**Name: GARGI GHOSAL**

**Class Roll No.: 301911001004**

**Department: B.E Information Technology**

**Year of study: 4th Year**

**Semester: 1st Sem**

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

**Git link:** [**https://github.com/GargiGhosal/MachineLearning\_Lab.git**](https://github.com/GargiGhosal/MachineLearning_Lab.git)

**Solutions::**

**Q1. Datasets used :**

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

**Q1. Answers::**

# WINE DATASET

# SVM(Without Tuning)[70-30 split]

import pandas as pd

import numpy as np

# Dataset Preparation

df = pd.read\_csv("wine.data",header=None)

col\_name = ['Class','Alcohol','Malic acid','Ash','Alcalinity of ash','Magnesium','Total phenols','Flavanoids',

'Nonflavanoid phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted wines','Proline']

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=0.3,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.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:")

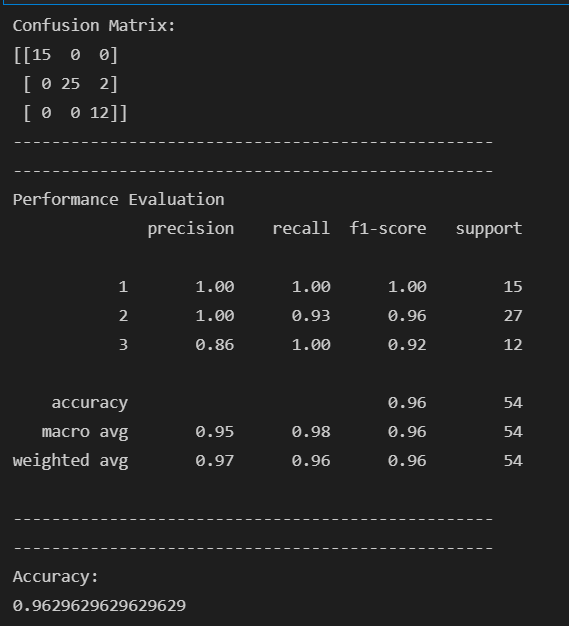
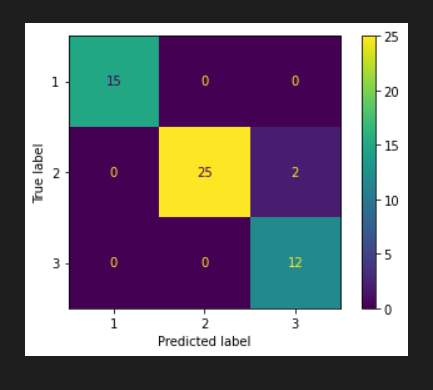
print(accuracy\_score(y\_test, y\_pred))

import matplotlib.pyplot as plt

from sklearn.metrics import plot\_confusion\_matrix

plot\_confusion\_matrix(classifier, X\_test, y\_test)

plt.show()

** **

# WINE DATASET

# SVM(Without Tuning)[60-40 split]

import pandas as pd

import numpy as np

# Dataset Preparation

df = pd.read\_csv("wine.data",header=None)

col\_name = ['Class','Alcohol','Malic acid','Ash','Alcalinity of ash','Magnesium','Total phenols','Flavanoids',

'Nonflavanoid phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted wines','Proline']

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=0.6,test\_size=0.4,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.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:")

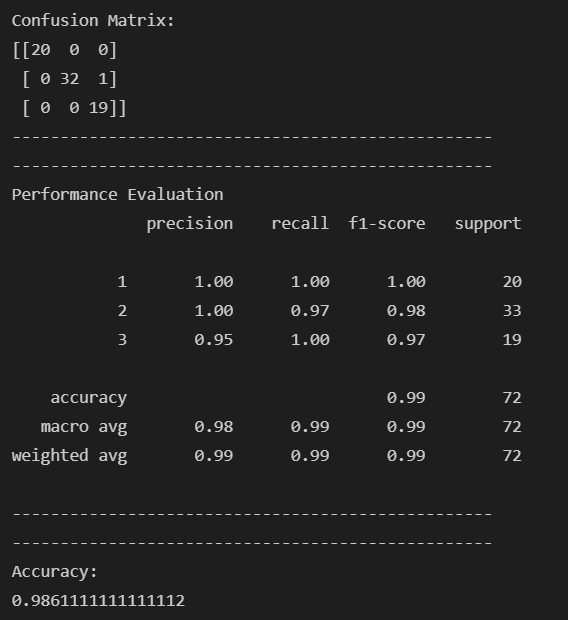
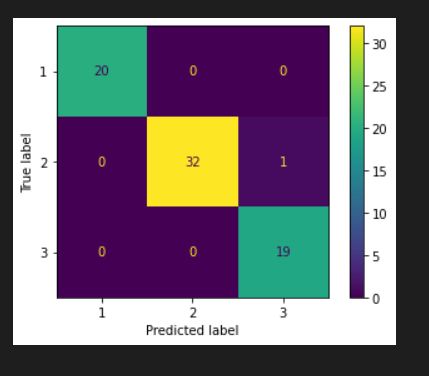
print(accuracy\_score(y\_test, y\_pred))

import matplotlib.pyplot as plt

from sklearn.metrics import plot\_confusion\_matrix

plot\_confusion\_matrix(classifier, X\_test, y\_test)

plt.show()

# WINE DATASET

# SVM(Without Tuning)[40-60 split]

import pandas as pd

import numpy as np

# Dataset Preparation

df = pd.read\_csv("wine.data",header=None)

col\_name = ['Class','Alcohol','Malic acid','Ash','Alcalinity of ash','Magnesium','Total phenols','Flavanoids',

'Nonflavanoid phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted wines','Proline']

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=0.4,test\_size=0.6,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.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:")

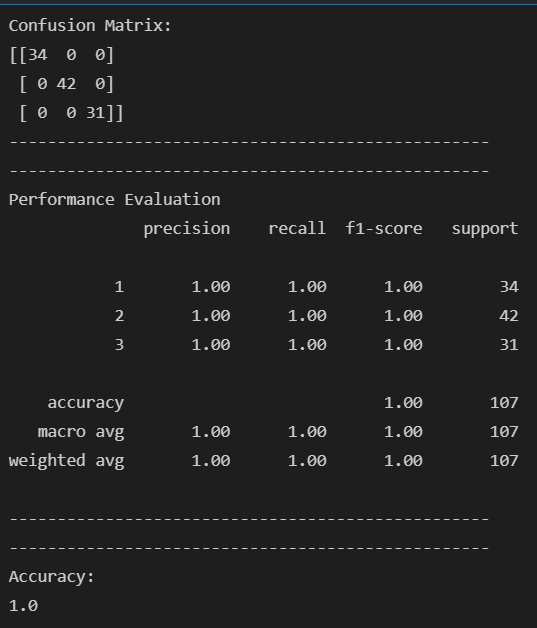
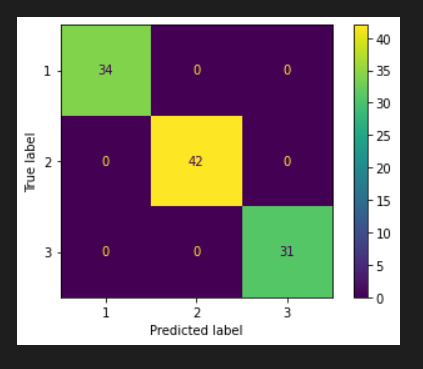
print(accuracy\_score(y\_test, y\_pred))

import matplotlib.pyplot as plt

from sklearn.metrics import plot\_confusion\_matrix

plot\_confusion\_matrix(classifier, X\_test, y\_test)

plt.show()

# WINE DATASET

# SVM(With Tuning)[40-60 split]

import pandas as pd

import numpy as np

# Dataset Preparation

df = pd.read\_csv("wine.data",header=None)

col\_name = ['Class','Alcohol','Malic acid','Ash','Alcalinity of ash','Magnesium','Total phenols','Flavanoids',

'Nonflavanoid phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted wines','Proline']

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=0.4,test\_size=0.6,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.svm import SVC

classifier = SVC()

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

# 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

param\_grid = {'C': [0.1,1, 10, 100], 'gamma': [1,0.1,0.01,0.001],'kernel': ['rbf', 'poly', 'sigmoid']}

pprint(param\_grid)

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

from sklearn.model\_selection import GridSearchCV

# Use the random grid to search for best hyperparameters

# First create the base model to tune

classifier = SVC()

# Random search of parameters, using 3 fold cross validation,

# search across 100 different combinations, and use all available cores

rf\_random = GridSearchCV(SVC(), param\_grid, refit=True, verbose=2)

