1. Write a program to implement the naïve Bayesian classifier for a sample training data set stored as a .CSV file. Compute the accuracy of the classifier, considering few test data sets.

Aim :

To write a Python program to implement a Naïve Bayesian classifier for a sample training dataset stored in a .CSV file. The program should accurately classify test data sets and compute the classifier's accuracy

1. Import necessary libraries

```
import pandas as pd
from sklearn import tree
from sklearn.preprocessing import LabelEncoder
from sklearn.naive_bayes import GaussianNB
```

2. Load data from CSV

```
data = pd.read_csv('tennisdata.csv')
print("The first 5 values of data is :\n",data.head())
```

Output:

```
The first 5 values of data is :
Outlook Temperature Humidity  Windy PlayTennis
0     Sunny        Hot     High  False        No
1     Sunny        Hot     High   True        No
2  Overcast        Hot     High  False       Yes
3     Rainy       Mild     High  False       Yes
4     Rainy       Cool   Normal  False       Yes
```

3. Obtain Train data and Train output

```
X = data.iloc[:,:-1]
print("\nThe First 5 values of train data is\n",X.head())
```

Output:

```
The First 5 values of train data is
Outlook Temperature Humidity  Windy
0     Sunny        Hot     High  False
1     Sunny        Hot     High   True
2  Overcast        Hot     High  False
3     Rainy       Mild     High  False
4     Rainy       Cool   Normal  False
```

```
y = data.iloc[:,-1]
print("\nThe first 5 values of Train output is\n",y.head())
```

Output:

```
The first 5 values of Train output is
 0    No
```

```
1    No
2    Yes
3    Yes
4    Yes
Name: PlayTennis, dtype: object
```

4. Convert then in numbers

```
le_outlook = LabelEncoder()
X.Outlook = le_outlook.fit_transform(X.Outlook)

le_Temperature = LabelEncoder()
X.Temperature = le_Temperature.fit_transform(X.Temperature)

le_Humidity = LabelEncoder()
X.Humidity = le_Humidity.fit_transform(X.Humidity)

le_Windy = LabelEncoder()
X.Windy = le_Windy.fit_transform(X.Windy)

print("\nNow the Train data is :\n",X.head())
```

Output:

```
Now the Train data is :
   Outlook  Temperature  Humidity  Windy
0       2           1        0     0
1       2           1        0     1
2       0           1        0     0
3       1           2        0     0
4       1           0        1     0
```

```
le_PlayTennis = LabelEncoder()
y = le_PlayTennis.fit_transform(y)
print("\nNow the Train output is\n",y)
```

```
Now the Train output is
 [0 0 1 1 1 0 1 0 1 1 1 1 1 0]
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.20)

classifier = GaussianNB()
classifier.fit(X_train,y_train)

from sklearn.metrics import accuracy_score
print("Accuracy is:",accuracy_score(classifier.predict(X_test),y_test))
```

Output:

```
Accuracy is: 0.6666666666666666
```

CSV FILE

```
Outlook,Temperature,Humidity,Windy,PlayTennis
Sunny,Hot,High,False,No
Sunny,Hot,High,True,No
Overcast,Hot,High,False,Yes
Rainy,Mild,High,False,Yes
Rainy,Cool,Normal,False,Yes
Rainy,Cool,Normal,True,No
Overcast,Cool,Normal,True,Yes
Sunny,Mild,High,False,No
Sunny,Cool,Normal,False,Yes
Rainy,Mild,Normal,False,Yes
Sunny,Mild,Normal,True,Yes
Overcast,Mild,High,True,Yes
Overcast,Hot,Normal,False,Yes
Rainy,Mild,High,True,No
```

2. Implement Linear Regression in a given business scenario and comment on its efficiency and performance.

Aim :

To Implement Linear Regression in a specified business context to analyse its applicability and performance, evaluating its efficiency and predictive capability.

