ML LAB EXPERIMENTS

- **B Rishitha**
192324130

1. Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples.

```
def get_input():
    n = int(input("Enter number of attributes: "))
    m = int(input("Enter number of training examples: "))
    data = []
    for _ in range(m):
        row = input(f"Enter example {_ + 1} (comma-separated, last is label): ").split(',')
        data.append(row)
    return n, data

def find_s_algorithm(n, data):
    hypothesis = ['Ø'] * n
    for row in data:
        if row[-1].lower() == 'yes':
            if hypothesis == ['Ø'] * n:
                hypothesis = row[:-1]
            else:
                for i in range(n):
                    if hypothesis[i] != row[i]:
                        hypothesis[i] = '?'
    return hypothesis

n, data = get_input()
result = find_s_algorithm(n, data)
print("Most specific hypothesis:", result)
```

```
= RESTART: C:\Users\rishi\AppData\Local\Programs\Python\Python312\FDS-1.py
Enter number of attributes: 6
Enter number of training examples: 4
Enter example 1 (comma-separated, last is label): sunny,warm,normal,strong,warm,same,yes

Enter example 2 (comma-separated, last is label): Enter example 3 (comma-separated, last is label): sunny,
cool, abnormal, weak, humid, different, no
Enter example 4 (comma-separated, last is label): rainy, warm, normal, humid, different, yes
Most specific hypothesis: ['sunny', 'warm', 'normal', 'strong', 'warm', 'same']
```

2. For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm in python to output a description of the set of all hypotheses consistent with the training examples.

```
import pandas as pd
```

```
def candidate_elimination(data):
    concepts = data.iloc[:, :-1].values
    target = data.iloc[:, -1].values


    specific_h = concepts[0].copy()
    general_h = [['?' for _ in range(len(specific_h))] for _ in range(len(specific_h))]

    for i, instance in enumerate(concepts):
        if target[i] == "Yes":
            for x in range(len(specific_h)):
                if instance[x] != specific_h[x]:
                    specific_h[x] = '?'
                    general_h[x][x] = '?'
        else:
            for x in range(len(specific_h)):
                if instance[x] != specific_h[x]:
                    general_h[x][x] = specific_h[x]
                else:
                    general_h[x][x] = '?'


    general_h = [h for h in general_h if h != ['?' for _ in range(len(specific_h))]]
    return specific_h, general_h


df = pd.read_csv('C:\\Users\\rishi\\OneDrive\\Documents\\data.csv')
specific, general = candidate_elimination(df)

print("Final Specific Hypothesis:", specific)
print("Final General Hypotheses:")
for g in general:
    print(g)
```

```
Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\rishi\AppData\Local\Programs\Python\Python312\FDS-1.py
Final Specific Hypothesis: ['Sunny' 'Warm' '?' 'Strong' '?' '?']
Final General Hypotheses:
['Sunny', '?', '?', '?', '?', '?']
['?', 'Warm', '?', '?', '?', '?']
>>>
```

3. Demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a

new sample.

```
from sklearn.tree import DecisionTreeClassifier
import pandas as pd

cols = input("Enter column names separated by commas (last one is label): ").split(',')
n = int(input("Enter number of training examples: "))
data = []
for i in range(n):
    row = input(f"Enter row {i+1} (comma-separated): ").split(',')
    data.append(row)

df = pd.DataFrame(data, columns=cols)
X = pd.get_dummies(df.iloc[:, :-1])
y = df.iloc[:, -1]

model = DecisionTreeClassifier(criterion='entropy')
model.fit(X, y)

test_input = input(f"Enter new sample values ({', '.join(cols[:-1])}): ").split(',')
test_df = pd.DataFrame([test_input], columns=cols[:-1])
test_encoded = pd.get_dummies(test_df)
test_encoded = test_encoded.reindex(columns=X.columns, fill_value=0)

prediction = model.predict(test_encoded)
print("Predicted class:", prediction[0])
```

```
= RESTART: C:\Users\rishi\AppData\Local\Programs\Python\Python312\FDS-1.py
Enter column names separated by commas (last one is label): outlook,temperature
humidity,wind,play
Enter number of training examples: 4
Enter row 1 (comma-separated): sunny,hot,high,strong,yes
Enter row 2 (comma-separated): sunny,cold,low,weak,no
Enter row 3 (comma-separated): rainy,hot,high,strong,yes
Enter row 4 (comma-separated): sunny,cold,low,weak,no
Enter new sample values (outlook, temperature, humidity, wind): sunny,cold,weak
strong
Predicted class: no
>>
```

4. Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, classification_report

X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
model = MLPClassifier(hidden_layer_sizes=(10,), activation='relu', solver='adam',
max_iter=1000)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
the optimization hasn't converged yet.
Accuracy: 0.9666666666666667
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.89      0.94         9
           2       0.92      1.00      0.96        11

    accuracy                           0.97        30
   macro avg       0.97      0.96      0.97        30
weighted avg       0.97      0.97      0.97        30
```

5. Write a program for Implementation of K-Nearest Neighbours (K-NN) in Python

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
= RESTART: C:/Users/rishi/AppData/Local/Programs/Py
Accuracy: 0.9666666666666667
>
```

6. Write a program to implement Naïve Bayes algorithm in python and to display the results using confusion matrix and accuracy.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix
```

```
X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
>
= RESTART: C:/Users/rishi/AppDat
Accuracy: 1.0
Confusion Matrix:
 [[10  0  0]
 [ 0 12  0]
 [ 0  0  8]]
>
```

7. Write a program to implement Logistic Regression (LR) algorithm in python

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
= RESTART: C:/Users/rish
Accuracy: 1.0
```

8. Write a program to implement Linear Regression (LR) algorithm in python

```
from sklearn.linear_model import LinearRegression
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

X, y = make_regression(n_samples=100, n_features=1, noise=10)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
model = LinearRegression()
model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)
print("Coefficient:", model.coef_[0])
print("Intercept:", model.intercept_)
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
```

```
= RESTART: C:/Users/rishi/AppData/Local/Programs/Py
Coefficient: 68.34502657992918
Intercept: -0.8771502789070631
Mean Squared Error: 76.36771022062729
>
```