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:")

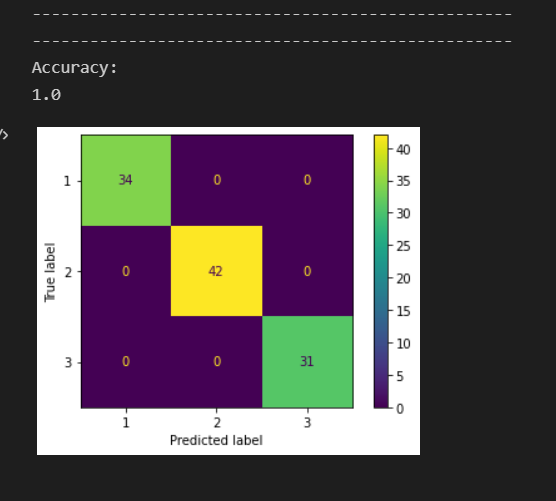
print(accuracy\_score(y\_test, y\_pred))

import matplotlib.pyplot as plt

from sklearn.metrics import plot\_confusion\_matrix

plot\_confusion\_matrix(rf\_random, X\_test, y\_test)

plt.show()



# WINE DATASET

# Decision Tree (Without Tuning)[70-30 split]

import pandas as pd

import numpy as np

# Dataset Preparation

df = pd.read\_csv("wine.data",header=None)

col\_name = ['Class','Alcohol','Malic acid','Ash','Alcalinity of ash','Magnesium','Total phenols','Flavanoids',

'Nonflavanoid phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted wines','Proline']

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=0.3,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().fit(X\_train,y\_train)

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

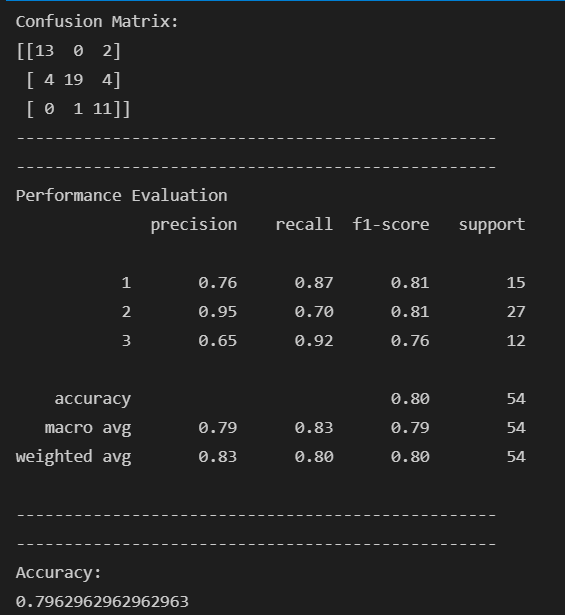
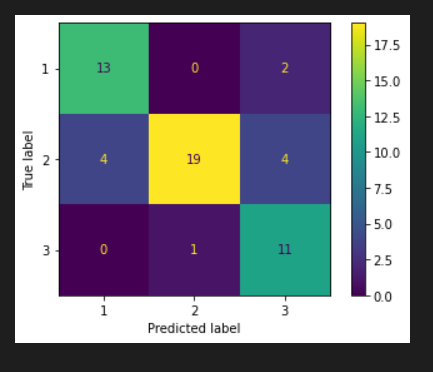
plot\_confusion\_matrix(classifier, X\_test, y\_test)

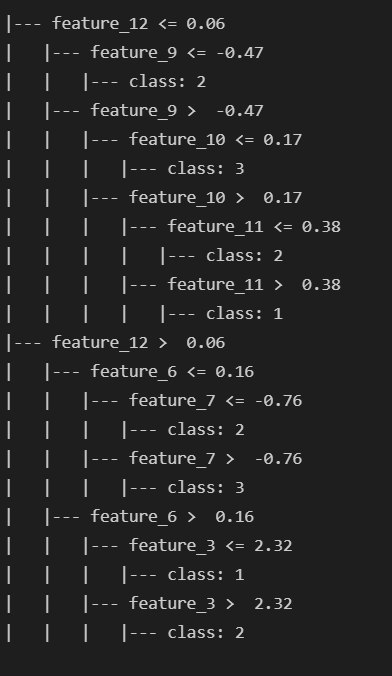
plt.show()

from sklearn import tree

text\_representation = tree.export\_text(classifier)

print(text\_representation)



# WINE DATASET

# Decision Tree (With Tuning)[70-30 split]

import pandas as pd

import numpy as np

# Dataset Preparation

df = pd.read\_csv("wine.data",header=None)

col\_name = ['Class','Alcohol','Malic acid','Ash','Alcalinity of ash','Magnesium','Total phenols','Flavanoids',

'Nonflavanoid phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted wines','Proline']

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=0.3,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(criterion="entropy",max\_depth=20).fit(X\_train,y\_train)

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

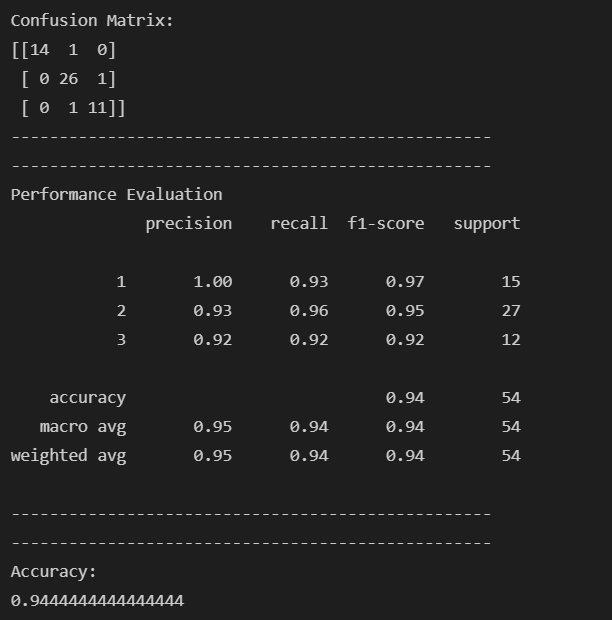
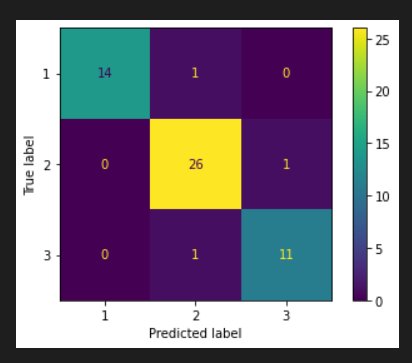
plot\_confusion\_matrix(classifier, X\_test, y\_test)

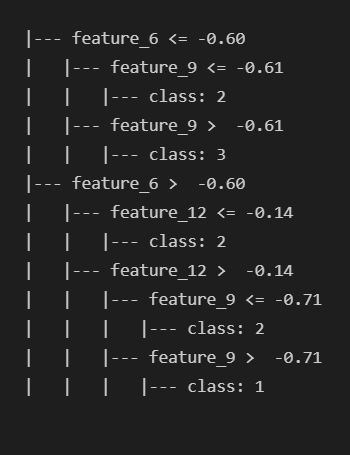
plt.show()

from sklearn import tree

text\_representation = tree.export\_text(classifier)

print(text\_representation)



# WINE DATASET

# Random Forest Classifier(Without Tuning)[70-30 split]

import pandas as pd

import numpy as np

# Dataset Preparation

df = pd.read\_csv("wine.data",header=None)

col\_name = ['Class','Alcohol','Malic acid','Ash','Alcalinity of ash','Magnesium','Total phenols','Flavanoids',

'Nonflavanoid phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted wines','Proline']

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=0.3,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:")