1. Import necessary libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

2. Load the advertising dataset

```
data = pd.read_csv('Ex2.csv')
```

3. Display the first few rows of the dataset

```
print(data.head())
```

4. Prepare the data

```
X = data[['TV', 'Radio', 'Newspaper']]
y = data['Sales']
```

5. Split the data into training and test sets (80% training, 20% test)

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

6. Create a linear regression model

```
model = LinearRegression()
```

7. Train the model using the training sets

```
model.fit(X_train, y_train)
```

8. Make predictions using the testing set

```
y_pred = model.predict(X_test)
```

9. Evaluate the model

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R^2 Score: {r2}')
```

10. Residual plot

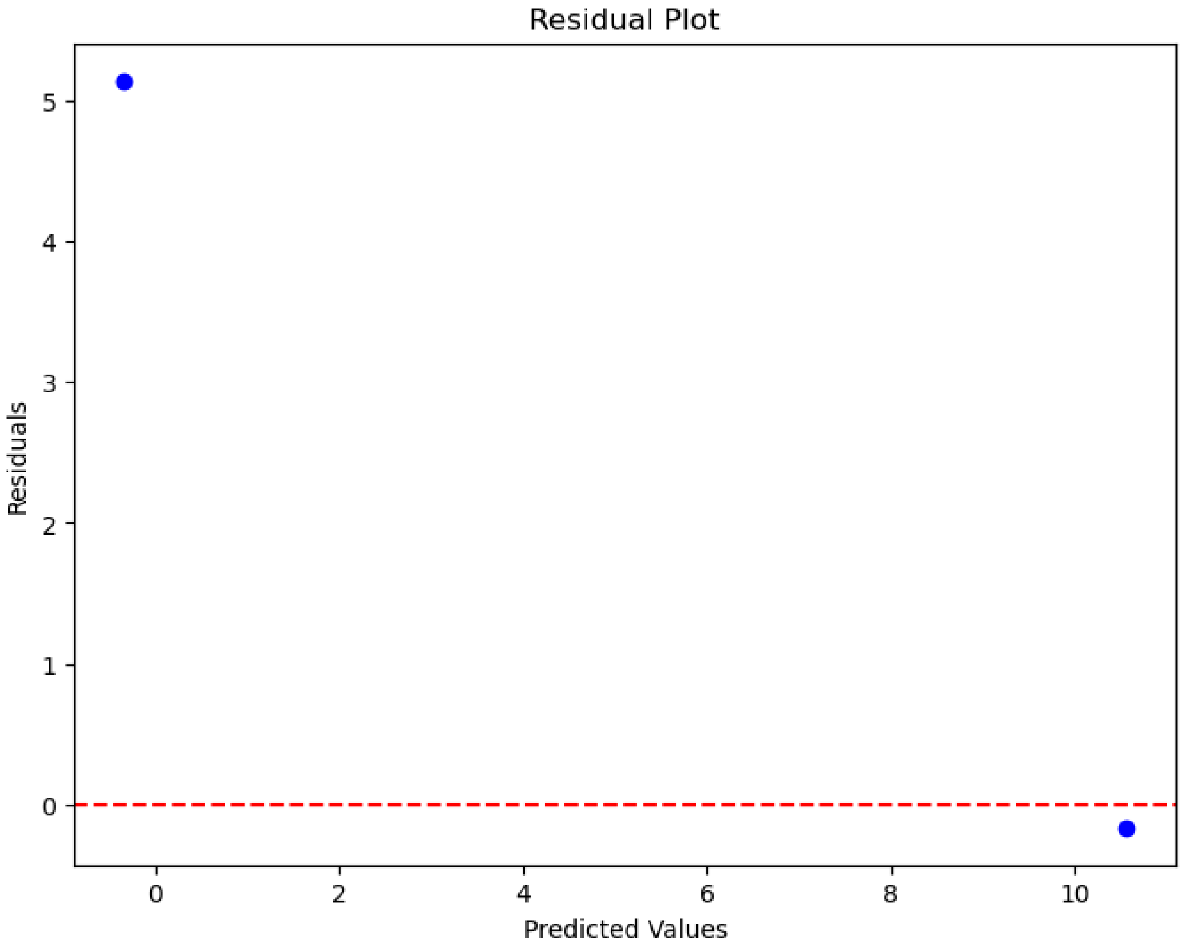
```
residuals = y_test - y_pred
plt.figure(figsize=(8, 6))
plt.scatter(y_pred, residuals, color='blue')
plt.title('Residual Plot')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.axhline(y=0, color='r', linestyle='--')
plt.show()
```

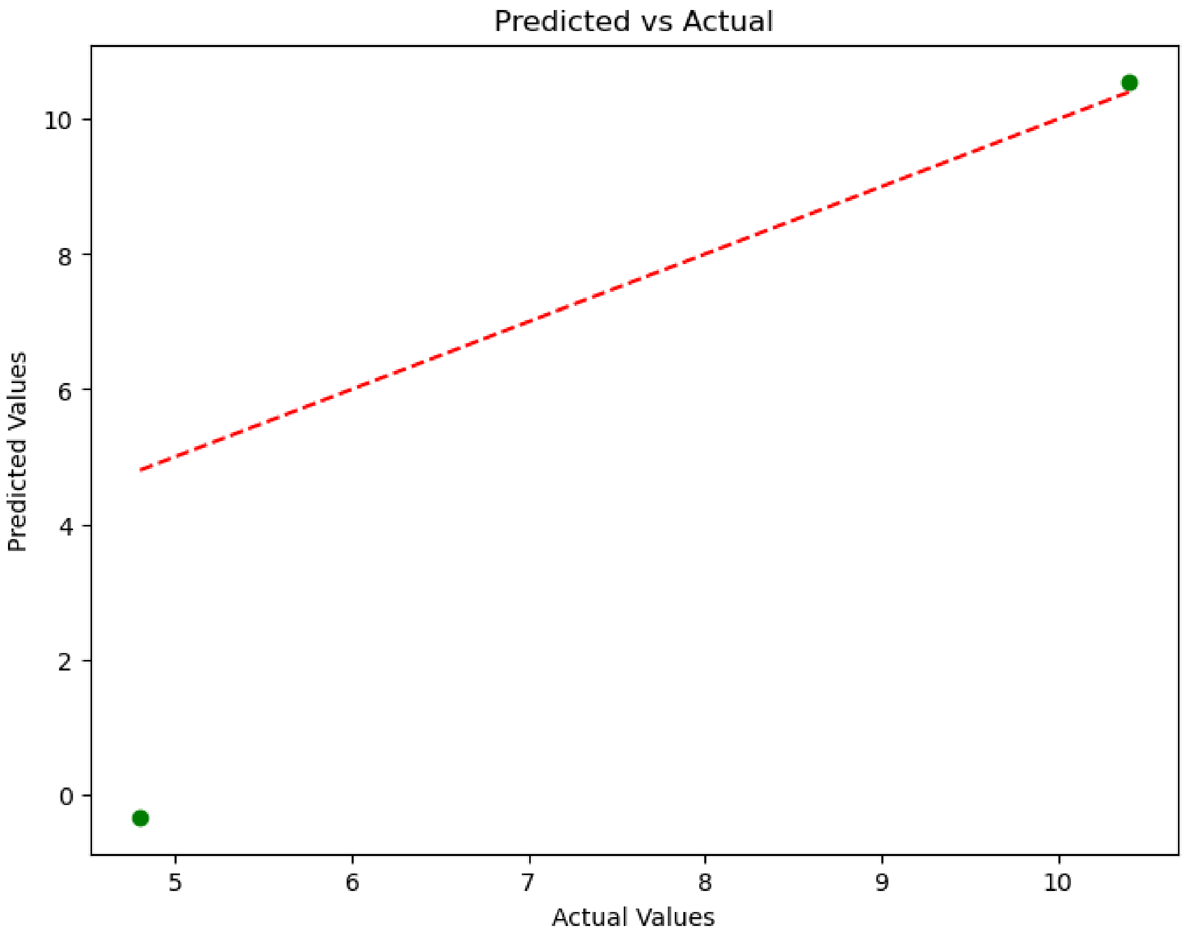
11. Predicted vs Actual plot

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, color='green')
plt.title('Predicted vs Actual')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--')
plt.show()
```

