In [1]:
```
#added all libraries
import numpy as np1 #for the numarray
import matplotlib.pyplot as plt1 # for the plotting between data set
import pandas as pd1 #for dataframe
from sklearn.tree import DecisionTreeClassifier #for the decision tree algorithm
from sklearn.linear_model import LogisticRegression  #for the LogisticRegression
from sklearn.svm import SVC  #for the support vector classification
from sklearn.neighbors import KNeighborsClassifier #for the  k-nearest neighbors
from sklearn.naive_bayes import GaussianNB # for the Gaussian Naive Bayes
from sklearn.ensemble import VotingClassifier #for Soft Voting and Majority clas
from sklearn import model_selection #for model selection
from sklearn.metrics import confusion_matrix #for confusion matrix
from sklearn. preprocessing import StandardScaler #for StandardScaler algorithm
from sklearn.model_selection import train_test_split # for model selection
```

In [ ]:
```
#Reading the dataset
data = pd1.read_csv('data.csv')
#Reading the Data
Data = pd1.read_csv('data.csv')
X1 = Data.iloc[:, 3:13].values
y1 = Data.iloc[:, 13].values
```

In [2]:
```
# categorical data
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
#LabelEncoder methof for encoding data
labelencoder_X1 = LabelEncoder()
#for holding data in X1
X1[:, 1] = labelencoder_X1.fit_transform(X1[:, 1])
labelencoder_X2 = LabelEncoder()
#for holding data from 2nd in datasetin X1
```

```
X1[:, 2] = labelencoder_X2.fit_transform(X1[:, 2])
```

In [3]:

```
#column transfrormer

from sklearn.compose import ColumnTransformer

ct1 = ColumnTransformer([("Geography1", OneHotEncoder(), [1])], remainder = 'pas

X1 = ct1.fit_transform(X1)

X1 = X1[:, 1:]


# dividing data into Training set and Test set

X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, y1, test_size = 0.20


#  Scaling data

sc1 = StandardScaler()

X_train1 = sc1.fit_transform(X_train1)

X_test1 = sc1.transform(X_test1)


#initialize  the machine learning models

model111 = LogisticRegression()

model211 = DecisionTreeClassifier(max_depth = 2)

model311 = SVC()

model411 = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)

model511 = GaussianNB()


#Training model

model111.fit(X_train1, y_train1)

model211.fit(X_train1, y_train1)

model311.fit(X_train1, y_train1)

model411.fit(X_train1, y_train1)

model511.fit(X_train1, y_train1)


#for the prediction

y_pred11 = model111.predict(X_test1)

y_pred21 = model211.predict(X_test1)
```

```
y_pred31 = model311.predict(X_test1)

y_pred41 = model411.predict(X_test1)

y_pred51= model511.predict(X_test1)

#built Confusion matrix

cm_LogisticRegression1 = confusion_matrix(y_test1, y_pred11)

cm_DecisionTree1 = confusion_matrix(y_test1, y_pred21)

cm_SupportVectorClass1 = confusion_matrix(y_test1, y_pred31)

cm_KNN1 = confusion_matrix(y_test1, y_pred41)

cm_NaiveBayes1 = confusion_matrix(y_test1, y_pred51)
```

In [4]:

```
#Kfold machine leanring model as classifier

kfold1 = model_selection.KFold(n_splits=10, random_state = 0,shuffle=True)

result11 = model_selection.cross_val_score(model111, X_train1, y_train1, cv=kfol

result21 = model_selection.cross_val_score(model211, X_train1, y_train1, cv=kfol

result31 = model_selection.cross_val_score(model311, X_train1, y_train1, cv=kfol

result41 = model_selection.cross_val_score(model411, X_train1, y_train1, cv=kfol

result51 = model_selection.cross_val_score(model511, X_train1, y_train1, cv=kfol

#displaying the accuracies using in cross-validation

print('Accuracy for Logistic Regression Model using mean method = ',result11.mea

print('Accuracy for Decision Tree Model using mean method = ',result21.mean())

print('Accuracy for Support Vector Machine Using mean method = ',result31.mean()

print('Accuracy for k-NN Model using mean method  = ',result41.mean())

print('Accuracy for Naive Bayes Model using mean method  = ',result51.mean())

# Hybrid Ensemble Learning Model

estimators1 = []
```

```
Accuracy for Logistic Regression Model using mean method =  0.812
Accuracy for Decision Tree Model using mean method =  0.828
Accuracy for Support Vector Machine Using mean method =  0.8542500000000001
Accuracy for k-NN Model using mean method  =  0.82875
Accuracy for Naive Bayes Model using mean method  =  0.8238749999999999
```

In [5]:

```
#declared 5 Logistic Regression Model
```

```
model111 = LogisticRegression(penalty = 'l2', random_state = 0)
estimators1.append(('logistic1', model111))
model121 = LogisticRegression(penalty = 'l2', random_state = 0)
estimators1.append(('logistic2', model121))
model131 = LogisticRegression(penalty = 'l2', random_state = 0)
estimators1.append(('logistic3', model131))
model141 = LogisticRegression(penalty = 'l2', random_state = 0)
estimators1.append(('logistic4', model141))
model151 = LogisticRegression(penalty = 'l2', random_state = 0)
estimators1.append(('logistic5', model151))

#declared 5 Decision Tree Classifier
model161 = DecisionTreeClassifier(max_depth = 3)
estimators1.append(('cart1', model161))
model171 = DecisionTreeClassifier(max_depth = 4)
estimators1.append(('cart2', model171))
model181 = DecisionTreeClassifier(max_depth = 5)
estimators1.append(('cart3', model181))
model191 = DecisionTreeClassifier(max_depth = 2)
estimators1.append(('cart4', model191))
model201 = DecisionTreeClassifier(max_depth = 3)
estimators1.append(('cart5', model201))

 #declared 5 Support Vector Classifier
model211 = SVC(kernel = 'linear')
estimators1.append(('svm1', model211))
model221 = SVC(kernel = 'poly')
estimators1.append(('svm2', model221))
model231 = SVC(kernel = 'rbf')
estimators1.append(('svm3', model231))
model241 = SVC(kernel = 'rbf')
```

```
estimators1.append(('svm4', model241))

model251 = SVC(kernel = 'linear')

estimators1.append(('svm5', model251))
```

In [6]:

```
# make the ensemble model

ensemble1 = VotingClassifier(estimators1)

ensemble1.fit(X_train1, y_train1)

y_pred1 = ensemble1.predict(X_test1)
```

In [7]:

```
#make Confisuin matrix

cm_HybridEnsembler = confusion_matrix(y_test1, y_pred1)
```

In [8]:

```
#make Cross-Validation

seed = 7

kfold1 = model_selection.KFold(n_splits=10, random_state=seed,shuffle=True)

results1 = model_selection.cross_val_score(ensemble1, X_train1, y_train1, cv=kfo

print(results1.mean())
```

```
0.8405000000000001
```