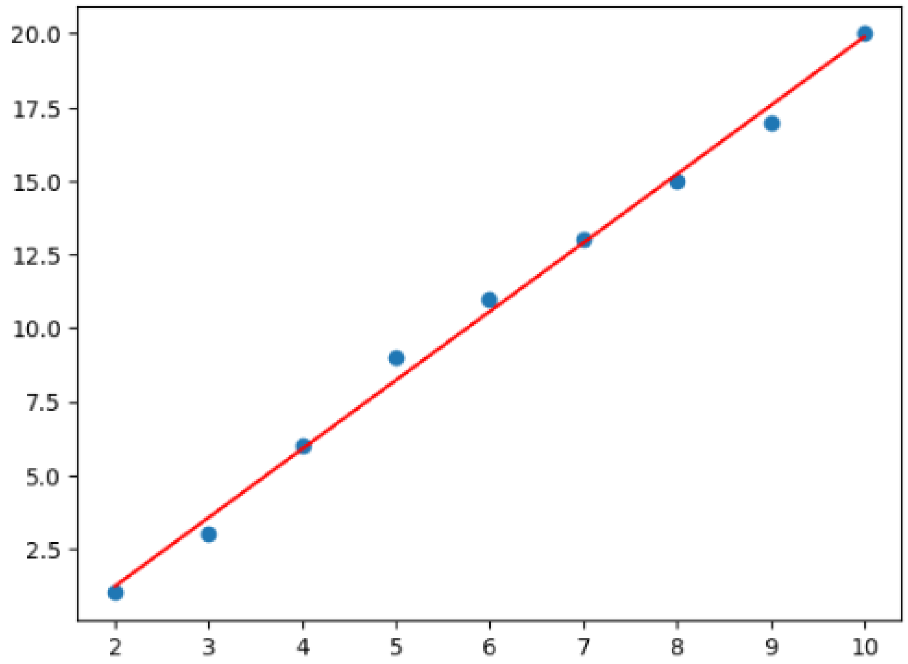
```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
X = np.array([[2], [3], [4], [5], [6], [7], [8], [9], [10]])
Y = np.array([[1], [3], [6], [9], [11], [13], [15], [17], [20]])
```

```
[ ] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
    linearregression_with_sklearn =LinearRegression().fit(X,Y)
    predictions =linearregression_with_sklearn.predict(X_test)

    mse = mean_squared_error(Y_test, predictions)
    rmse = np.sqrt(mse)
    print(f'Root Mean Squared Error: {rmse}')

    Root Mean Squared Error: 0.5555555555555554
```

```
[ ] linearregression_with_sklearn.coef_

    array([[2.33333333]])
```

```
[ ] linearregression_with_sklearn.intercept_

    array([-3.44444444])
```

```
[ ] import matplotlib.pyplot as plt
    plt.scatter(X, Y)
    plt.plot(X, linearregression_with_sklearn.predict(X), color='red')
    plt.show()
```




```
from sklearn.metrics import mean_squared_error
import math
y_pred = linearregression_with_sklearn.predict([[4]])
rmse = math.sqrt(mean_squared_error([6], y_pred))
print("Root mean squared error for x=4:", rmse)
```

