*#importing the libraries*

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

In [2]:

dataset**=**pd.read\_csv("C:\Prakash\AI\Learning\Machine\_learning\Classification\Random\_forest\_classifier\Social\_Network\_Ads.csv")

In [3]:

dataset**=**pd.get\_dummies(dataset,drop\_first**=True**)

In [4]:

dataset

Out[4]:

|  | **User ID** | **Age** | **EstimatedSalary** | **Purchased** | **Gender\_Male** |
| --- | --- | --- | --- | --- | --- |
| 0 | 15624510 | 19 | 19000 | 0 | 1 |
| 1 | 15810944 | 35 | 20000 | 0 | 1 |
| 2 | 15668575 | 26 | 43000 | 0 | 0 |
| 3 | 15603246 | 27 | 57000 | 0 | 0 |
| 4 | 15804002 | 19 | 76000 | 0 | 1 |
| ... | ... | ... | ... | ... | ... |
| 395 | 15691863 | 46 | 41000 | 1 | 0 |
| 396 | 15706071 | 51 | 23000 | 1 | 1 |
| 397 | 15654296 | 50 | 20000 | 1 | 0 |
| 398 | 15755018 | 36 | 33000 | 0 | 1 |
| 399 | 15594041 | 49 | 36000 | 1 | 0 |

400 rows × 5 columns

In [5]:

dataset.columns

Out[5]:

Index(['User ID', 'Age', 'EstimatedSalary', 'Purchased', 'Gender\_Male'], dtype='object')

In [6]:

dataset**=**dataset.drop("User ID",axis**=**1)

In [ ]:

dataset

In [7]:

dataset["Purchased"].value\_counts()

Out[7]:

0 257

1 143

Name: Purchased, dtype: int64

In [14]:

dataset.columns

Out[14]:

Index(['Age', 'EstimatedSalary', 'Purchased', 'Gender\_Male'], dtype='object')

In [15]:

independ**=**dataset[["Age", "EstimatedSalary", "Gender\_Male"]]

depend**=**dataset[["Purchased"]]

In [16]:

independ.shape

Out[16]:

(400, 3)

In [17]:

depend

Out[17]:

|  | **Purchased** |
| --- | --- |
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| ... | ... |
| 395 | 1 |
| 396 | 1 |
| 397 | 1 |
| 398 | 0 |
| 399 | 1 |

400 rows × 1 columns

In [20]:

*#split in to training and test set*

**from** sklearn.model\_selection **import** train\_test\_split

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(independ,depend,test\_size**=**1**/**3,random\_state**=**0)

#Multinominal Navie Bayes

from sklearn.naive\_bayes import MultinomialNB

classifier = MultinomialNB()

classifier.fit(x\_train, y\_train)

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

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

from sklearn.metrics import classification\_report

clf\_report = classification\_report(y\_test, y\_pred)

print(clf\_report)

print(cm)

precision recall f1-score support

0 0.63 1.00 0.78 85

1 0.00 0.00 0.00 49

accuracy 0.63 134

macro avg 0.32 0.50 0.39 134

weighted avg 0.40 0.63 0.49 134

[[85 0]

[49 0]]

TP=85, TN=0, FP=0, FN=49

#Bernoulli Naive\_bayes

from sklearn.naive\_bayes import BernoulliNB

classifier = BernoulliNB()

classifier.fit(x\_train, y\_train)

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

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

from sklearn.metrics import classification\_report

clf\_report = classification\_report(y\_test, y\_pred)

print(clf\_report)

print(cm)

precision recall f1-score support

0 0.63 1.00 0.78 85

1 0.00 0.00 0.00 49

accuracy 0.63 134

macro avg 0.32 0.50 0.39 134

weighted avg 0.40 0.63 0.49 134

[[85 0]

[49 0]]

TP=85, TN=0, FP=0, FN=49

#CategoricalNaive bayes

from sklearn.naive\_bayes import CategoricalNB

classifier =CategoricalNB()

classifier.fit(x\_train, y\_train)

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

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

from sklearn.metrics import classification\_report

clf\_report = classification\_report(y\_test, y\_pred)

print(clf\_report)

print(cm)

precision recall f1-score support

0 0.82 0.96 0.89 85

1 0.91 0.63 0.75 49

accuracy 0.84 134

macro avg 0.87 0.80 0.82 134

weighted avg 0.85 0.84 0.84 134

[[82 3]

[18 31]]