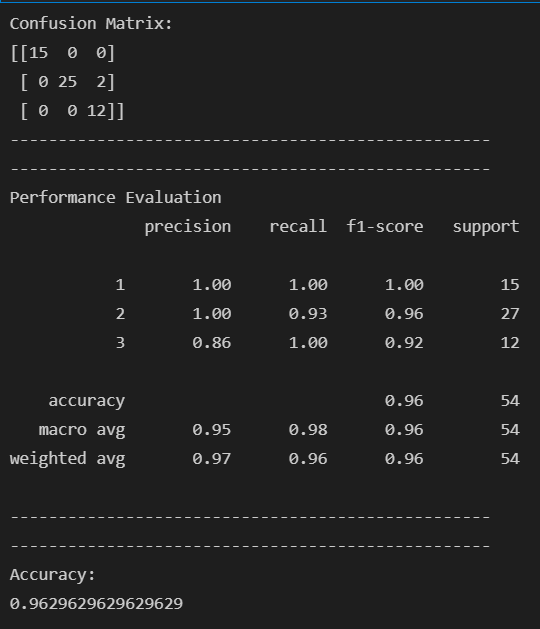
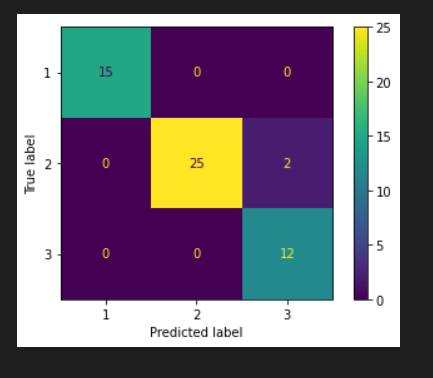
print(accuracy\_score(y\_test, y\_pred))

import matplotlib.pyplot as plt

from sklearn.metrics import plot\_confusion\_matrix

plot\_confusion\_matrix(classifier, X\_test, y\_test)

plt.show()

# WINE DATASET

# Random Forest Classifier(With Tuning)[70-30 split]

import pandas as pd

import numpy as np

# Dataset Preparation

df = pd.read\_csv("wine.data",header=None)

col\_name = ['Class','Alcohol','Malic acid','Ash','Alcalinity of ash','Magnesium','Total phenols','Flavanoids',

'Nonflavanoid phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted wines','Proline']

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()

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

# 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

from sklearn.model\_selection import RandomizedSearchCV

# Number of trees in random forest

n\_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]

# Number of features to consider at every split

max\_features = ['auto', 'sqrt']

# Maximum number of levels in tree

max\_depth = [int(x) for x in np.linspace(10, 110, num = 11)]

max\_depth.append(None)

# Minimum number of samples required to split a node

min\_samples\_split = [2, 5, 10]

# Minimum number of samples required at each leaf node

min\_samples\_leaf = [1, 2, 4]

# Method of selecting samples for training each tree

bootstrap = [True, False]

# Create the random grid

random\_grid = {'n\_estimators': n\_estimators,

'max\_features': max\_features,

'max\_depth': max\_depth,

'min\_samples\_split': min\_samples\_split,

'min\_samples\_leaf': min\_samples\_leaf,

'bootstrap': bootstrap}

pprint(random\_grid)

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

# Use the random grid to search for best hyperparameters

# First create the base model to tune

classifier = RandomForestClassifier()

# Random search of parameters, using 3 fold cross validation,

# search across 100 different combinations, and use all available cores

rf\_random = RandomizedSearchCV(estimator = classifier, param\_distributions = random\_grid, n\_iter = 100, cv = 3, verbose=2, random\_state=42, n\_jobs = -1)

# Fit the random search model

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:")

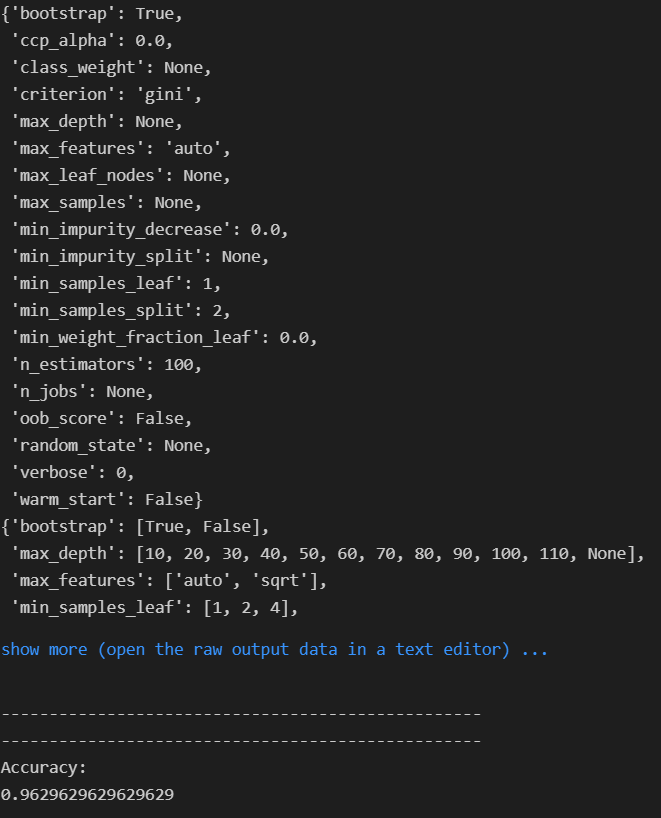
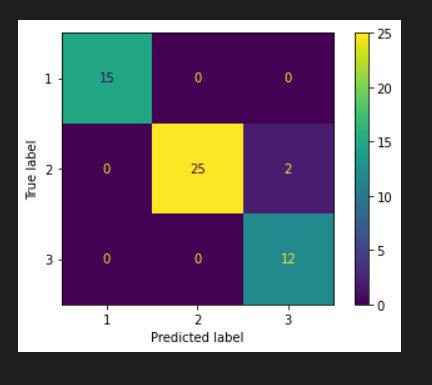
print(accuracy\_score(y\_test, y\_pred))

import matplotlib.pyplot as plt

from sklearn.metrics import plot\_confusion\_matrix

plot\_confusion\_matrix(rf\_random, X\_test, y\_test)

plt.show()

# WINE DATASET

# Gaussian Naive Bayes (Without Tuning)[70-30 split]

import pandas as pd

import numpy as np

# Dataset Preparation

df = pd.read\_csv("wine.data",header=None)

col\_name = ['Class','Alcohol','Malic acid','Ash','Alcalinity of ash','Magnesium','Total phenols','Flavanoids',

'Nonflavanoid phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted wines','Proline']

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=0.3,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.naive\_bayes import GaussianNB

classifier = GaussianNB().fit(X\_train,y\_train)

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:")

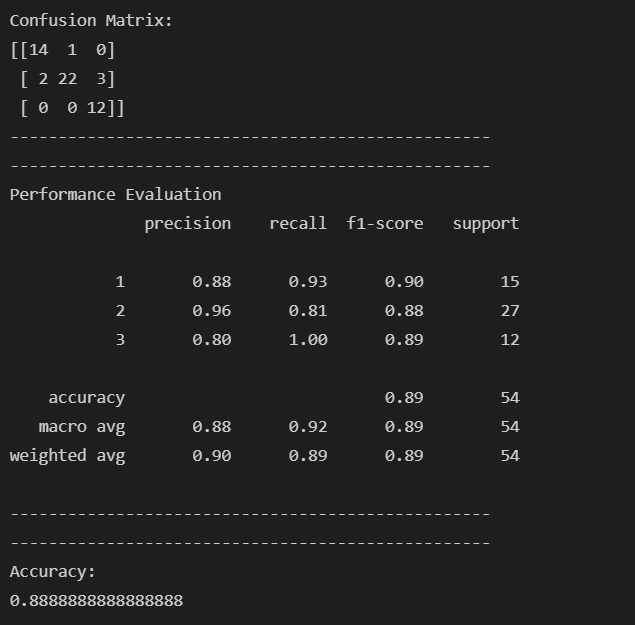
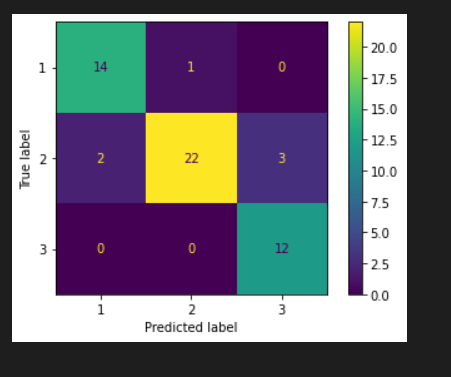
print(accuracy\_score(y\_test, y\_pred))

import matplotlib.pyplot as plt

from sklearn.metrics import plot\_confusion\_matrix

plot\_confusion\_matrix(classifier, X\_test, y\_test)

plt.show()

# WINE DATASET

# Gaussian Naive Bayes (With Tuning)[70-30 split]

import pandas as pd

import numpy as np

# Dataset Preparation

df = pd.read\_csv("wine.data",header=None)

col\_name = ['Class','Alcohol','Malic acid','Ash','Alcalinity of ash','Magnesium','Total phenols','Flavanoids',

