MACHINE LEARNING LAB ANSWER SCRIPT

Name : Sudipta Biswas

Roll No: 001811001093

Subject: Machine Learning Lab (IT/PC/B/S/411)

Year: 4

Semester: 1

Department : Information Technology

GitHub Link: https://github.com/sudiptajuit/ML-lab/tree/main/Final%20Evaluation%20of%20ML%20Lab

Question1:

Apply different types of following machine learning models: **(CO1) (10)**

- Support Vector Machine (SVM)
- Decision Tree
- Random Forest
- Naive Bayes

And compare as well as discuss the performances in terms of Accuracy, Precision, Recall and F-measure on the following datasets:

1. Wine Dataset: https://archive.ics.uci.edu/ml/datasets/wine
2. Ionosphere Dataset: https://archive.ics.uci.edu/ml/datasets/Ionosphere

Generate the respective confusion matrices with class labels (heat map).

IMPORT THE REQUIRED HEADERS

In [60]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scikitplot as skplt

from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier from
sklearn.ensemble import RandomForestClassifier from
sklearn.naive_bayes import GaussianNB, BernoulliNB
```

Setting up the dataset

In [61]:

```
def preprocess(X,y,te_size,label=False,scale=False,pca=False):
   if
label:
    from sklearn.preprocessing import LabelEncoder
y = LabelEncoder().fit_transform(y)

  from sklearn.model_selection import train_test_split
  X_tr,X_te,y_tr,y_te = train_test_split(X,y,test_size=te_size)
   if
scale:
    from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
    X_tr = sc.fit_transform(X_tr)
    X_te = sc.transform(X_te)
   if
pca:
    from sklearn.decomposition import PCA
    pca = PCA(n_components='mle')
    X_tr = pca.fit_transform(X_tr)
    X_te = pca.transform(X_te)

  return X_tr,X_te,y_tr,y_te
```

Tester function to display

In [62]:

```
def tester(classi,X_t,y_t,y_p):

  from sklearn.metrics import
classification_report,confusion_matrix,accuracy_score,   print("Confusion
Matrix")   print(confusion_matrix(y_t,y_p))
  print('-----------------------------------')
print('-----------------------------------\n\n')
print('Preformance Evaluation:')
print(classification_report(y_t,y_p))   print('------
-----------------------------')   print('------------
-----------------------\n\n')   print('Accuracy
Score:')   print(accuracy_score(y_t,y_p))
plot_confusion_matrix(classi,X_t,y_t)
plt.title('Heat map for confusion matrix')
plt.show()
  y_p_proba = classifier.predict_proba(X_t)
skplt.metrics.plot_roc(y_t,y_p_proba)
plt.show()
```

IMPORTING THE WINE DATASET

In [63]:

```
df1 = pd.read_csv('wine.data', header=None)
 X = df1.iloc[:,1:] y = df1.iloc[:,0]
```

TRAIN-TEST SPLIT OF 50:50

In [64]:

```
X_train,X_test,y_train,y_test = preprocess(X,y,0.5,scale=True,pca=True)
```

SVM CLASSIFIERS with Linear, Polynomial(degree2), Polynomial(degree3), Gaussian, Sigmoid

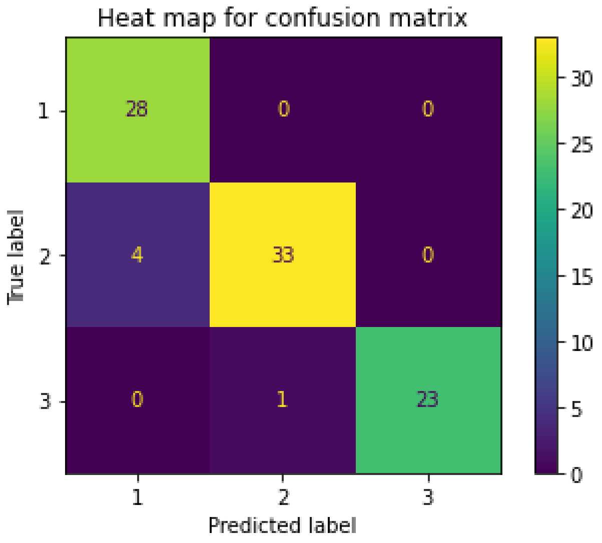
In [65]:

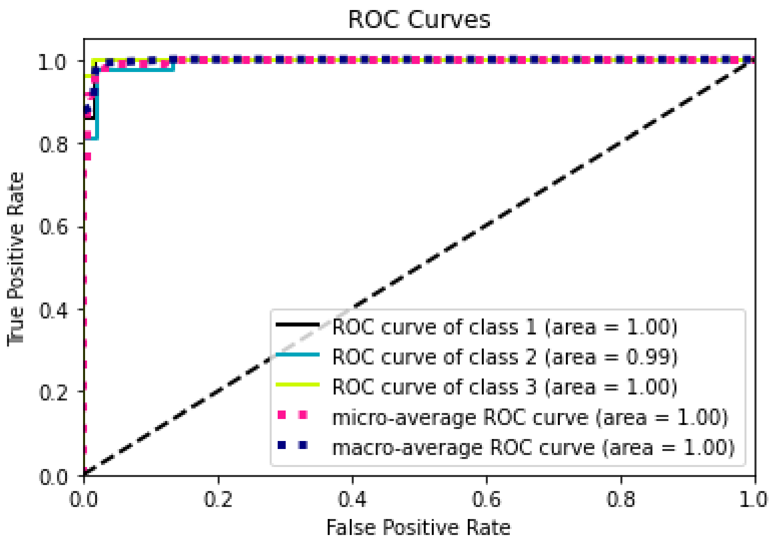
```
#SVM linear model
classifier = SVC(kernel='linear',
probability=True) classifier.fit(X_train,y_train)
y_pred = classifier.predict(X_test) print('SVC
Linear:')
tester(classifier,X_test,y_test,y_pred) print('----------
---------------------------------\n\n\n')
#SVM polynomial model degree 2
classifier = SVC(kernel='poly', degree=2,
probability=True) classifier.fit(X_train,y_train) y_pred
= classifier.predict(X_test) print('SVC Polynomial degree
2:') tester(classifier,X_test,y_test,y_pred) print('-----
--------------------------------------\n\n\n')
#SVM polynomial model degree 3
classifier = SVC(kernel='poly', degree=3,
probability=True) classifier.fit(X_train,y_train) y_pred
= classifier.predict(X_test) print('SVC Polynomial degree
3:') tester(classifier,X_test,y_test,y_pred) print('-----
--------------------------------------\n\n\n')
#SVM gaussian model
classifier = SVC(kernel='rbf',
probability=True)
classifier.fit(X_train,y_train) y_pred =
classifier.predict(X_test) print('SVC
Gaussian:')
tester(classifier,X_test,y_test,y_pred) print('----------
---------------------------------\n\n\n')
#SVM sigmoid model
classifier = SVC(kernel='sigmoid',
probability=True) classifier.fit(X_train,y_train)
y_pred = classifier.predict(X_test) print('SVC
Sigmoid:')
tester(classifier,X_test,y_test,y_pred) print('---------------------------------
----------\n\n\n')
```

```
SVC Linear:
Confusion Matrix
[[28  0  0]
 [ 4 33  0]
 [ 0  1 23]]
-----------------------------------
-----------------------------------
Preformance Evaluation:               precision
recall  f1-score   support

1       0.88      1.00      0.93        28
2       0.97      0.89      0.93        37
        3       1.00      0.96      0.98        24

    accuracy                           0.94
89     macro avg       0.95      0.95      0.95
89 weighted avg       0.95      0.94      0.94
89
```

```
----------------------------------- ---------------
--------------------
```

```
Accuracy Score:
0.9438202247191011
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarnin g: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: Confusio nMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.   warnings.warn(msg,
category=FutureWarning)
```

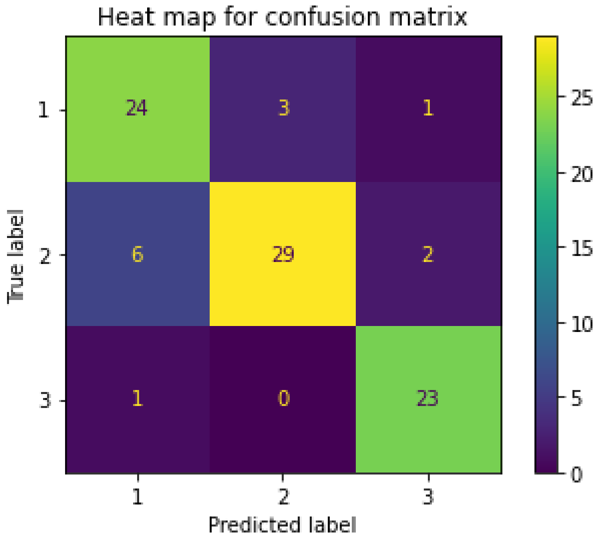


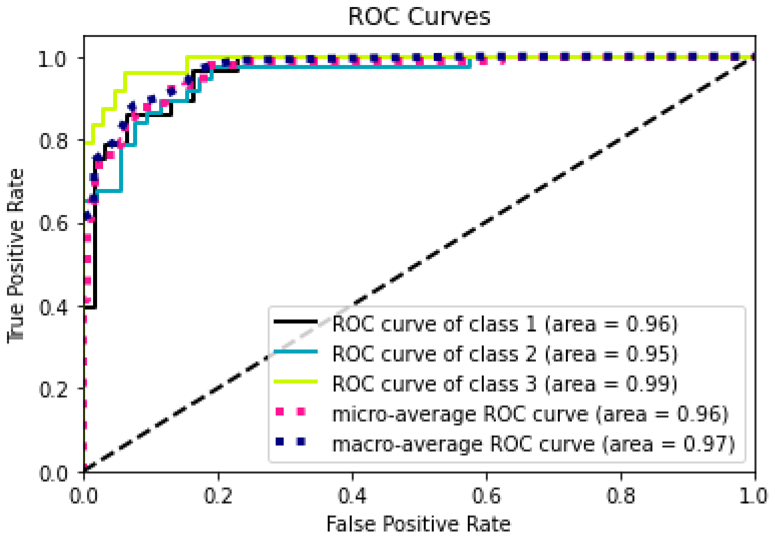


```
-------------------------------------------
```

```
SVC Polynomial degree 2:
Confusion Matrix
[[24  3  1]
 [ 6 29  2]
 [ 1  0 23]]
-----------------------------------
-----------------------------------
```

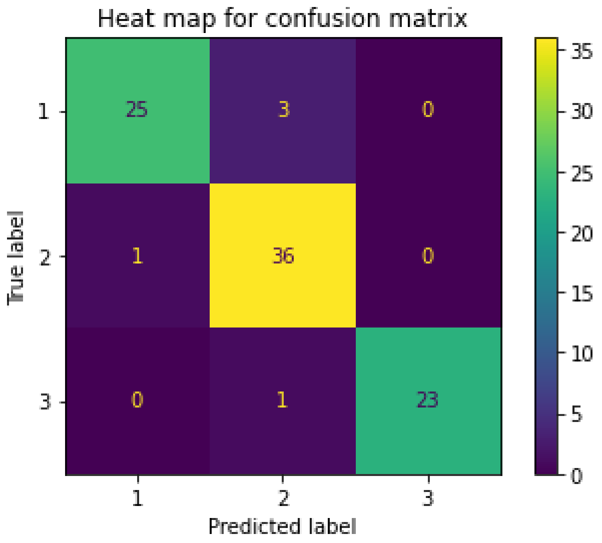
```
Preformance Evaluation:                   precision
recall  f1-score   support

1       0.77      0.86      0.81        28
2       0.91      0.78      0.84        37
        3       0.88      0.96      0.92        24

    accuracy                           0.85
89    macro avg       0.86      0.87      0.86
89 weighted avg       0.86      0.85      0.85
89

----------------------------------- ---------------
--------------------


Accuracy Score:
0.8539325842696629
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarnin g: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: Confusio nMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.   warnings.warn(msg,
category=FutureWarning)
```

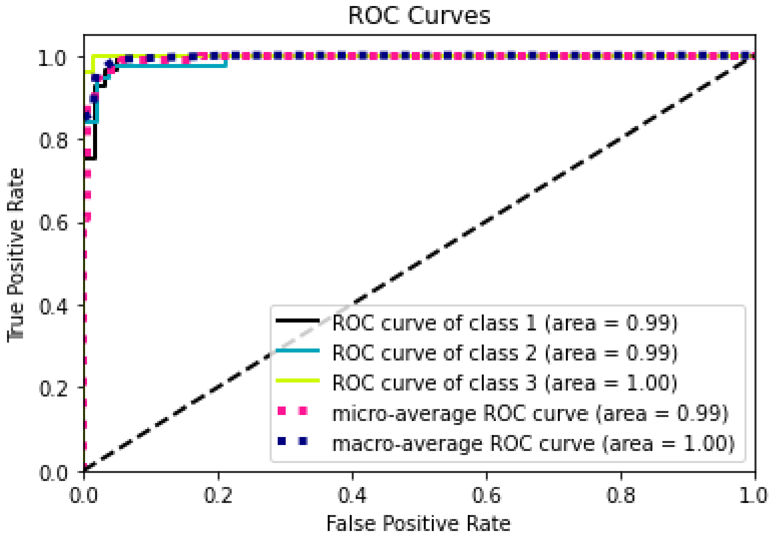




```
-------------------------------------------


SVC Polynomial degree 3:


Confusion Matrix
[[25  3  0]
 [ 1 36  0]
 [ 0  1 23]]
-----------------------------------
-----------------------------------


