ITA0628 Machine Learning - Lab Experiments

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

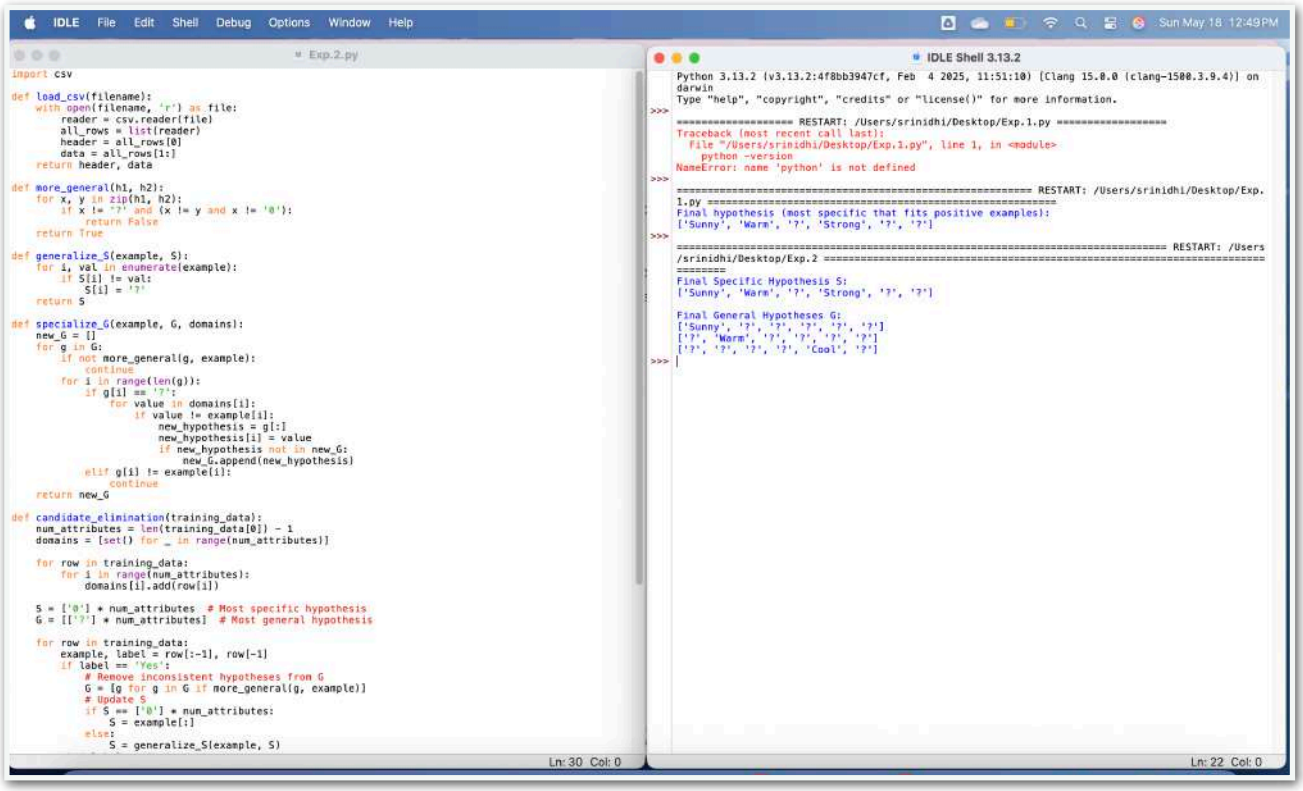
```python
# FIND-S Algorithm Implementation
def find_s_algorithm(training_data):
    # Get the number of attributes (excluding the target label)
    num_attributes = len(training_data[0]) - 1

    # Step 1: Initialize the hypothesis to the first positive example
    for example in training_data:
        if example[-1].lower() == 'yes':
            hypothesis = example[:-1]  # Exclude the label
            break

    # Step 2: Generalize hypothesis using remaining positive examples
    for example in training_data:
        if example[-1].lower() == 'yes':  # Only consider positive examples
            for i in range(num_attributes):
                if hypothesis[i] != example[i]:
                    hypothesis[i] = '?'  # Generalize differing attribute

    return hypothesis

# Sample training dataset
# Format: [Sky, AirTemp, Humidity, Wind, Water, Forecast, EnjoySport]
training_data = [
    ['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same', 'Yes'],
    ['Sunny', 'Warm', 'High',   'Strong', 'Warm', 'Same', 'Yes'],
    ['Rainy', 'Cold', 'High',   'Strong', 'Warm', 'Change', 'No'],
    ['Sunny', 'Warm', 'High',   'Strong', 'Cool', 'Change', 'Yes']
]

# Run FIND-S Algorithm
hypothesis = find_s_algorithm(training_data)

# Output the result
print("Final hypothesis (most specific that fits positive examples):")
print(hypothesis)
```

Output:

```
IDLE Shell 3.13.2
Python 3.13.2 (v3.13.2:4f8bb3947cf, Feb  4 2025, 11:51:10) [Clang 15.0.0 (clang-1500.3.9.4)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=================== RESTART: /Users/srinidhi/Desktop/Exp.1.py ==================
Traceback (most recent call last):
  File "/Users/srinidhi/Desktop/Exp.1.py", line 1, in <module>
    python -version
NameError: name 'python' is not defined
>>>
========================================================== RESTART: /Users/srinidhi/Desktop/Exp.1.py ==========================================================
Final hypothesis (most specific that fits positive examples):
['Sunny', 'Warm', '?', 'Strong', '?', '?']
>>> |
```

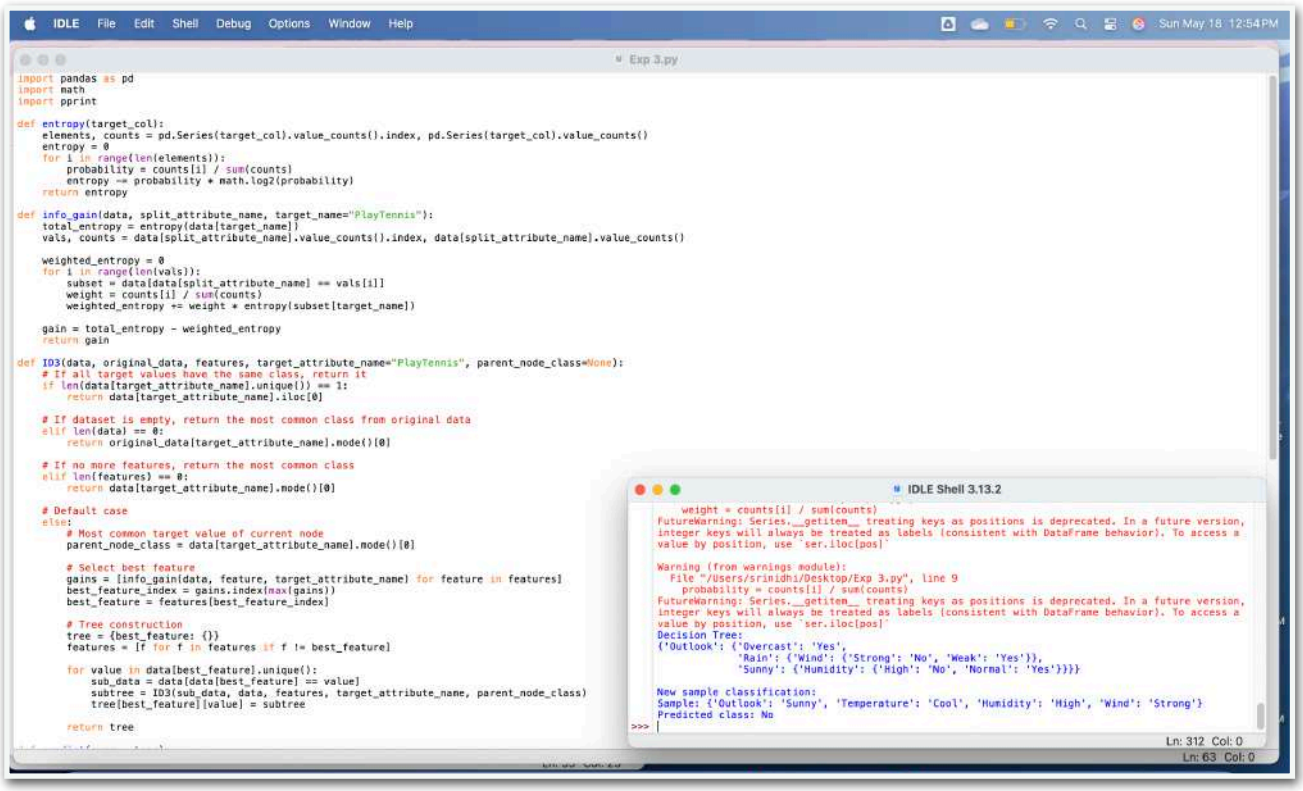
2. **For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm in python to output a description of the set of all hypotheses consistent with the training examples**