'Nonflavanoid phenols','Proanthocyanins','Color intensity','Hue','OD280/OD315 of diluted wines','Proline']

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=0.3,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.naive\_bayes import GaussianNB

classifier = GaussianNB(priors=None,var\_smoothing=1e-5).fit(X\_train,y\_train)

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:")

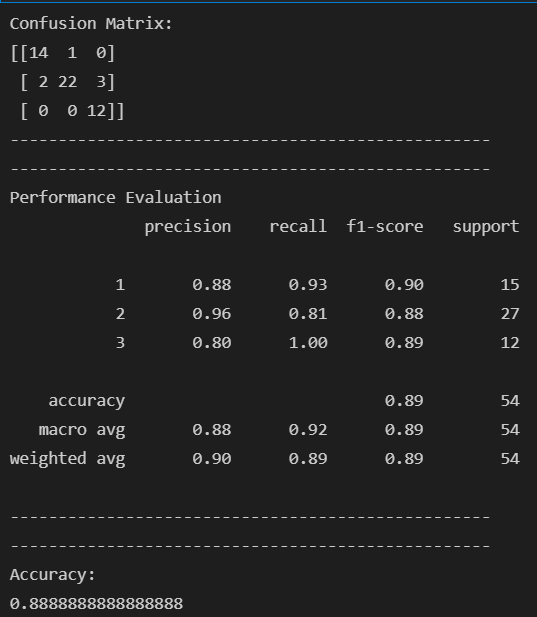
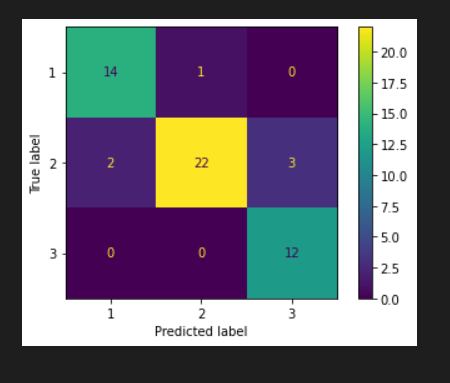
print(accuracy\_score(y\_test, y\_pred))

import matplotlib.pyplot as plt

from sklearn.metrics import plot\_confusion\_matrix

plot\_confusion\_matrix(classifier, X\_test, y\_test)

plt.show()

# IONOSPHERE DATASET

# SVM(Without Tuning)[70-30 split]

import pandas as pd

import numpy as np

# Dataset Preparation

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=0.3,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.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:")

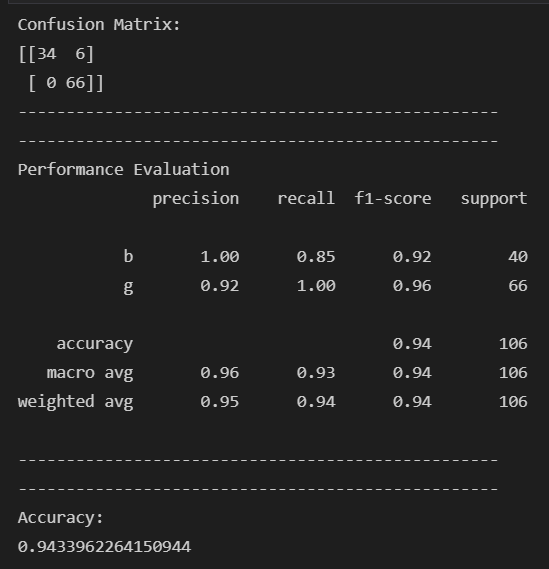
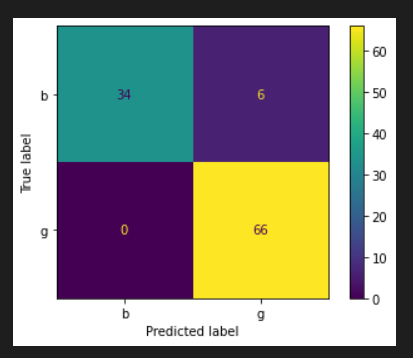
print(accuracy\_score(y\_test, y\_pred))

import matplotlib.pyplot as plt

from sklearn.metrics import plot\_confusion\_matrix

plot\_confusion\_matrix(classifier, X\_test, y\_test)

plt.show()

# IONOSPHERE DATASET

# SVM(With Tuning)[70-30 split]

import pandas as pd

import numpy as np

# Dataset Preparation

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=0.3,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.svm import SVC

classifier = SVC()

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

# 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

param\_grid = {'C': [0.1,1, 10, 100], 'gamma': [1,0.1,0.01,0.001],'kernel': ['rbf', 'poly', 'sigmoid']}

pprint(param\_grid)

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

from sklearn.model\_selection import GridSearchCV

# Use the random grid to search for best hyperparameters

# First create the base model to tune

classifier = SVC()

# Random search of parameters, using 3 fold cross validation,

# search across 100 different combinations, and use all available cores

rf\_random = GridSearchCV(SVC(), param\_grid, refit=True, verbose=2)

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:")

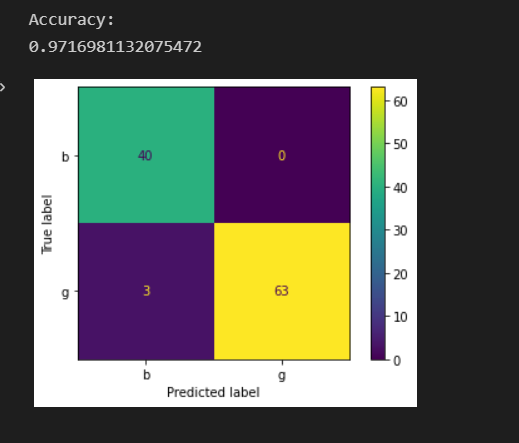
print(accuracy\_score(y\_test, y\_pred))

import matplotlib.pyplot as plt

from sklearn.metrics import plot\_confusion\_matrix

plot\_confusion\_matrix(rf\_random, X\_test, y\_test)

plt.show()



# Ionosphere DATASET

# Decision Tree (Without Tuning)[70-30 split]

import pandas as pd

import numpy as np

# Dataset Preparation

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=0.3,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().fit(X\_train,y\_train)

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

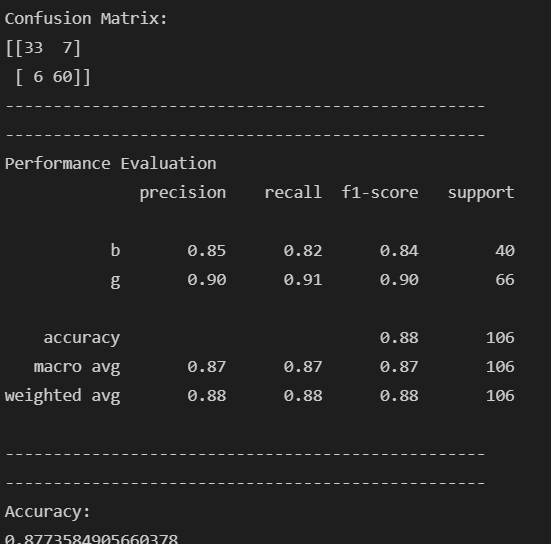
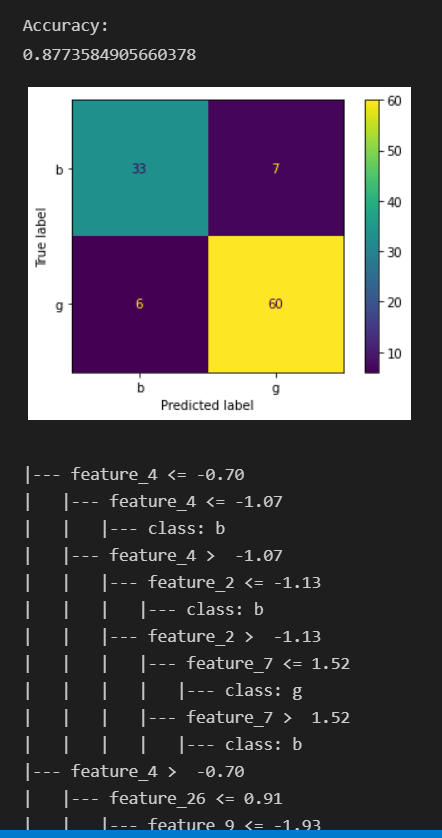
plot\_confusion\_matrix(classifier, X\_test, y\_test)

plt.show()

from sklearn import tree

text\_representation = tree.export\_text(classifier)

print(text\_representation)