9. Compare Linear and Polynomial Regression using Python

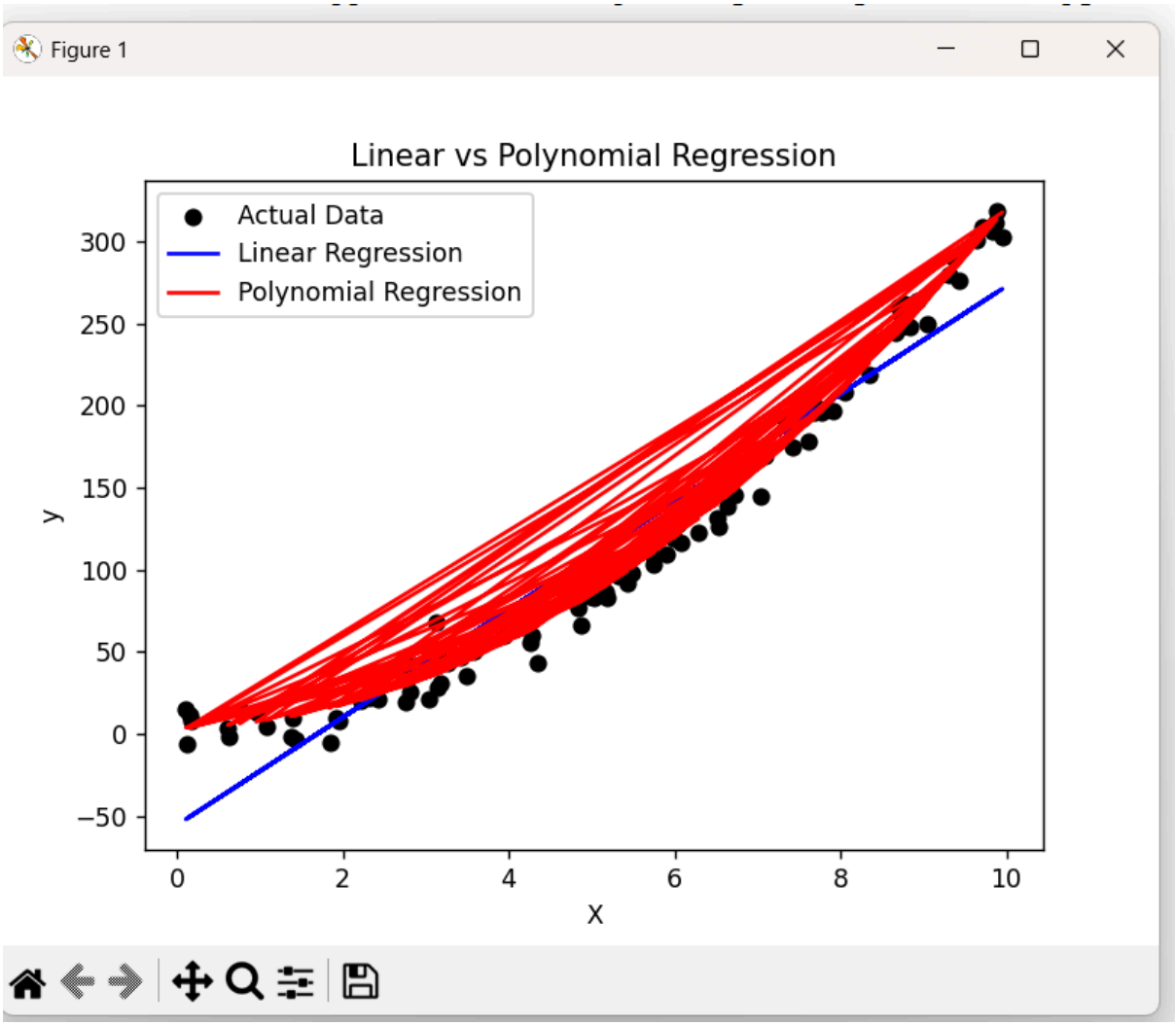
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

X = np.random.rand(100, 1) * 10
y = 3 * X.squeeze() ** 2 + 2 * X.squeeze() + np.random.randn(100) * 10

linear_model = LinearRegression()
linear_model.fit(X, y)
y_lin_pred = linear_model.predict(X)

poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)
poly_model = LinearRegression()
poly_model.fit(X_poly, y)
y_poly_pred = poly_model.predict(X_poly)