Output ScreenShot:



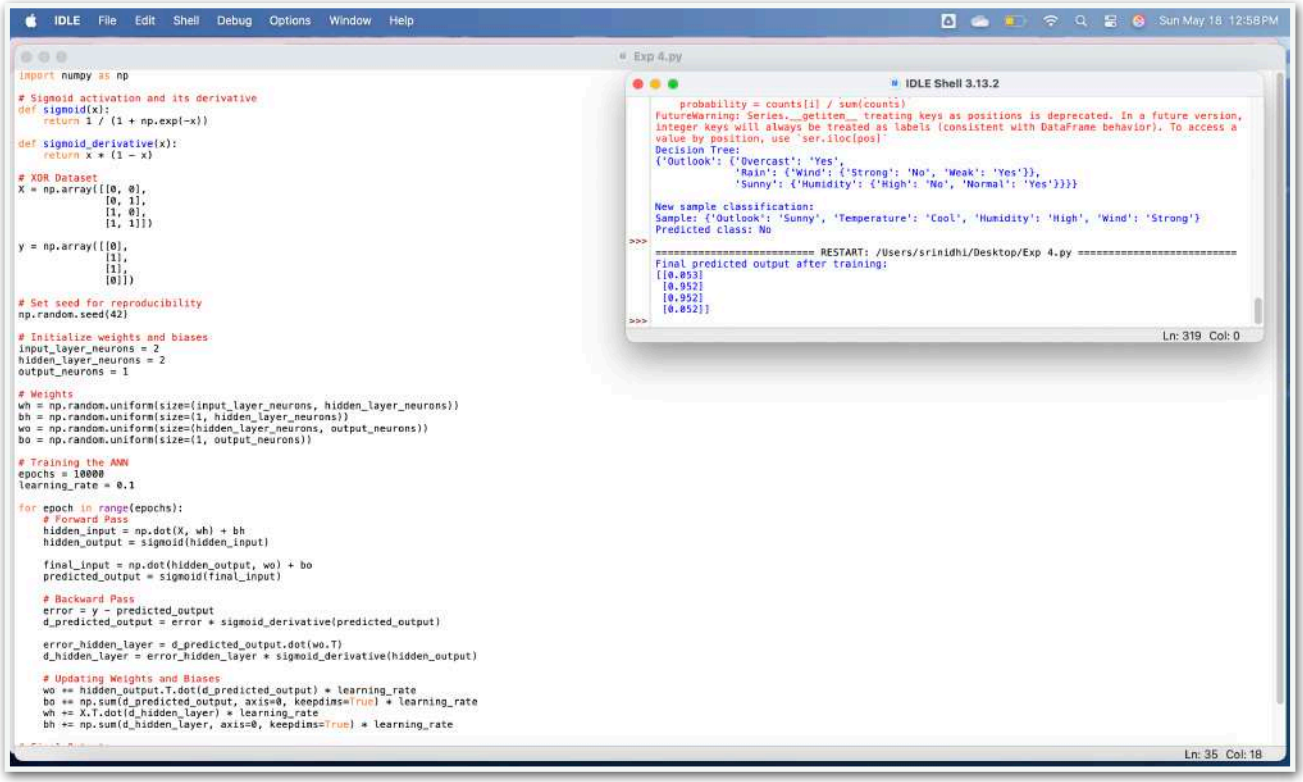
3. Demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

Output:



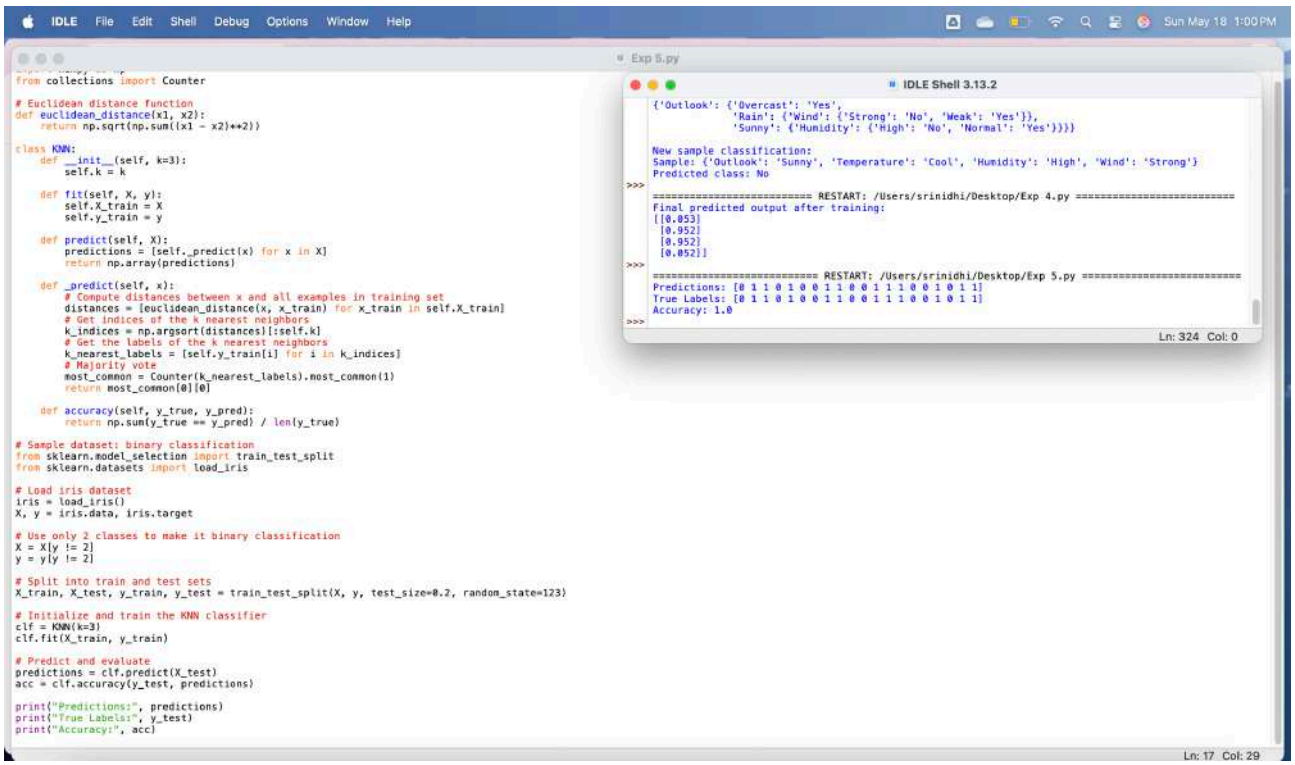
4. Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.

Output:



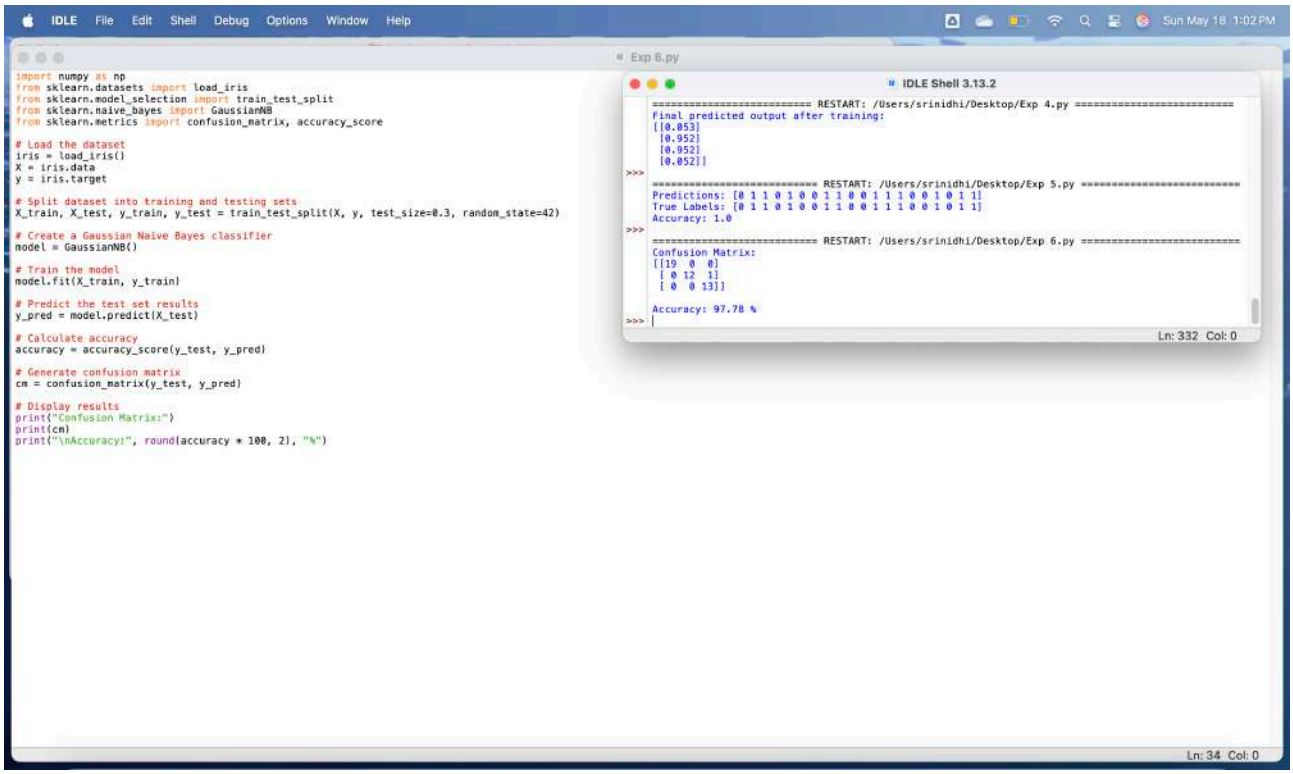
5. Write a program for Implementation of K-Nearest Neighbours (K-NN) in Python.