Preformance Evaluation:               precision
recall  f1-score   support

1       0.96      0.89      0.93        28
2       0.90      0.97      0.94        37

3       1.00      0.96      0.98        24

    accuracy                           0.94
89    macro avg       0.95      0.94      0.95
89 weighted avg       0.95      0.94      0.94
89

-----------------------------------
-----------------------------------


Accuracy Score:
0.9438202247191011
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
```



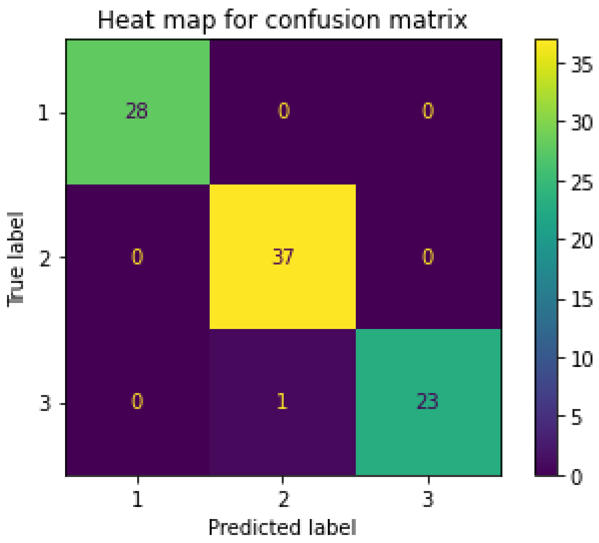


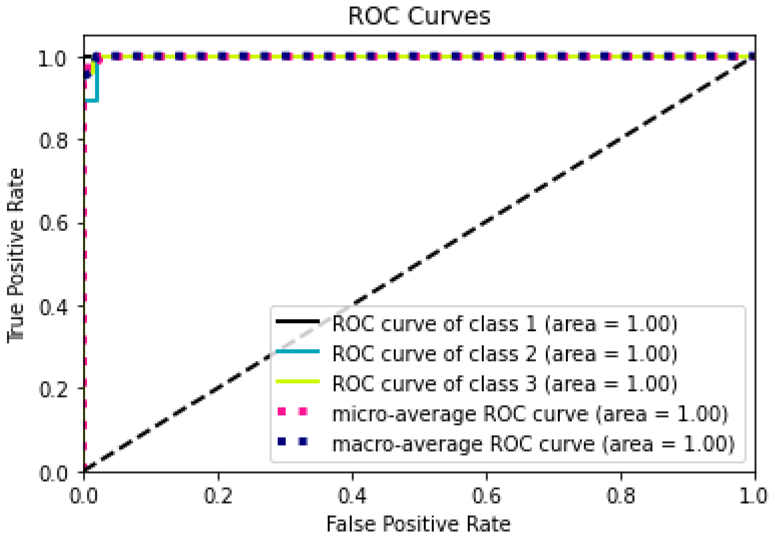
```
FutureWarnin g: Function

plot_confusion_matrix is

deprecated; Function

`plot_confusion_matrix` is

deprecated in 1.0 and will be removed in 1.2. Use one of the class methods:

Confusio nMatrixDisplay.from_predictions or
```

```
ConfusionMatrixDisplay.from_estimator.   warnings.warn(msg,
```

```
category=FutureWarning) ------------------------------------------
```

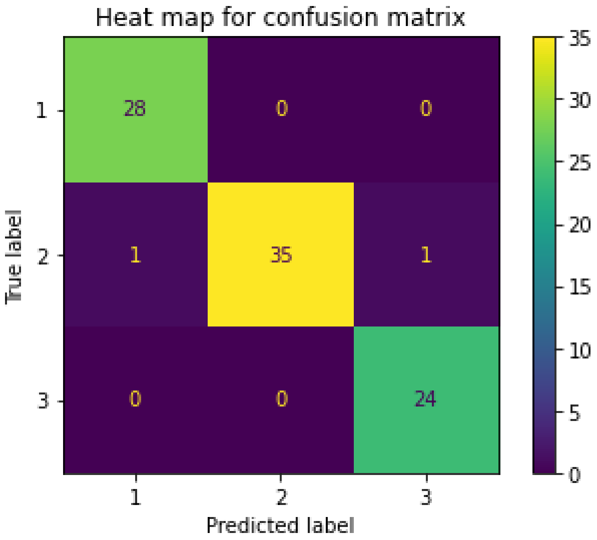
```
SVC Gaussian:
Confusion Matrix
[[28  0  0]
 [ 0 37  0]
 [ 0  1 23]]
----------------------------------- ---------------
--------------------


Preformance Evaluation:               precision
recall  f1-score   support

1       1.00      1.00      1.00        28
2       0.97      1.00      0.99        37
        3       1.00      0.96      0.98        24

    accuracy                           0.99
89    macro avg       0.99      0.99      0.99
89 weighted avg       0.99      0.99      0.99
89

-----------------------------------
-----------------------------------


Accuracy Score:
0.9887640449438202
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarnin g: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: Confusio nMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.   warnings.warn(msg,
category=FutureWarning)
```

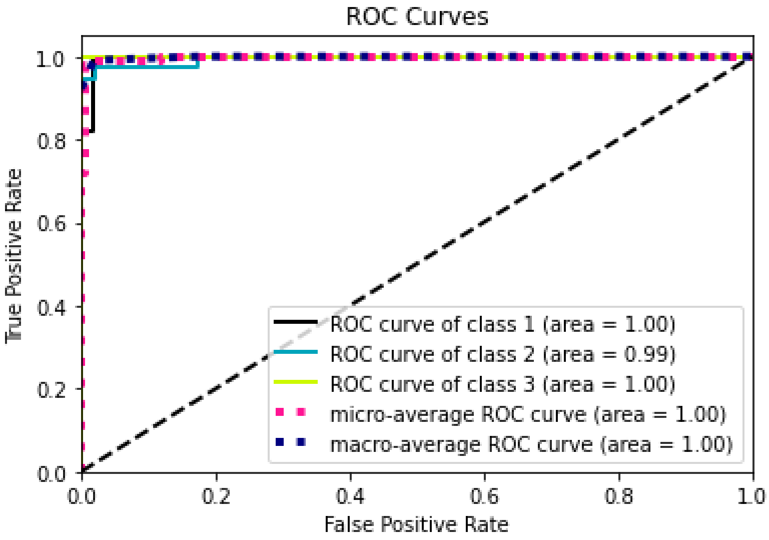




```
-------------------------------------------


SVC Sigmoid:
Confusion Matrix
[[28  0  0]
 [ 1 35  1]
 [ 0  0 24]]
----------------------------------- ---------------
--------------------


Preformance Evaluation:               precision
recall  f1-score   support

1       0.97      1.00      0.98        28
2       1.00      0.95      0.97        37
        3        0.96      1.00      0.98        24

    accuracy                           0.98
89     macro avg       0.98      0.98      0.98
89 weighted avg       0.98      0.98      0.98
89

-----------------------------------
-----------------------------------
```

```
Accuracy Score:
0.9775280898876404
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarnin g: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: Confusio nMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```





```
-------------------------------------------
```

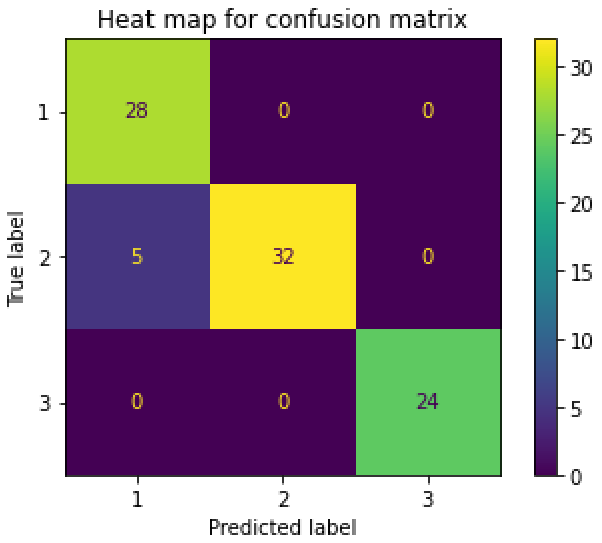
DECISION TREE CLASSIFIER

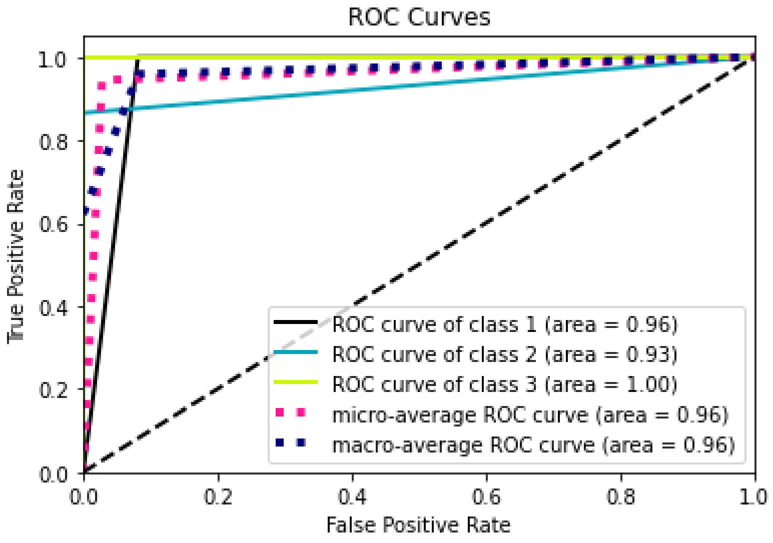
In [66]:

```
#Decision tree classifier
classifier=DecisionTreeClassifier()
classifier.fit(X_train,y_train) y_pred
= classifier.predict(X_test)
print('Decision Tree Classifier:')
tester(classifier,X_test,y_test,y_pred)
print('-------------------------------------------\n\n\n')
```

```
Decision Tree Classifier:
Confusion Matrix
[[28  0  0]
```

```
 [ 5 32  0]
 [ 0  0 24]]
-----------------------------------
-----------------------------------


Preformance Evaluation:              precision
recall  f1-score   support

1       0.85      1.00      0.92        28
2       1.00      0.86      0.93        37
3       1.00      1.00      1.00        24
    accuracy                          0.94
89    macro avg       0.95      0.95      0.95
89 weighted avg       0.95      0.94      0.94
89

----------------------------------- ---------------
--------------------


Accuracy Score:
0.9438202247191011
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarnin g: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: Confusio nMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.   warnings.warn(msg,
category=FutureWarning)
```





```
-------------------------------------------
```