# Ionosphere DATASET

# Decision Tree (With Tuning)[70-30 split]

import pandas as pd

import numpy as np

# Dataset Preparation

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=0.3,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(criterion="entropy",max\_depth=20).fit(X\_train,y\_train)

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

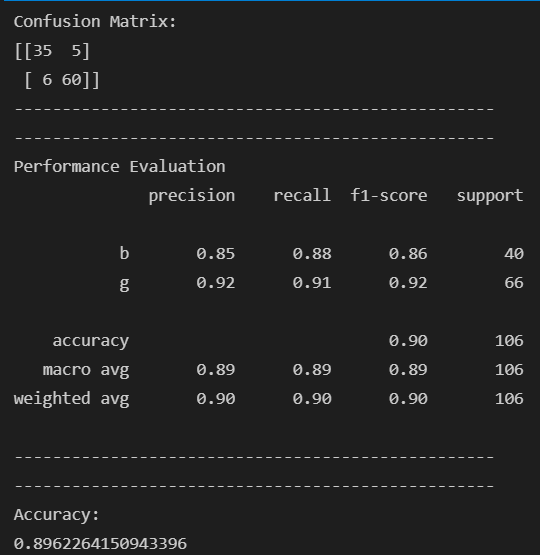
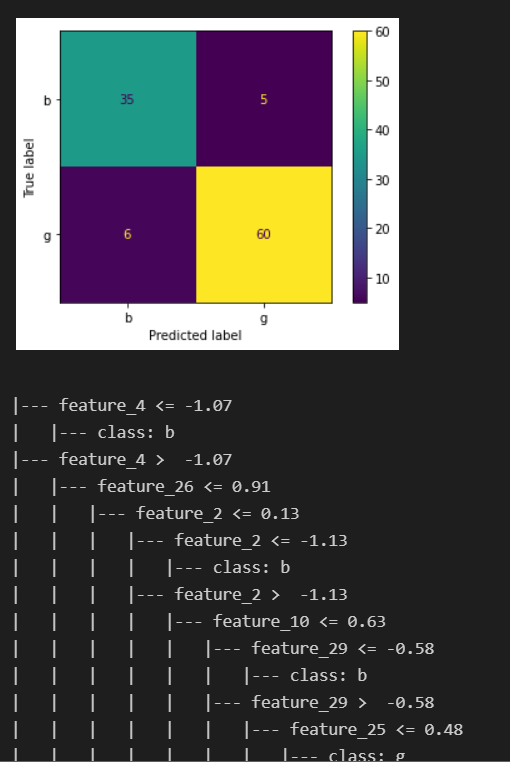
plot\_confusion\_matrix(classifier, X\_test, y\_test)

plt.show()

from sklearn import tree

text\_representation = tree.export\_text(classifier)

print(text\_representation)

# IONOSPHERE DATASET

# Random Forest Classifier(Without Tuning)[70-30 split]

import pandas as pd

import numpy as np

# Dataset Preparation

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:")

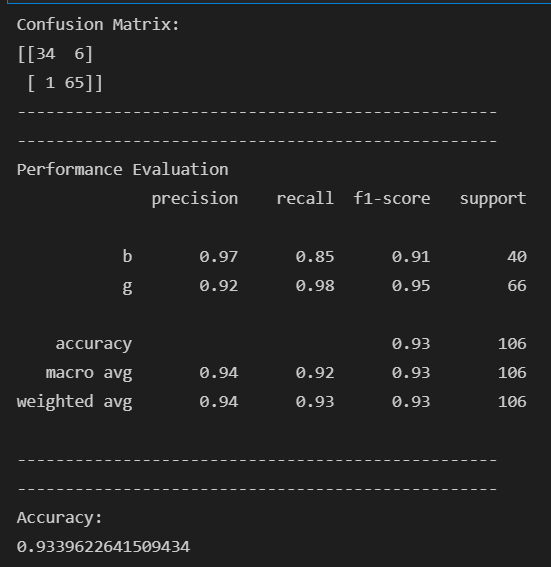
print(accuracy\_score(y\_test, y\_pred))

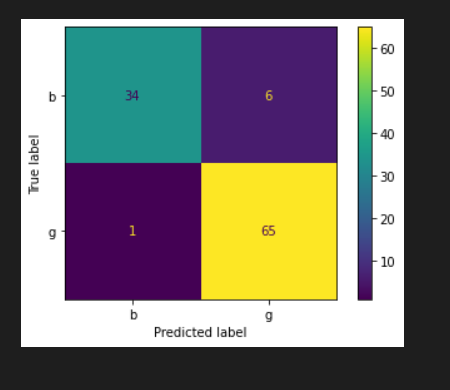
import matplotlib.pyplot as plt

from sklearn.metrics import plot\_confusion\_matrix

plot\_confusion\_matrix(classifier, X\_test, y\_test)

plt.show()





# IONOSPHERE DATASET

# Random Forest Classifier(With Tuning)[70-30 split]

import pandas as pd

import numpy as np

# Dataset Preparation

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=0.3,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()

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

# 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

from sklearn.model\_selection import RandomizedSearchCV

# Number of trees in random forest

n\_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]

# Number of features to consider at every split

max\_features = ['auto', 'sqrt']

# Maximum number of levels in tree

max\_depth = [int(x) for x in np.linspace(10, 110, num = 11)]

max\_depth.append(None)

# Minimum number of samples required to split a node

min\_samples\_split = [2, 5, 10]

# Minimum number of samples required at each leaf node

min\_samples\_leaf = [1, 2, 4]

# Method of selecting samples for training each tree

bootstrap = [True, False]

# Create the random grid

random\_grid = {'n\_estimators': n\_estimators,

'max\_features': max\_features,

'max\_depth': max\_depth,

'min\_samples\_split': min\_samples\_split,

'min\_samples\_leaf': min\_samples\_leaf,

'bootstrap': bootstrap}

pprint(random\_grid)

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

# Use the random grid to search for best hyperparameters

# First create the base model to tune

classifier = RandomForestClassifier()

# Random search of parameters, using 3 fold cross validation,

# search across 100 different combinations, and use all available cores

rf\_random = RandomizedSearchCV(estimator = classifier, param\_distributions = random\_grid, n\_iter = 100, cv = 3, verbose=2, random\_state=42, n\_jobs = -1)

# Fit the random search model

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:")

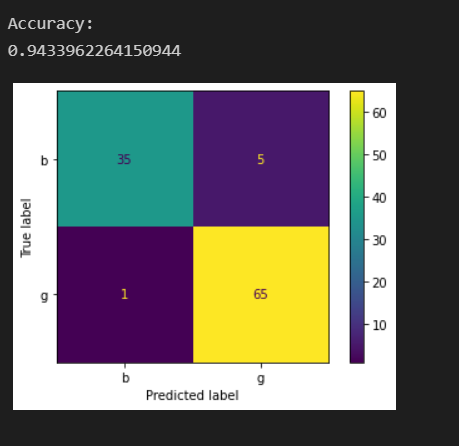
print(accuracy\_score(y\_test, y\_pred))

import matplotlib.pyplot as plt

from sklearn.metrics import plot\_confusion\_matrix

plot\_confusion\_matrix(rf\_random, X\_test, y\_test)

plt.show()



# Ionosphere DATASET

# Gaussian Naive Bayes (Without Tuning)[70-30 split]

import pandas as pd

import numpy as np

# Dataset Preparation

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=0.3,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.naive\_bayes import GaussianNB

classifier = GaussianNB().fit(X\_train,y\_train)

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:")

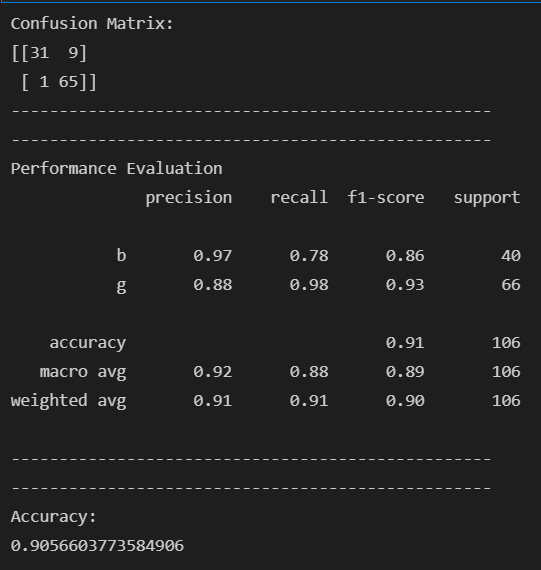
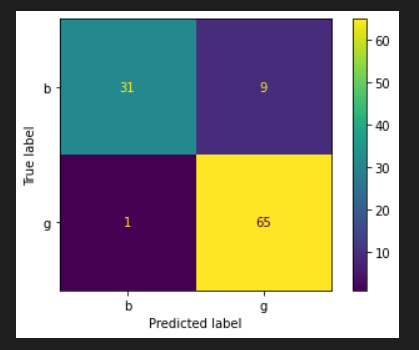
print(accuracy\_score(y\_test, y\_pred))

import matplotlib.pyplot as plt

from sklearn.metrics import plot\_confusion\_matrix

plot\_confusion\_matrix(classifier, X\_test, y\_test)

plt.show()