plt.scatter(X, y, color='black', label='Actual Data')
plt.plot(X, y_lin_pred, color='blue', label='Linear Regression')
plt.plot(X, y_poly_pred, color='red', label='Polynomial Regression')
plt.legend()
plt.title("Linear vs Polynomial Regression")
plt.xlabel("X")
plt.ylabel("y")
plt.show()
```



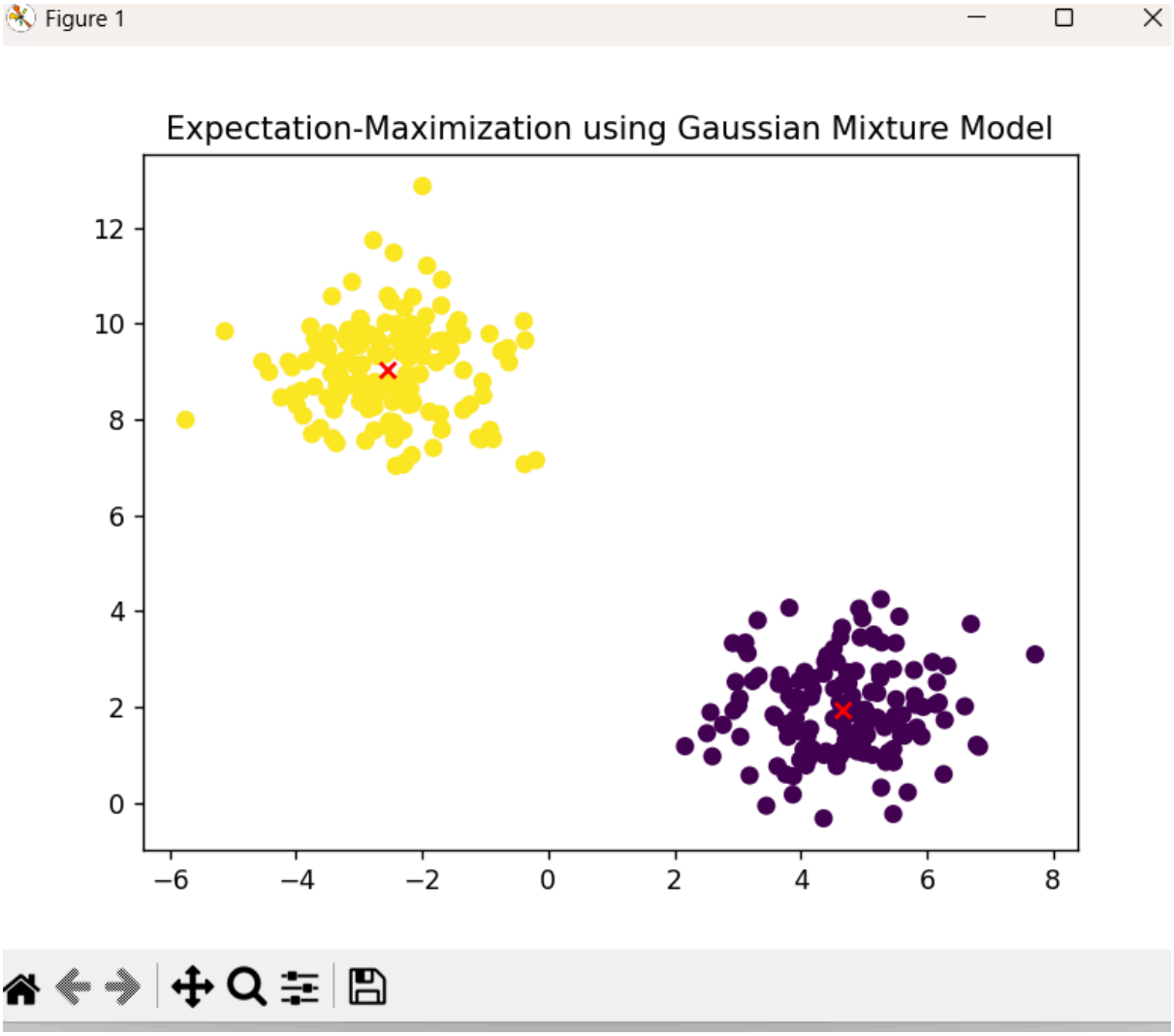
10. Write a Python Program to Implement Expectation & Maximization Algorithm

```
from sklearn.mixture import GaussianMixture
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt

X, _ = make_blobs(n_samples=300, centers=2, random_state=42)

model = GaussianMixture(n_components=2)
model.fit(X)
labels = model.predict(X)

plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.scatter(model.means_[:, 0], model.means_[:, 1], c='red', marker='x')
plt.title("Expectation-Maximization using Gaussian Mixture Model")
plt.show()
```



11. Write a program for the task of Credit Score Classification

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score

data = {
    'Income': [50000, 60000, 35000, 90000, 45000, 70000, 30000, 100000],
    'Age': [25, 45, 35, 52, 23, 40, 30, 50],
    'LoanAmount': [2000, 5000, 1200, 7000, 2500, 4000, 1000, 8000],
    'CreditScore': ['Good', 'Good', 'Poor', 'Good', 'Average', 'Average', 'Poor', 'Good']
}

df = pd.DataFrame(data)
df['CreditScore'] = df['CreditScore'].map({'Poor': 0, 'Average': 1, 'Good': 2})

X = df[['Income', 'Age', 'LoanAmount']]
y = df['CreditScore']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       1.00      1.00      1.00         1
           2       1.00      0.50      0.67         2

    accuracy                           0.67         3
   macro avg       0.67      0.50      0.56         3
weighted avg       1.00      0.67      0.78         3
```