TP=82, TN=31, FP=3, FN=18

#complement naive bayes

from sklearn.naive\_bayes import ComplementNB

classifier =ComplementNB()

classifier.fit(x\_train, y\_train)

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

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

from sklearn.metrics import classification\_report

clf\_report = classification\_report(y\_test, y\_pred)

print(clf\_report)

print(cm)

precision recall f1-score support

0 0.67 0.47 0.55 85

1 0.39 0.59 0.47 49

accuracy 0.51 134

macro avg 0.53 0.53 0.51 134

weighted avg 0.57 0.51 0.52 134

[[40 45]

[20 29]]

TP=40, TN=29, FP=45, FN=20

#Gaussian naive bayes

from sklearn.naive\_bayes import GaussianNB

classifier =GaussianNB()

classifier.fit(x\_train, y\_train)

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

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

from sklearn.metrics import classification\_report

clf\_report = classification\_report(y\_test, y\_pred)

print(clf\_report)

print(cm)

precision recall f1-score support

0 0.91 0.94 0.92 85

1 0.89 0.84 0.86 49

accuracy 0.90 134

macro avg 0.90 0.89 0.89 134

weighted avg 0.90 0.90 0.90 134

[[80 5]

[ 8 41]]

TP=80, TN=41, FP=5, FN=8

:

|  | | **Actual** | |
| --- | --- | --- | --- |
| **Purchased** | **Not Purchased** |
| **Predicted** | **Purchased** | True Positive | False Positive |
| **Not Purchased** | False Negative | True Negative |

1. What is the Accuracy of the model

🡪Accuracy is used to measure the performance of the model. It is the ratio of Total correct instances to the total instances.

Accuracy=(TP+TN)/(FP+FN+TP+TN​)

1. What is the Precision of the model.

🡪Precision is a measure of how accurate a model’s positive predictions are. It is defined as the ratio of true positive predictions to the total number of positive predictions made by the model.

Precision= TP/TP+FP

1. What I the Recall/Sensitivity value of the model?

🡪Recall measures the effectiveness of a classification model in identifying all relevant instances from a dataset. It is the ratio of the number of true positive (TP) instances to the sum of true positive and false negative (FN) instances.

Recall= TP/TP+FN

1. What is the F1-Score of the model

🡪F1-score is used to evaluate the overall performance of a classification model. It is the harmonic mean of precision and recall,

F1-Score=(2\*Precision\*recall)/(Precision+Recall)

1. What is the specificity of the model?

🡪Specificity is another important metric in the evaluation of classification models, particularly in binary classification. It measures the ability of a model to correctly identify negative instances. Specificity is also known as the True Negative Rate.

Specificity =TN/(TN+FP)

1. Type 1 and Type 2 error

* Type 1 error occurs when the model predicts a positive instance, but it is actually negative. Precision is affected by false positives, as it is the ratio of true positives to the sum of true positives and false positives.

=FP/TN+FP

* Type 2 error occurs when the model fails to predict a positive instance. Recall is directly affected by false negatives, as it is the ratio of true positives to the sum of true positives and false negatives.

=FP/TP+FN

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Paremeter | Precision | | Recall | | F1-Score | |  |  |  |
| Module | Confusion Matrix | Accuracy | Not Purchased | Purchased | Not Purchased | Purchased | Not Purchased | Purchased | Specificity TN/(TN+FP) | Type 1 error FP/TN+FP | Type 2 error FP/TP+FN |
| Multinominal NB | [[85 0]  [49 0]] | 0.63 | 0.63 | 0 | 1 | 0 | 0.78 | 0 | 0 | 0 | 0 |
| Categorical NB | [[82 3]  [18 31]] | 0.84 | 0.82 | 0.91 | 96 | 63 | 89 | 75 | 0.9117647 | 0.08823529 | 0.06122449 |
| Gausian NB | [[80 5]  [ 8 41]] | 0.9 | 0.91 | 0.89 | 0.94 | 0.84 | 0.92 | 0.86 | 0.8913043 | 0.10869565 | 0.05681818 |
| complement NB | [[40 45]  [20 29]] | 0.51 | 0.67 | 0.39 | 0.47 | 0.59 | 0.55 | 0.47 | 0.3918919 | 0.60810811 | 0.91836735 |
| Bernoulli NB | [[85 0]  [49 0]] | 0.63 | 0.63 | 0 | 1 | 0 | 0.78 | 0 | 0 | 0 | 0 |