Q3. Used datasets

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>

# IONOSPHERE DATASET

# GaussianHMM(Without Tuning)[70-30 split]

print("Split: 70-30")

import pandas as pd

import numpy as np

# Dataset Preparation

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(['1','2','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=0.3,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 hmmlearn import hmm

classifier = hmm.GaussianHMM(n\_components=2, covariance\_type="full")

classifier.fit(X\_train)

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

size = len(y\_pred)

strings = np.empty(size, np.unicode\_)

for i in range (size):

if y\_pred[i] == 1:

strings[i] = ("g")

else:

strings[i] = ("b")

strings

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

print("Confusion Matrix:")

print(confusion\_matrix(y\_test, strings))

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

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

print("Performance Evaluation")

print(classification\_report(y\_test, strings))

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

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

print("Accuracy:")

print(accuracy\_score(y\_test, strings))

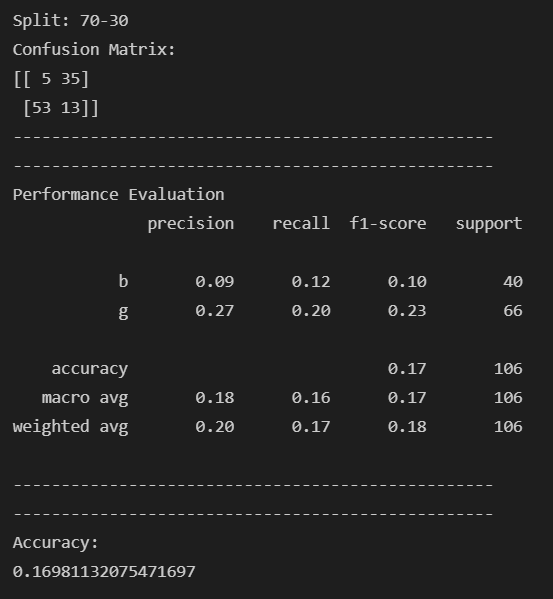
import matplotlib.pyplot as plt

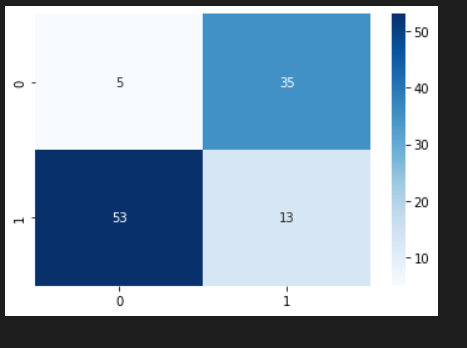
import seaborn as sns

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

sns.heatmap(cm, annot=True, fmt="d",cmap='Blues')

plt.show()





# IONOSPHERE DATASET

# GaussianHMM(With Tuning)[70-30 split]

import pandas as pd

import numpy as np

# Dataset Preparation

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(['1','2','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=0.3,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 hmmlearn import hmm

classifier = hmm.GaussianHMM(n\_components=2, covariance\_type="full",n\_iter=5,algorithm='viterbi',verbose=False)

classifier.fit(X\_train)

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

size = len(y\_pred)

strings = np.empty(size, np.unicode\_)

for i in range (size):

if y\_pred[i] == 1:

strings[i] = ("g")

else:

strings[i] = ("b")

strings

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

print("Confusion Matrix:")

print(confusion\_matrix(y\_test, strings))

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

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

print("Performance Evaluation")

print(classification\_report(y\_test, strings))

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

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

print("Accuracy:")

print(accuracy\_score(y\_test, strings))

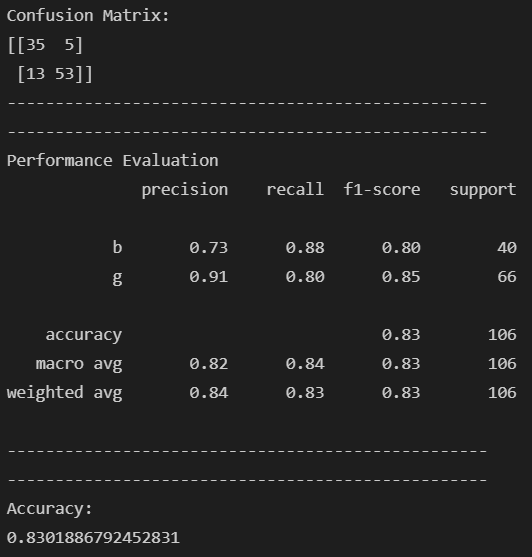
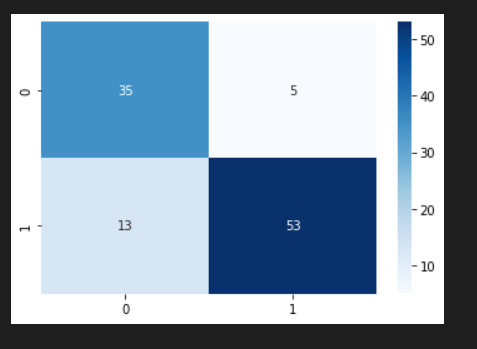
import matplotlib.pyplot as plt

import seaborn as sns

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

sns.heatmap(cm, annot=True, fmt="d",cmap='Blues')

plt.show()

# IONOSPHERE DATASET

# GMMHMM(Without Tuning)[60-40 split]

print("Split: 60-40")

import pandas as pd

import numpy as np

# Dataset Preparation

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(['1','2','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=0.6,test\_size=0.4,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 hmmlearn import hmm

import hmmlearn

classifier = hmmlearn.hmm.GMMHMM(n\_components=2, random\_state=10,covariance\_type='full',algorithm='viterbi',n\_iter=10)

classifier.fit(X\_train)

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

size = len(y\_pred)

strings = np.empty(size, np.unicode\_)

for i in range (size):

if y\_pred[i] == 1:

strings[i] = ("g")

else:

strings[i] = ("b")

strings

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

print("Confusion Matrix:")

print(confusion\_matrix(y\_test, strings))

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

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

print("Performance Evaluation")

print(classification\_report(y\_test, strings))

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

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

print("Accuracy:")

print(accuracy\_score(y\_test, strings))

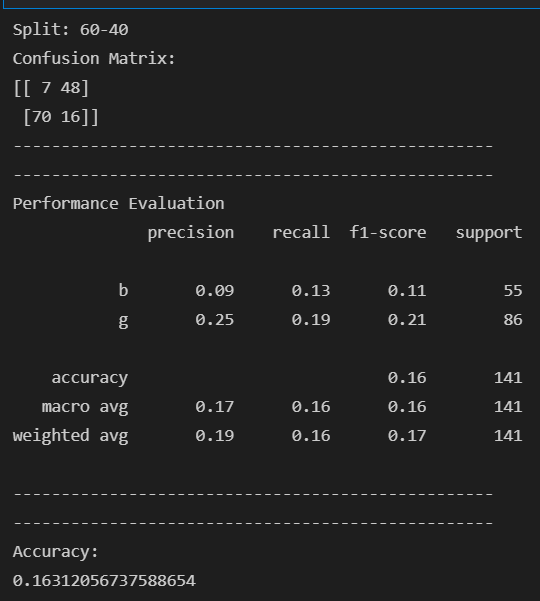
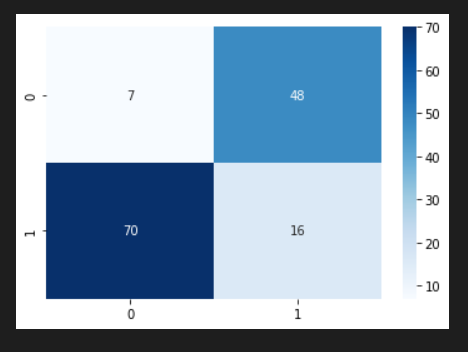
import matplotlib.pyplot as plt

import seaborn as sns

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

sns.heatmap(cm, annot=True, fmt="d",cmap='Blues')

plt.show()

