Name : Sonu Kumar Mahanty

Roll No: 001811001038

Subject: Machine Learning Lab

Year: 4 Semester: 1

Department : Information Technology

Final Lab Evaluation

Google Drive Link :
https://drive.google.com/drive/folders/1nJCnggam6h8Ce1frFN8VgJWcyoCvEoGU?usp=sharing
Comparision Table Link :
https://docs.google.com/spreadsheets/d/1wvaUnJpm7WbMIivlLQ17O16yiSmEVPU-/edit?usp=sharing&ouid=103758038152179595317&rtpof=true&sd=true
Github Link-https://github.com/Mahantyjusl/ML_Assignments

Ans 1

Decision Tree

Wine Dataset

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

# preparing the dataset

from sklearn.datasets import load_wine

dataset = load_wine()

X = dataset.data
y = dataset.target

from sklearn.model_selection import  train_test_split

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

# Classification

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(max_depth=3)
classifier.fit(X_train , y_train)
```

```
y_pred = classifier.predict(X_test)

# Evaluating The Performances

from sklearn.metrics import classification_report , confusion_matrix ,
accuracy_score

print('Confusion Matrix \n')
print(confusion_matrix(y_test,y_pred),'\n')
print("==========================================================
==============")
print("==========================================================
==============")
print("\nEvaluation Metrics \n" )
print(classification_report(y_test,y_pred))
print("==========================================================
==============")
print("==========================================================
==============")
print("Accuarcy", 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

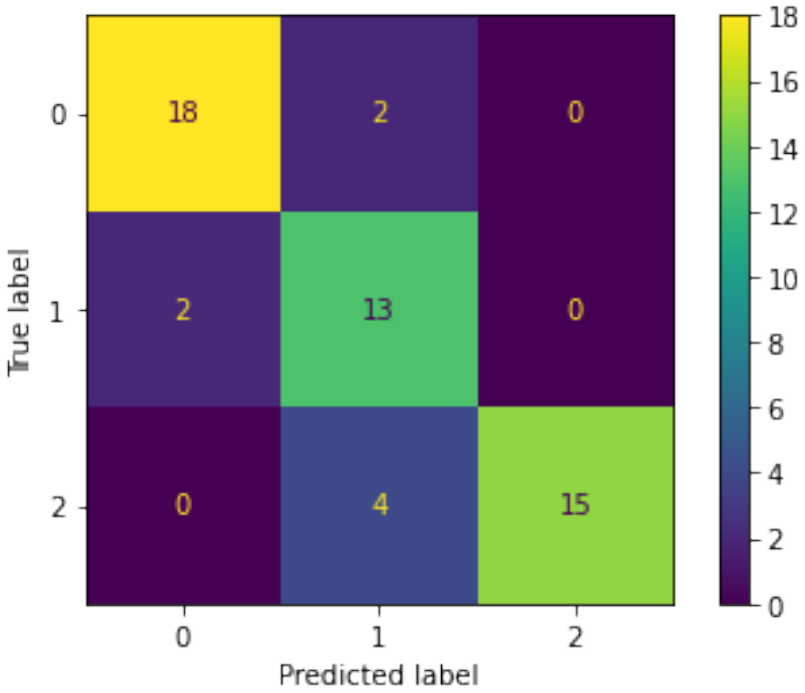
```
[[18  2  0]
 [ 2 13  0]
 [ 0  4 15]]
```

```
==========================================================================
========
==========================================================================
========
```