12. Implement IRIS Flower classification using KNN

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, classification_report

X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = MLPClassifier(hidden_layer_sizes=(10,), activation='relu', solver='adam',
max_iter=1000)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

>
```

13. Implement the Car price prediction model using Python

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

data = {
   'Year': [2015, 2012, 2018, 2010, 2016, 2013, 2019, 2011],
   'Mileage': [35000, 50000, 20000, 80000, 40000, 60000, 15000, 75000],
   'EngineSize': [1.2, 1.4, 1.5, 1.3, 1.6, 1.4, 1.8, 1.2],
   'Price': [320000, 280000, 400000, 200000, 360000, 300000, 450000, 220000]
}

df = pd.DataFrame(data)

X = df[['Year', 'Mileage', 'EngineSize']]
y = df['Price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Predicted Prices:", y_pred)
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
```

```
= RESTART: C:\Users\rishi\AppData\Local\Programs\Python\Pytho
Predicted Prices: [268203.12500033 278085.93750089]
Mean Squared Error: 309696197.4864689
>>
```

14. Implement House price Prediction using appropriate machine learning algorithm

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

data = {
   'Area': [1200, 1500, 1000, 1800, 1100, 1600, 2000, 1700],
   'Bedrooms': [2, 3, 2, 4, 2, 3, 4, 3],
   'Bathrooms': [1, 2, 1, 2, 1, 2, 3, 2],
   'Age': [10, 5, 15, 3, 20, 4, 2, 6],
   'Price': [3000000, 4000000, 2500000, 5000000, 2600000, 4200000, 5500000, 4500000]
}

df = pd.DataFrame(data)

X = df[['Area', 'Bedrooms', 'Bathrooms', 'Age']]
y = df['Price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Predicted Prices:", y_pred)
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
```

```
= RESTART: C:\Users\rishi\AppData\Local\Programs\Python\Py
Predicted Prices: [4066000. 4302000.]
Mean Squared Error: 7380000000.0005665
```

15. Implement Iris Flower Classification using Naive Bayes classifier

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report

X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
>
= RESTART: C:\Users\rishi\AppData\Local\Programs\Python\Python312\ml.py
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

>
```

16. Compare different types Classification Algorithms and evaluate their performance.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

models = {
    "Logistic Regression": LogisticRegression(max_iter=200),
    "K-Nearest Neighbors": KNeighborsClassifier(n_neighbors=3),
    "Naive Bayes": GaussianNB(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier()
}
```

```
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"Model: {name}")
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("Classification Report:\n", classification_report(y_test, y_pred))
    print("-" * 60)
```

```
Model: Logistic Regression
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

------------------------------------------------------------
Model: K-Nearest Neighbors
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

------------------------------------------------------------
Model: Naive Bayes
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

17. Implement Mobile Price Prediction using appropriate machine learning algorithm

```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

```
data = {
   'RAM_GB': [4, 6, 8, 3, 2, 12, 6, 4],
   'ROM_GB': [64, 128, 256, 32, 16, 512, 128, 64],
   'Battery_mAh': [4000, 5000, 4500, 3000, 2800, 6000, 5000, 4000],
   'Camera_MP': [12, 48, 64, 8, 5, 108, 50, 13],
   'Price': [12000, 18000, 25000, 8000, 6000, 45000, 22000, 13000]
}

df = pd.DataFrame(data)

X = df[['RAM_GB', 'ROM_GB', 'Battery_mAh', 'Camera_MP']]
y = df['Price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

model = RandomForestRegressor()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Predicted Prices:", y_pred)
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
```