Output:



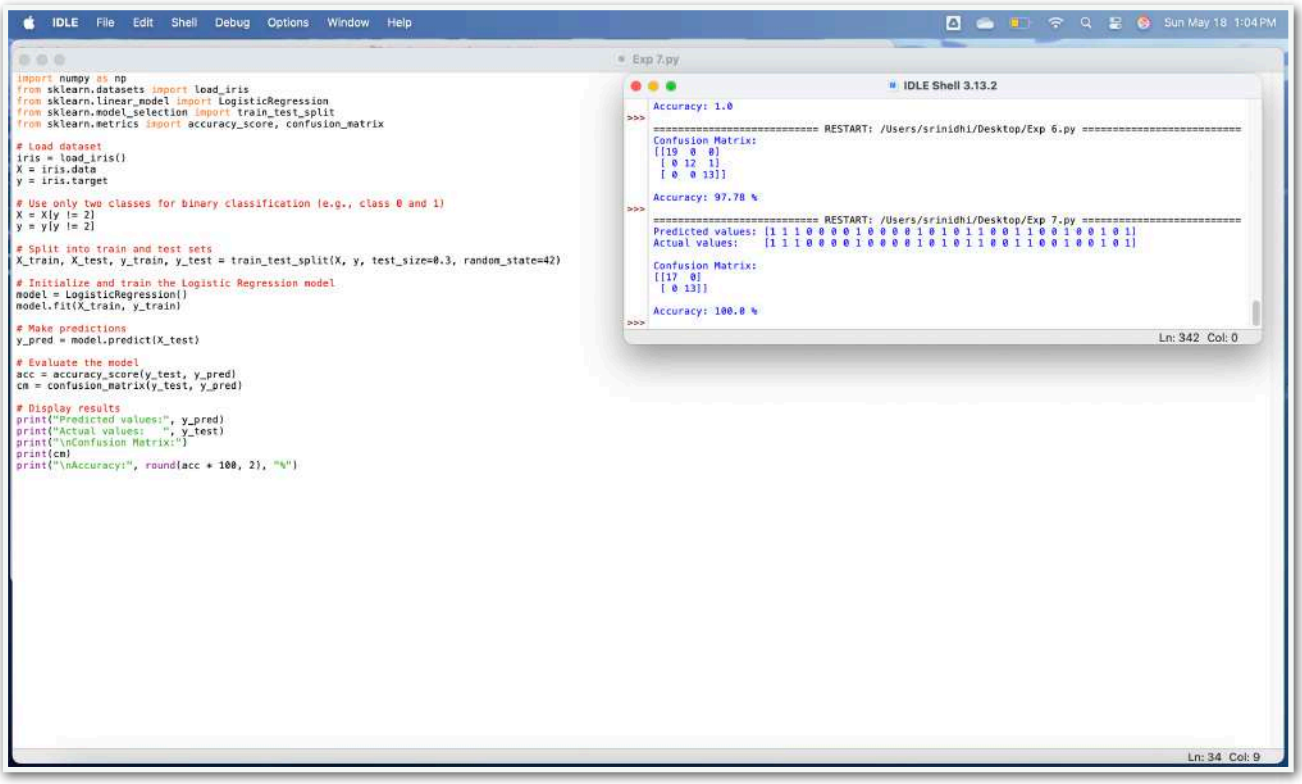
6. **Write a program to implement Naïve Bayes algorithm in python and to display the results using confusion matrix and accuracy.**

Output:



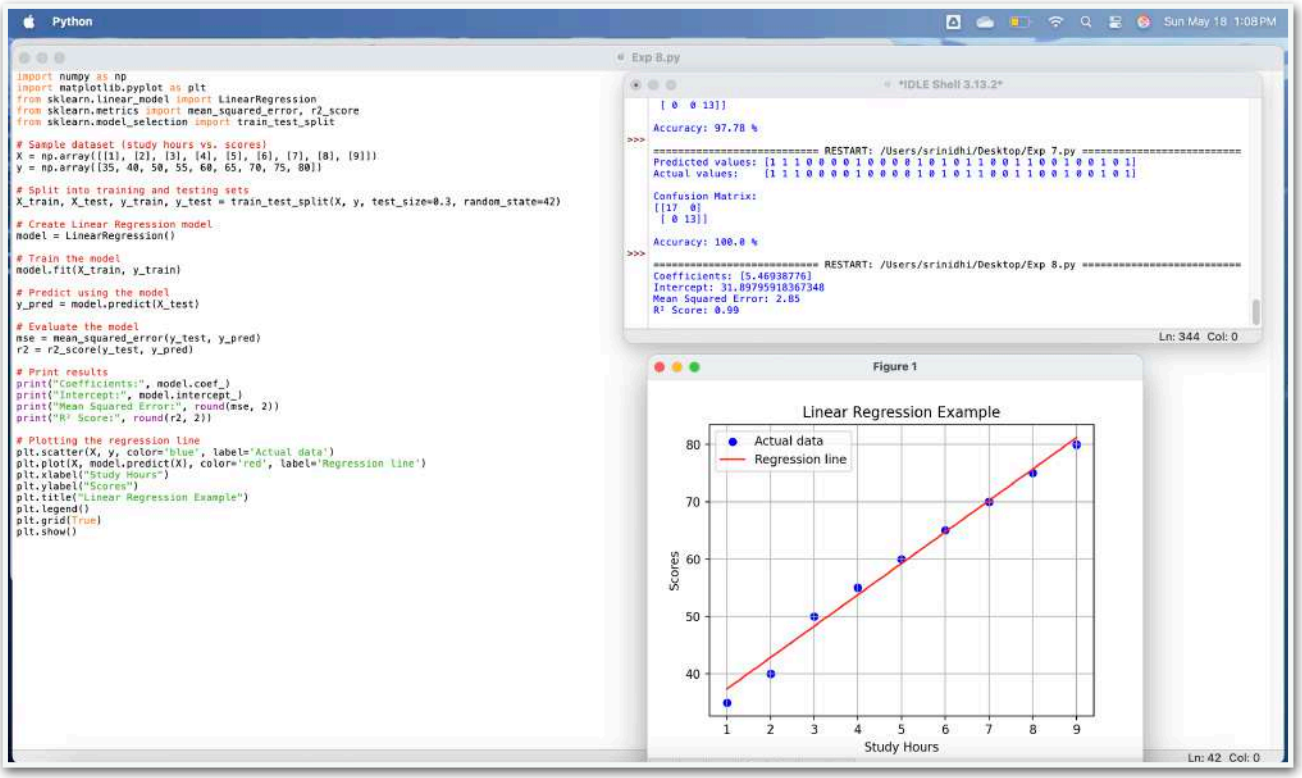
7. **Write a program to implement Logistic Regression (LR) algorithm in python**

Output:



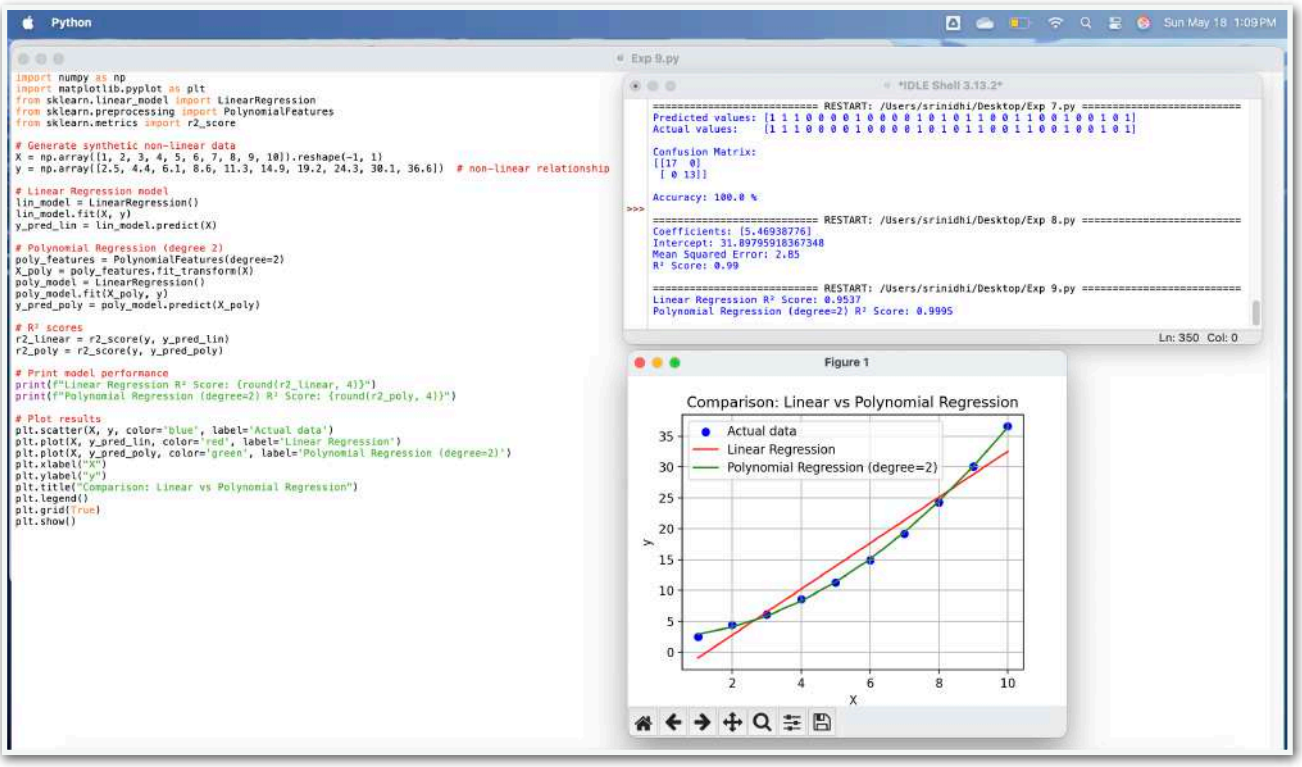
8. Write a program to implement Linear Regression (LR) algorithm in python

Output:



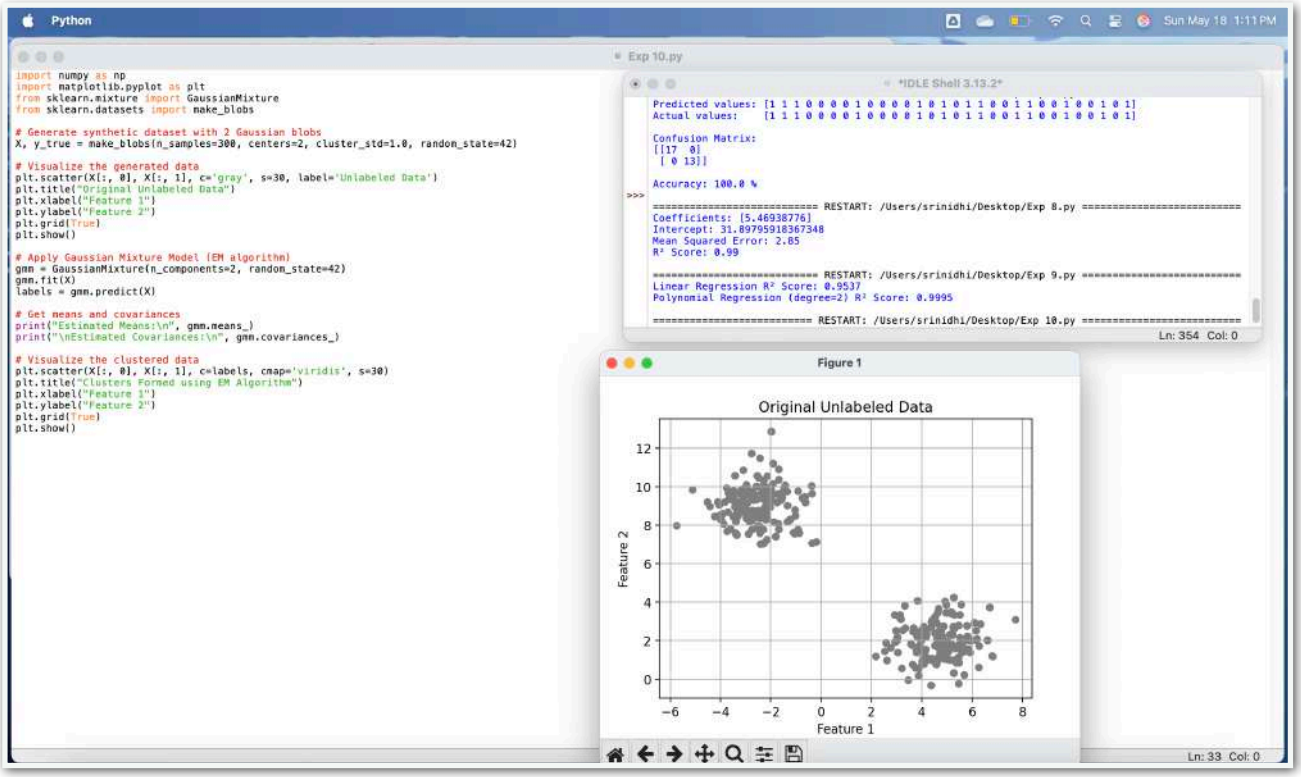
9. Compare Linear and Polynomial Regression using Python

Output:



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

Output:

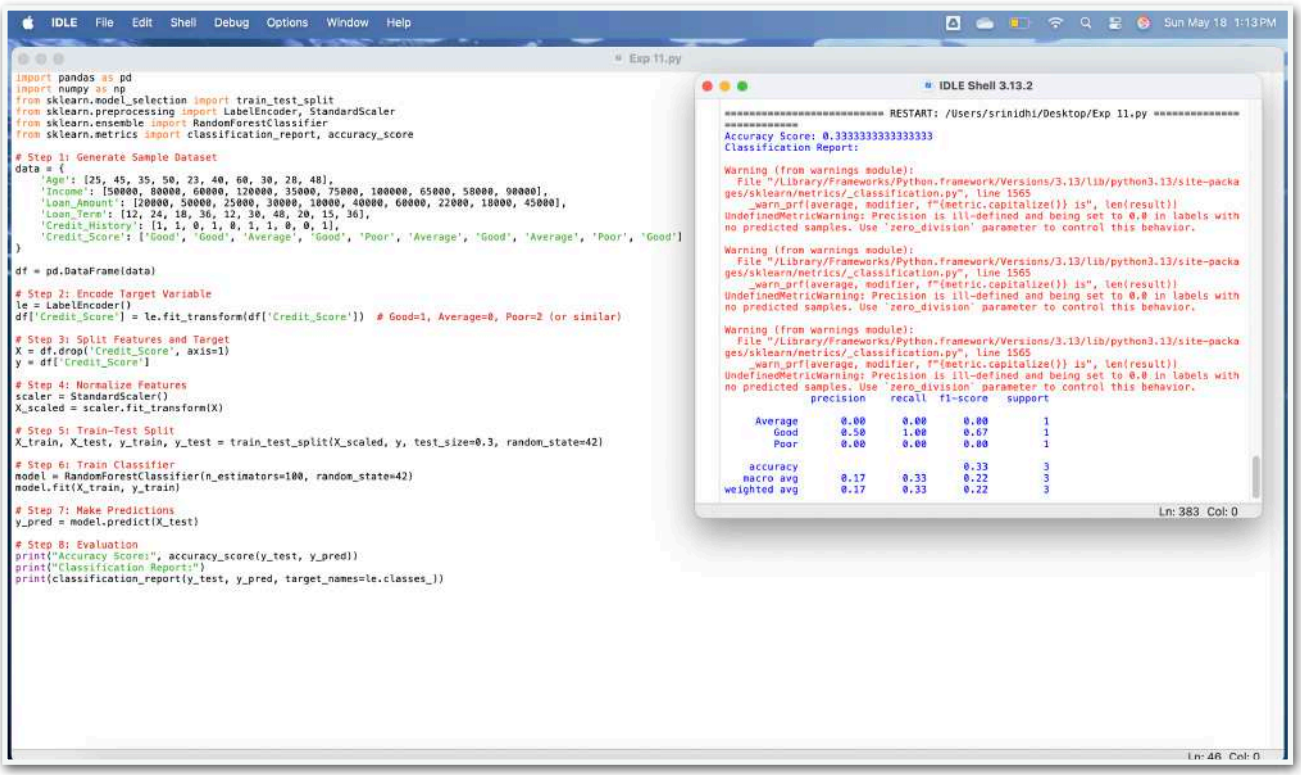


```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.mixture import GaussianMixture
from sklearn.datasets import make_blobs

# Generate synthetic dataset with 2 Gaussian blobs
X, y_true = make_blobs(n_samples=300, centers=2, cluster_std=1.0, random_state=42)

# Visualize the generated data
plt.scatter(X[:, 0], X[:, 1], c='gray', s=30, label='Unlabeled Data')
plt.title("Original Unlabeled Data")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.grid(True)
plt.show()

# Apply Gaussian Mixture Model (EM algorithm)
gmm = GaussianMixture(n_components=2, random_state=42)
gmm.fit(X)
labels = gmm.predict(X)

# Get means and covariances
print("Estimated Means:\n", gmm.means_)
print("\nEstimated Covariances:\n", gmm.covariances_)

# Visualize the clustered data
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis', s=30)
plt.title("Clusters Formed using EM Algorithm")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.grid(True)
plt.show()
```

```
Predicted values: [1 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1 1 0 0 1 1 0 0 1 0 0 1 0 1]
Actual values:    [1 1 1 0 0 0 0 1 0 0 0 0 1 0 1 0 1 1 0 0 1 1 0 0 1 0 0 1 0 1]

Confusion Matrix:
[[17  0]
 [ 0 13]]

Accuracy: 100.0 %
>>>
========================== RESTART: /Users/srinidhi/Desktop/Exp 8.py ==========================
Coefficients: [5.46938776]
Intercept: 31.89795918367348
Mean Squared Error: 2.85
R² Score: 0.99

========================== RESTART: /Users/srinidhi/Desktop/Exp 9.py ==========================
Linear Regression R² Score: 0.9537
Polynomial Regression (degree=2) R² Score: 0.9995

========================= RESTART: /Users/srinidhi/Desktop/Exp 10.py ==========================
```