Output:

```
     TV  Radio  Newspaper  Sales
0  230.1   37.8       69.2   22.1
1   44.5   39.3       45.1   10.4
2   17.2   45.9       69.3    9.3
3  151.5   41.3       58.5   18.5
4  180.8   10.8       58.4   12.9
Mean Squared Error: 13.201451790963432
R^2 Score: -0.6838586468065599
```


Residual Plot
Residuals
Predicted Values
0
1
2
3
4
5
0
2
4
6
8
10



In [ ]:

CSV File

```
TV,Radio,Newspaper,Sales
230.1,37.8,69.2,22.1
44.5,39.3,45.1,10.4
17.2,45.9,69.3,9.3
151.5,41.3,58.5,18.5
180.8,10.8,58.4,12.9
8.7,48.9,75,7.2
57.5,32.8,23.5,11.8
120.2,19.6,11.6,13.2
8.6,2.1,1,4.8
199.8,2.6,21.2,10.6
```

3.**Implement SVM algorithm in a given business scenario and comment on its efficiency and performance.**

Aim:

To Implement SVM algorithm in a given business scenario and comment on its efficiency and performance.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score,
f1_score
from sklearn.svm import SVC
```

1. Load dataset

```
data = pd.read_csv('Ex3.csv')
```

2. Handling missing values (if any)

```
data.fillna(method='ffill', inplace=True)
```

3. Encoding categorical variables

```
le = LabelEncoder()
data['Gender'] = le.fit_transform(data['Gender'])
```

4. Splitting the dataset into the Training set and Test set

```
X = data.drop('Purchase', axis=1)
y = data['Purchase']
```

5. Feature scaling

```
sc = StandardScaler()
X_scaled = sc.fit_transform(X)
```

6. Define SVM classifiers

```
classifiers = {
    'Linear SVM': SVC(kernel='linear', random_state=0),
    'Polynomial SVM': SVC(kernel='poly', degree=3, random_state=0),
    'RBF SVM': SVC(kernel='rbf', random_state=0)
}
```

7. Evaluate each classifier using k-fold cross-validation

```
for clf_name, clf in classifiers.items():
    scores = cross_val_score(clf, X_scaled, y, cv=5, scoring='accuracy')
    print(f"{clf_name} Cross-Validation Accuracy: {scores.mean():.2f} (+/- {scores.std() * 2:.2f})")
```

8. Train a selected SVM model (e.g., Linear SVM) on the entire dataset

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.25, random_state=0)
classifier = SVC(kernel='linear', random_state=0)
classifier.fit(X_train, y_train)
```

9. Make Predictions

```
y_pred = classifier.predict(X_test)
```

10. Performance Evaluation

```
cm = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)


print(f"Performance Score: {f1:.2f}")
```

Output:

```
Linear SVM Cross-Validation Accuracy: 0.80 (+/- 0.33)
Polynomial SVM Cross-Validation Accuracy: 0.80 (+/- 0.53)
RBF SVM Cross-Validation Accuracy: 0.80 (+/- 0.53)

Performance Score: 1.00
```

CSV File

```
Age,Income,Gender,Purchase
25,50000,Male,0
45,64000,Female,1
35,58000,Female,0
50,72000,Male,1
23,48000,Male,0
31,52000,Female,0
46,60000,Female,1
29,55000,Male,0
52,75000,Male,1
48,68000,Female,1
36,59000,Male,0
28,53000,Female,0
27,52000,Female,0
44,61000,Male,1
33,57000,Female,1
```

4.Implement Decision Tree algorithm in a given business scenario and comment on its efficiency and performance.

Aim:

To Implement Decision Tree algorithm in a given business scenario and comment on its efficiency and performance.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score, f1_score
```

1. Load dataset

```
data = pd.read_csv('Ex4.csv')
```

2. Data preprocessing

```
# One-Hot Encoding for categorical variable 'Gender'
data = pd.get_dummies(data, columns=['Gender'], drop_first=True)
```

3. Verify columns after one-hot encoding

```
print(data.columns)
```

4. Splitting the dataset into the Training set and Test set

```
X = data.drop('TargetVariable', axis=1)  # Adjust to your actual target
variable name
y = data['TargetVariable']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=0)
```

5. Building the Decision Tree model

```
classifier = DecisionTreeClassifier(random_state=0)
classifier.fit(X_train, y_train)
```

6. Making predictions

```
y_pred = classifier.predict(X_test)
```

7. Performance evaluation

```
cm = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
```

8. Outputting results

```
print("Confusion Matrix:\n", cm)
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"F1 Score: {f1:.2f}")
```

```
Index(['Age', 'Income', 'TargetVariable', 'Gender_Male'], dtype='object')
Confusion Matrix:
 [[0 0]
 [1 3]]
Accuracy: 0.75
Precision: 1.00
Recall: 0.75
F1 Score: 0.86
```

CSV File

```
Age,Income,Gender,TargetVariable
25,50000,Male,0
45,64000,Female,1
35,58000,Female,0
50,72000,Male,1
23,48000,Male,0
31,52000,Female,0
46,60000,Female,1
29,55000,Male,0
52,75000,Male,1
48,68000,Female,1
36,59000,Male,0
28,53000,Female,0
27,52000,Female,0
44,61000,Male,1
33,57000,Female,1
```