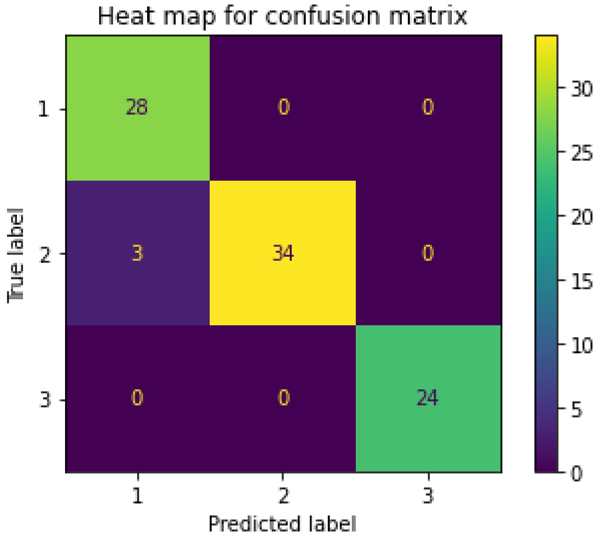
RANDOM FOREST MODEL

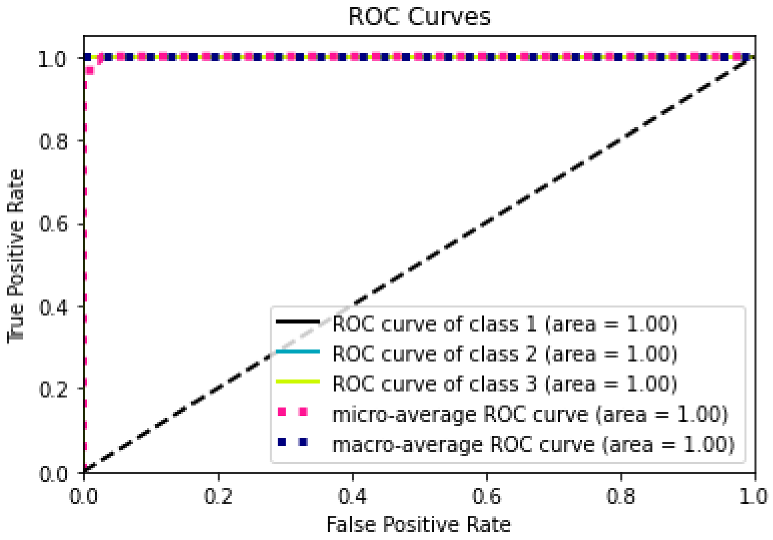
In [67]:

```
#Random forest model
classifier=RandomForestClassifier()
classifier.fit(X_train,y_train) y_pred
= classifier.predict(X_test)
print('Random Forest Classifier:')
tester(classifier,X_test,y_test,y_pred)
print('-------------------------------------------\n\n\n')
```

```
Random Forest Classifier:
Confusion Matrix
[[28  0  0]
 [ 3 34  0]
 [ 0  0 24]]
----------------------------------- ---------------
--------------------
```

```
Preformance Evaluation:                   precision
recall  f1-score   support

1       0.90      1.00      0.95        28
2       1.00      0.92      0.96        37
        3       1.00      1.00      1.00        24

    accuracy                          0.97
89    macro avg       0.97      0.97      0.97
89 weighted avg       0.97      0.97      0.97
89

----------------------------------- ---------------
--------------------


Accuracy Score:
0.9662921348314607
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarnin g: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: Confusio nMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```





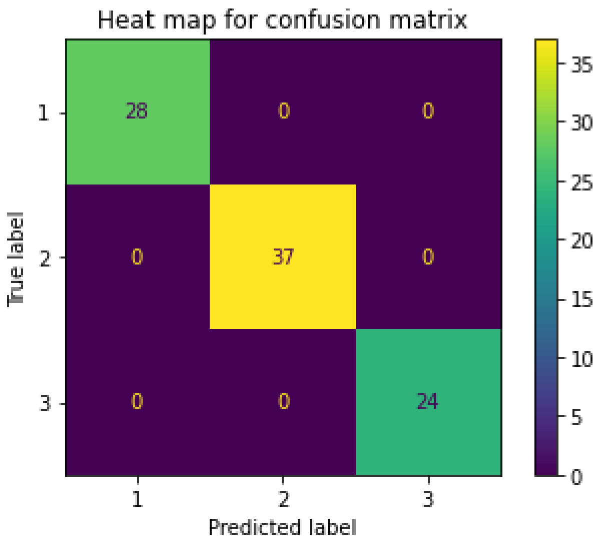
```
-------------------------------------------
```

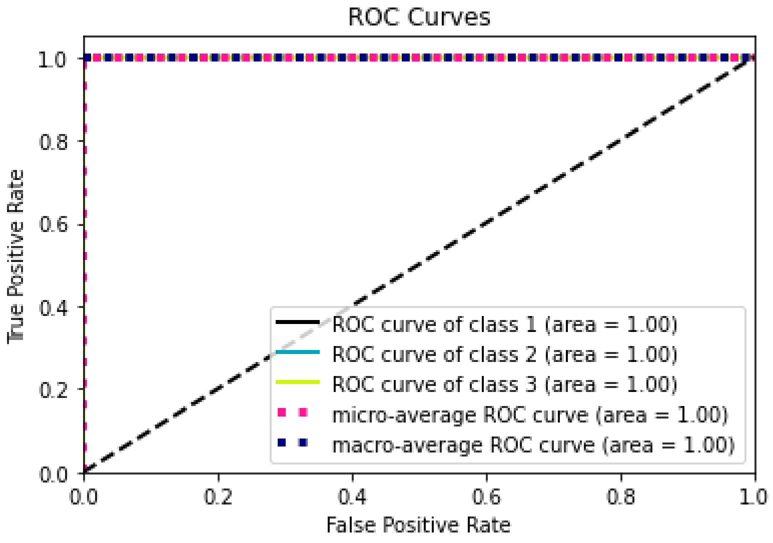
NAIVE BAYES CLASSIFIER

In [68]:

```
#Gaussian naive bayes
classifier=GaussianNB()
classifier.fit(X_train,y_train) y_pred
= classifier.predict(X_test)
print('Gaussian Naive Bayes:')
tester(classifier,X_test,y_test,y_pred)
print('-------------------------------------------\n\n\n')
#Bernoulli naive bayes
classifier=BernoulliNB()
classifier.fit(X_train,y_train) y_pred
= classifier.predict(X_test)
print('Bernoulli Naive Bayes:')
tester(classifier,X_test,y_test,y_pred)
print('-------------------------------------------\n\n\n')
```

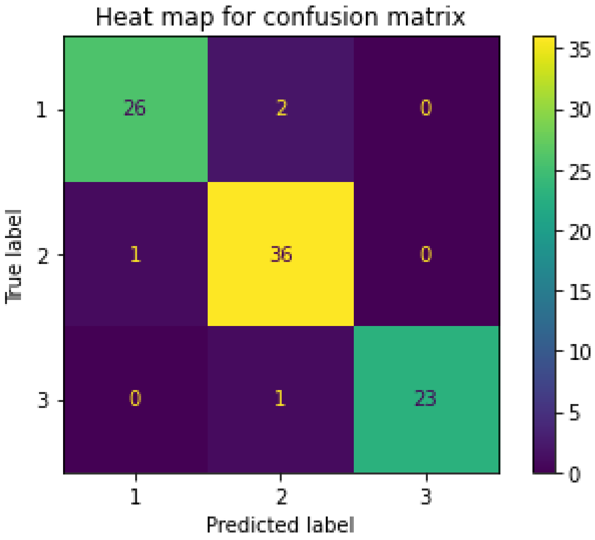
```
Gaussian Naive Bayes:
Confusion Matrix
[[28  0  0]
 [ 0 37  0]
 [ 0  0 24]]
-----------------------------------
-----------------------------------


Preformance Evaluation:               precision
recall  f1-score   support

1       1.00      1.00      1.00        28
2       1.00      1.00      1.00        37
        3       1.00      1.00      1.00        24

    accuracy                           1.00
89     macro avg       1.00      1.00      1.00
89 weighted avg       1.00      1.00      1.00
89

-----------------------------------
-----------------------------------


Accuracy Score:
1.0
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarnin g: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: Confusio nMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.   warnings.warn(msg,
category=FutureWarning)
```

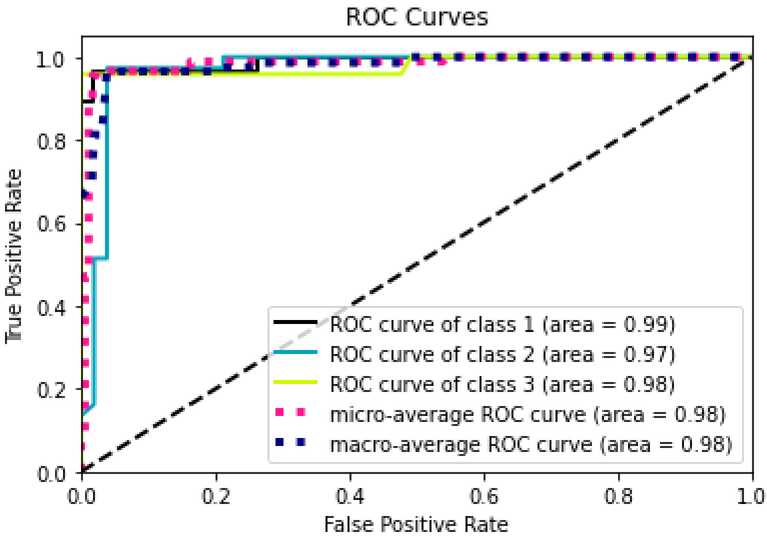




```
-------------------------------------------


Bernoulli Naive Bayes:
Confusion Matrix
[[26  2  0]
 [ 1 36  0]
 [ 0  1 23]]
-----------------------------------
-----------------------------------


Preformance Evaluation:               precision
recall  f1-score   support

1       0.96      0.93      0.95        28
2       0.92      0.97      0.95        37
        3       1.00      0.96      0.98        24

    accuracy                          0.96
89     macro avg       0.96      0.95      0.96
89 weighted avg       0.96      0.96      0.96
89

----------------------------------- ---------------
--------------------
```

```
Accuracy Score:
0.9550561797752809
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
```





```
FutureWarnin g: Function
plot_confusion_matrix is
deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: Confusio nMatrixDisplay.from_predictions or
```

```
ConfusionMatrixDisplay.from_estimator.                          warnings.warn(msg,



category=FutureWarning) -------------------------------------------
```

WINE DATASET

	Accuracy	**Precision**	**Recall**	**F-Measure**
SVM CLASSIFIER				
Linear	0.943	0.95	0.95	0.95
Polynomial (2)	0.853	0.86	0.87	0.86
Polynomial (3)	0.943	0.95	0.94	0.95
Gaussian	**0.988**	0.99	0.99	0.99
Sigmoid	0.977	0.98	0.98	0.98
DECISION TREE CLASSIFIER				
Decision Tree	0.943	0.95	0.94	0.95
RANDOM FOREST CLASSIFIER				
Random Forest	0.966	0.97	0.97	0.97
NAÏVE BAYES CLASSIFIER				
Gaussian	**1**	1	1	1
Bernoulli	0.955	0.96	0.95	0.96

From the above tabulated data, we can conclude that at **50:50** train test split ratio in the **WINE** dataset, **Gaussian Naïve Bayes Classifier** produces the maximum accuracy followed by the **SVM Gaussian classifier**.

IMPORT THE REQUIRED HEADERS

In [41]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scikitplot as skplt

from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier from
sklearn.ensemble import RandomForestClassifier from
sklearn.naive_bayes import GaussianNB, BernoulliNB
```

Setting up the dataset

In [26]:

```
def preprocess(X,y,te_size,label=False,scale=False,pca=False):
   if
label:
    from sklearn.preprocessing import LabelEncoder
y = LabelEncoder().fit_transform(y)

  from sklearn.model_selection import train_test_split
  X_tr,X_te,y_tr,y_te = train_test_split(X,y,test_size=te_size)
   if
scale:
    from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
    X_tr = sc.fit_transform(X_tr)
    X_te = sc.transform(X_te)
   if
pca:
    from sklearn.decomposition import PCA
    pca = PCA(n_components='mle')
    X_tr = pca.fit_transform(X_tr)
    X_te = pca.transform(X_te)

  return X_tr,X_te,y_tr,y_te
```

Tester function to display

```
In [42]:

def tester(classi,X_t,y_t,y_p):

  from sklearn.metrics import
classification_report,confusion_matrix,accuracy_score,   print("Confusion
Matrix")   print(confusion_matrix(y_t,y_p))
  print('-----------------------------------')
print('-----------------------------------\n\n')
print('Preformance Evaluation:')
print(classification_report(y_t,y_p))   print('------
-----------------------------')   print('------------
-----------------------\n\n')   print('Accuracy
Score:')   print(accuracy_score(y_t,y_p))
plot_confusion_matrix(classi,X_t,y_t)
plt.title('Heat map for confusion matrix')
plt.show()
  y_p_proba = classifier.predict_proba(X_t)
skplt.metrics.plot_roc(y_t,y_p_proba)
plt.show()
```

IMPORTING THE IONOSPHERE DATASET

```
In [28]: df1 = pd.read_csv('ionosphere.data', header=None)
          X = df1.iloc[:,:-1] y = df1.iloc[:,-1]
```

TRAIN-TEST SPLIT OF 50:50

```
In [35]:

X_train,X_test,y_train,y_test = preprocess(X,y,0.5,scale=True,pca=True)
```

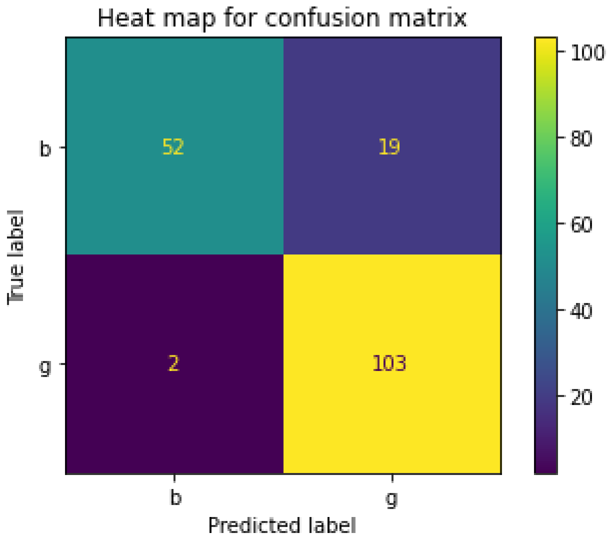
SVM CLASSIFIERS with Linear, Polynomial(degree2), Polynomial(degree3), Gaussian, Sigmoid

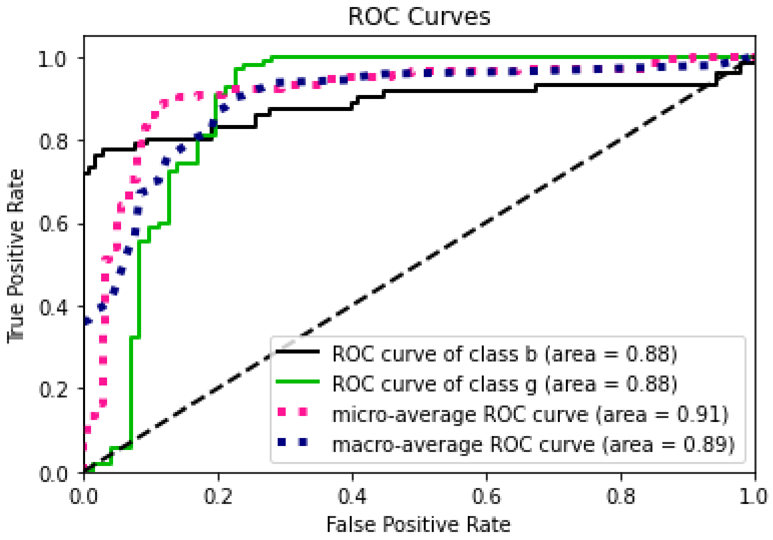
In [43]:

```
#SVM linear model
classifier = SVC(kernel='linear',
probability=True) classifier.fit(X_train,y_train)
y_pred = classifier.predict(X_test) print('SVC
Linear:')
tester(classifier,X_test,y_test,y_pred) print('----------
---------------------------------\n\n\n')
#SVM polynomial model degree 2
classifier = SVC(kernel='poly', degree=2,
probability=True) classifier.fit(X_train,y_train) y_pred
= classifier.predict(X_test) print('SVC Polynomial degree
2:') tester(classifier,X_test,y_test,y_pred) print('-----
--------------------------------------\n\n\n')
#SVM polynomial model degree 3
classifier = SVC(kernel='poly', degree=3,
probability=True) classifier.fit(X_train,y_train) y_pred
= classifier.predict(X_test) print('SVC Polynomial degree
3:') tester(classifier,X_test,y_test,y_pred) print('-----
--------------------------------------\n\n\n')
#SVM gaussian model
classifier = SVC(kernel='rbf',
probability=True)
classifier.fit(X_train,y_train) y_pred =
classifier.predict(X_test) print('SVC
Gaussian:')
tester(classifier,X_test,y_test,y_pred) print('----------
---------------------------------\n\n\n')
#SVM sigmoid model
classifier = SVC(kernel='sigmoid',
probability=True) classifier.fit(X_train,y_train)
y_pred = classifier.predict(X_test) print('SVC
Sigmoid:')
tester(classifier,X_test,y_test,y_pred) print('---------------------------------
----------\n\n\n')
```

```
SVC Linear:
Confusion Matrix
[[ 52  19]
 [  2 103]]
-----------------------------------
-----------------------------------



Preformance Evaluation:
              precision    recall  f1-score
support

           b       0.96      0.73      0.83
71           g       0.84      0.98      0.91
105

    accuracy                           0.88
176    macro avg       0.90      0.86      0.87
176 weighted avg       0.89      0.88      0.88
176
```