Evaluation Metrics

	precision	recall	f1-score	support
0	0.90	0.90	0.90	20
1	0.68	0.87	0.76	15
2	1.00	0.79	0.88	19
accuracy			0.85	54
macro avg	0.86	0.85	0.85	54
weighted avg	0.88	0.85	0.86	54

```
================================================================================
========
================================================================================
========
Accuarcy 0.8518518518518519

/usr/local/lib/python3.7/dist-packages/sklearn/utils/
deprecation.py:87: FutureWarning: 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
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



Ionosphere Dataset

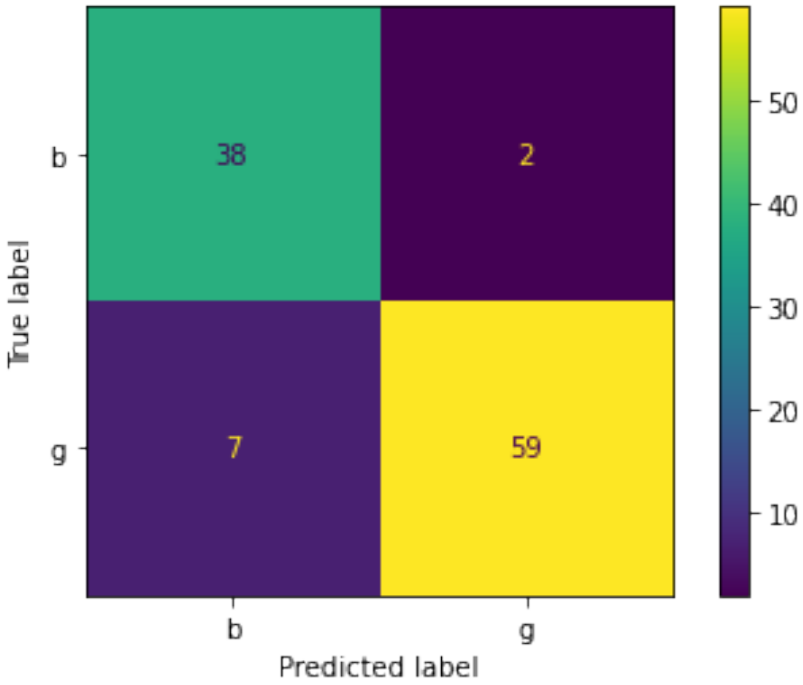
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# preparing the dataset
df = pd.read_csv("ionosphere.data",header=None)

col_name =
['1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16
','17','18','19'
           ,'20','21','22','23','24','25','26','27','28','29','30','31
','32','33','34','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,random_state=10)

# Feature Scaling
from sklearn.preprocessing import StandardScaler

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

# Classification

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(max_depth=3)
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:
[[38  2]
 [ 7 59]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.84      0.95      0.89        40
           g       0.97      0.89      0.93        66

    accuracy                           0.92       106
   macro avg       0.91      0.92      0.91       106
weighted avg       0.92      0.92      0.92       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9150943396226415
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/
deprecation.py:87: FutureWarning: 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
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



Naive Bayes

Wine Dataset

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

# preparing the dataset

from sklearn.datasets import load_wine

dataset = load_wine()

X = dataset.data
y = dataset.target

from sklearn.model_selection import  train_test_split

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

# Classification

from sklearn.naive_bayes import GaussianNB

classifier =GaussianNB(priors=None,var_smoothing=1e-10)
classifier.fit(X_train , y_train)
```

```
y_pred = classifier.predict(X_test)

# Evaluating The Performances

from sklearn.metrics import classification_report , confusion_matrix ,
accuracy_score

print('Confusion Matrix \n')
print(confusion_matrix(y_test,y_pred),'\n')
print("==========================================================
==============")
print("==========================================================
==============")
print("\nEvaluation Metrics \n" )
print(classification_report(y_test,y_pred))
print("==========================================================
==============")
print("==========================================================
==============")
print("Accuarcy", 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

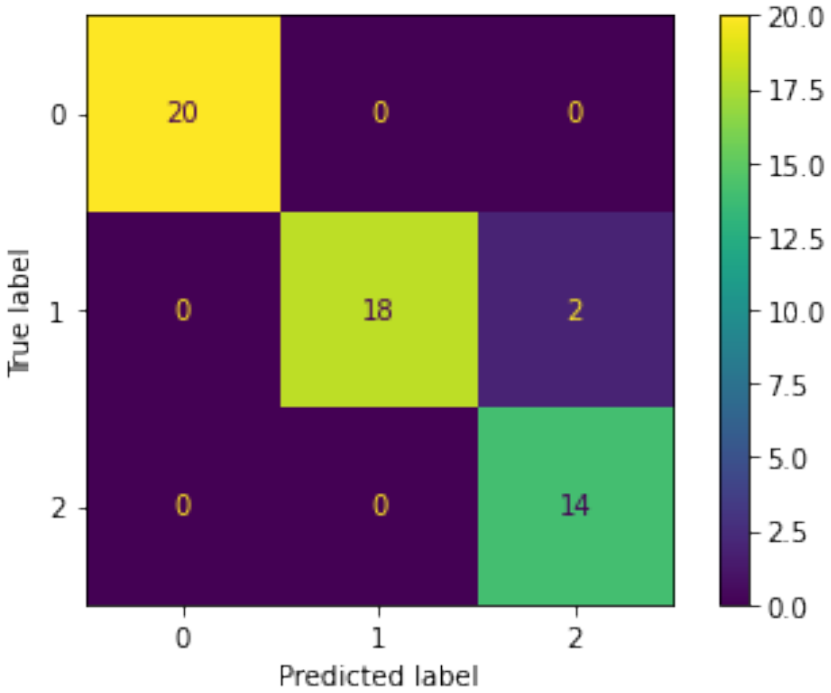
```
[[20  0  0]
 [ 0 18  2]
 [ 0  0 14]]
```

```
==========================================================================
========
==========================================================================
========
```

Evaluation Metrics

	precision	recall	f1-score	support
0	1.00	1.00	1.00	20
1	1.00	0.90	0.95	20
2	0.88	1.00	0.93	14
accuracy			0.96	54
macro avg	0.96	0.97	0.96	54
weighted avg	0.97	0.96	0.96	54

```
================================================================================
========
================================================================================
========
Accuarcy 0.9629629629629629

/usr/local/lib/python3.7/dist-packages/sklearn/utils/
deprecation.py:87: FutureWarning: 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
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```

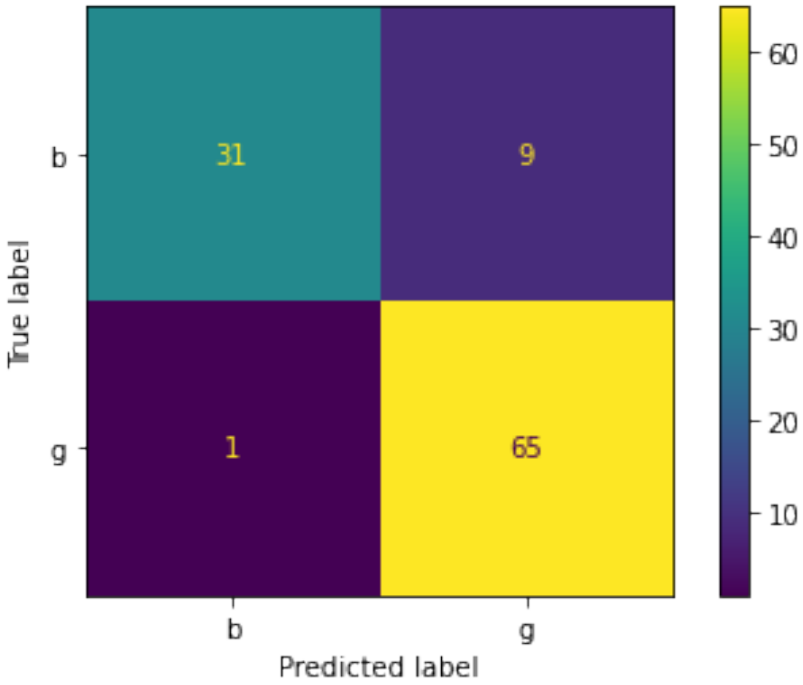


Ionosphere Dataset

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

# preparing the dataset
df = pd.read_csv("ionosphere.data",header=None)

col_name =
['1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16
','17','18','19'
           ,'20','21','22','23','24','25','26','27','28','29','30','31
','32','33','34','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,random_state=10)