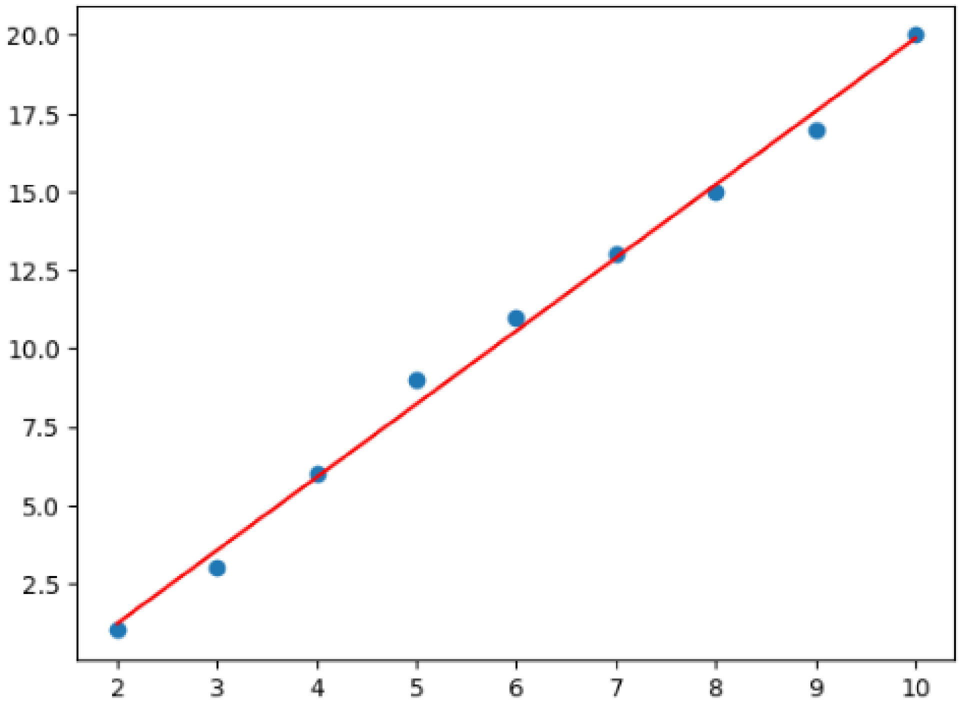
```
Root mean squared error for x=4: 0.11111111111111249
```

```
X = np.array([[2], [3], [4], [5], [6], [7], [8], [9], [10]])
Y = np.array([[1], [3], [6], [9], [11], [13], [15], [17], [20]])
```

[ ]
```
X_mean = np.mean(X)
Y_mean = np.mean(Y)

num = 0
den = 0
for i in range(len(X)):
    num += (X[i] - X_mean)*(Y[i] - Y_mean)
    den += (X[i] - X_mean)**2
b1 = num / den
b0 = Y_mean - b1*X_mean

print (b1, b0)
```

```
[2.33333333] [-3.44444444]
```

```
Y_pred = b1*X + b0

plt.scatter(X, Y) # actual
# plt.scatter(X, Y_pred, color='red')
plt.plot([min(X), max(X)], [min(Y_pred), max(Y_pred)], color='red') # predicted
plt.show()
```



```
def rmse(predictions,target):
  return np.sqrt(((predictions-target)**2).mean())
print(rmse(Y_pred,Y))
```

```
0.41573970964154894
```

```
x=[4]
Y_pred = b1*x +b0
print(rmse(Y_pred,[6]))
```

```
0.11111111111111072
```

case 2

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
X = np.array([[2], [3], [4], [5], [6], [7], [8], [9], [10]])
Y = np.array([[1], [3], [6], [9], [10], [13], [14], [17], [21]])
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
linearregression_with_sklearn =LinearRegression().fit(X,Y)
predictions =linearregression_with_sklearn.predict(X_test)

mse = mean_squared_error(Y_test, predictions)
rmse = np.sqrt(mse)
print(f'Root Mean Squared Error: {rmse}')
```