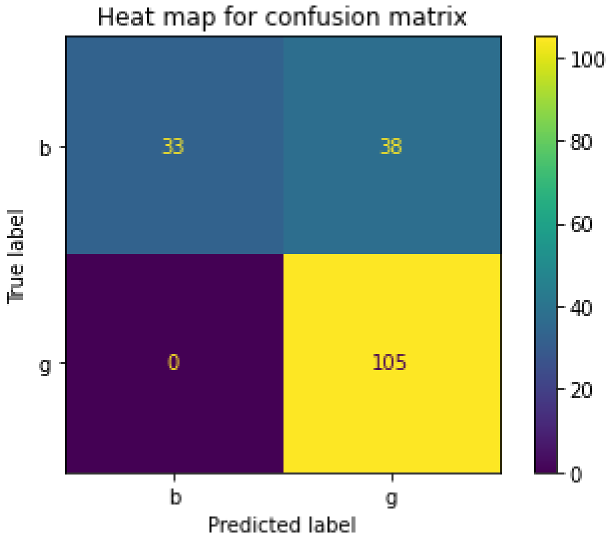
```
-----------------------------------
-----------------------------------


Accuracy Score:
0.8806818181818182
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarnin g: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: Confusio nMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.   warnings.warn(msg,
category=FutureWarning)
```

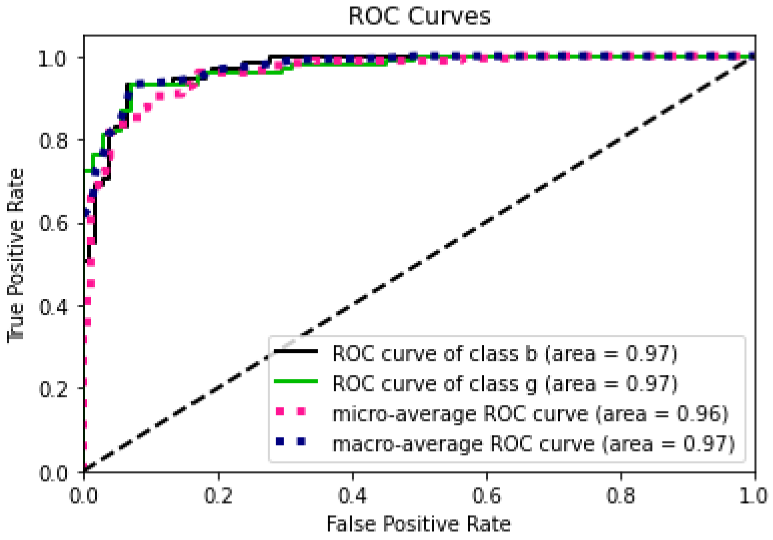




```
-------------------------------------------


SVC Polynomial degree 2:
Confusion Matrix
[[ 33  38]
 [  0 105]]
-----------------------------------
-----------------------------------
Preformance Evaluation:               precision
recall   f1-score   support
```

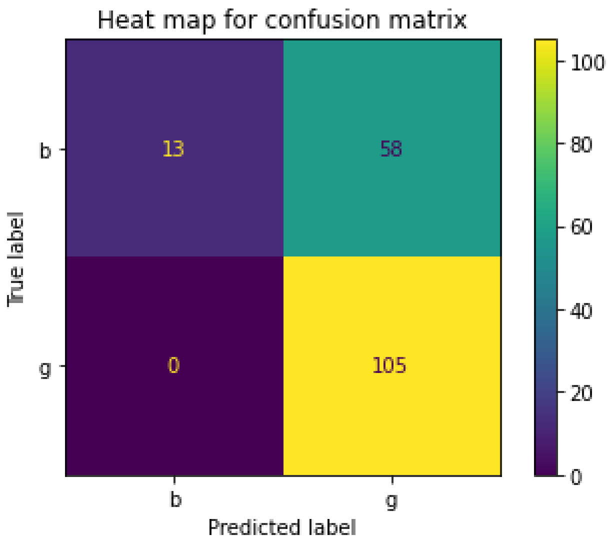
```
           b       1.00      0.46      0.63
71            g       0.73      1.00      0.85
105

    accuracy                          0.78
176    macro avg       0.87      0.73      0.74
176 weighted avg       0.84      0.78      0.76
176

-----------------------------------
-----------------------------------


Accuracy Score:
0.7840909090909091
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarnin g: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: Confusio nMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.   warnings.warn(msg,
category=FutureWarning)
```

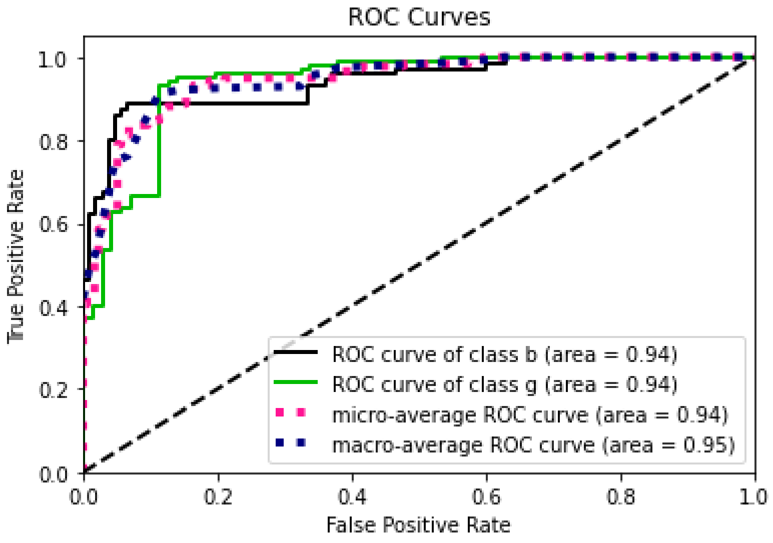




```
-------------------------------------------


SVC Polynomial degree 3:
```

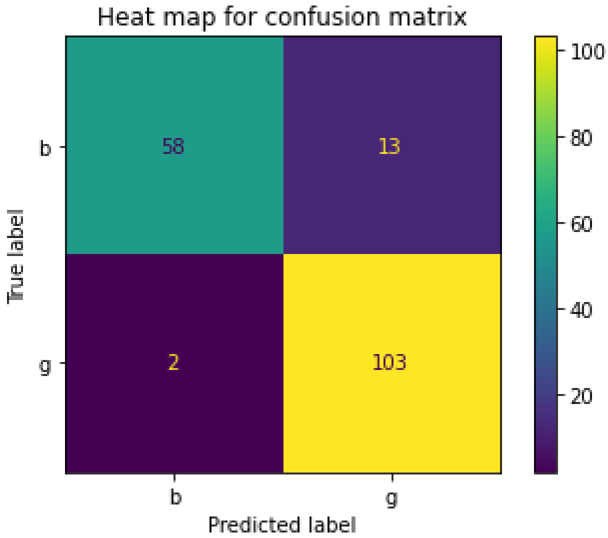
```
Confusion Matrix
[[ 13  58]
 [  0 105]]
-----------------------------------
-----------------------------------


Preformance Evaluation:               precision
recall  f1-score   support

           b       1.00      0.18      0.31
71            g       0.64      1.00      0.78
105

    accuracy                           0.67
176    macro avg       0.82      0.59      0.55
176 weighted avg       0.79      0.67      0.59
176

----------------------------------- ---------------
--------------------


Accuracy Score:
0.6704545454545454
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarnin g: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: Confusio nMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.   warnings.warn(msg,
category=FutureWarning)
```

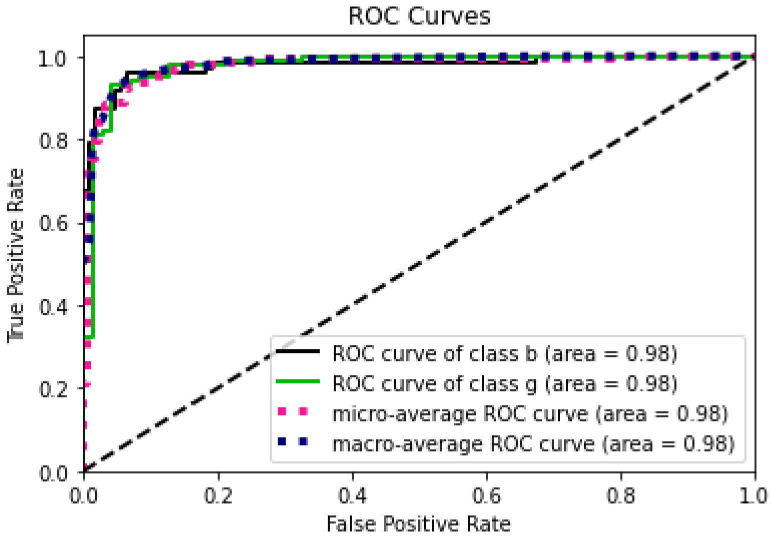




```
-------------------------------------------


SVC Gaussian:
Confusion Matrix
[[ 58  13]
 [  2 103]]
----------------------------------- ---------------
--------------------


Preformance Evaluation:                    precision
recall  f1-score   support

          b       0.97      0.82      0.89
71              g       0.89      0.98      0.93
105

    accuracy                           0.91
176     macro avg       0.93      0.90      0.91
176 weighted avg       0.92      0.91      0.91
176

----------------------------------- ---------------
--------------------
```

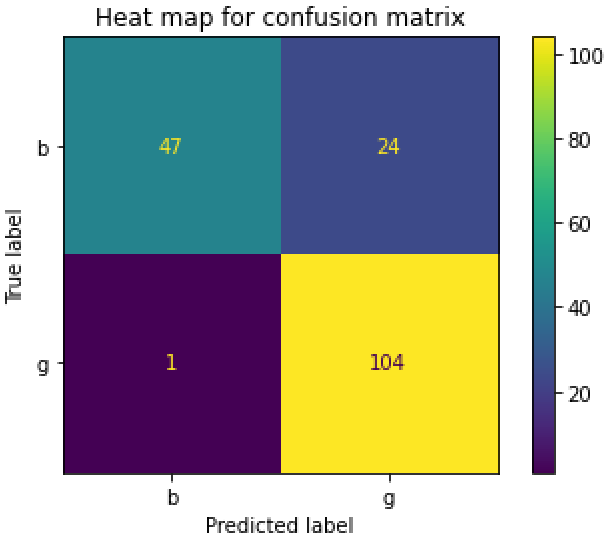
```
Accuracy Score:
0.9147727272727273
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarnin g: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: Confusio nMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.   warnings.warn(msg,
category=FutureWarning)
```

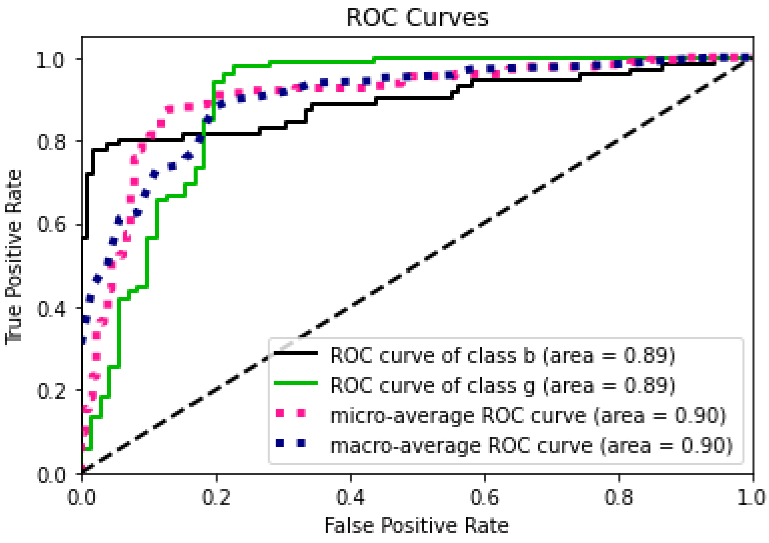




```
------------------------------------------


SVC Sigmoid:


Confusion Matrix
[[ 47  24]
 [  1 104]]
---------------------------------- --------------
--------------------


Preformance Evaluation:               precision
recall   f1-score   support
```

```
          b       0.98      0.66      0.79
71             g       0.81      0.99      0.89
105

    accuracy                          0.86
176    macro avg       0.90      0.83      0.84
176 weighted avg       0.88      0.86      0.85
176

----------------------------------- ---------------
--------------------


Accuracy Score:
0.8579545454545454
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
```





```
FutureWarnin g: Function

plot_confusion_matrix is

deprecated; Function

`plot_confusion_matrix` is

deprecated in 1.0 and will be removed in 1.2. Use one of the class methods:
```

```
Confusio nMatrixDisplay.from_predictions or
```

```
ConfusionMatrixDisplay.from_estimator.   warnings.warn(msg,
```

```
category=FutureWarning) ------------------------------------------
```

