

AMAN KUMAR PANDEY

RA1911003010685

ARTIFICIAL INTELLIGENCE LAB

EXPERIMENT NO: 10

IMPLEMENTATION OF A LEARNING
ALGORITHM – LINEAR REGRESSION

Working Principle:

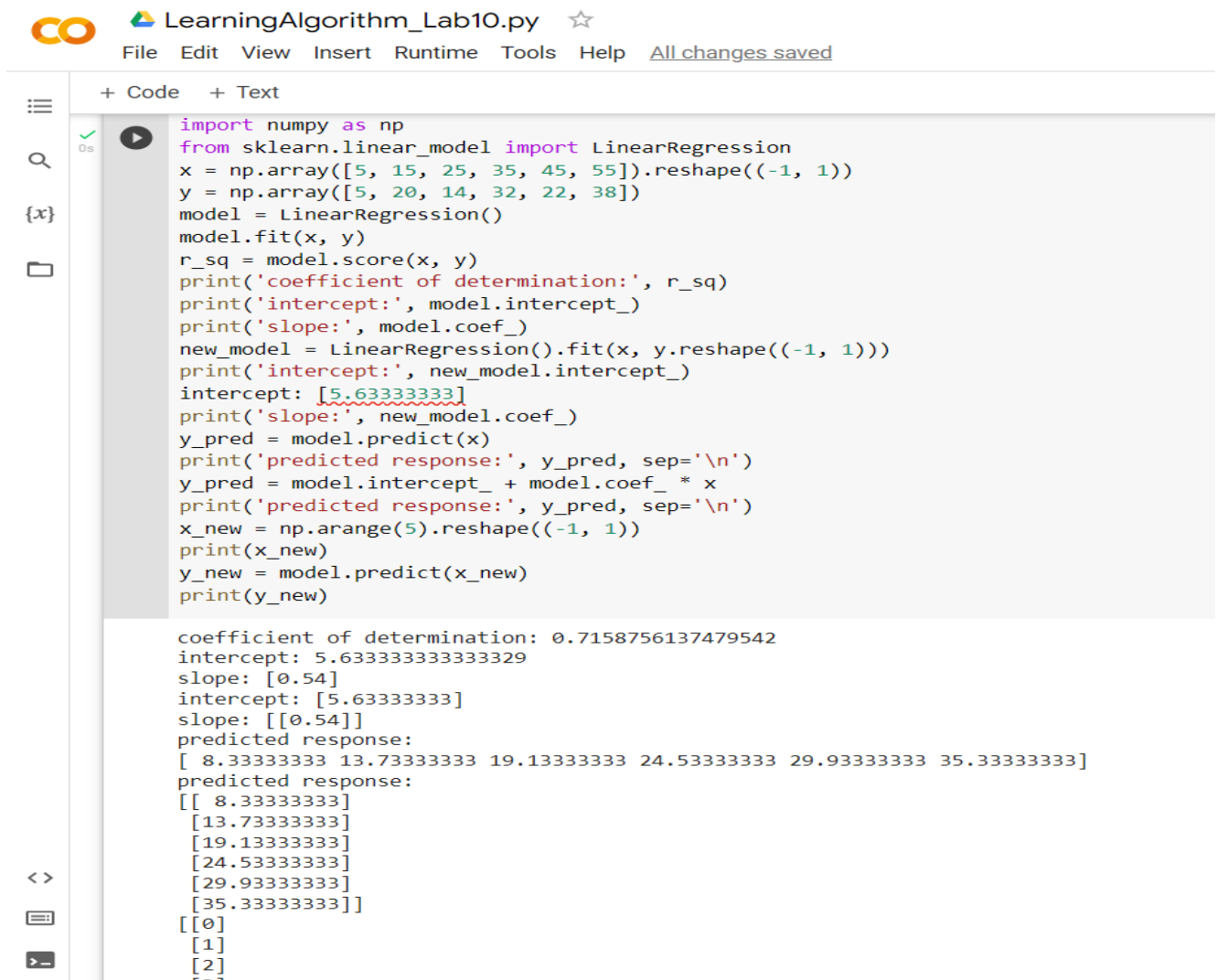
Linear regression shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis). To calculate best-fit line linear regression uses a traditional slope-intercept form. A regression line can be a Positive Linear Relationship or a Negative Linear Relationship.

The goal of the linear regression algorithm is to get the best values for a_0 and a_1 to find the best fit line and the best fit line should have the least error. In Linear Regression, **Mean Squared Error (MSE)** cost function is used, which helps to figure out the best possible values for a_0 and a_1 , which provides the best fit line for the data points. Using the MSE function, we will change the values of a_0 and a_1 such that the MSE value settles at the minima. Gradient descent is a method of updating a_0 and a_1 to minimize the cost function (MSE)

Source code:

```
import numpy as np
from sklearn.linear_model import LinearRegression
x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))
y = np.array([5, 20, 14, 32, 22, 38])
model = LinearRegression() model.fit(x, y)
r_sq = model.score(x, y)
print('coefficient of determination:', r_sq)
print('intercept:', model.intercept_)
print('slope:', model.coef_)
new_model = LinearRegression().fit(x, y.reshape((-1, 1)))
print('intercept:', new_model.intercept_)
intercept: [5.63333333] print('slope:', new_model.coef_)
y_pred = model.predict(x)
print('predicted response:', y_pred, sep='\n')
y_pred = model.intercept_ + model.coef_ * x
print('predicted response:', y_pred, sep='\n')
x_new = np.arange(5).reshape((-1, 1))
print(x_new)
y_new = model.predict(x_new)
print(y_new)
```

Output:



LearningAlgorithm_Lab10.py ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

```
import numpy as np
from sklearn.linear_model import LinearRegression
x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))
y = np.array([5, 20, 14, 32, 22, 38])
model = LinearRegression()
model.fit(x, y)
r_sq = model.score(x, y)
print('coefficient of determination:', r_sq)
print('intercept:', model.intercept_)
print('slope:', model.coef_)
new_model = LinearRegression().fit(x, y.reshape((-1, 1)))
print('intercept:', new_model.intercept_)
intercept: [5.63333333]
print('slope:', new_model.coef_)
y_pred = model.predict(x)
print('predicted response:', y_pred, sep='\n')
y_pred = model.intercept_ + model.coef_ * x
print('predicted response:', y_pred, sep='\n')
x_new = np.arange(5).reshape((-1, 1))
print(x_new)
y_new = model.predict(x_new)
print(y_new)
```

coefficient of determination: 0.7158756137479542
intercept: 5.633333333333329
slope: [0.54]
intercept: [5.63333333]
slope: [[0.54]]
predicted response:
[8.33333333 13.73333333 19.13333333 24.53333333 29.93333333 35.33333333]
predicted response:
[[8.33333333
[13.73333333
[19.13333333
[24.53333333
[29.93333333
[35.33333333]
[[0]
[1]
[2]
[3]
[4]

Result:

Hence, the Implementation of Linear Regression as a machine learning algorithm is done successfully.