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

Output:

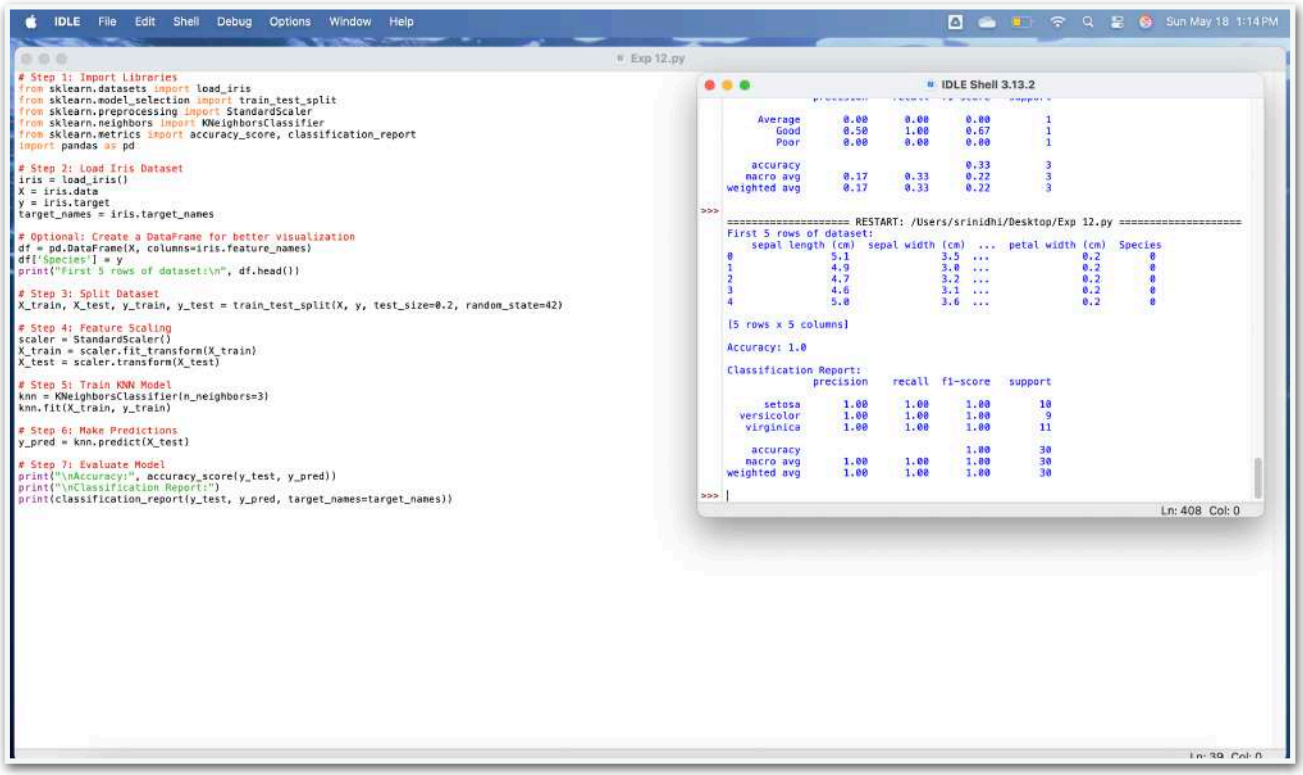


```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score

# Step 1: Generate Sample Dataset
data = {
    'Age': [25, 45, 35, 50, 23, 40, 60, 30, 28, 48],
    'Income': [50000, 80000, 60000, 120000, 35000, 75000, 100000, 65000, 58000, 90000],
    'Loan_Amount': [20000, 50000, 25000, 30000, 10000, 40000, 60000, 22000, 18000, 45000],
    'Loan_Term': [12, 24, 18, 36, 12, 30, 48, 20, 15, 36],
    'Credit_History': [1, 1, 0, 1, 0, 1, 1, 0, 0, 1],
    'Credit_Score': ['Good', 'Good', 'Average', 'Good', 'Poor', 'Average', 'Good', 'Average', 'Poor', 'Good']
}

df = pd.DataFrame(data)

# Step 2: Encode Target Variable
le = LabelEncoder()
df['Credit_Score'] = le.fit_transform(df['Credit_Score'])  # Good=1, Average=0, Poor=2 (or similar)

# Step 3: Split Features and Target
X = df.drop('Credit_Score', axis=1)
y = df['Credit_Score']

# Step 4: Normalize Features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 5: Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)

# Step 6: Train Classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Step 7: Make Predictions
y_pred = model.predict(X_test)

# Step 8: Evaluation
print("Accuracy Score:", accuracy_score(y_test, y_pred))
print("Classification Report:")
print(classification_report(y_test, y_pred, target_names=le.classes_))
```

```
========================== RESTART: /Users/srinidhi/Desktop/Exp 11.py ==============
============
Accuracy Score: 0.3333333333333333
Classification Report:

Warning (from warnings module):
  File "/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packa
ges/sklearn/metrics/_classification.py", line 1565
    _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.

Warning (from warnings module):
  File "/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packa
ges/sklearn/metrics/_classification.py", line 1565
    _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.

Warning (from warnings module):
  File "/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packa
ges/sklearn/metrics/_classification.py", line 1565
    _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with
no predicted samples. Use `zero_division` parameter to control this behavior.
              precision    recall  f1-score   support

     Average       0.00      0.00      0.00         1
        Good       0.50      1.00      0.67         1
        Poor       0.00      0.00      0.00         1

    accuracy                           0.33         3
   macro avg       0.17      0.33      0.22         3
weighted avg       0.17      0.33      0.22         3
```