```
===== RESTART: C:\Users\rishi\AppData\L
Predicted Prices: [19750. 22890.]
Mean Squared Error: 245957300.0
>>>
```

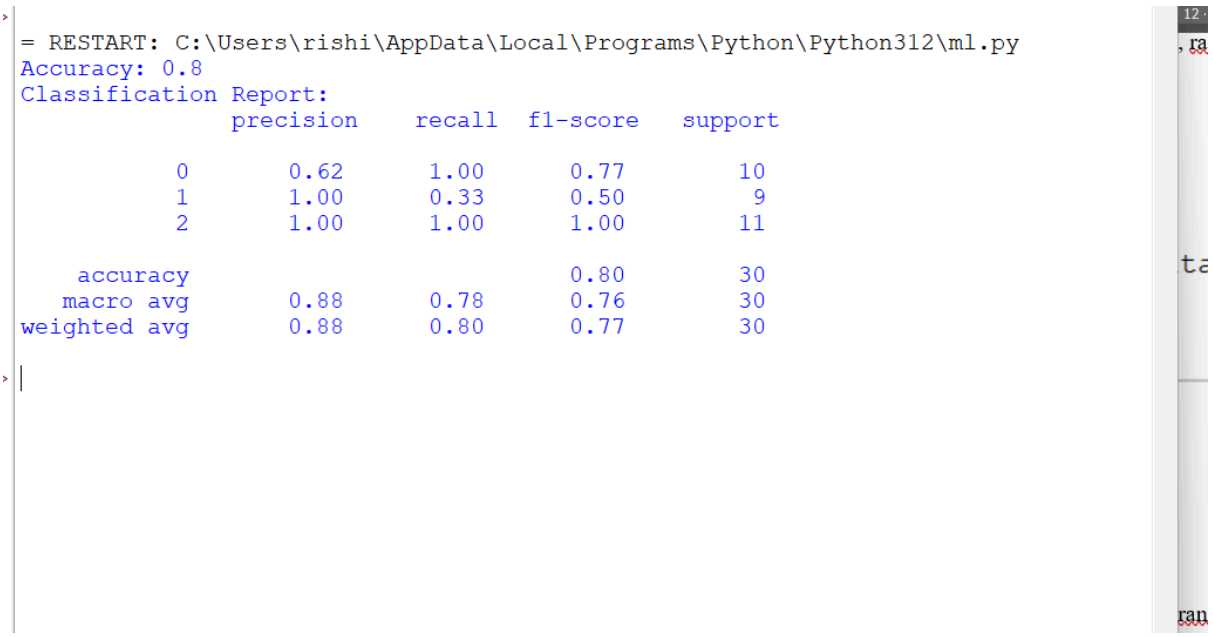
18. Implement Perceptron based IRIS classification

```
from sklearn.datasets import load_iris
from sklearn.linear_model import Perceptron
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report

X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = Perceptron(max_iter=1000, eta0=0.1)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```



19. Implementation of Naive Bayes classification for Bank Loan prediction

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report

data = {
   'Age': [25, 40, 35, 28, 50, 45, 23, 38],
   'Income': [30000, 80000, 60000, 40000, 100000, 85000, 32000, 72000],
   'CreditScore': [650, 720, 700, 580, 800, 750, 610, 690],
   'LoanApproved': ['No', 'Yes', 'Yes', 'No', 'Yes', 'Yes', 'No', 'Yes']
}

df = pd.DataFrame(data)
df['LoanApproved'] = df['LoanApproved'].map({'No': 0, 'Yes': 1})

X = df[['Age', 'Income', 'CreditScore']]
y = df['LoanApproved']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
>>
= RESTART: C:\Users\rishi\AppData\Local\Programs\Python\Python312\m
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         2

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2

>>
```

20. Implement Future Sales Prediction using a suitable machine learning algorithm

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

data = {
   'Month': [1, 2, 3, 4, 5, 6, 7, 8],
   'Sales': [1200, 1500, 1600, 1800, 2000, 2300, 2500, 2700]
}

df = pd.DataFrame(data)

X = df[['Month']]
y = df['Sales']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Predicted Sales:", y_pred)
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
```

```
----- RESTART: C:\Users\ITSHI\AppData\Local\Program
Predicted Sales: [1388. 2256.]
Mean Squared Error: 7240.0
```