**AIM: IMPLEMENTATION OF LINEAR REGRESSION & LOGISTIC REGRESSION**

1. **LINEAR REGRESSION:**

**THEORY:**

Linear regression is a basic and commonly used type of predictive analysis. The overall idea of regression is to examine two things:

1. does a set of predictor variables do a good job in predicting an outcome (dependent) variable?
2. Which variables in particular are significant predictors of the outcome variable, and in what way do they–indicated by the magnitude and sign of the beta estimates–impact the outcome variable?

These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables. The simplest form of the regression equation with one dependent and one independent variable is defined by the formula y = c + b\*x, where y = estimated dependent variable score, c = constant, b = regression coefficient, and x = score on the independent variable

**SOURCE CODE & OUTPUT:**

1. **IMPORTING LIBRARIES:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

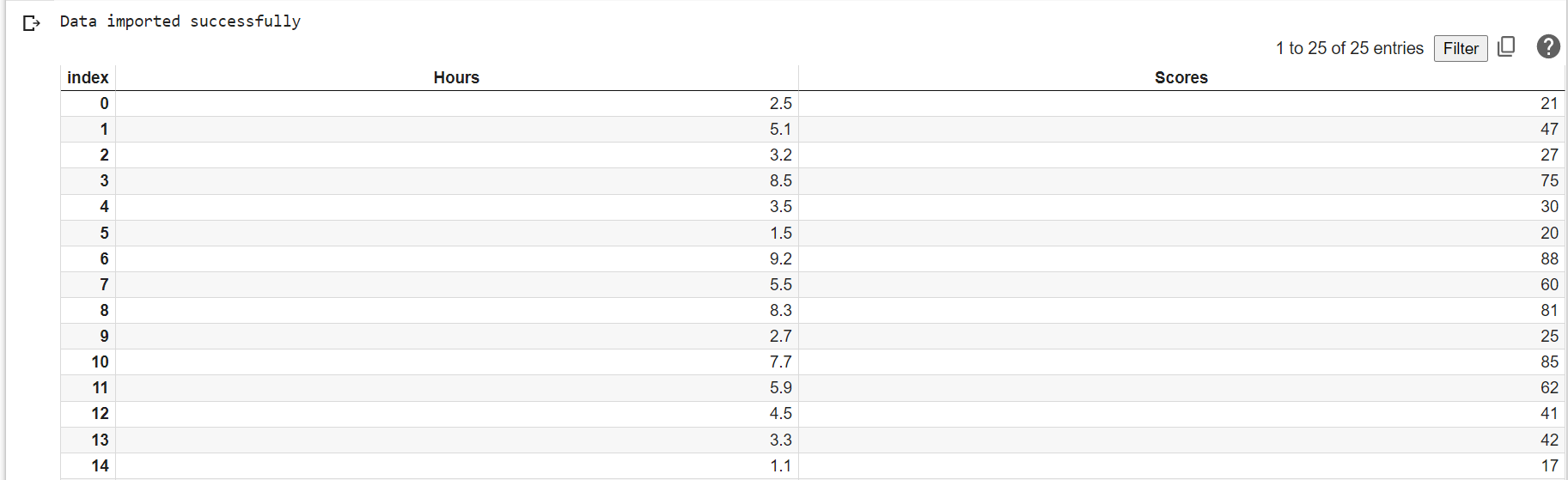
1. **COLLECTING DATA:**

url = "http://bit.ly/w-data"

data = pd.read\_csv(url)

print("Data imported successfully")

data



1. **Plotting the distribution of scores:**

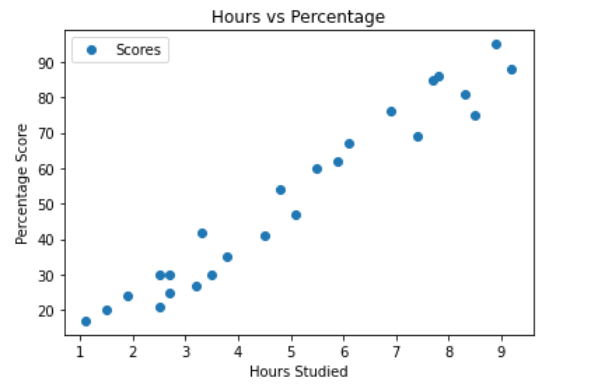
data.plot(x='Hours', y='Scores', style='o')

plt.title('Hours vs Percentage')

plt.xlabel('Hours Studied')

plt.ylabel('Percentage Score')

plt.show()



1. **TRAINING DATA:**

#Training Data

X = data.iloc[:, :-1].values

y = data.iloc[:, 1].values

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X\_train, y\_train)

print("Training complete.")