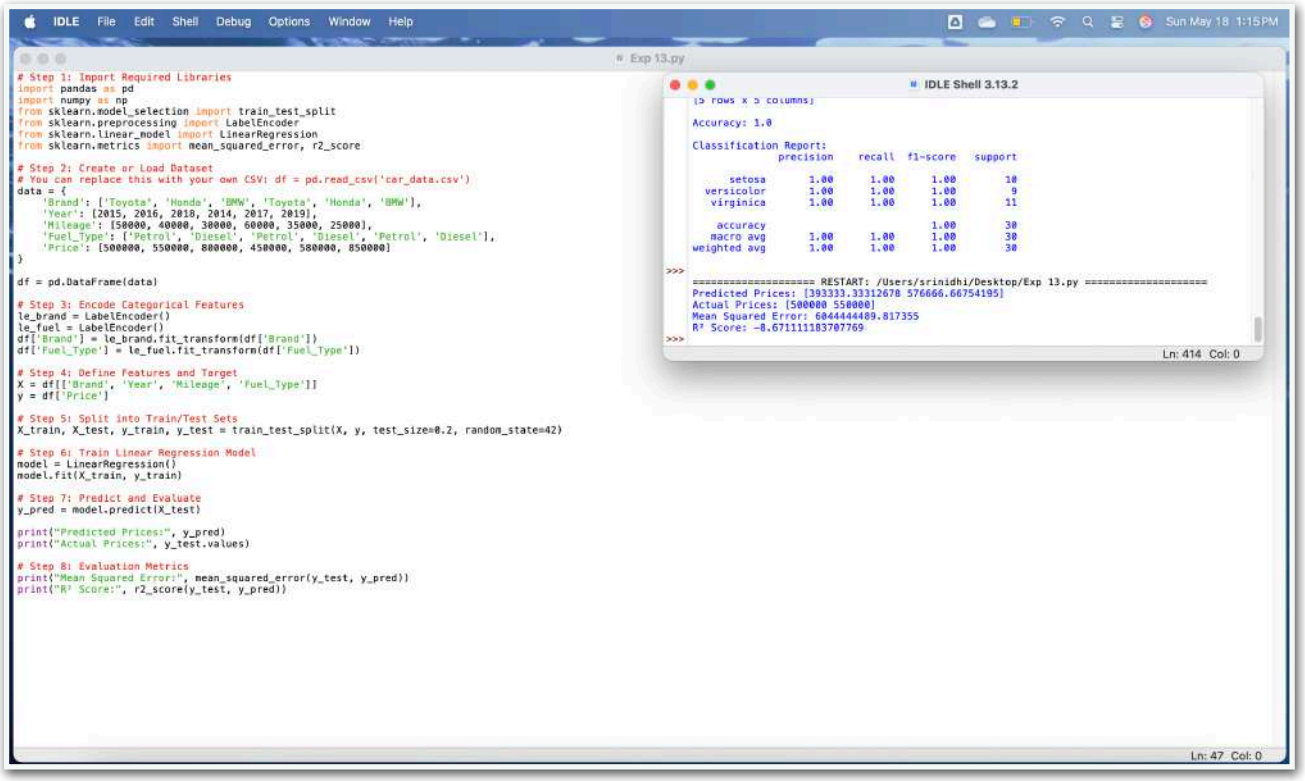
12. Implement Iris Flower Classification using KNN

Output:



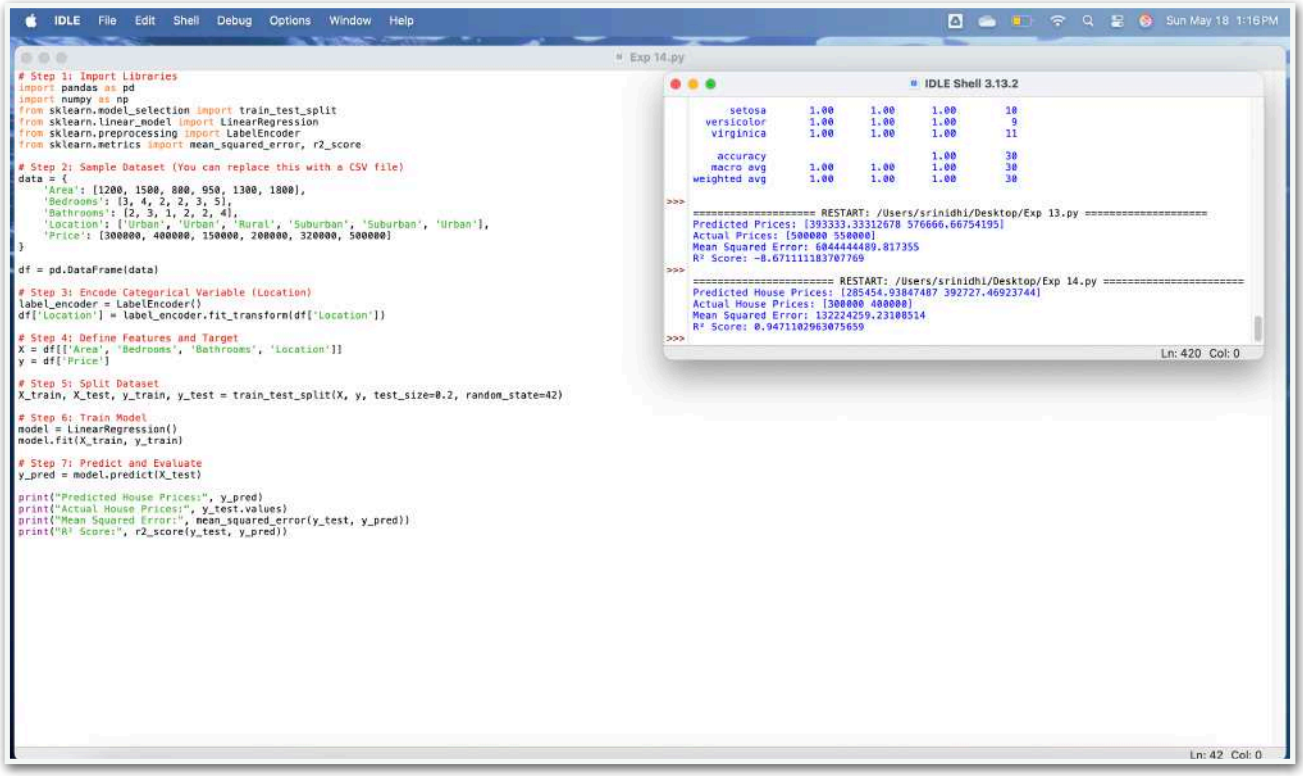
13. Implement the Car Price Prediction Model using Python.

Output:



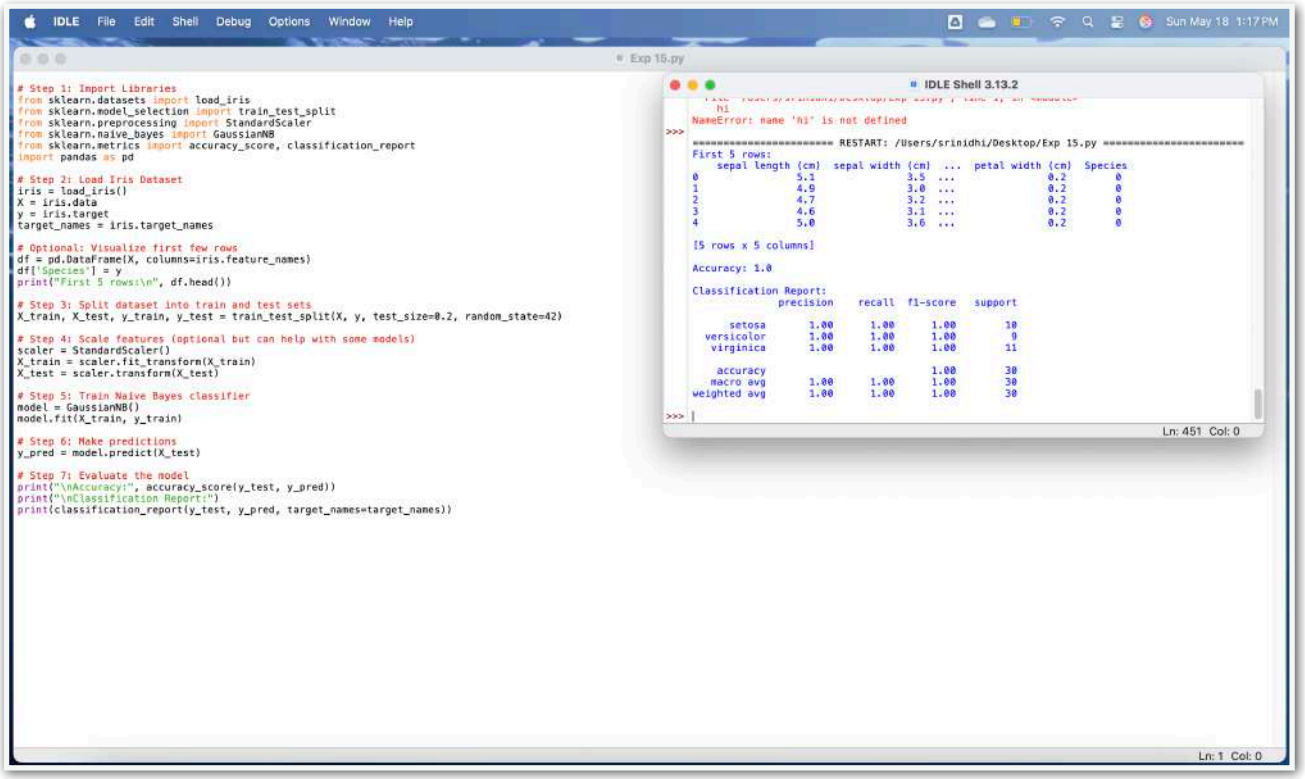
14. Implement House price Prediction using appropriate machine learning algorithm

