```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import sklearn
```

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.

Importing of dataset

```
dataset = pd.read_csv('/content/drive/MyDrive/My_csv_files/Social_Network_Ads.csv')
X = dataset.iloc[:, [1, 2, 3]].values
y = dataset.iloc[:, -1].values
```

X

```
array([['Male', 19, 19000],
       ['Male', 35, 20000],
       ['Female', 26, 43000],
       ...,
       ['Female', 50, 20000],
       ['Male', 36, 33000],
       ['Female', 49, 36000]], dtype=object)
```

y

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 0, 1])
```

Since our dataset containing character variables we have to encode it using LabelEncoder

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:,0] = le.fit_transform(X[:,0])
```

```
X
```

```
array([[1, 19, 19000],
       [1, 35, 20000],
       [0, 26, 43000],
       ...,
       [0, 50, 20000],
       [1, 36, 33000],
       [0, 49, 36000]], dtype=object)
```

Train test Split We are performing a train test split on our dataset. We are providing the test size as 0.20, that means our training sample contains 320 training set and test sample contains 80 test set

```
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, random_state = 0)
```

Feature scaling Next, we are doing feature scaling to the training and test set of independent variables

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
X_train
```

```
[ 1.02532046e+00,  7.59458956e-02,  1.04507694e+00],
[-9.75304830e-01, -1.18475597e-01, -3.74554562e-01],
[-9.75304830e-01, -1.18779381e+00,  6.00265106e-02],
[-9.75304830e-01, -3.12897090e-01, -1.35960499e+00],
[-9.75304830e-01,  1.53410709e+00,  1.10302108e+00],
[ 1.02532046e+00, -7.98950822e-01, -1.53343742e+00],
[ 1.02532046e+00,  7.59458956e-02,  1.85629494e+00],
[ 1.02532046e+00, -8.96161568e-01, -7.80163563e-01],
[ 1.02532046e+00, -5.07318583e-01, -7.80163563e-01],
[ 1.02532046e+00, -3.12897090e-01, -9.25023920e-01],
[ 1.02532046e+00,  2.70367388e-01, -7.22219420e-01],
[-9.75304830e-01,  2.70367388e-01,  6.00265106e-02],
[-9.75304830e-01,  7.59458956e-02,  1.85629494e+00],
[-9.75304830e-01, -1.09058306e+00,  1.94321116e+00],
[-9.75304830e-01, -1.67384754e+00, -1.56240949e+00],
[ 1.02532046e+00, -1.18779381e+00, -1.09885635e+00],
[ 1.02532046e+00, -7.01740076e-01, -1.13805918e-01],
[-9.75304830e-01,  7.59458956e-02,  8.89985821e-02],
```

```
       [ 1.02532046e+00,  2.70367388e-01,  2.62831011e-01],
       [-9.75304830e-01,  8.53631867e-01, -5.77359062e-01],
       [-9.75304830e-01,  2.70367388e-01, -1.15680049e+00],
       [-9.75304830e-01, -1.18475597e-01,  6.68440012e-01],
       [-9.75304830e-01,  2.11737157e+00, -6.93247348e-01],
       [ 1.02532046e+00, -1.28500455e+00, -1.38857706e+00],
       [-9.75304830e-01, -9.93372315e-01, -9.53995992e-01],
       [-9.75304830e-01, -2.12648508e-02, -4.32498705e-01],
       [-9.75304830e-01, -2.15686344e-01, -4.61470776e-01],
       [-9.75304830e-01, -1.77105829e+00, -9.82968063e-01],
       [-9.75304830e-01,  1.72852858e+00,  9.87132798e-01],
       [ 1.02532046e+00,  1.73156642e-01, -3.74554562e-01],
       [-9.75304830e-01,  3.67578135e-01,  1.10302108e+00],
       [-9.75304830e-01, -1.77105829e+00, -1.35960499e+00],
       [ 1.02532046e+00,  1.73156642e-01, -1.42777990e-01],
       [ 1.02532046e+00,  8.53631867e-01, -1.44652121e+00],
       [-9.75304830e-01, -1.96547978e+00,  4.65635512e-01],
       [ 1.02532046e+00, -3.12897090e-01,  2.62831011e-01],
       [-9.75304830e-01,  1.82573933e+00, -1.06988428e+00],
       [-9.75304830e-01, -4.10107836e-01,  6.00265106e-02],
       [-9.75304830e-01,  1.04805336e+00, -8.96051849e-01],
       [-9.75304830e-01, -1.09058306e+00, -1.12782842e+00],
       [ 1.02532046e+00, -1.86826903e+00,  2.08236764e-03],
       [-9.75304830e-01,  7.59458956e-02,  2.62831011e-01],
       [ 1.02532046e+00, -1.18779381e+00,  3.20775154e-01],
       [ 1.02532046e+00, -1.28500455e+00,  2.91803083e-01],
       [-9.75304830e-01, -9.93372315e-01,  4.36663440e-01],
       [ 1.02532046e+00,  1.63131784e+00, -8.96051849e-01],
       [-9.75304830e-01,  1.14526411e+00,  5.23579655e-01],
       [ 1.02532046e+00,  1.04805336e+00,  5.23579655e-01],
       [ 1.02532046e+00,  1.33968560e+00,  2.31984809e+00],
       [-9.75304830e-01, -3.12897090e-01, -1.42777990e-01],
       [ 1.02532046e+00,  3.67578135e-01, -4.61470776e-01],
       [ 1.02532046e+00, -4.10107836e-01, -7.80163563e-01],
       [ 1.02532046e+00, -1.18475597e-01, -5.19414919e-01],
       [-9.75304830e-01,  9.50842613e-01, -1.15680049e+00],
       [ 1.02532046e+00, -8.96161568e-01, -7.80163563e-01],
       [ 1.02532046e+00, -2.15686344e-01, -5.19414919e-01],
       [-9.75304830e-01, -1.09058306e+00, -4.61470776e-01],
       [-9.75304830e-01, -1.18779381e+00,  1.39274180e+00]])
```

Training the Naive Bayes model on the training set

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

```
GaussianNB()
```

predict the test results

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

Predicted and actual value –

```
 y_pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1])
```

```
y_test
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1])
```

```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test, y_pred)
ac = accuracy_score(y_test,y_pred)
```

```
cm
```

```
array([[56,  2],
       [ 4, 18]])
```

```
ac
```

```
0.925
```

Double-click (or enter) to edit

Colab paid products - Cancel contracts here