1. **PLOTTING THE REGRESSION LINE:**

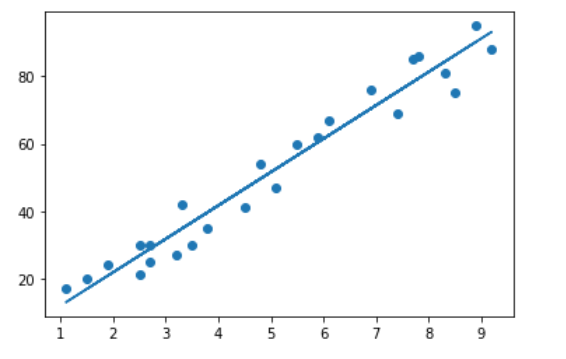
line = regressor.coef\_\*X+regressor.intercept\_

# Plotting for the test data

plt.scatter(X, y)

plt.plot(X, line);

plt.show()



1. **PREDICTING THE SCORES:**

y\_pred = regressor.predict(X\_test)

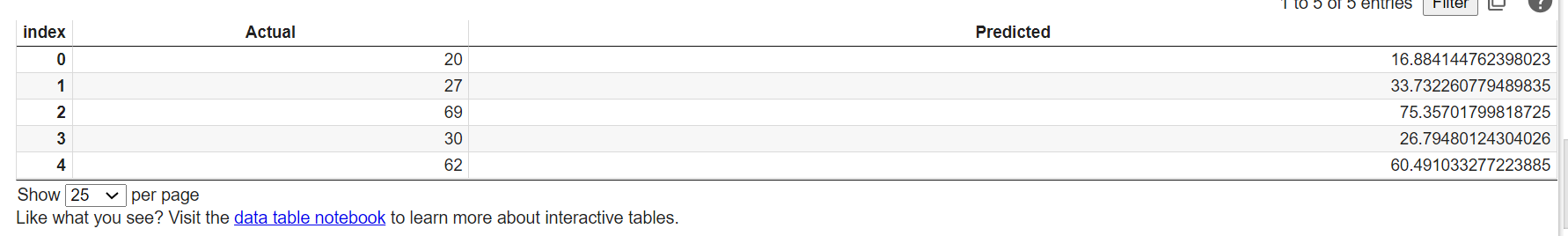
y\_pred



1. **COMPARING ACTUAL VS PREDICTED:**

df = pd.DataFrame({'Actual': y\_test, 'Predicted': y\_pred})

df



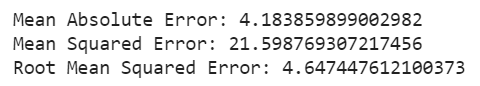
1. **EVALUATING THE ALGORITHM:**

from sklearn import metrics

print('Mean Absolute Error:', metrics.mean\_absolute\_error(y\_test, y\_pred))

print('Mean Squared Error:', metrics.mean\_squared\_error(y\_test, y\_pred))

print('Root Mean Squared Error:', np.sqrt(metrics.mean\_squared\_error(y\_test, y\_pred)))



1. **LOGISTIC REGRESSION:**

**THEORY:**

In statistics, the logistic model (or logit model) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. Each object being detected in the image would be assigned a probability between 0 and 1, with a sum of one.

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression[1] (or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1".

**SOURCE CODE & OUTPUT:**

1. **COLLECTING DATA:**

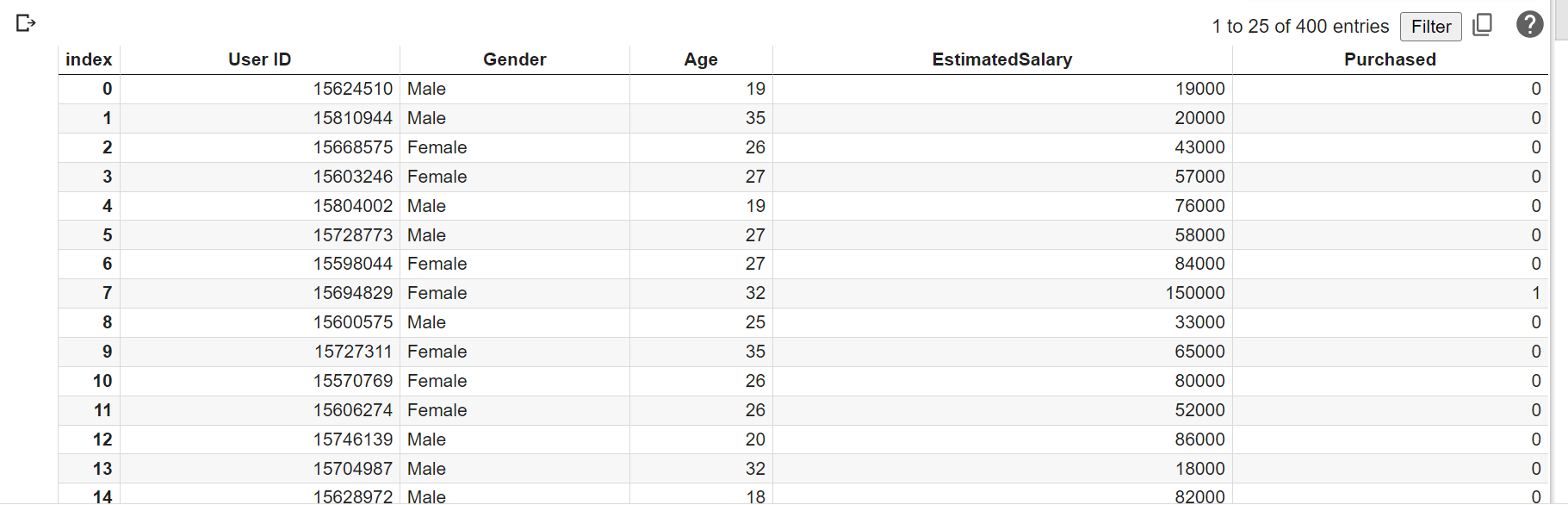
import numpy as nm

import matplotlib.pyplot as mtp

import pandas as pd

data\_set= pd.read\_csv('suv\_data.csv')

data\_set



1. **EXTRACTING INDEPENDENT AND DEPENDENT VARIABLE:**

x = data\_set.iloc[:, [2,3]].values

y = data\_set.iloc[:, 4].values

1. **SPLITTING THE DATASET INTO TRAINING AND TEST SET:**

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test= train\_test\_split(x, y, test\_size= 0.25, random\_state=0)

1. **FEATURE SCALING:**

from sklearn.preprocessing import StandardScaler

st\_x= StandardScaler()

x\_train= st\_x.fit\_transform(x\_train)

x\_test= st\_x.transform(x\_test)

1. **FITTING LOGISTIC REGRESSION TO THE TRAINING SET:**

from sklearn.linear\_model import LogisticRegression

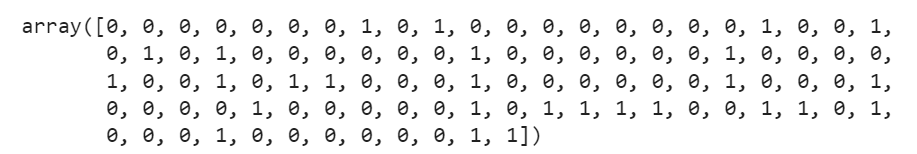
classifier= LogisticRegression(random\_state=0)

classifier.fit(x\_train, y\_train)

1. **PREDICTING THE TEST SET RESULT:**

y\_pred= classifier.predict(x\_test)

y\_pred

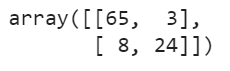


1. **CREATING THE CONFUSION MATRIX:**

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

cm



1. **VISUALIZING THE TRAINING SET RESULT:**

from matplotlib.colors import ListedColormap

x\_set, y\_set = x\_train, y\_train

x1, x2 = nm.meshgrid(nm.arange(start = x\_set[:, 0].min() - 1, stop = x\_set[:, 0].max() + 1, step =0.01),

                     nm.arange(start = x\_set[:, 1].min() - 1, stop = x\_set[:, 1].max() + 1, step = 0.01))

mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),

             alpha = 0.75, cmap = ListedColormap(('purple','green' )))

mtp.xlim(x1.min(), x1.max())

mtp.ylim(x2.min(), x2.max())

for i, j in enumerate(nm.unique(y\_set)):

  mtp.scatter(x\_set[y\_set == j, 0], x\_set[y\_set == j, 1],

              c = ListedColormap(('purple', 'green'))(i), label = j)

mtp.title('Logistic Regression (Training set)')

mtp.xlabel('Age')

mtp.ylabel('Estimated Salary')

mtp.legend()

mtp.show()



1. **VISULAIZING THE TEST SET RESULT:**

from matplotlib.colors import ListedColormap

x\_set, y\_set = x\_test, y\_test

x1, x2 = nm.meshgrid(nm.arange(start = x\_set[:, 0].min() - 1, stop = x\_set[:, 0].max() + 1, step =0.01),

                     nm.arange(start = x\_set[:, 1].min() - 1, stop = x\_set[:, 1].max() + 1, step = 0.01))

mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.

shape), alpha = 0.75, cmap = ListedColormap(('purple','green' )))

mtp.xlim(x1.min(), x1.max())

mtp.ylim(x2.min(), x2.max())

for i, j in enumerate(nm.unique(y\_set)):

  mtp.scatter(x\_set[y\_set == j, 0], x\_set[y\_set == j, 1],

              c = ListedColormap(('purple', 'green'))(i), label = j)

mtp.title('Logistic Regression (Test set)')

mtp.xlabel('Age')

mtp.ylabel('Estimated Salary')

mtp.legend()

mtp.show()



**CONCLUSION:**

From this practical, I have learned about Implementation of Linear and logistic regression in python.