```
Root Mean Squared Error: 0.4555555555555566
```

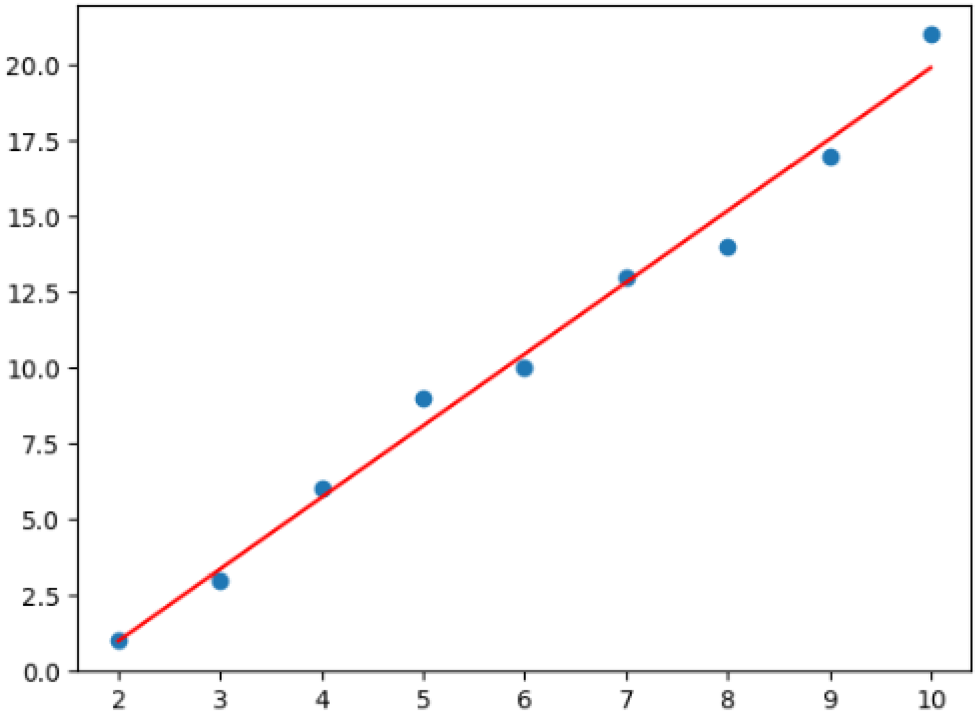
```
linearregression_with_sklearn.coef_
```

```
array([[2.36666667]])
```

```
linearregression_with_sklearn.intercept_
```

```
array([-3.75555556])
```

```
import matplotlib.pyplot as plt
plt.scatter(X, Y)
plt.plot(X, linearregression_with_sklearn.predict(X), color='red')
plt.show()
```




```
[ ] from sklearn.metrics import mean_squared_error
    import math
    y_pred = linearregression_with_sklearn.predict([[4]])
    rmse = math.sqrt(mean_squared_error([6], y_pred))
    print("Root mean squared error for x=4:", rmse)
```

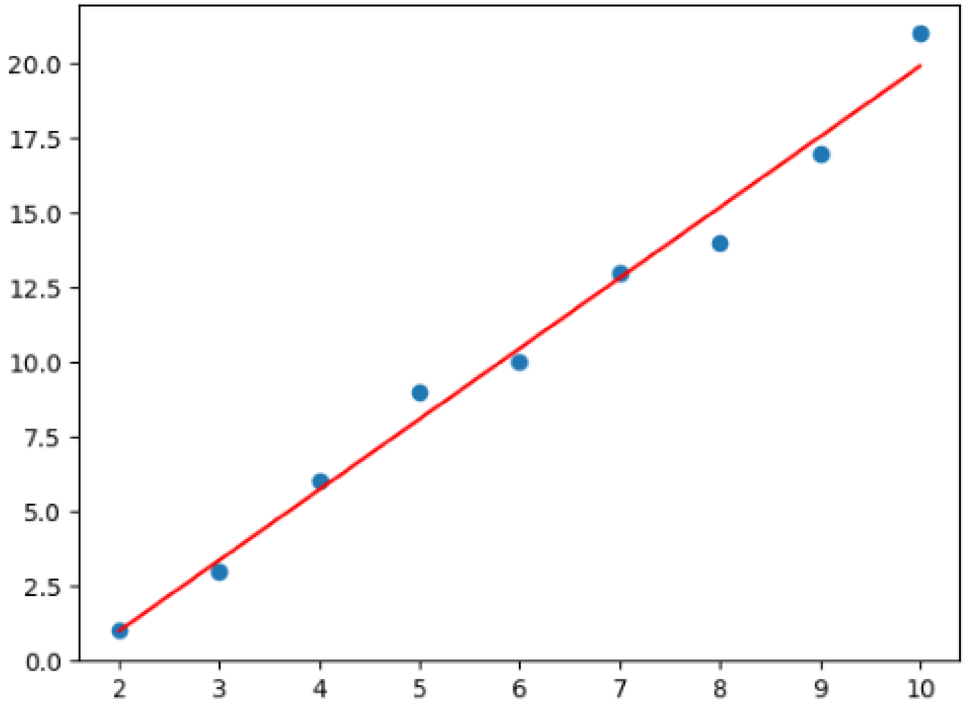
```
Root mean squared error for x=4: 0.28888888888888786
```