# Classification

from sklearn.naive_bayes import GaussianNB

classifier =GaussianNB(priors=None,var_smoothing=1e-10)
classifier.fit(X_train , y_train)

y_pred = classifier.predict(X_test)

# Evaluating The Performances

from sklearn.metrics import classification_report , confusion_matrix ,
accuracy_score

print('Confusion Matrix \n')
print(confusion_matrix(y_test,y_pred),'\n')
print("==========================================================
==========")
print("==========================================================
==========")
print("\nEvaluation Metrics \n" )
print(classification_report(y_test,y_pred))
print("==========================================================
==========")
print("==========================================================
==========")
print("Accuarcy", 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

[[31  9]
 [ 1 65]]
```

```
================================================================================
========
================================================================================
========

Evaluation Metrics

              precision    recall  f1-score   support

           b       0.97      0.78      0.86        40
           g       0.88      0.98      0.93        66

    accuracy                           0.91       106
   macro avg       0.92      0.88      0.89       106
weighted avg       0.91      0.91      0.90       106

================================================================================
========
================================================================================
========
Accuarcy 0.9056603773584906

/usr/local/lib/python3.7/dist-packages/sklearn/utils/
deprecation.py:87: FutureWarning: 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
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



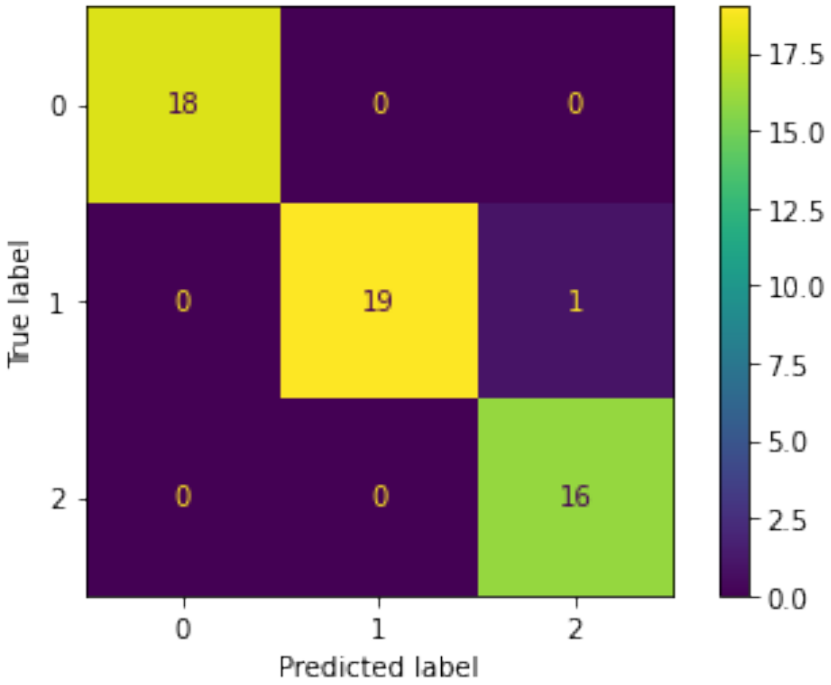
Random Forest Classifier

Wine Dataset

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

# preparing the dataset

from sklearn.datasets import load_wine

dataset = load_wine()

