**Experiment-4**

4.AIM: Exercises to solve the real-world problems using the following machine learning methods:

a) Linear regression:

import numpy as np

from sklearn.linear\_model import LinearRegression

import matplotlib.pyplot as plt

# Sample data

X = np.array([2, 1, 3]).reshape(-1, 1) # Feature (reshape to a column vector)

y = np.array([92, 86, 89]) # Target

# Create a linear regression model

model = LinearRegression()

# Fit the model to the data

model.fit(X, y)

# Make predictions

predictions = model.predict(X)

# Print the coefficients

print("Intercept:", model.intercept\_)

print("Coefficient:", model.coef\_[0])

# Print actual and predicted outputs

print("Actual Output:", y)

print("Predicted Output:", predictions)

# Plot the original data and the regression line

plt.scatter(X, y, color='black')

plt.plot(X, predictions, color='blue', linewidth=3)

plt.xlabel('X')

plt.ylabel('y')

plt.title('Linear Regression')

plt.show()

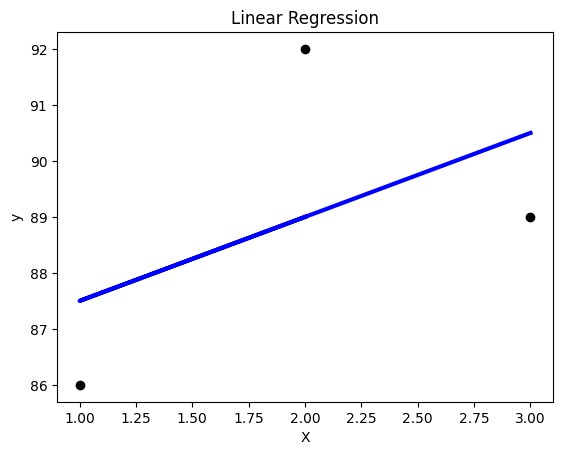
Output:

Intercept: 86.0

Coefficient: 1.4999999999999993

Actual Output: [92 86 89]

Predicted Output: [89. 87.5 90.5]



B) **Logistic Regression:**

import matplotlib.pyplot as plt

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)

def myfunc(x):

return slope \* x + intercept

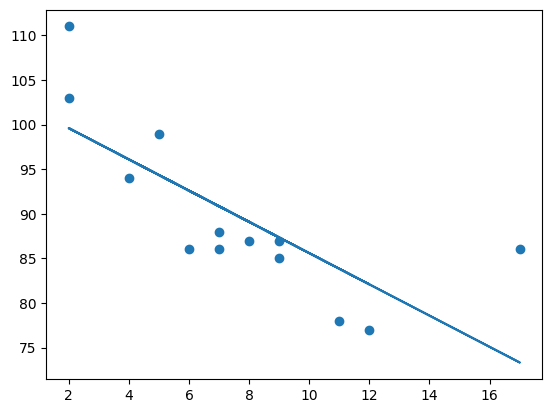
mymodel = list(map(myfunc, x))

plt.scatter(x, y)

plt.plot(x, mymodel)

plt.show()

Output:



**c) Binary Classifier:**

import warnings

import pandas as pd

from sklearn.linear\_model import LogisticRegression

from sklearn import svm

from sklearn.ensemble import RandomForestClassifier

from sklearn.neural\_network import MLPClassifier

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

from sklearn.metrics import confusion\_matrix, classification\_report

# Create the heart dataset

heart\_data = {

'sbp': [160, 144, 118, 170, 134],

'tobacco': [12, 0, 0, 7, 0],

'ldl': [5.73, 4.41, 3.48, 6.41, 3.5],

'adiposity': [23.11, 28.61, 32.28, 38.03, 27.78],

'famhist': ['Present', 'Absent', 'Present', 'Present', 'Absent'],

'typea': [49, 55, 52, 51, 60],

'obesity': [25.3, 30.4, 27.7, 42.4, 29.6],

'alcohol': [97.2, 2.06, 3.81, 4.84, 0.0],

'age': [52, 63, 46, 58, 49],

'chd': [1, 1, 0, 1, 0]

}

heart = pd.DataFrame(heart\_data)

# Convert 'famhist' to numerical using one-hot encoding

heart = pd.get\_dummies(heart, columns=['famhist'], drop\_first=True)

# Separate target variable and features for heart dataset

y\_heart = heart['chd']

X\_heart = heart.drop('chd', axis=1)

# Split the heart dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_heart, y\_heart, test\_size=0.2, random\_state=42)

# Logistic Regression

logistic\_regression\_model = LogisticRegression()

logistic\_regression\_model.fit(X\_train, y\_train)

logistic\_regression\_predictions = logistic\_regression\_model.predict(X\_test)

# SVM

svm\_model = svm.SVC(decision\_function\_shape="ovo").fit(X\_train, y\_train)

svm\_predictions = svm\_model.predict(X\_test)

# Random Forest

rf\_model = RandomForestClassifier(n\_estimators=1000, max\_depth=10, random\_state=0).fit(X\_train, y\_train)

rf\_predictions = rf\_model.predict(X\_test)

# Neural Network

nn\_model = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden\_layer\_sizes=(150, 10), random\_state=1).fit(X\_train, y\_train)

nn\_predictions = nn\_model.predict(X\_test)

# Print results for all models

models = ['Logistic Regression', 'SVM', 'Random Forest', 'Neural Network']

for i, model in enumerate([logistic\_regression\_model, svm\_model, rf\_model, nn\_model]):

predictions = model.predict(X\_test)

accuracy = model.score(X\_test, y\_test)

confusion\_mat = confusion\_matrix(y\_test, predictions)

# Suppress warnings and display the classification report

with warnings.catch\_warnings():

warnings.simplefilter("ignore")

classification\_rep = classification\_report(y\_test, predictions, zero\_division=1)

print(f"\nModel: {models[i]}")

print(f"Accuracy: {round(accuracy, 4)}")

print("Confusion Matrix:\n", confusion\_mat)

print("Classification Report:\n", classification\_rep)

Output:

Model: Logistic Regression

Accuracy: 0.0

Confusion Matrix:

[[0 0]

[1 0]]

Classification Report:

precision recall f1-score support

0 0.00 1.00 0.00 0.0

1 1.00 0.00 0.00 1.0

accuracy 0.00 1.0

macro avg 0.50 0.50 0.00 1.0

weighted avg 1.00 0.00 0.00 1.0

Model: SVM

Accuracy: 0.0

Confusion Matrix:

[[0 0]

[1 0]]

Classification Report:

precision recall f1-score support

0 0.00 1.00 0.00 0.0

1 1.00 0.00 0.00 1.0

accuracy 0.00 1.0

macro avg 0.50 0.50 0.00 1.0

weighted avg 1.00 0.00 0.00 1.0

Model: Random Forest

Accuracy: 0.0

Confusion Matrix:

[[0 0]

[1 0]]

Classification Report:

precision recall f1-score support

0 0.00 1.00 0.00 0.0

1 1.00 0.00 0.00 1.0

accuracy 0.00 1.0

macro avg 0.50 0.50 0.00 1.0

weighted avg 1.00 0.00 0.00 1.0

Model: Neural Network

Accuracy: 0.0

Confusion Matrix:

[[0 0]

[1 0]]

Classification Report:

precision recall f1-score support

0 0.00 1.00 0.00 0.0

1 1.00 0.00 0.00 1.0

accuracy 0.00 1.0

macro avg 0.50 0.50 0.00 1.0

weighted avg 1.00 0.00 0.00 1.0