```
[ ] X = np.array([[2], [3], [4], [5], [6], [7], [8], [9], [10]])
    Y = np.array([[1], [3], [6], [9], [10], [13], [14], [17], [21]])
```

```
[ ] X_mean = np.mean(X)
    Y_mean = np.mean(Y)

    num = 0
    den = 0
    for i in range(len(X)):
        num += (X[i] - X_mean)*(Y[i] - Y_mean)
        den += (X[i] - X_mean)**2
    b1 = num / den
    b0 = Y_mean - b1*X_mean

    print (b1, b0)
```

```
[2.36666667] [-3.75555556]
```

```
[ ] Y_pred = b1*X + b0

    plt.scatter(X, Y) # actual
    # plt.scatter(X, Y_pred, color='red')
    plt.plot([min(X), max(X)], [min(Y_pred), max(Y_pred)], color='red') # predicted
    plt.show()
```



```
[ ] def rmse(predictions,target):
      return np.sqrt(((predictions-target)**2).mean())
    print(rmse(Y_pred,Y))
```

```
0.6795059933964734
```

```
[ ] x=[4]
    Y_pred = b1*x +b0
    print(rmse(Y_pred,[6]))
```

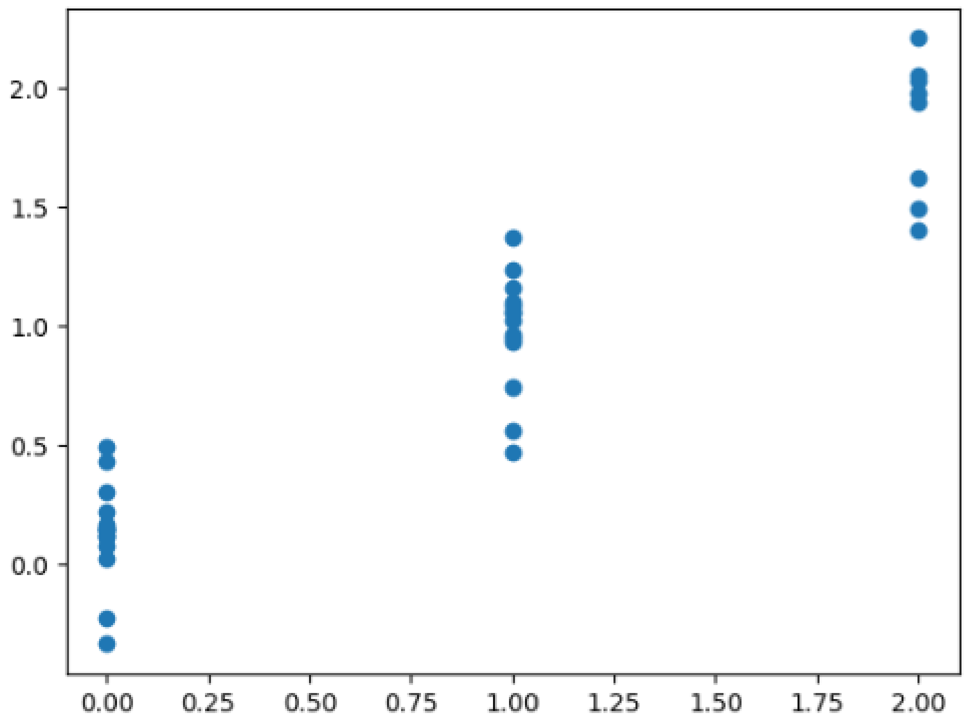
```
0.28888888888888786
```

using dataset

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import datasets

wine = datasets.load_wine()
X = wine.data
y = wine.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()

model.fit(X_train, y_train)

predictions = model.predict(X_test)

mse = mean_squared_error(y_test, predictions)
rmse = np.sqrt(mse)
print(rmse)
```

```
0.2617890078719129
```

```
[ ] import matplotlib.pyplot as plt
    plt.scatter( y_test,predictions)
    plt.show()
```


2.0
1.5
1.0
0.5
0.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00