X = dataset.data
y = dataset.target

from sklearn.model_selection import  train_test_split

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

# Feature Scaling
from sklearn.preprocessing import StandardScaler

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

```
from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n_estimators=20, random_state=0)
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:
[[18  0  0]
 [ 0 19  1]
 [ 0  0 16]]
--------------------------------------------------------
--------------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        18
           1       1.00      0.95      0.97        20
           2       0.94      1.00      0.97        16

    accuracy                           0.98        54
   macro avg       0.98      0.98      0.98        54
weighted avg       0.98      0.98      0.98        54
```

```
--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9814814814814815

/usr/local/lib/python3.7/dist-packages/sklearn/utils/
deprecation.py:87: FutureWarning: 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
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



Ionosphere Dataset

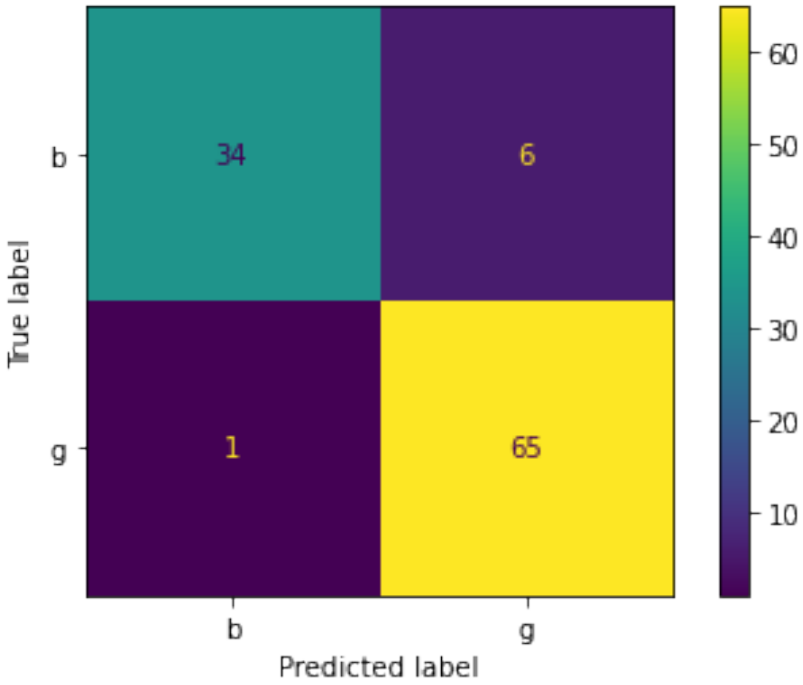
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# preparing the dataset
df = pd.read_csv("ionosphere.data",header=None)

col_name =
['1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16
','17','18','19'
           ,'20','21','22','23','24','25','26','27','28','29','30','31
','32','33','34','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,random_state=10)

# Feature Scaling
from sklearn.preprocessing import StandardScaler

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

# Classification
from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n_estimators=20, random_state=0)
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:
[[34  6]
 [ 1 65]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.97      0.85      0.91        40
           g       0.92      0.98      0.95        66

    accuracy                           0.93       106
   macro avg       0.94      0.92      0.93       106
weighted avg       0.94      0.93      0.93       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9339622641509434
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/
deprecation.py:87: FutureWarning: 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
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



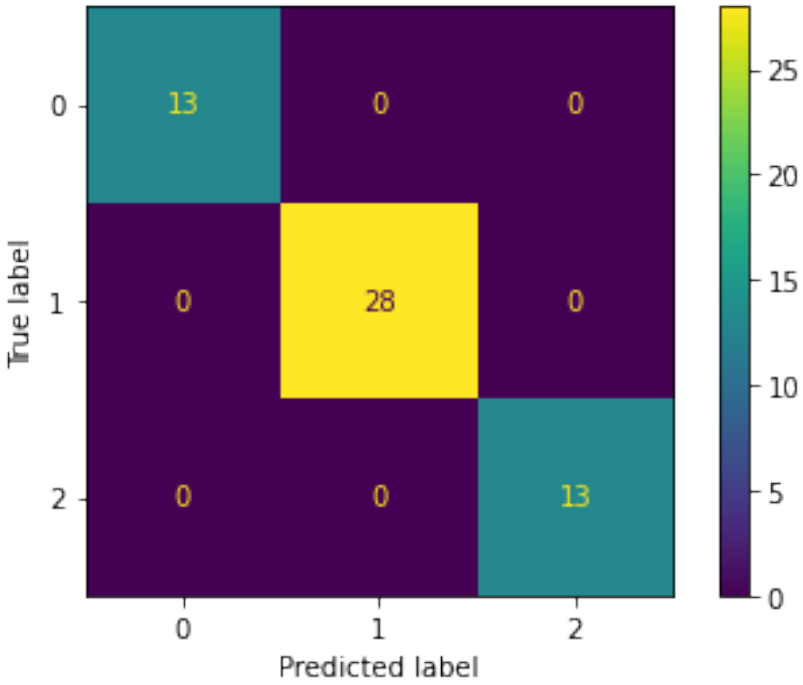
SVM

Wine Dataset

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

# preparing the dataset

from sklearn.datasets import load_wine

dataset = load_wine()

X = dataset.data
y = dataset.target

from sklearn.model_selection import  train_test_split

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

# Feature Scaling
from sklearn.preprocessing import StandardScaler

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

```
from sklearn.svm import SVC

classifier = SVC()
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:
[[13  0  0]
 [ 0 28  0]
 [ 0  0 13]]
----------------------------------------------------
----------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        13
           1       1.00      1.00      1.00        28
           2       1.00      1.00      1.00        13

    accuracy                           1.00        54
   macro avg       1.00      1.00      1.00        54
weighted avg       1.00      1.00      1.00        54
```

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

/usr/local/lib/python3.7/dist-packages/sklearn/utils/
deprecation.py:87: FutureWarning: 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
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



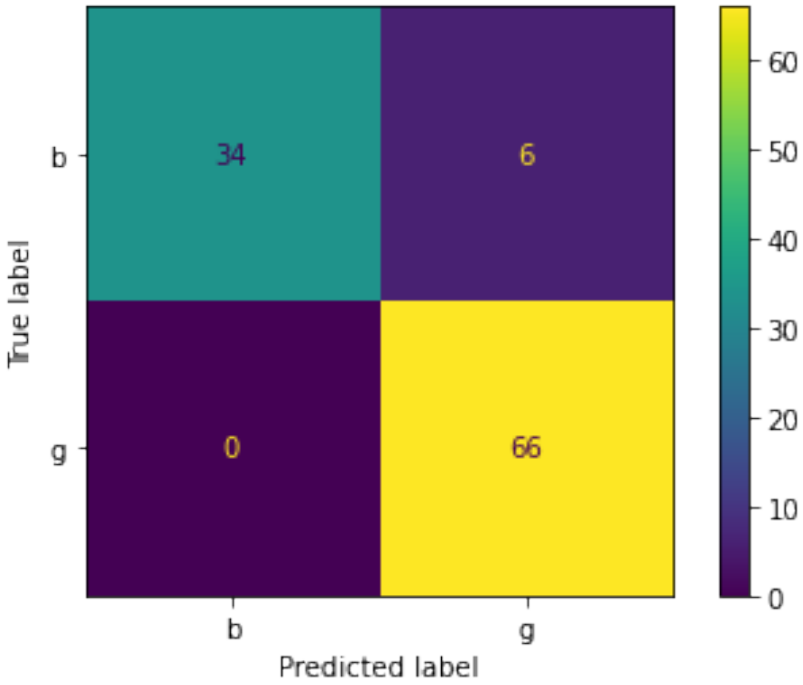
Ionosphere Dataset

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

# preparing the dataset
df = pd.read_csv("ionosphere.data",header=None)

col_name =
['1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16
','17','18','19'
           ,'20','21','22','23','24','25','26','27','28','29','30','31
','32','33','34','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,random_state=10)

# Feature Scaling
from sklearn.preprocessing import StandardScaler

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

from sklearn.preprocessing import StandardScaler

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

# Classification
from sklearn.svm import SVC

classifier = SVC()
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:
[[34  6]
 [ 0 66]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       1.00      0.85      0.92        40
           g       0.92      1.00      0.96        66

    accuracy                           0.94       106
   macro avg       0.96      0.93      0.94       106
weighted avg       0.95      0.94      0.94       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9433962264150944

/usr/local/lib/python3.7/dist-packages/sklearn/utils/
deprecation.py:87: FutureWarning: 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
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



Ans2

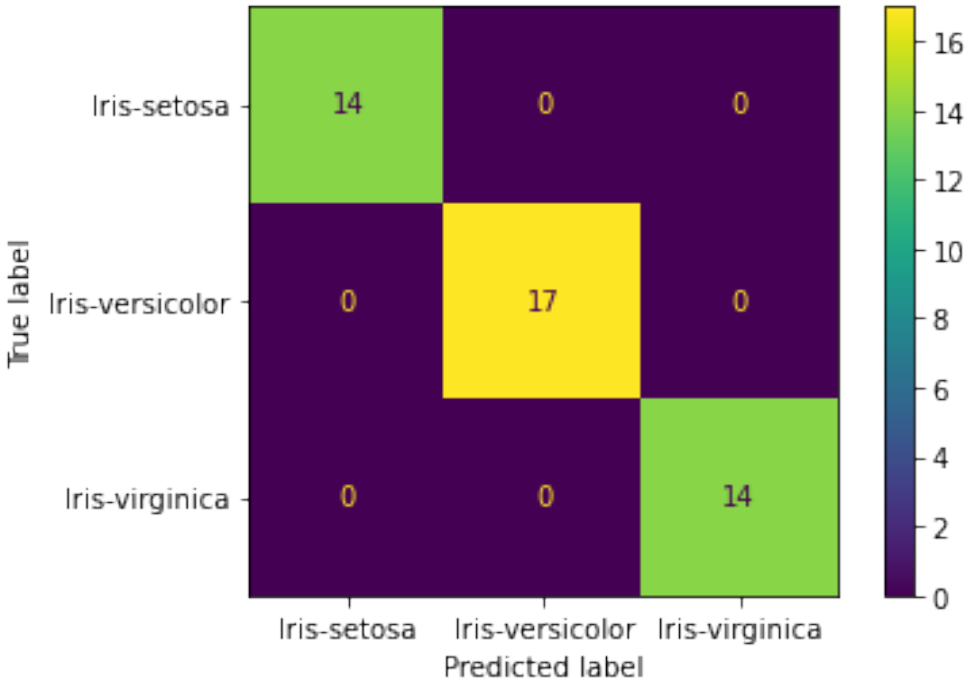
Iris Dataset