Output:



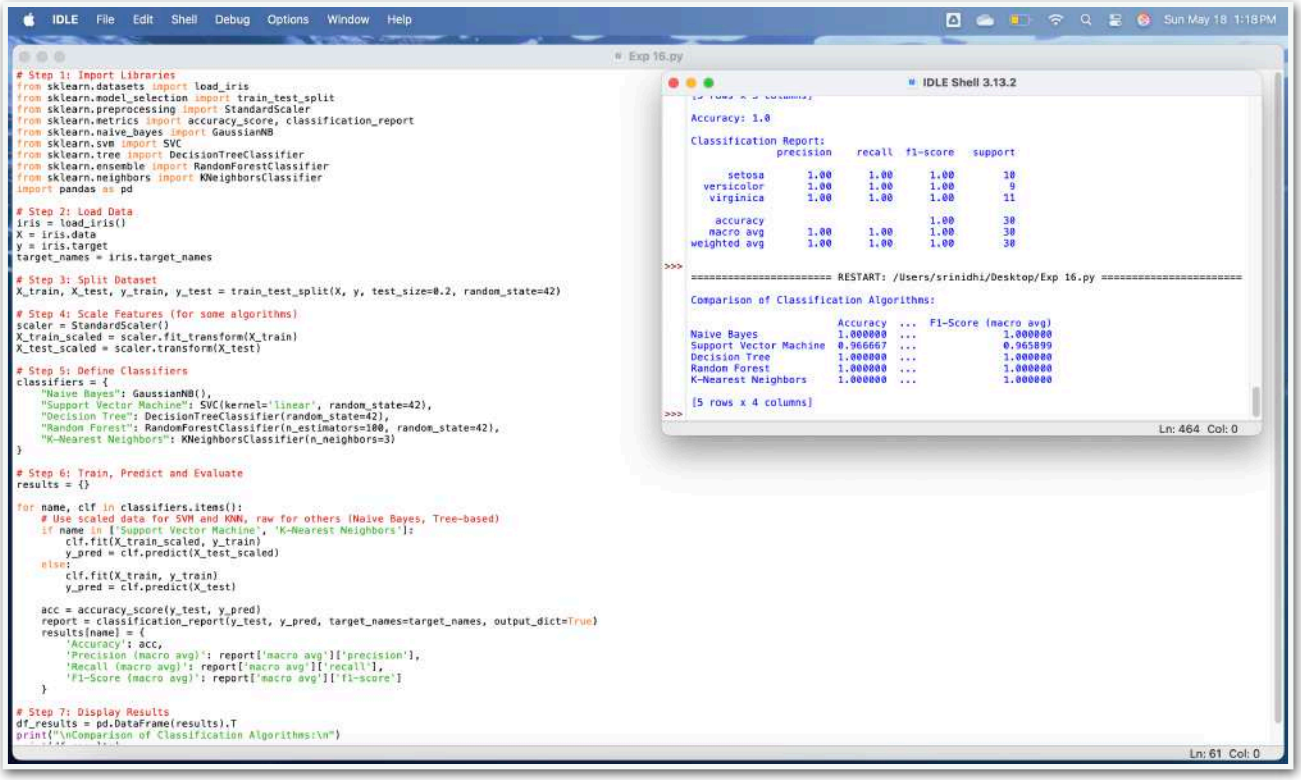
15. Implement Iris Flower Classification using Naive Bayes classifier

Output:



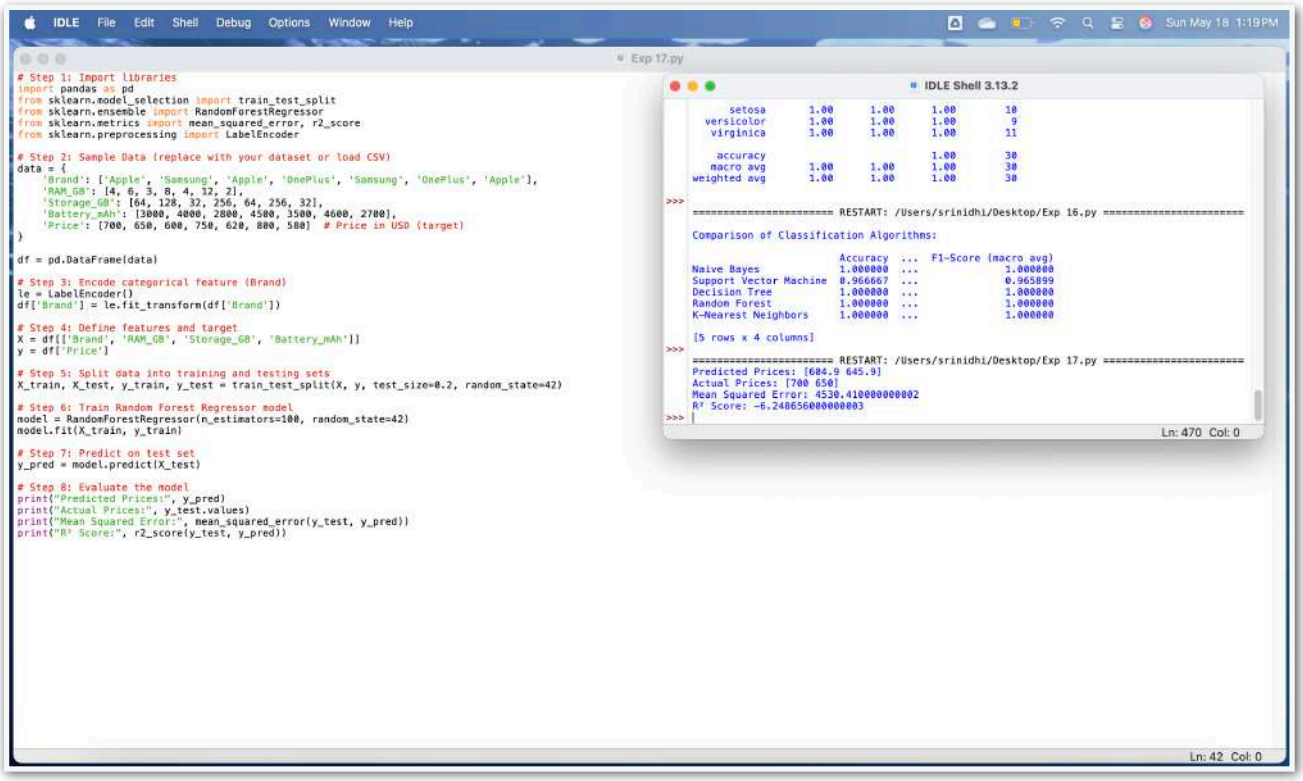
16. Compare different types Classification Algorithms and evaluate their performance.

Output:



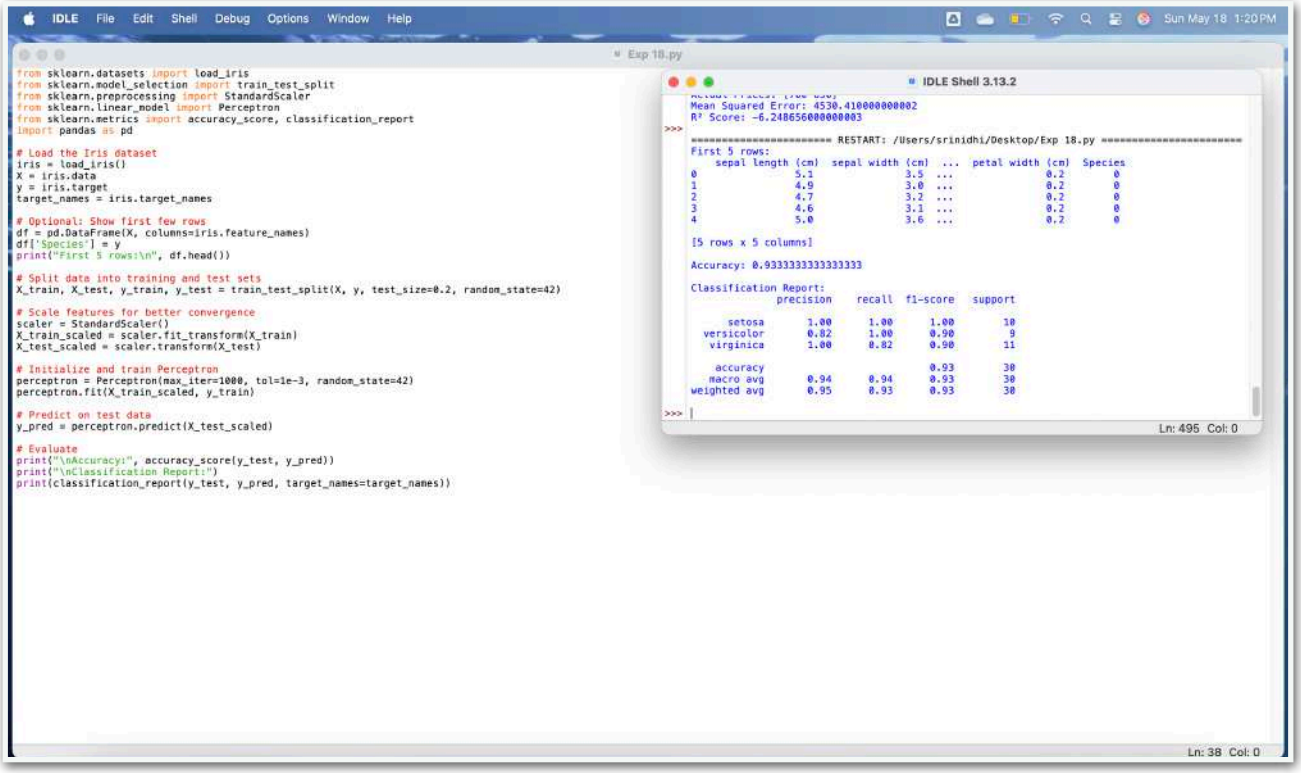
17. Implement Mobile Price Prediction using appropriate machine learning algorithm

Output:



18. Implement Perceptron based IRIS classification

Output:

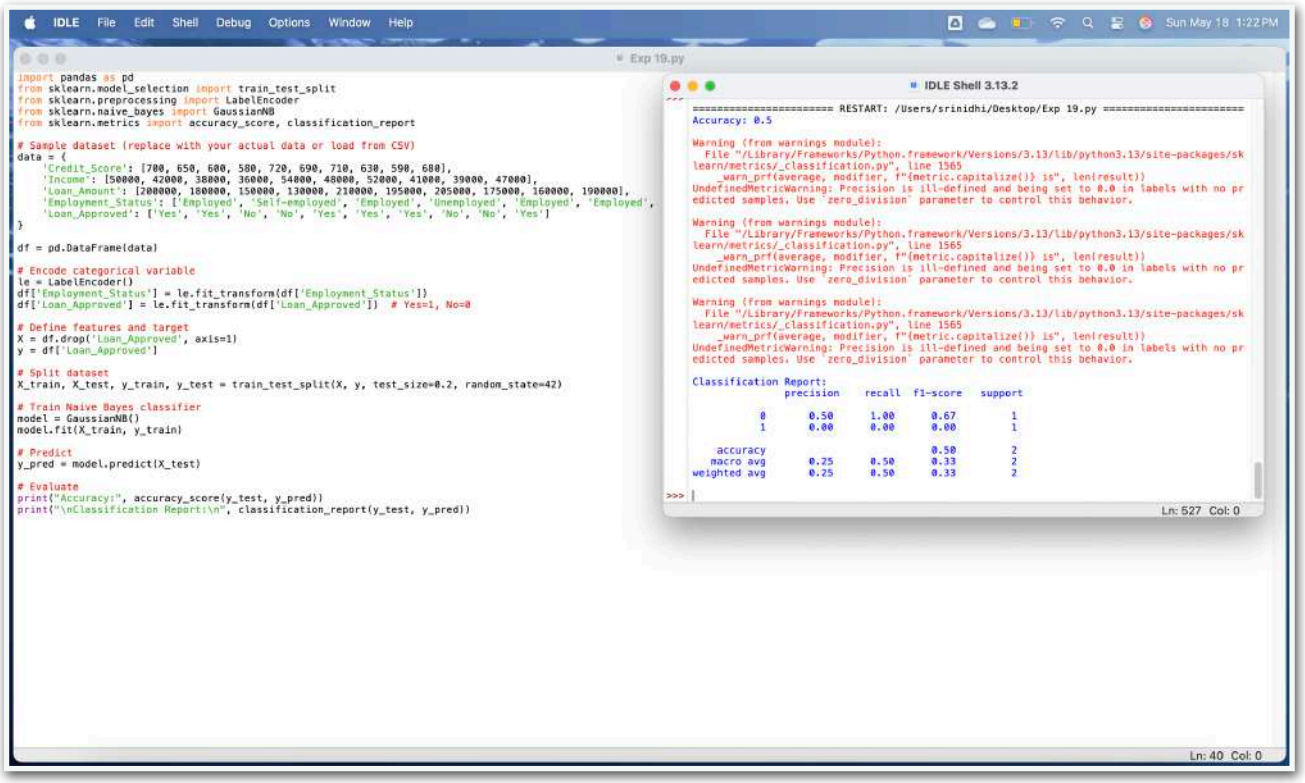


```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target
target_names = iris.target_names

# Optional: Show first few rows
df = pd.DataFrame(X, columns=iris.feature_names)
df['Species'] = y
print("First 5 rows:\n", df.head())

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale features for better convergence
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize and train Perceptron
perceptron = Perceptron(max_iter=1000, tol=1e-3, random_state=42)
perceptron.fit(X_train_scaled, y_train)

# Predict on test data
y_pred = perceptron.predict(X_test_scaled)

# Evaluate
print("\nAccuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=target_names))
```

```
Mean Squared Error: 4530.410000000002
R² Score: -6.248656000000003
>>> 
======================= RESTART: /Users/srinidhi/Desktop/Exp 18.py ======================
First 5 rows:
    sepal length (cm)  sepal width (cm)  ...  petal width (cm)  Species
0                5.1               3.5  ...               0.2        0
1                4.9               3.0  ...               0.2        0
2                4.7               3.2  ...               0.2        0
3                4.6               3.1  ...               0.2        0
4                5.0               3.6  ...               0.2        0

[5 rows x 5 columns]

Accuracy: 0.9333333333333333

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       0.82      1.00      0.90         9
   virginica       1.00      0.82      0.90        11

    accuracy                           0.93        30
   macro avg       0.94      0.94      0.93        30
weighted avg       0.95      0.93      0.93        30

>>> |
```

19. Implementation of Naive Bayes classification for Bank Loan prediction

Output:



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

# Sample dataset (replace with your actual data or load from CSV)
data = {
    'Credit_Score': [700, 650, 600, 580, 720, 690, 710, 630, 590, 680],
    'Income': [50000, 42000, 38000, 36000, 54000, 48000, 52000, 41000, 39000, 47000],
    'Loan_Amount': [200000, 180000, 150000, 130000, 210000, 195000, 205000, 175000, 160000, 190000],
    'Employment_Status': ['Employed', 'Self-employed', 'Employed', 'Unemployed', 'Employed', 'Employed',
    'Loan_Approved': ['Yes', 'Yes', 'No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'No', 'Yes']
}

df = pd.DataFrame(data)

# Encode categorical variable
le = LabelEncoder()
df['Employment_Status'] = le.fit_transform(df['Employment_Status'])
df['Loan_Approved'] = le.fit_transform(df['Loan_Approved'])  # Yes=1, No=0

# Define features and target
X = df.drop('Loan_Approved', axis=1)
y = df['Loan_Approved']

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Naive Bayes classifier
model = GaussianNB()
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)

# Evaluate
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
======================= RESTART: /Users/srinidhi/Desktop/Exp 19.py ======================
Accuracy: 0.5

Warning (from warnings module):
  File "/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages/sk
learn/metrics/_classification.py", line 1565
    _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pr
edicted samples. Use `zero_division` parameter to control this behavior.

Warning (from warnings module):
  File "/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages/sk
learn/metrics/_classification.py", line 1565
    _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pr
edicted samples. Use `zero_division` parameter to control this behavior.

Warning (from warnings module):
  File "/Library/Frameworks/Python.framework/Versions/3.13/lib/python3.13/site-packages/sk
learn/metrics/_classification.py", line 1565
    _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no pr
edicted samples. Use `zero_division` parameter to control this behavior.

Classification Report:
               precision    recall  f1-score   support

           0       0.50      1.00      0.67         1
           1       0.00      0.00      0.00         1

    accuracy                           0.50         2
   macro avg       0.25      0.50      0.33         2
weighted avg       0.25      0.50      0.33         2

>>> |
```

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

Output: