**Experiment-4:**

Exercises to solve real world problems using the following machine learning methods:

a) Linear regression b) Logistic regression c) Binary classifier

1. Linear regression

**#Implementation**

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])

print('x',x)

print('y',y)

model = LinearRegression()

model.fit(x, y)

#model = LinearRegression().fit(x, y)

r\_sq = model.score(x, y)

print(f"coefficient of determination: {r\_sq}")

print(f"intercept: {model.intercept\_}")

print(f"slope: {model.coef\_}")

new\_model = LinearRegression().fit(x, y.reshape((-1, 1)))

print(f"intercept: {new\_model.intercept\_}")

print(f"slope: {new\_model.coef\_}")

y\_pred = model.predict(x)

print(f"predicted response:\n{y\_pred}")

y\_pred = model.intercept\_ + model.coef\_ \* x

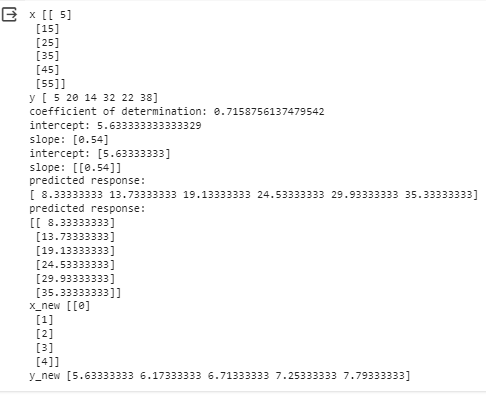
print(f"predicted response:\n{y\_pred}")

x\_new = np.arange(5).reshape((-1, 1))

print('x\_new',x\_new)

y\_new = model.predict(x\_new)

print('y\_new',y\_new)



**a1)Linear model coefficients**

from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]

y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std\_err = stats.linregress(x, y)

print(r)

print(slope)

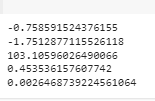
print(intercept)

print(std\_err)

print(p)

#The r value ranges from -1 to 1, where 0 means no relationship, and 1 (and -1) means 100% related.

Output



1. **Logistic Regression**

import numpy

from sklearn import linear\_model

X = numpy.array([3.78, 2.44, 2.09, 0.14, 1.72, 1.65, 4.92, 4.37, 4.96, 4.52, 3.69, 5.88]).reshape(-1,1)

y = numpy.array([0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1])

logr = linear\_model.LogisticRegression()

logr.fit(X,y)

predicted = logr.predict(numpy.array([3.46]).reshape(-1,1))

print(predicted)

Output:

[0]

1. **Binary classifier**

**#Implementation**

import pandas as pd

import numpy as np

df = pd.read\_csv('/content/sample\_data/Pokemon.csv')

df.head()

df.drop(columns=['#','Name',  'Type 1', 'Type 2'],inplace=True)

df

df.Legendary.value\_counts()

from sklearn import preprocessing

label\_encoder = preprocessing.LabelEncoder()

df['Legendary']= label\_encoder.fit\_transform(df['Legendary'])

df['Legendary'].unique()

df

df.Legendary.value\_counts()

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

pokemon\_features = df.drop("Legendary",axis=1)

target = df["Legendary"]

X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(pokemon\_features,target,test\_size=0.20,random\_state=0)

from sklearn.metrics import accuracy\_score

from sklearn.linear\_model import LogisticRegression

lr = LogisticRegression()

lr.fit(X\_train,Y\_train)

Y\_pred\_lr = lr.predict(X\_test)

score\_lr = round(accuracy\_score(Y\_pred\_lr,Y\_test)\*100,2)

print("The accuracy score achieved using Logistic Regression is: "+str(score\_lr)+" %")