```
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,random_state=10)


# 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:
[[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 iterations (200) reached and the optimization
hasn't converged yet.
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87
: FutureWarning: 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
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



Diabetes Dataset

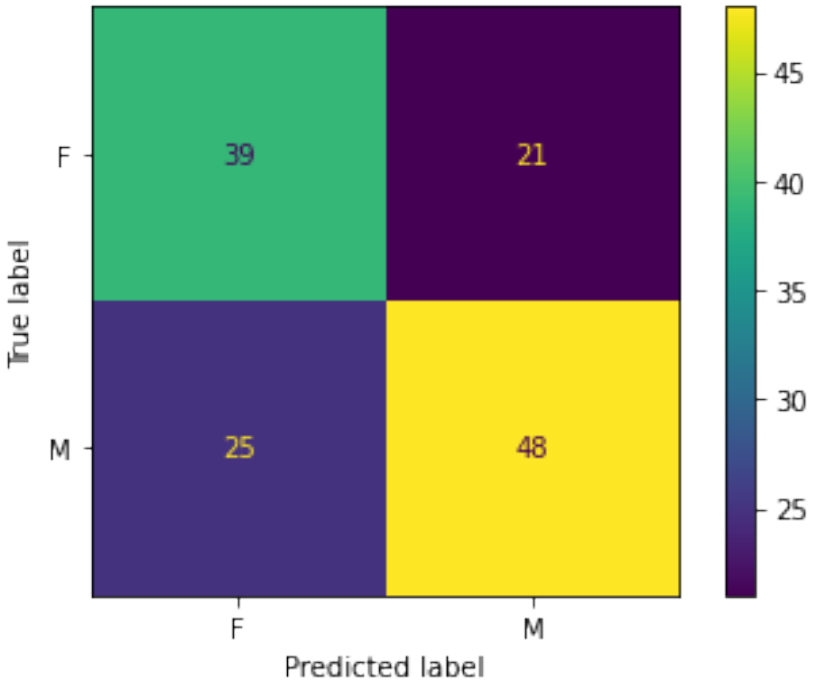
```
import pandas as pd
import numpy as np
```

```
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 to be 'M' and the other to be 'F'

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)

# 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:
[[39 21]
 [25 48]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           F       0.61      0.65      0.63        60
           M       0.70      0.66      0.68        73

    accuracy                           0.65       133
   macro avg       0.65      0.65      0.65       133
weighted avg       0.66      0.65      0.65       133

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.6541353383458647

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

```
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)
```



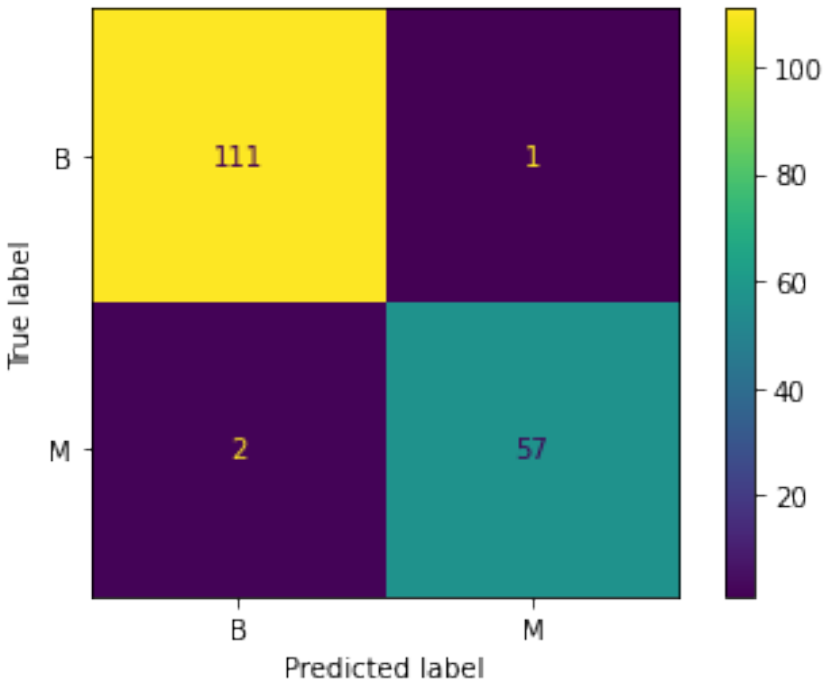
Wisconsin Breast Cancer Dataset

