import numpy as np import pandas as pd

import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings('ignore')

dataset=pd.read\_csv(r"D:\DS\_NIT\Machine Learning\Classifications\Social\_Network\_Ads.csv") x=dataset.iloc[:,[2,3]].values

y=dataset.iloc[:,-1].values

***#Splitting the dataset into training and 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=0.2,random\_state=0)

***#Feature scaling***

from sklearn.preprocessing import MinMaxScaler sc=MinMaxScaler()

***#from sklearn.preprocessing import Normalizer #sc=Normalizer()* x\_train=sc.fit\_transform(x\_train) x\_test=sc.transform(x\_test)**

***#Training the naive bayes modele on the training set* from sklearn.naive\_bayes import MultinomialNB classifier=MultinomialNB() classifier.fit(x\_train,y\_train)**

***#predict the test set results***

y\_pred=classifier.predict(x\_test)

***#Making the confusion matrix***

from sklearn.metrics import confusion\_matrix cm=confusion\_matrix(y\_test,y\_pred) print(cm)

from sklearn.metrics import accuracy\_score ac=accuracy\_score(y\_test,y\_pred) print(ac)

bias=classifier.score(x\_train,y\_train) print(bias)

variance=classifier.score(x\_test,y\_test) print(variance)

from sklearn.metrics import classification\_report cr=classification\_report(y\_test,y\_pred) print(cr)

***#bernoulli Naive Bayes***

from sklearn.naive\_bayes import BernoulliNB bernouli\_cl=BernoulliNB() bernouli\_cl.fit(x\_train,y\_train)

bernouli\_ypred=bernouli\_cl.predict(x\_test) *#Confusion matrix* bern\_cm=confusion\_matrix(y\_test,bernouli\_ypred) print(bern\_cm)

*#accuracy* bern\_ac=accuracy\_score(y\_test,bernouli\_ypred) print(bern\_ac)

***#bias* bern\_bias=bernouli\_cl.score(x\_train,y\_train) print(bern\_bias)**

***#variance* bern\_var=bernouli\_cl.score(x\_test,y\_test) print(bern\_var)**

**from sklearn.metrics import classification\_report bern\_cr=classification\_report(y\_test,bernouli\_ypred) print(bern\_cr)**

**from sklearn.naive\_bayes import GaussianNB gausian\_cl=GaussianNB() gausian\_cl.fit(x\_train,y\_train)**

**gausian\_ypred=gausian\_cl.predict(x\_test)**

***#accuracy* gausian\_ac=accuracy\_score(y\_test,gausian\_ypred) print(gausian\_ac)**

***#bias* gausian\_bias=gausian\_cl.score(x\_train,y\_train) gausian\_bias()**

***#variance* gausian\_variance=gausian\_cl.score(x\_test,y\_test) print(gausian\_variance)**

***#classification report* gausian\_cr=classification\_report(y\_test,gausian\_ypred) print(gausian\_cr)**

|  |  |  |  |
| --- | --- | --- | --- |
| **without scaling** | | | |
|  | accuracy | bias | variance |
|  | 0.5625 | 0.67 | 0.56 |
| **with scaling-Normalizer** | | | |
|  | 0.72 | 0.62 | 0.72 |
| **with scaling-StandardScaler** | | | |
| MultinomialNB does not work with negative values. Since StandardScaler can produce negative values | | | |
| **with scaling-MinMaxScaler** | | | |
|  | 0.72 | 0.62 | 0.72 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Multinominal** | | | | |
|  | **precision** | **recall** | **f1-score** | **support** |
| 0 | 0.72 | 1 | 0.94 | 58 |
| 1 | 0 | 0 | 0.83 | 22 |
| **accuracy** |  |  | 0.72 | 80 |
| **macro avg** | 0.36 | 0.5 | 0.42 | 80 |
| **weighted\_avg** | 0.53 | 0.72 | 0.61 | 80 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bernouli** | | | | |
|  | **precision** | **recall** | **f1-score** | **support** |
| 0 | 0.72 | 1 | 0.84 | 58 |
| 1 | 0 | 0 | 0.83 | 22 |
| **accuracy** |  |  | 0.72 | 80 |
| **macro avg** | 0.36 | 0.5 | 0.42 | 80 |
| **weighted\_avg** | 0.53 | 0.72 | 0.61 | 80 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Gaussian** | | | | |
|  | **precision** | **recall** | **f1-score** | **support** |
| 0 | 0.93 | 0.95 | 0.94 | 58 |
| 1 | 0.86 | 0.82 | 0.84 | 22 |
| **accuracy** |  |  | 0.91 | 80 |
| **macro avg** | 0.89 | 0.88 | 0.89 | 80 |
| **weighted\_avg** | 0.91 | 0.91 | 0.91 | 80 |