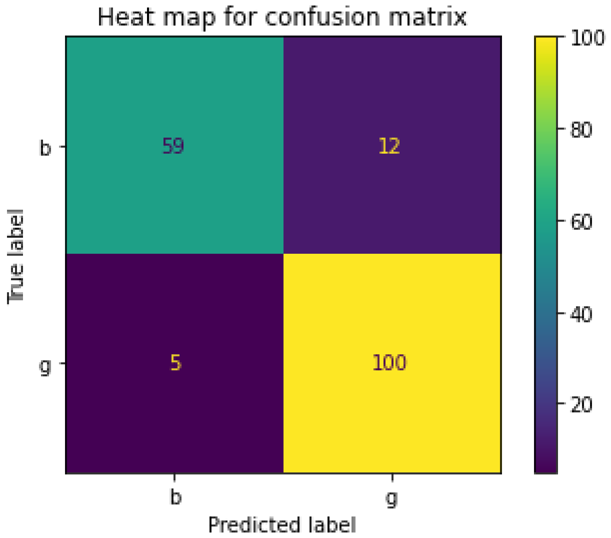
DECISION TREE CLASSIFIER

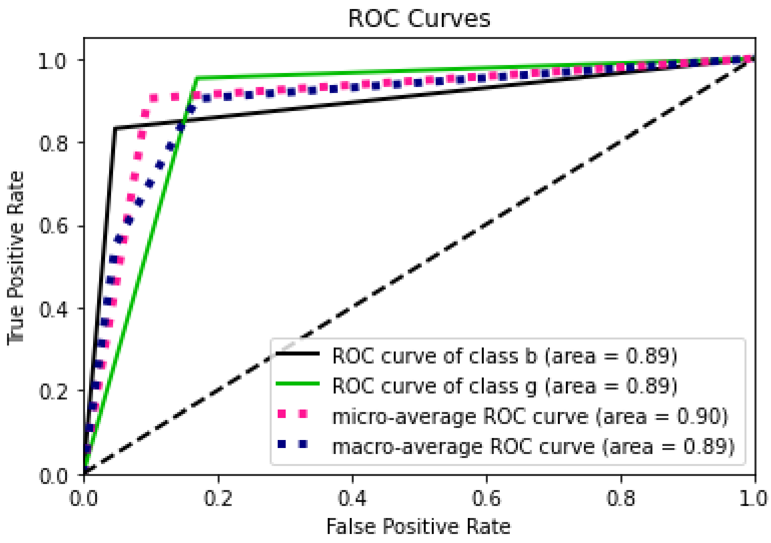
In [46]:

```
#Decision tree classifier
classifier=DecisionTreeClassifier()
classifier.fit(X_train,y_train) y_pred
= classifier.predict(X_test)
print('Decision Tree Classifier:')
tester(classifier,X_test,y_test,y_pred)
print('-------------------------------------------\n\n\n')
```

```
Decision Tree Classifier:
Confusion Matrix
[[ 59  12]
 [  5 100]]
----------------------------------- ---------------
--------------------


Preformance Evaluation:               precision
recall  f1-score   support

          b       0.92      0.83      0.87
71             g       0.89      0.95      0.92
105

    accuracy                           0.90
176     macro avg       0.91      0.89      0.90
176 weighted avg       0.90      0.90      0.90
176

-----------------------------------
-----------------------------------


Accuracy Score:
0.9034090909090909
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarnin g: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: Confusio nMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```





```
------------------------------------------
```

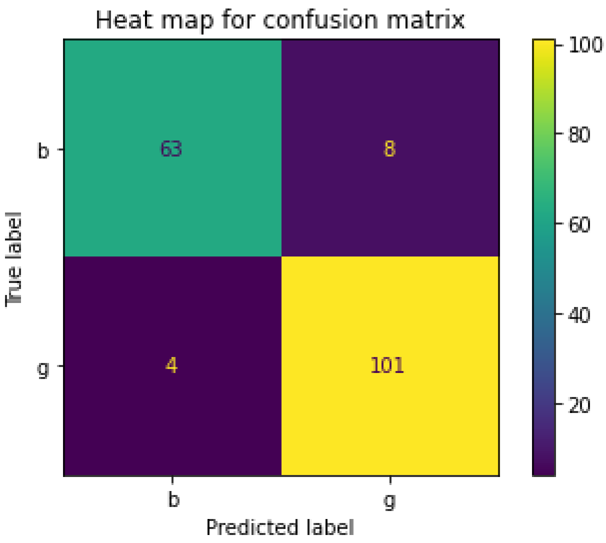
RANDOM FOREST MODEL

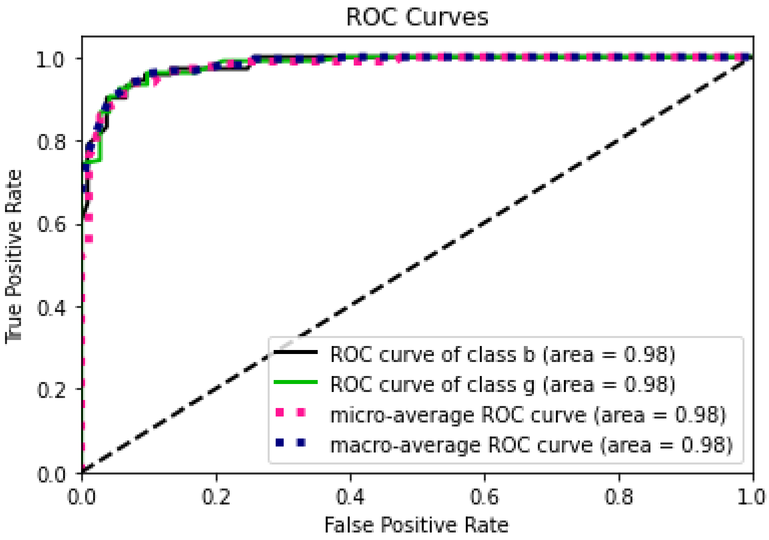
In [45]:

```
#Random forest model
classifier=RandomForestClassifier()
classifier.fit(X_train,y_train) y_pred
= classifier.predict(X_test)
print('Random Forest Classifier:')
tester(classifier,X_test,y_test,y_pred)
print('------------------------------------------\n\n\n')
```

```
Random Forest Classifier:
Confusion Matrix
[[ 63   8]
 [  4 101]]
```

```
---------------------------------- ---------------
--------------------


Preformance Evaluation:               precision
recall  f1-score   support

           b       0.94      0.89      0.91
71            g       0.93      0.96      0.94
105

    accuracy                           0.93
176    macro avg       0.93      0.92      0.93
176 weighted avg       0.93      0.93      0.93
176

---------------------------------- ---------------
--------------------


Accuracy Score:
0.9318181818181818
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarnin g: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: Confusio nMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```





```
-------------------------------------------
```

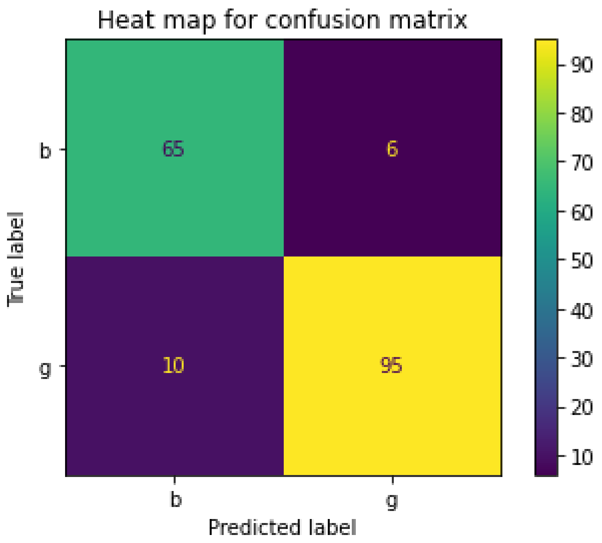
NAIVE BAYES CLASSIFIER

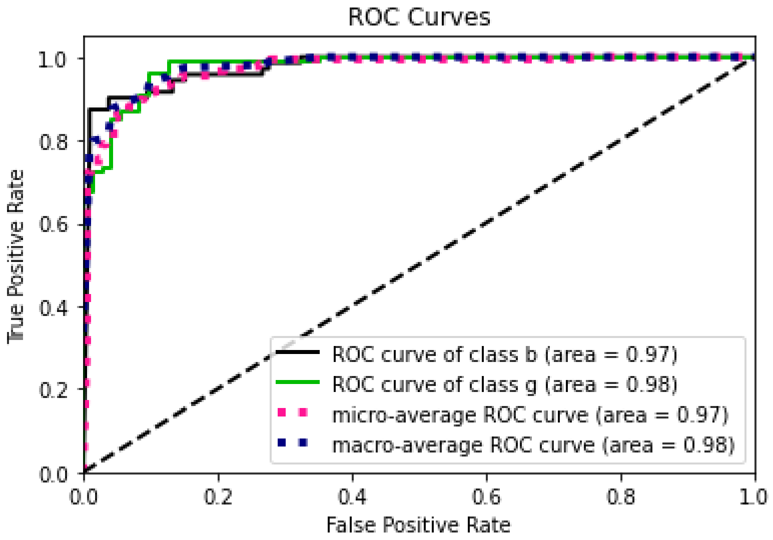
In [47]:

```
#Gaussian naive bayes
classifier=GaussianNB()
classifier.fit(X_train,y_train) y_pred
= classifier.predict(X_test)
print('Gaussian Naive Bayes:')
tester(classifier,X_test,y_test,y_pred)
print('-------------------------------------------\n\n\n')
#Bernoulli naive bayes
classifier=BernoulliNB()
classifier.fit(X_train,y_train) y_pred
= classifier.predict(X_test)
print('Bernoulli Naive Bayes:')
tester(classifier,X_test,y_test,y_pred)
print('-------------------------------------------\n\n\n')
```

```
Gaussian Naive Bayes:
Confusion Matrix
[[65  6]
```

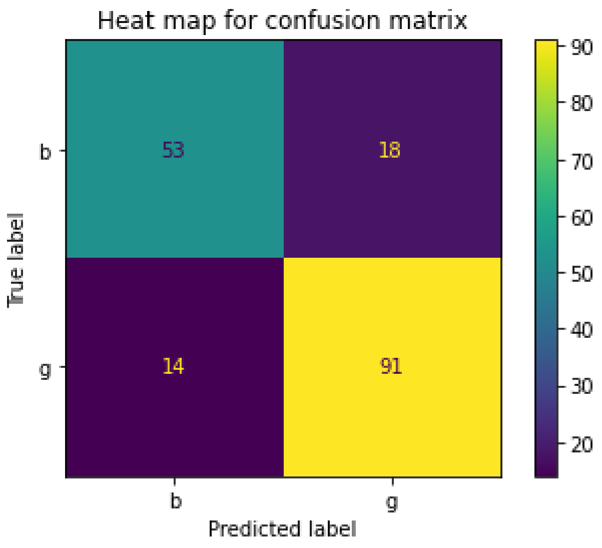
```
 [10 95]]
-----------------------------------
-----------------------------------
Preformance Evaluation:               precision
recall  f1-score   support

           b       0.87      0.92      0.89
71            g       0.94      0.90      0.92
105

    accuracy                           0.91
176    macro avg       0.90      0.91      0.91
176 weighted avg       0.91      0.91      0.91
176

----------------------------------- ---------------
--------------------


Accuracy Score:
0.9090909090909091
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
FutureWarnin g: Function plot_confusion_matrix is deprecated; Function
`plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: Confusio nMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.   warnings.warn(msg,
category=FutureWarning)
```