```
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','18','19'
           ,'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,random_state=10)
```

```
# 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()
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/
_multilayer_perceptron.py:696: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (200) reached and the optimization
hasn't converged yet.
  ConvergenceWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87
: FutureWarning: Function plot_confusion_matrix is deprecated;
Function `plot_confusion_matrix` is deprecated in 1.0 and will be
```

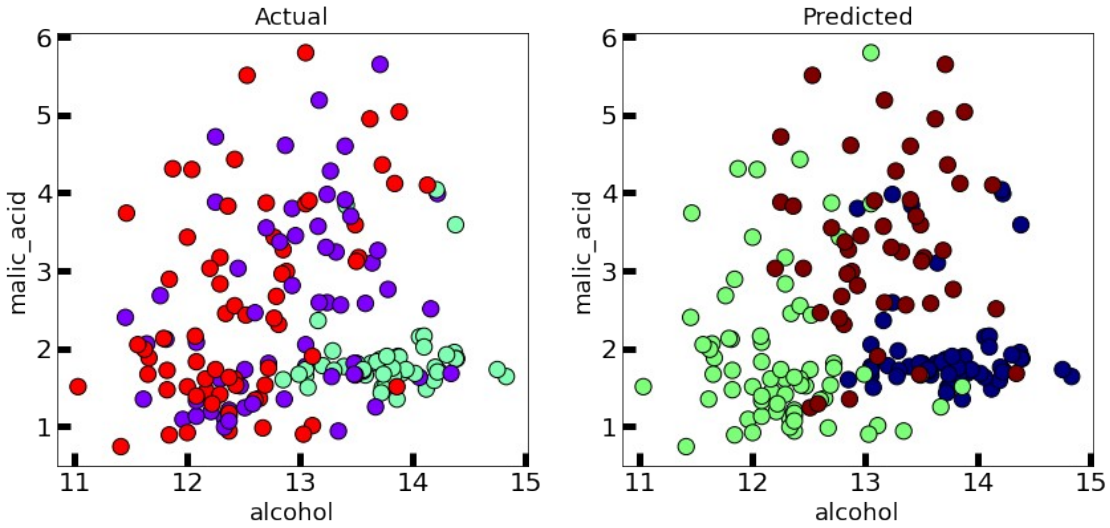
```
removed in 1.2. Use one of the class methods:
ConfusionMatrixDisplay.from_predictions or
ConfusionMatrixDisplay.from_estimator.
  warnings.warn(msg, category=FutureWarning)

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
```



Ans 5

K-Means

```
#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

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)

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=150) #you can also try cmap='rainbow'
axes[1].scatter(x=df['alcohol'], y=df['malic_acid'], c=wine.target,
cmap='jet',edgecolor='k', s=150)
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)

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




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

```
The silhouette score is :

0.5711381937868844
```

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

```
The calinski harabasz score is :

561.815657860671
```

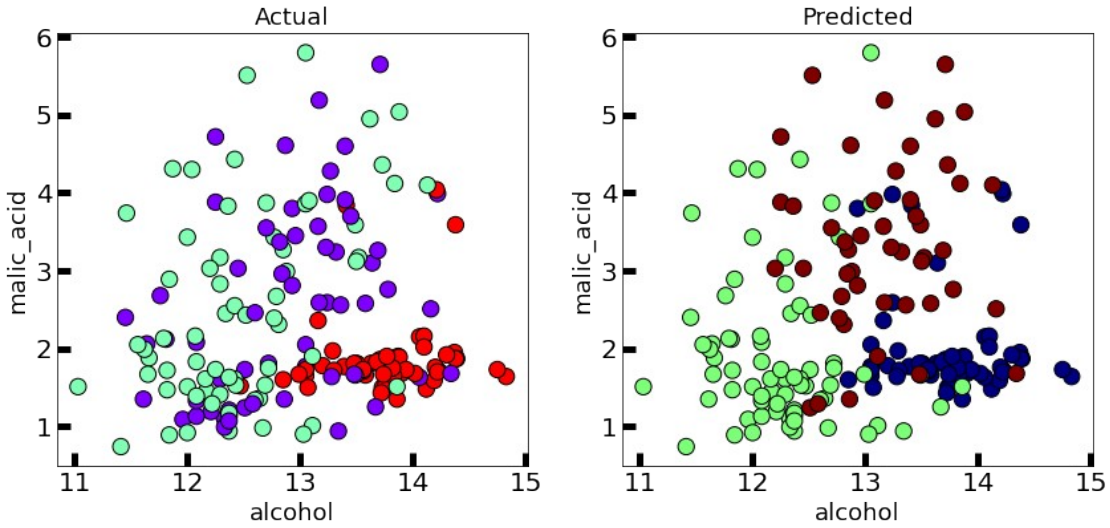
K-medoids

```
!pip install scikit-learn-extra
```

```
#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

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)
```

```
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=150) #you can also try cmap='rainbow'
axes[1].scatter(x=df['alcohol'], y=df['malic_acid'], c=wine.target,
cmap='jet',edgecolor='k', s=150)
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)
```

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




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

```
The silhouette score is :
```

```
0.5666480408636575
```