# IONOSPHERE DATASET

# GMMHMM(With Tuning)[60-40 split]

print("Split size: 60-40")

import pandas as pd

import numpy as np

# Dataset Preparation

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(['1','2','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=0.6,test\_size=0.4,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 hmmlearn import hmm

import hmmlearn

classifier = hmmlearn.hmm.GMMHMM(n\_components=2, random\_state=10,covariance\_type='full',algorithm='viterbi',n\_iter=10)

classifier.fit(X\_train)

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

size = len(y\_pred)

strings = np.empty(size, np.unicode\_)

for i in range (size):

if y\_pred[i] == 1:

strings[i] = ("g")

else:

strings[i] = ("b")

strings

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

print("Confusion Matrix:")

print(confusion\_matrix(y\_test, strings))

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

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

print("Performance Evaluation")

print(classification\_report(y\_test, strings))

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

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

print("Accuracy:")

print(accuracy\_score(y\_test, strings))

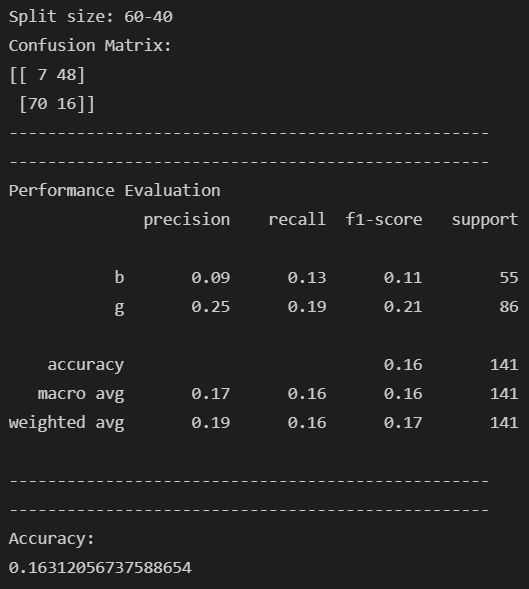
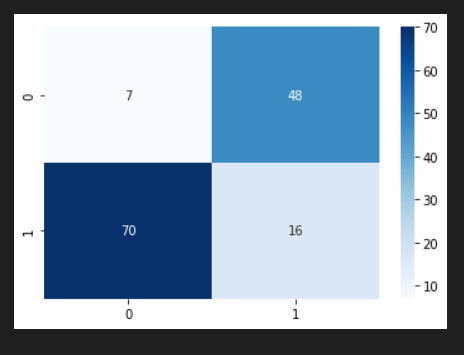
import matplotlib.pyplot as plt

import seaborn as sns

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

sns.heatmap(cm, annot=True, fmt="d",cmap='Blues')

plt.show()

# IONOSPHERE DATASET

# MultinomialHMM(Without Tuning)[40-60 split]

print("Split: 40-60")

import pandas as pd

import numpy as np

# Dataset Preparation

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(['1','2','Class'], axis=1)

y = df['Class']

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=0.4,test\_size=0.6,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 hmmlearn import hmm

import hmmlearn

classifier = hmmlearn.hmm.MultinomialHMM(n\_components=4, random\_state=15,n\_iter=10,algorithm='viterbi',params='ste')

import math

row = len(X\_train)

col = len(X\_train[0])

new = [1] \* 33

for i in range(row):

for j in range(col):

X\_train[i][j] = X\_train[i][j]\*10

X\_train[i][j] = math.floor(X\_train[i][j])

x = X\_train[i].astype(int)

new = np.vstack([new,x])

y = new

y = np.absolute(y)

X\_train = y

import math

row = len(X\_test)

col = len(X\_test[0])

new

for i in range(row):

for j in range(col):

X\_test[i][j] = X\_test[i][j]\*10

X\_test[i][j] = math.floor(X\_test[i][j])

x = X\_test[i].astype(int)

new = np.vstack([new,x])

y = new

y = np.absolute(y)

X\_test = y

classifier.fit(X\_train)

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

size = len(y\_pred)

strings = np.empty(size, np.unicode\_)

for i in range (size):

if y\_pred[i] == 1:

strings[i] = ("b")

else:

strings[i] = ("g")

strings

strings = strings[0:211]

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

print("Confusion Matrix:")

print(confusion\_matrix(y\_test, strings))

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

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

print("Performance Evaluation")

print(classification\_report(y\_test, strings))

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

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

print("Accuracy:")

print(accuracy\_score(y\_test, strings))

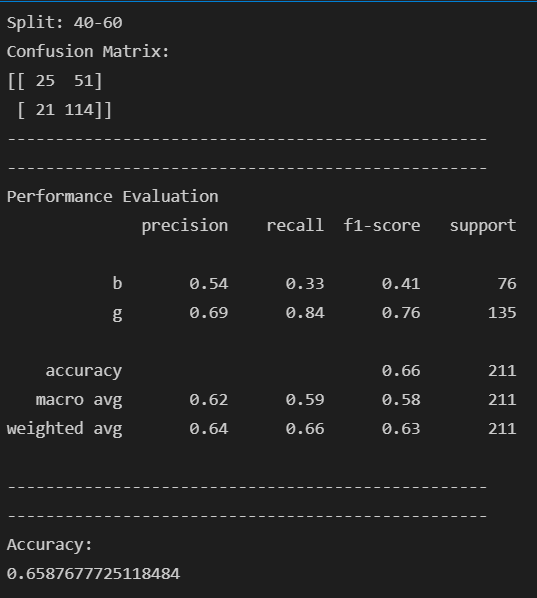
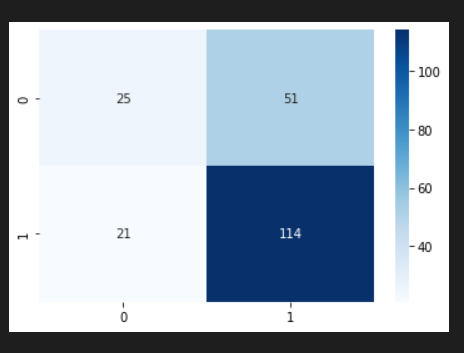
import matplotlib.pyplot as plt

import seaborn as sns

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

sns.heatmap(cm, annot=True, fmt="d",cmap='Blues')

plt.show()

# Irish DATASET

# GaussianHMM(Without Tuning)[30-70 split]

print("Split: 30-70")

import pandas as pd

import numpy as np

import sklearn

from sklearn import datasets

iris = datasets.load\_iris()

X=iris.data

y=iris.target

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,train\_size=0.3,test\_size=0.7,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 hmmlearn import hmm

classifier = hmm.GaussianHMM(n\_components=2, covariance\_type="full")

classifier.fit(X\_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))

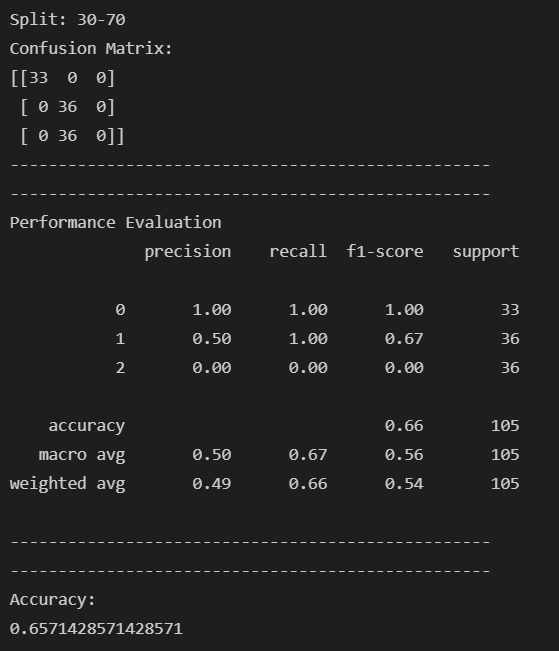
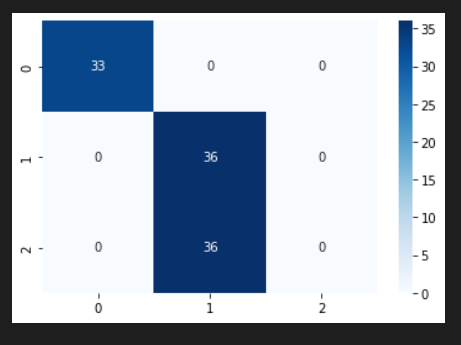
import matplotlib.pyplot as plt

import seaborn as sns

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

sns.heatmap(cm, annot=True, fmt="d",cmap='Blues')

plt.show()