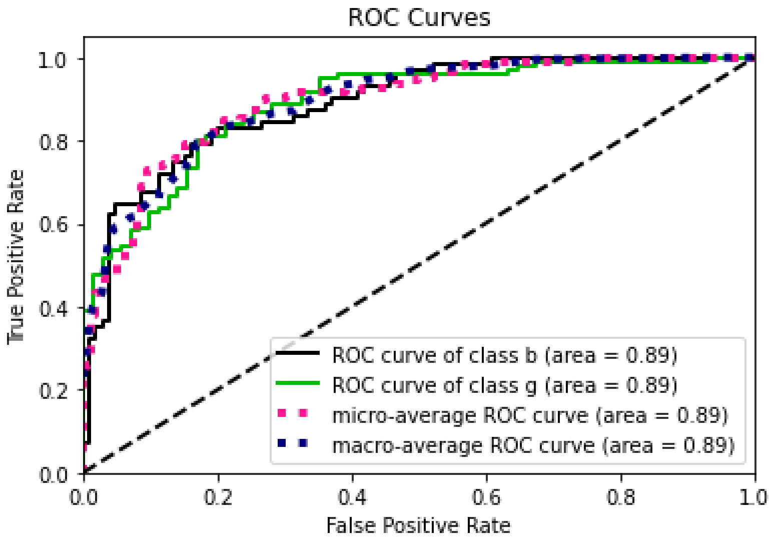




```
-------------------------------------------


Bernoulli Naive Bayes:
Confusion Matrix
[[53 18]
 [14 91]]
-----------------------------------
-----------------------------------


Preformance Evaluation:               precision
recall  f1-score   support

           b       0.79      0.75      0.77
71             g       0.83      0.87      0.85
105

    accuracy                           0.82
176     macro avg       0.81      0.81      0.81
176 weighted avg       0.82      0.82      0.82
176

----------------------------------- ---------------
--------------------
```

```
Accuracy Score:
0.8181818181818182
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
```





```
FutureWarnin g: Function
plot_confusion_matrix is
deprecated; Function
`plot_confusion_matrix` is
deprecated in 1.0 and will be removed in 1.2. Use one of the class methods:
Confusio nMatrixDisplay.from_predictions or
```

```
ConfusionMatrixDisplay.from_estimator.   warnings.warn(msg,


category=FutureWarning) -------------------------------------------
```

IONOSPHERE DATASET

	Accuracy	**Precision**	**Recall**	**F-Measure**
SVM CLASSIFIER				
Linear	0.880	0.90	0.86	0.87
Polynomial (2)	0.784	0.87	0.73	0.74
Polynomial (3)	0.670	0.82	0.59	0.55
Gaussian	**0.91**	0.93	0.90	0.91
Sigmoid	0.857	0.90	0.83	0.84
DECISION TREE CLASSIFIER				
Decision Tree	0.903	0.91	0.89	0.90
RANDOM FOREST CLASSIFIER				
Random Forest	**0.931**	0.93	0.92	0.93
NAÏVE BAYES CLASSIFIER				
Gaussian	0.909	0.90	0.91	0.91
Bernoulli	0.818	0.81	0.81	0.81

From the above tabulated data, we can conclude that at **50:50** train test split ratio in the

IONOSPHERE dataset, **Random Forest Classifier** produces the maximum accuracy followed by the **SVM Gaussian classifier**.

Question2:

Implement an ANN based model for classification task on the following datasets: **(CO2)** (10)

1. Iris plants dataset:
   https://archive.ics.uci.edu/ml/datasets/Iris/
2. Diabetes dataset:
   https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html
3. Wisconsin Breast Cancer Dataset:
4. https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

Iris Dataset

In [4]:

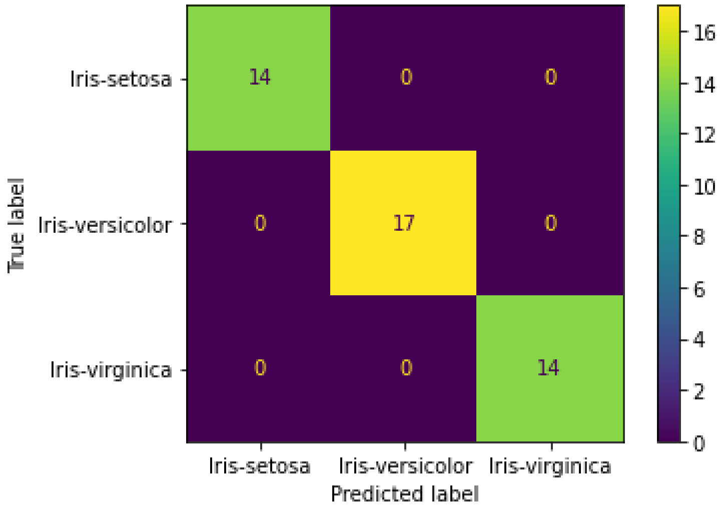
```
import pandas as pd
import numpy as np

# Dataset Preparation
df = pd.read_csv("iris.data",header=None)

col_name = ['Sepal Length','Sepal Width','Petal Length','Petal Width','Class']


df.columns = col_name


X = df.drop(['Class'], axis=1)
y = df['Class']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.7,test_size=0.30,rando


# Feature Scaling
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [5]:

```
# Classification using MLP
from sklearn.neural_network import MLPClassifier

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

y_pred = classifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("--------------------------------------------------")
print("--------------------------------------------------")


print("Performance Evaluation")
print(classification_report(y_test, y_pred))


print("--------------------------------------------------")
print("--------------------------------------------------")

print("Accuracy:")
print(accuracy_score(y_test, y_pred))

import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

```
Confusion Matrix:
[[14  0  0]
 [ 0 17  0]
 [ 0  0 14]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation                     precision
 recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       1.00      1.00      1.00        17 Iris-virginica
 1.00      1.00      1.00        14

       accuracy                           1.00        45
  macro avg       1.00      1.00      1.00        45
  weighted avg       1.00      1.00      1.00        45

--------------------------------------------------
--------------------------------------------------
Accuracy:
1.0
```

```
/usr/local/lib/python3.7/dist-
packages/sklearn/neural_network/_multilayer_ perceptron.py:696:
ConvergenceWarning: Stochastic Optimizer: Maximum itera tions (200)
reached and the optimization hasn't converged yet.  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: Fu
tureWarning: Function plot_confusion_matrix is deprecated; Function `plot_
confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one
of the class methods: ConfusionMatrixDisplay.from_predictions or Confusion
MatrixDisplay.from_estimator.  warnings.warn(msg, category=FutureWarning)
```



Diabetes Dataset

In [6]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import  train_test_split
from sklearn.datasets import load_diabetes

# preparing the dataset

dataset = load_diabetes()

X = np.delete(dataset.data,1,1)
y = dataset.data[:,1]

# as in the dataset Male or Female is not mentioned properly so we assume the first unique

data_sex_type = np.unique(y);
y = list(map(lambda x : 'M' if x == data_sex_type[0] else 'F' , y));

target_name = ['M','F']
feature_name = list(filter(lambda x : x != 'sex',dataset.feature_names));

X_train , X_test , y_train , y_test = train_test_split(X,y,test_size = 0.3)
```

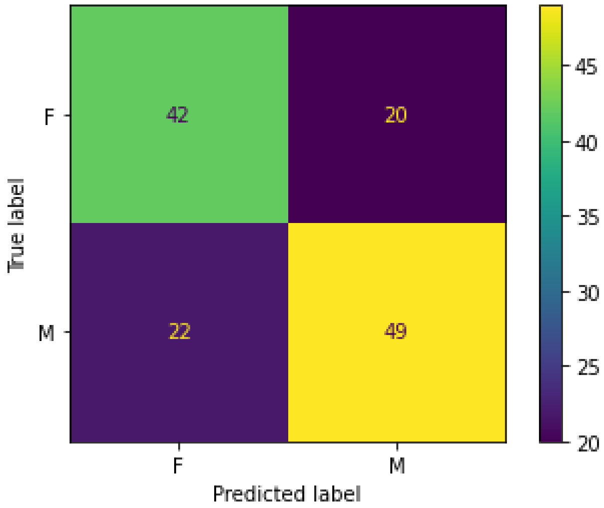
In [7]:

```
# Classification
from sklearn.neural_network import MLPClassifier
classifier = MLPClassifier(max_iter=100)

############################################################################
# Showing all the parameters

from pprint import pprint
# Look at parameters used by our current forest
print('Parameters currently in use:\n')
pprint(classifier.get_params())

############################################################################
# Creating a set of important sample features

parameter_space = {
    'hidden_layer_sizes': [(50,50,50), (50,100,50), (100,)],
    'activation': ['tanh', 'relu'],
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.05],
    'learning_rate': ['constant','adaptive'],
}
pprint(parameter_space)

############################################################################

from sklearn.model_selection import GridSearchCV

# Use the random grid to search for best hyperparameters
# First create the base model to tune
classifier = MLPClassifier(max_iter=100)
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores

rf_random = GridSearchCV(classifier, parameter_space, n_jobs=-1, cv=3)
rf_random.fit(X_train, y_train)

y_pred = rf_random.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("--------------------------------------------------")
print("--------------------------------------------------")


print("Performance Evaluation")
print(classification_report(y_test, y_pred))


print("--------------------------------------------------")
print("--------------------------------------------------")

print("Accuracy:")
print(accuracy_score(y_test, y_pred))
```

```
import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(rf_random, X_test, y_test)
plt.show()
```

```
Parameters currently in use:

{'activation': 'relu', 'alpha':
 0.0001,
 'batch_size': 'auto',
 'beta_1': 0.9,
 'beta_2': 0.999,
 'early_stopping': False,
 'epsilon': 1e-08,
 'hidden_layer_sizes': (100,),
 'learning_rate': 'constant',
 'learning_rate_init': 0.001,
 'max_fun': 15000,
 'max_iter': 100,
 'momentum': 0.9,
 'n_iter_no_change': 10,
 'nesterovs_momentum': True,
 'power_t': 0.5,
 'random_state': None,
 'shuffle': True,
 'solver': 'adam',
 'tol': 0.0001,
 'validation_fraction': 0.1,
 'verbose': False,
 'warm_start': False}
{'activation': ['tanh', 'relu'],
 'alpha': [0.0001, 0.05],
 'hidden_layer_sizes': [(50, 50, 50), (50, 100, 50), (100,)],
 'learning_rate': ['constant', 'adaptive'],
 'solver': ['sgd', 'adam']}
Confusion Matrix:
[[42 20]
 [22 49]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation              precision
 recall  f1-score   support

           F       0.66      0.68      0.67        62
 M       0.71      0.69      0.70        71

    accuracy                           0.68       133
macro avg       0.68      0.68      0.68       133 weighted
avg       0.69      0.68      0.68       133

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.6842105263157895
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87:
Fu tureWarning: Function plot_confusion_matrix is deprecated; Function
`plot_ confusion_matrix` is deprecated in 1.0 and will be removed in 1.2.
Use one of the class methods: ConfusionMatrixDisplay.from_predictions or
Confusion MatrixDisplay.from_estimator.  warnings.warn(msg,
category=FutureWarning)
```



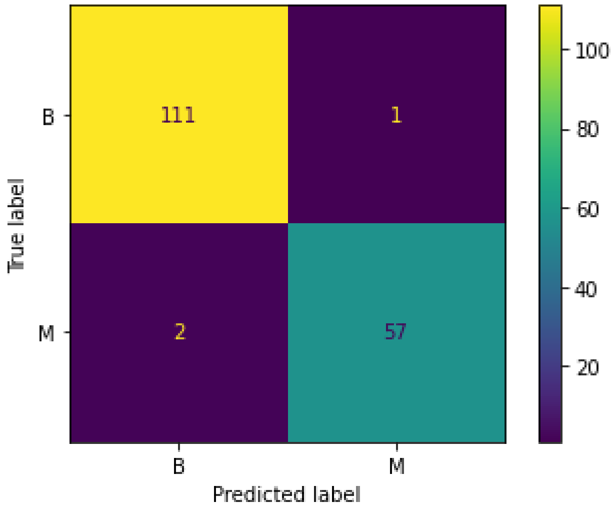
Wisconsin Breast Cancer Dataset

In [9]:

```
import pandas as pd
import numpy as np

# Dataset Preparation
df = pd.read_csv("wdbc.data",header=None)

col_name = ['1','Class','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17'
           ,'20','21','22','23','24','25','26','27','28','29','30','31','32']


df.columns = col_name


X = df.drop(['1','Class'], axis=1)
y = df['Class']

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.7,test_size=0.30,rando


# Feature Scaling
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Classification using MLP
from sklearn.neural_network import MLPClassifier

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

y_pred = classifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("--------------------------------------------------")
print("--------------------------------------------------")


print("Performance Evaluation")
print(classification_report(y_test, y_pred))


print("--------------------------------------------------")
print("--------------------------------------------------")

print("Accuracy:")
print(accuracy_score(y_test, y_pred))

import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(classifier, X_test, y_test)
plt.show()
```

```
Confusion Matrix:
[[111   1]
 [  2  57]]
-------------------------------------------------- -------------------------
-------------------------
Performance Evaluation                precision
 recall  f1-score   support

           B       0.98      0.99      0.99       112
 M       0.98      0.97      0.97        59

    accuracy                           0.98       171
macro avg       0.98      0.98      0.98       171 weighted
avg       0.98      0.98      0.98       171

-------------------------------------------------- -------------------------
-------------------------
Accuracy:
0.9824561403508771
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_multilayer_pe
rceptron.py:696: ConvergenceWarning: Stochastic Optimizer: Maximum iteration
s (200) reached and the optimization hasn't converged yet.
ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: Futu
reWarning: Function plot_confusion_matrix is deprecated; Function `plot_conf
usion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of th
e class methods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixD
isplay.from_estimator.  warnings.warn(msg, category=FutureWarning)
```



Question5:

Apply any two of the comparisons for the clustering task: **(CO4) (10)**

- K-means versus, K-medoids/PAM,
- Dendrogram versus AGNES versus BIRCH
- DBSCAN versus OPTICS

on **the Wine Dataset**:
https://archive.ics.uci.edu/ml/datasets/wine
and use the following performance measures

1. Silhouette Coefficient
2. Calinski-Harabasz Index
3. Davies-Bouldin Index

K-Means

In [1]:

```
#importing libraries
import numpy as np
import pandas as pd
import sklearn as sk
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.cluster import KMeans
from sklearn.datasets import load_wine
```

In [2]:

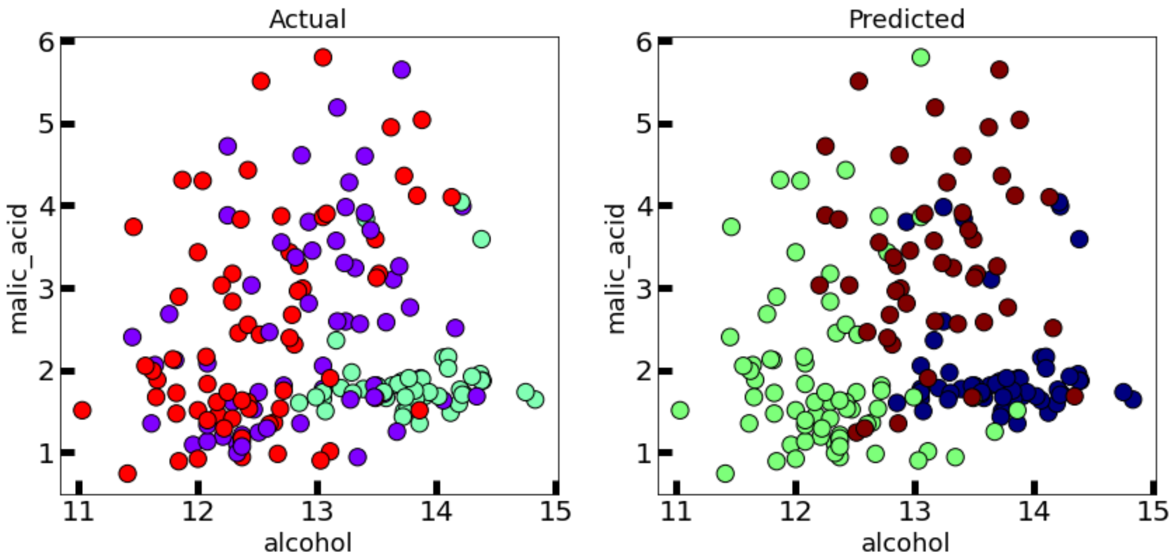
```
wine=load_wine()
x = wine.data
df=pd.DataFrame(data=x, columns=wine.feature_names)
kmeans = KMeans(init="random", n_clusters=3, n_init=10, max_iter=300, random_state=42)
y = kmeans.fit_predict(x)
```

In [3]:

```
fig, axes = plt.subplots(1, 2, figsize=(14,6))
axes[0].scatter(x=df['alcohol'], y=df['malic_acid'], c=y, cmap='rainbow',edgecolor='k', s=1
axes[1].scatter(x=df['alcohol'], y=df['malic_acid'], c=wine.target,
cmap='jet',edgecolor='k axes[0].set_xlabel('alcohol', fontsize=18)
axes[0].set_ylabel('malic_acid', fontsize=18) axes[1].set_xlabel('alcohol', fontsize=18)
axes[1].set_ylabel('malic_acid', fontsize=18)
axes[0].tick_params(direction='in', length=10, width=5, colors='k', labelsize=20)
axes[1].tick_params(direction='in', length=10, width=5, colors='k',
labelsize=20) axes[0].set_title('Actual', fontsize=18)
axes[1].set_title('Predicted', fontsize=18)
```

Out[3]:

```
Text(0.5, 1.0, 'Predicted')
```



[4]:

In

```
from sklearn.metrics import silhouette_score
print("The silhouette score is :")
silhouette_score(x, kmeans.labels_)
```

```
The silhouette score is :
```

Out[4]:

```
0.5711381937868844
```

In [5]:

```
from sklearn.metrics import calinski_harabasz_score
print("The calinski harabasz score is :")
calinski_harabasz_score(x, kmeans.labels_)
```

```
The calinski harabasz score is :
```

Out[5]:

```
561.815657860671
```

K-medoids

In [7]:

```
#importing libraries
import numpy as np
import pandas as pd
import sklearn as sk
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn_extra.cluster import KMedoids
from sklearn.datasets import load_wine
```

In [8]:

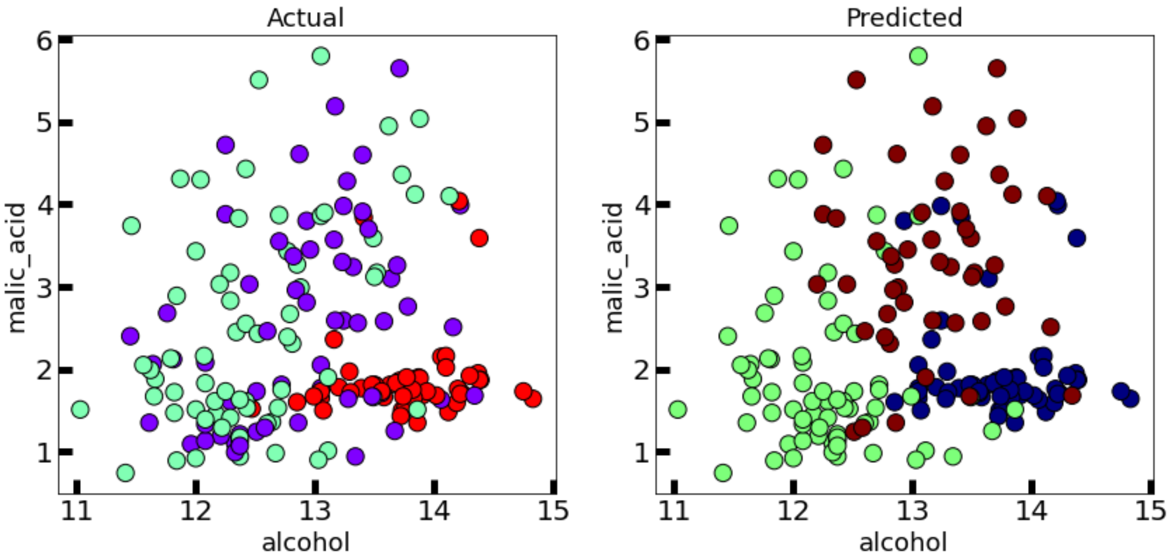
```
wine=load_wine()
x = wine.data
df=pd.DataFrame(data=x, columns=wine.feature_names)
kmedoid = KMedoids(init="heuristic", n_clusters=3, max_iter=300, random_state=42)
y = kmedoid.fit_predict(x)
```

[9]:

```
fig, axes = plt.subplots(1, 2, figsize=(14,6))
axes[0].scatter(x=df['alcohol'], y=df['malic_acid'], c=y, cmap='rainbow',edgecolor='k', s=1
axes[1].scatter(x=df['alcohol'], y=df['malic_acid'], c=wine.target,
cmap='jet',edgecolor='k axes[0].set_xlabel('alcohol', fontsize=18)
axes[0].set_ylabel('malic_acid', fontsize=18) axes[1].set_xlabel('alcohol', fontsize=18)
axes[1].set_ylabel('malic_acid', fontsize=18)
axes[0].tick_params(direction='in', length=10, width=5, colors='k', labelsize=20)
axes[1].tick_params(direction='in', length=10, width=5, colors='k',
labelsize=20) axes[0].set_title('Actual', fontsize=18)
axes[1].set_title('Predicted', fontsize=18)
```

In

Out[9]:
```
Text(0.5, 1.0,

'Predicted')
```



In [10]:

```
from sklearn.metrics import silhouette_score
print("The silhouette score is :")
silhouette_score(x, kmedoid.labels_)
```

```
The silhouette score is :
```

Out[10]:

```
0.5666480408636575
```

[11]:

```
from sklearn.metrics import calinski_harabasz_score
print("The calinski harabasz score is :")
calinski_harabasz_score(x, kmedoid.labels_)
```

```
The calinski harabasz score is :
```

Out[11]:

```
539.3792353535451
```

In

Dendrogram

In [12]:

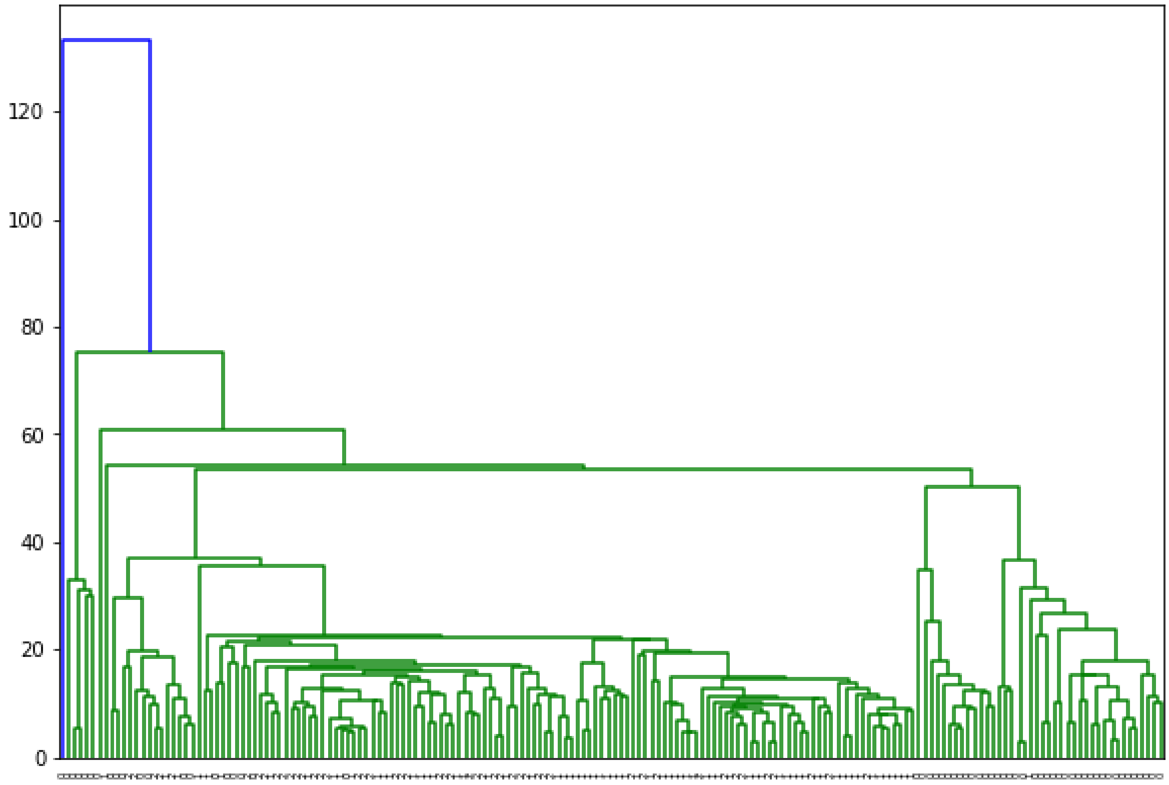
```
#importing libraries
import numpy as np
import pandas as pd
import sklearn as sk
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.datasets import load_wine
```

In [13]:

```
wine=load_wine()
x = wine.data
df=pd.DataFrame(data=x, columns=wine.feature_names)
```

[14]:

```
from scipy.cluster.hierarchy import dendrogram, linkage

linked = linkage(x, 'single')
plt.figure(figsize=(10,7))

dendrogram(linked,
           orientation='top',
           labels=wine.target,
           distance_sort='descending',
           show_leaf_counts=True)

plt.show()
```



In

Since dendrogram illustrates how each cluster is composed by drawing a U-shaped link between a nonsingleton cluster and its children, evaltion mterics cannot be applied on this

Agnes

In [15]:

```
#importing libraries
import numpy as np
import pandas as pd
import sklearn as sk
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.datasets import load_wine
```

[16]:

```
wine=load_wine()
x = wine.data
df=pd.DataFrame(data=x, columns=wine.feature_names)
```

In [17]:

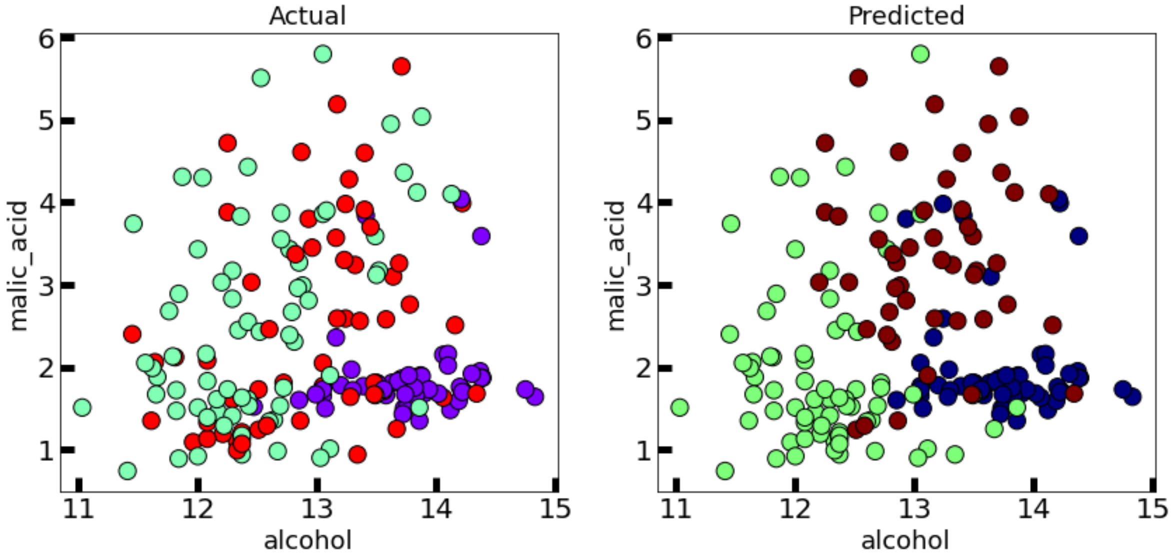
```
from sklearn.cluster import AgglomerativeClustering

cluster = AgglomerativeClustering(n_clusters=3, affinity='euclidean', linkage='ward')
y = cluster.fit_predict(x)
```

In [18]:

```
fig, axes = plt.subplots(1, 2, figsize=(14,6))
axes[0].scatter(x=df['alcohol'], y=df['malic_acid'], c=y, cmap='rainbow',edgecolor='k', s=1
axes[1].scatter(x=df['alcohol'], y=df['malic_acid'], c=wine.target,
cmap='jet',edgecolor='k axes[0].set_xlabel('alcohol', fontsize=18)
axes[0].set_ylabel('malic_acid', fontsize=18) axes[1].set_xlabel('alcohol', fontsize=18)
axes[1].set_ylabel('malic_acid', fontsize=18)
axes[0].tick_params(direction='in', length=10, width=5, colors='k', labelsize=20)
axes[1].tick_params(direction='in', length=10, width=5, colors='k',
labelsize=20) axes[0].set_title('Actual', fontsize=18)
axes[1].set_title('Predicted', fontsize=18)
```

```
Out[18]: Text(0.5, 1.0,

'Predicted')
```

In



In [19]:

```
from sklearn.metrics import silhouette_score
print("The silhouette score is :")
silhouette_score(x, cluster.labels_)
```

```
The silhouette score is :
```

Out[19]:

```
0.5644796401732074
```

   [20]:

```
from sklearn.metrics import calinski_harabasz_score
print("The calinski harabasz score is :")
calinski_harabasz_score(x, cluster.labels_)
```

```
The calinski harabasz score is :
```

Out[20]:

```
552.851711505718
```

Birch

In [21]:

```
#importing libraries
import numpy as np
import pandas as pd
import sklearn as sk
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.datasets import load_wine
```

In [22]:

```
wine=load_wine()
x = wine.data
df=pd.DataFrame(data=x, columns=wine.feature_names)
```

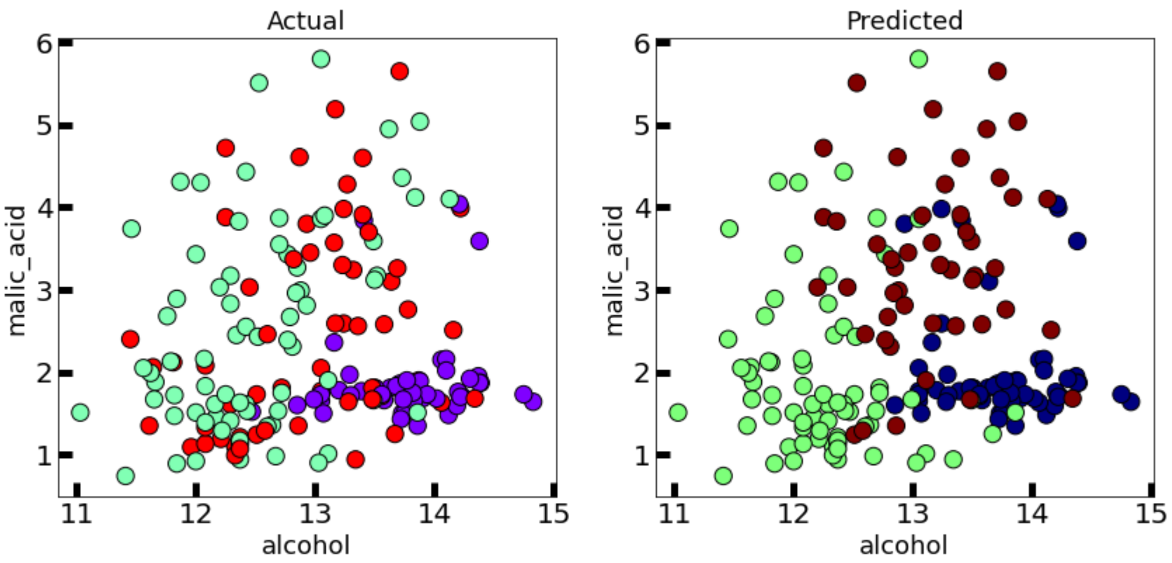
In
In [23]:

```
from sklearn.cluster import Birch

birch = Birch(n_clusters=3, compute_labels=True, branching_factor=50)
y = birch.fit_predict(x)
```

[24]:

```
fig, axes = plt.subplots(1, 2, figsize=(14,6))
axes[0].scatter(x=df['alcohol'], y=df['malic_acid'], c=y, cmap='rainbow',edgecolor='k', s=1
axes[1].scatter(x=df['alcohol'], y=df['malic_acid'], c=wine.target,
cmap='jet',edgecolor='k axes[0].set_xlabel('alcohol', fontsize=18)
axes[0].set_ylabel('malic_acid', fontsize=18) axes[1].set_xlabel('alcohol', fontsize=18)
axes[1].set_ylabel('malic_acid', fontsize=18)
axes[0].tick_params(direction='in', length=10, width=5, colors='k', labelsize=20)
axes[1].tick_params(direction='in', length=10, width=5, colors='k',
labelsize=20) axes[0].set_title('Actual', fontsize=18)
axes[1].set_title('Predicted', fontsize=18)
```

Out[24]:
```
Text(0.5, 1.0,

'Predicted')
```



In [25]:

```
from sklearn.metrics import silhouette_score
print("The silhouette score is :")
silhouette_score(x, birch.labels_)
```

```
The silhouette score is :
```

Out[25]:

```
0.5644796401732074
```

In [26]:

```
from sklearn.metrics import calinski_harabasz_score
print("The calinski harabasz score is :")
calinski_harabasz_score(x, birch.labels_)
```

```
The calinski harabasz score is :
```

In
Out[26]:

```
552.851711505718
```

DBSCAN

[27]:

```
#importing libraries
import numpy as np
import pandas as pd
import sklearn as sk
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.datasets import load_wine

wine=load_wine()
x = wine.data
df=pd.DataFrame(data=x, columns=wine.feature_names)
```

In [28]:

```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=35, algorithm='auto', metric='euclidean')
y = dbscan.fit_predict(x)
```

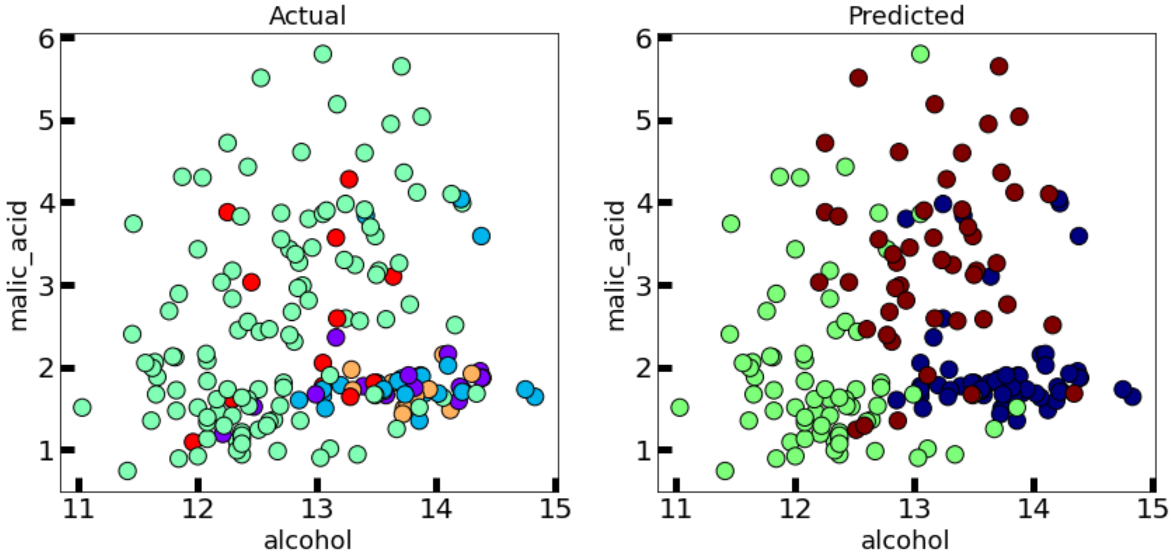
In [29]:

```
fig, axes = plt.subplots(1, 2, figsize=(14,6))
axes[0].scatter(x=df['alcohol'], y=df['malic_acid'], c=y, cmap='rainbow',edgecolor='k', s=1
axes[1].scatter(x=df['alcohol'], y=df['malic_acid'], c=wine.target,
cmap='jet',edgecolor='k axes[0].set_xlabel('alcohol', fontsize=18)
axes[0].set_ylabel('malic_acid', fontsize=18) axes[1].set_xlabel('alcohol', fontsize=18)
axes[1].set_ylabel('malic_acid', fontsize=18)
axes[0].tick_params(direction='in', length=10, width=5, colors='k', labelsize=20)
axes[1].tick_params(direction='in', length=10, width=5, colors='k',
labelsize=20) axes[0].set_title('Actual', fontsize=18)
axes[1].set_title('Predicted', fontsize=18)
```

Out[29]:

```
Text(0.5, 1.0, 'Predicted')
```

In



[30]:

```
from sklearn.metrics import silhouette_score
print("The silhouette score is :")
silhouette_score(x, dbscan.labels_)
```

```
The silhouette score is :
```

Out[30]:

```
0.4413295944891938
```

In [31]:

```
from sklearn.metrics import calinski_harabasz_score
print("The calinski harabasz score is :")
calinski_harabasz_score(x, dbscan.labels_)
```

```
The calinski harabasz score is :
```

Out[31]:

```
208.9449395725058
```

OPTICS

In [32]:

```
#importing libraries
import numpy as np
import pandas as pd
import sklearn as sk
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.datasets import load_wine

wine=load_wine()
x = wine.data
df=pd.DataFrame(data=x, columns=wine.feature_names)
```

In [33]:

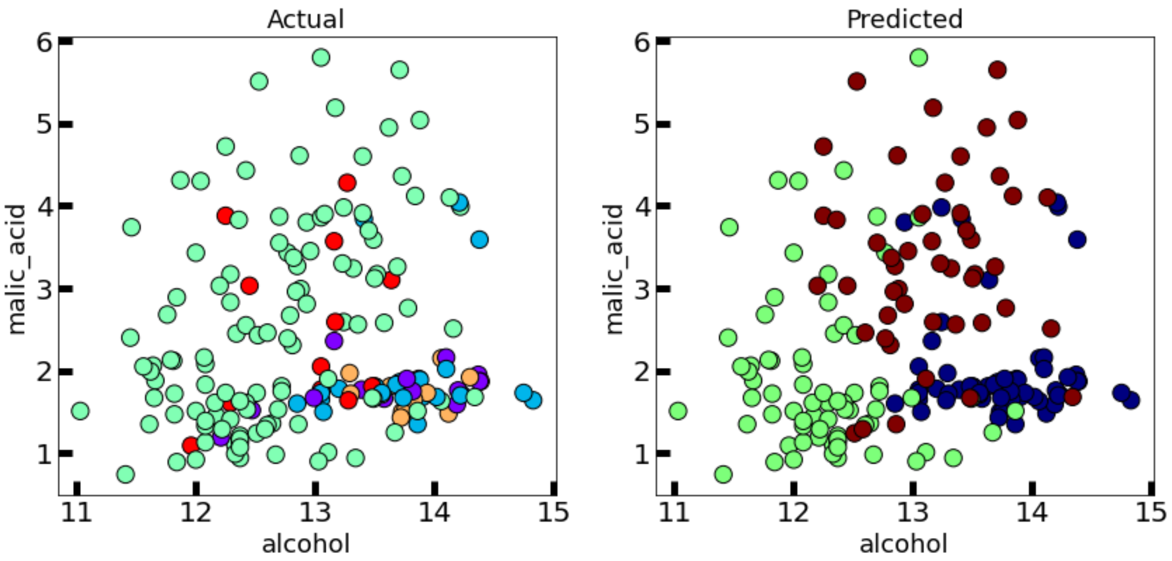
In

```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=35, algorithm='auto', metric='euclidean')
y = dbscan.fit_predict(x)
```

[34]:

```
fig, axes = plt.subplots(1, 2, figsize=(14,6))
axes[0].scatter(x=df['alcohol'], y=df['malic_acid'], c=y, cmap='rainbow',edgecolor='k', s=1
axes[1].scatter(x=df['alcohol'], y=df['malic_acid'], c=wine.target,
cmap='jet',edgecolor='k axes[0].set_xlabel('alcohol', fontsize=18)
axes[0].set_ylabel('malic_acid', fontsize=18) axes[1].set_xlabel('alcohol', fontsize=18)
axes[1].set_ylabel('malic_acid', fontsize=18)
axes[0].tick_params(direction='in', length=10, width=5, colors='k', labelsize=20)
axes[1].tick_params(direction='in', length=10, width=5, colors='k',
labelsize=20) axes[0].set_title('Actual', fontsize=18)
axes[1].set_title('Predicted', fontsize=18)
```

```
Out[34]: Text(0.5, 1.0,

'Predicted')
```



In [35]:

```
from sklearn.metrics import silhouette_score
print("The silhouette score is :")
silhouette_score(x, dbscan.labels_)
```

```
The silhouette score is :
```

In

Out[35]:

```
0.4413295944891938
```

[36]:

```
from sklearn.metrics import calinski_harabasz_score
print("The calinski harabasz score is :")
calinski_harabasz_score(x, dbscan.labels_)
```

```
The calinski harabasz score is :
```

Out[36]:

```
208.9449395725058
```