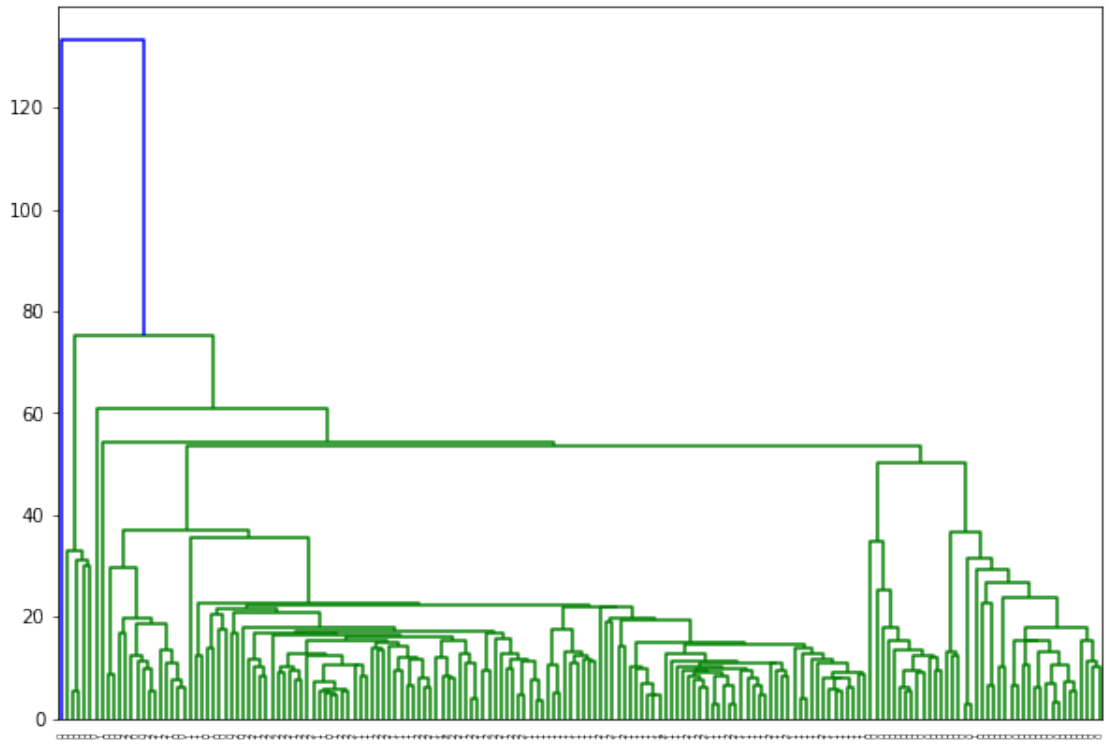
```
from sklearn.metrics import calinski_harabasz_score
print("The calinski harabasz score is :")
calinski_harabasz_score(x, kmedoid.labels_)
```

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

```
539.3792353535451
```

Dendrogram

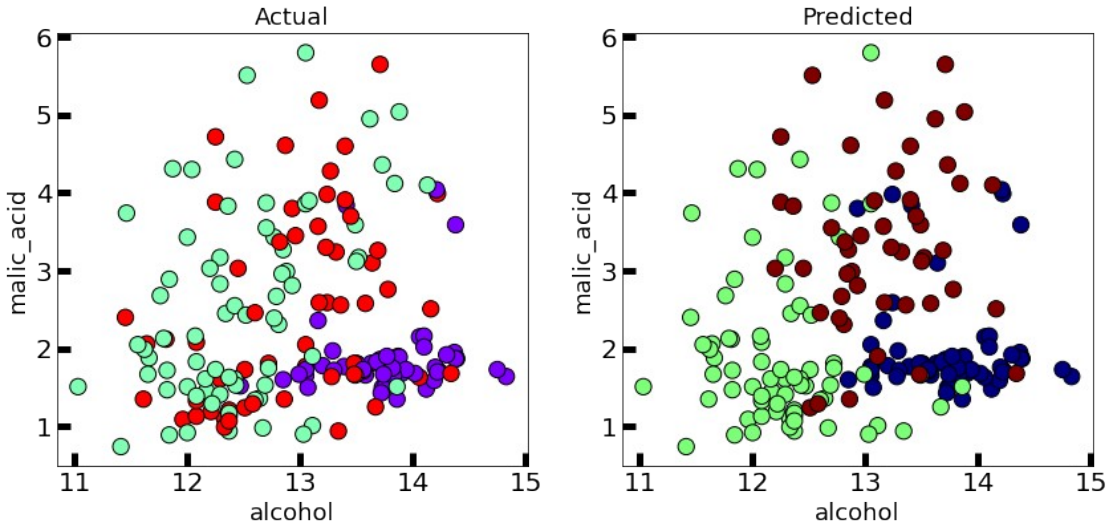
```
#importing libraries
import numpy as np
import pandas as pd
import sklearn as sk
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.datasets import load_wine

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

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()
```



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

Agnes

```
#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)

from sklearn.cluster import AgglomerativeClustering

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

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=150) #you can also try cmap='rainbow'
axes[1].scatter(x=df['alcohol'], y=df['malic_acid'], c=wine.target,
cmap='jet',edgecolor='k', s=150)
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)

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




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

```
The silhouette score is :

0.5644796401732074
```

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

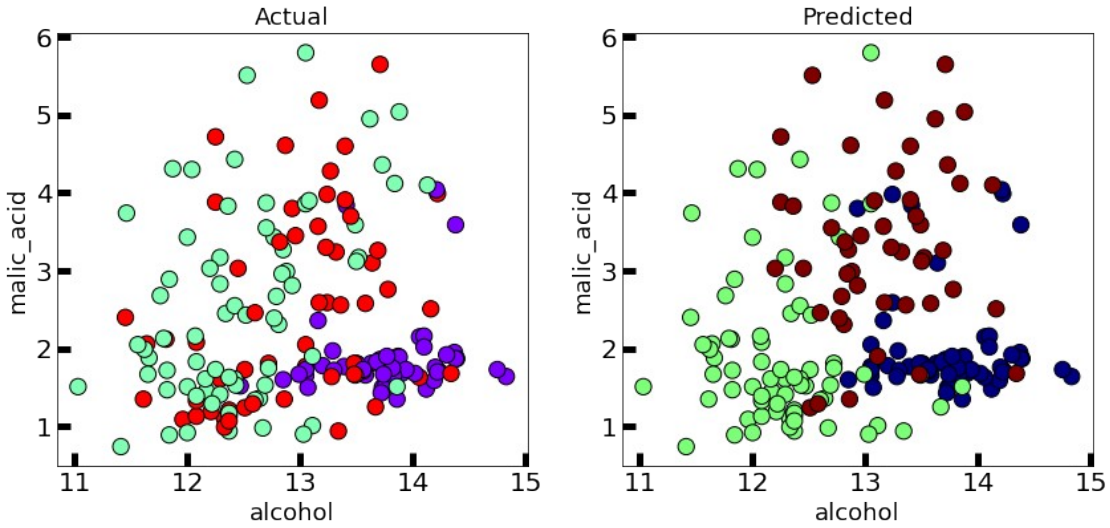
```
The calinski harabasz score is :

