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()

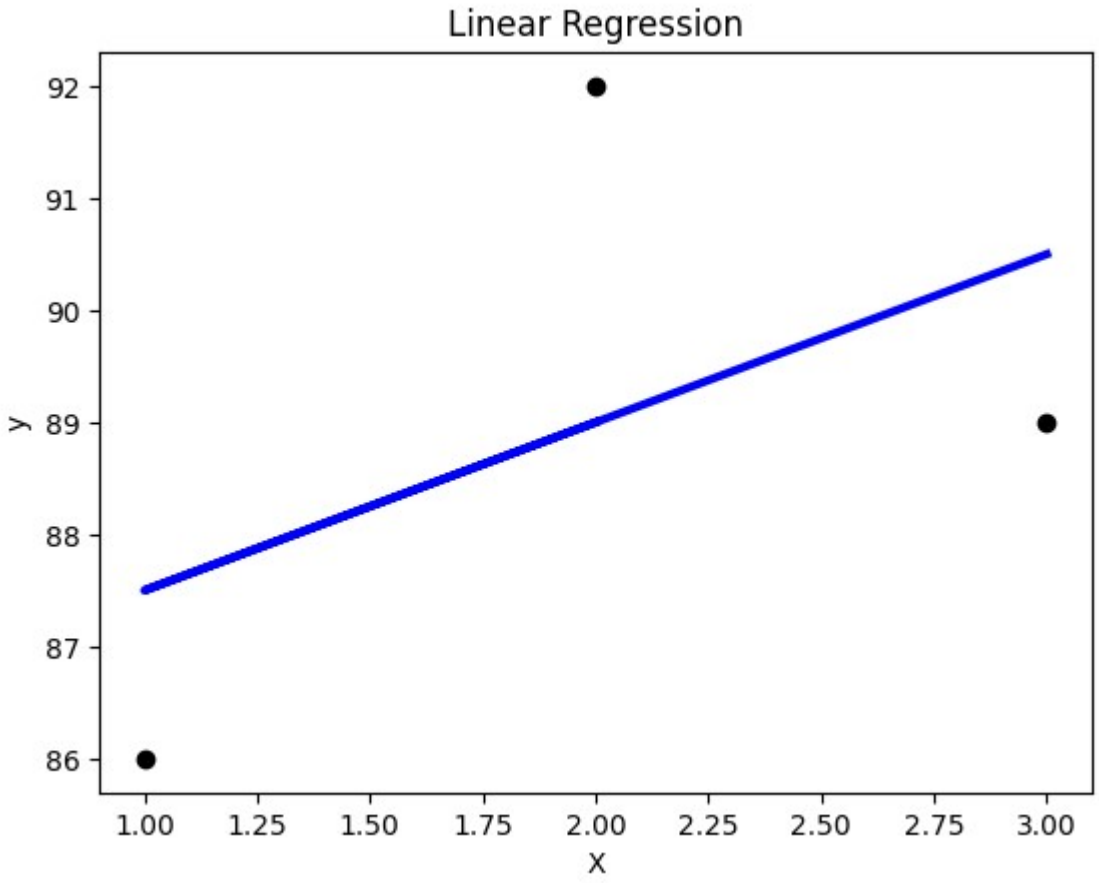
Output:

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
```

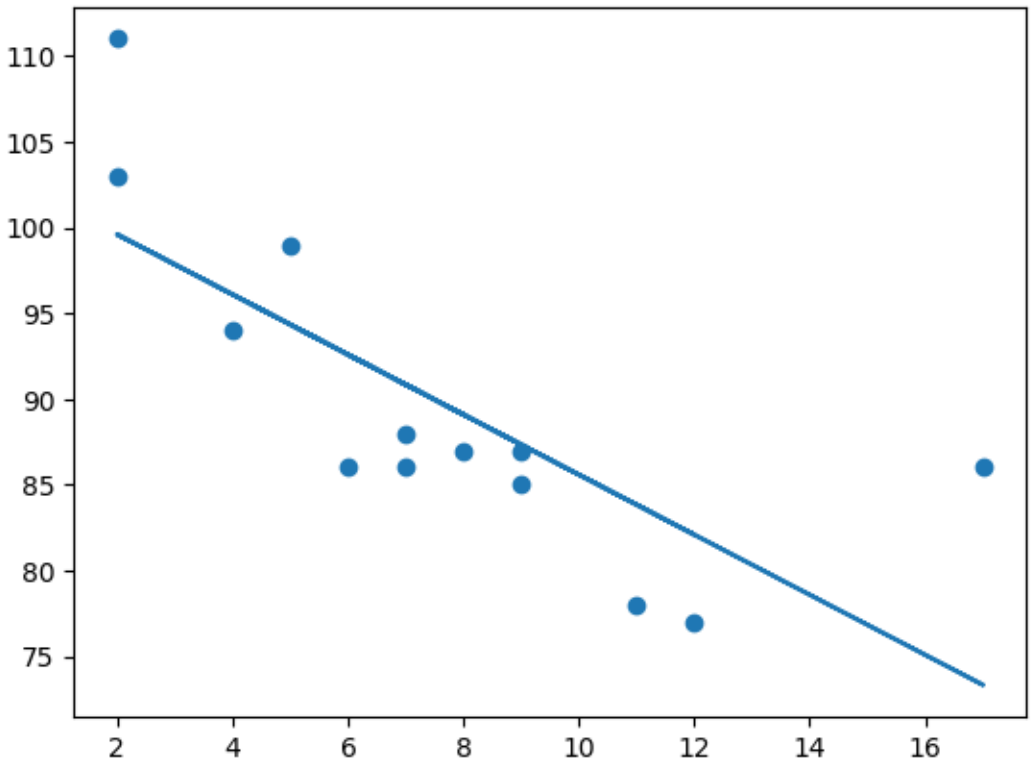
```
mymodel = list(map(myfunc, x))

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
```