552.851711505718
```

Birch

```
#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)

from sklearn.cluster import Birch

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

```
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=150) #you can also try cmap='rainbow'
axes[1].scatter(x=df['alcohol'], y=df['malic_acid'], c=wine.target,
cmap='jet',edgecolor='k', s=150)
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)
```

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




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

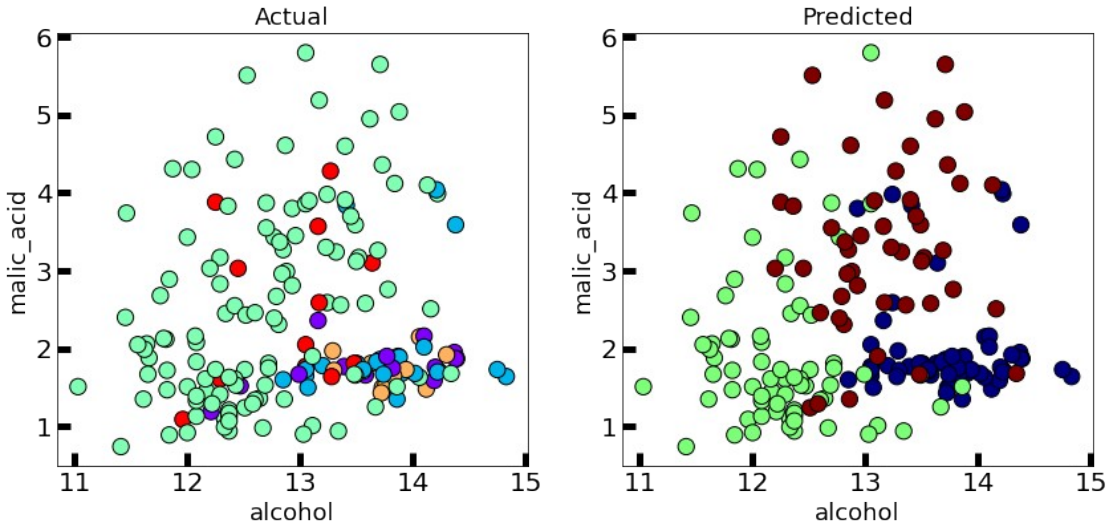
```
The silhouette score is :

0.5644796401732074
```

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

```
The calinski harabasz score is :

552.851711505718
```

DBSCAN

```
#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)

from sklearn.cluster import DBSCAN

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

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=150) #you can also try cmap='rainbow'
axes[1].scatter(x=df['alcohol'], y=df['malic_acid'], c=wine.target,
cmap='jet',edgecolor='k', s=150)
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)

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




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

```
The silhouette score is :

0.4413295944891938
```

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

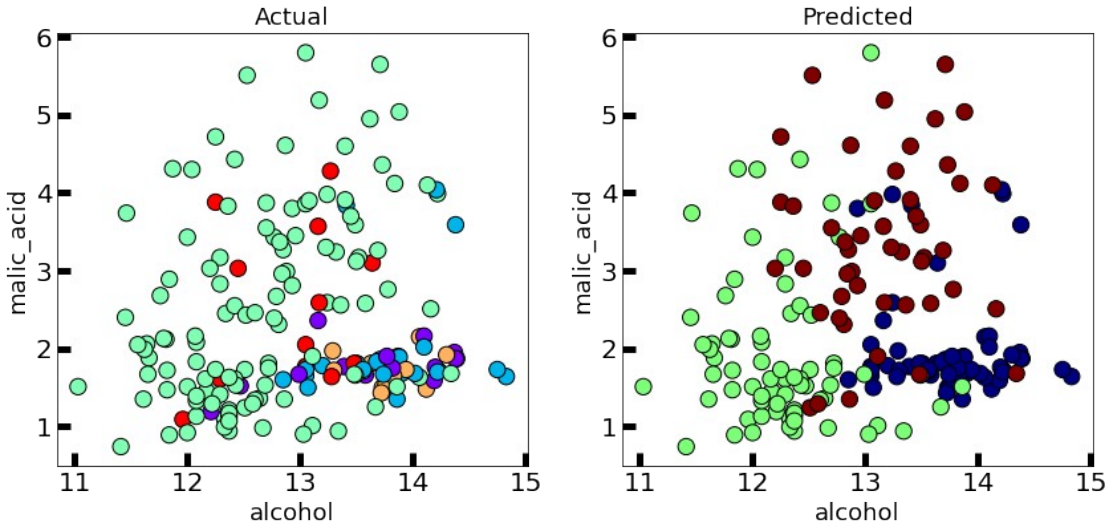
```
The calinski harabasz score is :

208.9449395725058
```

OPTICS

```
#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)

from sklearn.cluster import DBSCAN

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

```
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=150) #you can also try cmap='rainbow'
axes[1].scatter(x=df['alcohol'], y=df['malic_acid'], c=wine.target,
cmap='jet',edgecolor='k', s=150)
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)
```

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




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

```
The silhouette score is :

0.4413295944891938
```

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

```
The calinski harabasz score is :

208.9449395725058
```