# Irish DATASET

# GaussianHMM(With Tuning)[70-30 split]

print("Split: 70-30")

import pandas as pd

import numpy as np

import sklearn

from sklearn import datasets

iris = datasets.load\_iris()

X=iris.data

y=iris.target

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=0.3,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 hmmlearn import hmm

classifier = hmm.GaussianHMM(n\_components=2, covariance\_type="full",n\_iter=5,algorithm='viterbi',verbose=False)

classifier.fit(X\_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))

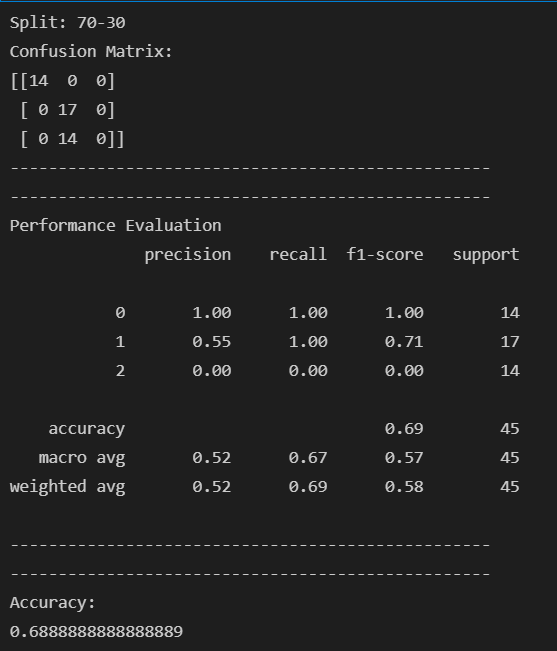
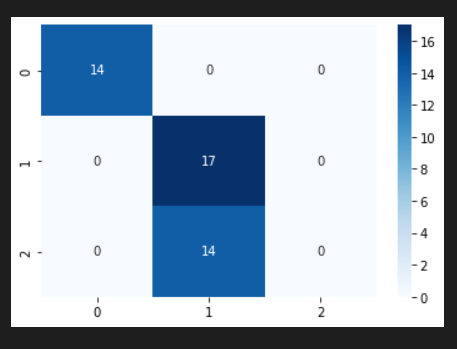
import matplotlib.pyplot as plt

import seaborn as sns

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

sns.heatmap(cm, annot=True, fmt="d",cmap='Blues')

plt.show()

# Irish DATASET

# GMMHMM(With Tuning)[30-70 split]

print("Split: 30-70")

import pandas as pd

import numpy as np

import sklearn

from sklearn import datasets

iris = datasets.load\_iris()

X=iris.data

y=iris.target

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,train\_size=0.3,test\_size=0.7,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

import hmmlearn

classifier = hmmlearn.hmm.GMMHMM(n\_components=2, random\_state=10,covariance\_type='full',algorithm='viterbi',n\_iter=10)

classifier.fit(X\_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))

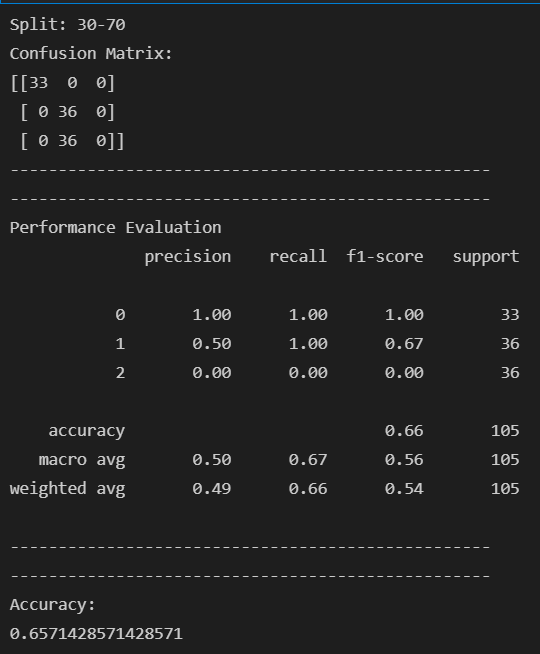
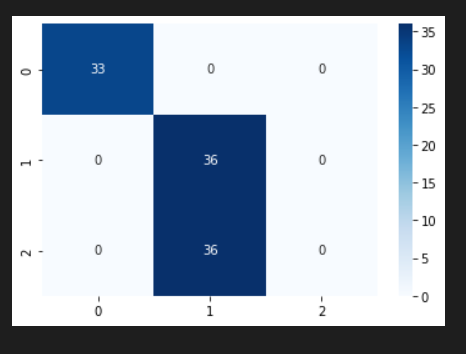
import matplotlib.pyplot as plt

import seaborn as sns

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

sns.heatmap(cm, annot=True, fmt="d",cmap='Blues')

plt.show()

Q5. Used Dataset

Wine Dataset: <https://archive.ics.uci.edu/ml/datasets/wine>

Kmeans vs Kmedoids

#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.cluster import KMeans

from sklearn.datasets import load\_wine

wine=load\_wine() #loading iris dataset from sklearn.datasets

x=wine.data

df=pd.DataFrame(data=wine.data, columns=['alcohol',

'malic\_acid',

'ash',

'alcalinity\_of\_ash',

'magnesium',

'total\_phenols',

'flavanoids',

'nonflavanoid\_phenols',

'proanthocyanins',

'color\_intensity',

'hue',

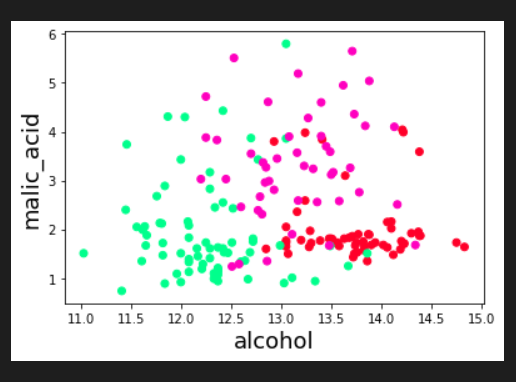
'od280/od315\_of\_diluted\_wines',

'proline'])

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=wine.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('alcohol', fontsize=18)

plt.ylabel('malic\_acid', fontsize=18)



kmeans = KMeans(init="random", n\_clusters=3, n\_init=10, max\_iter=300, random\_state=42)

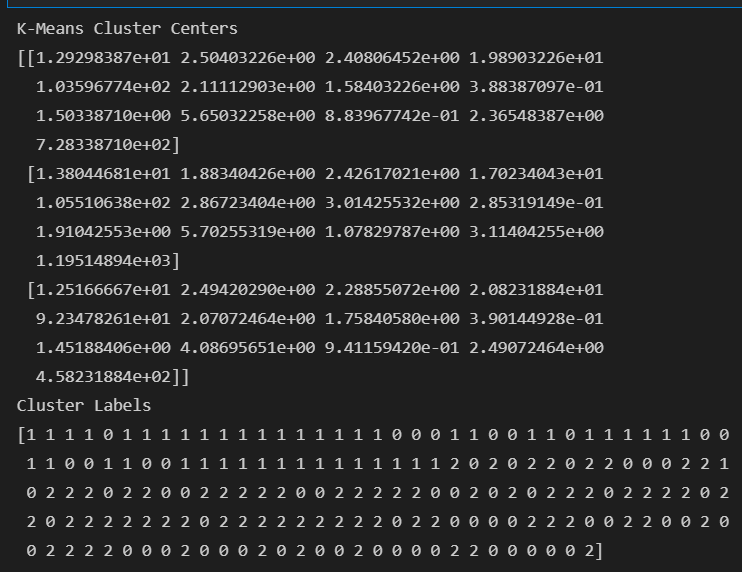
y = kmeans.fit\_predict(x)

print("K-Means Cluster Centers")

print(kmeans.cluster\_centers\_)

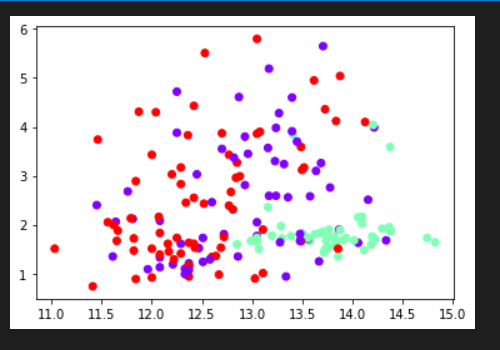
print("Cluster Labels")

print(kmeans.labels\_)



plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=kmeans.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()



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)

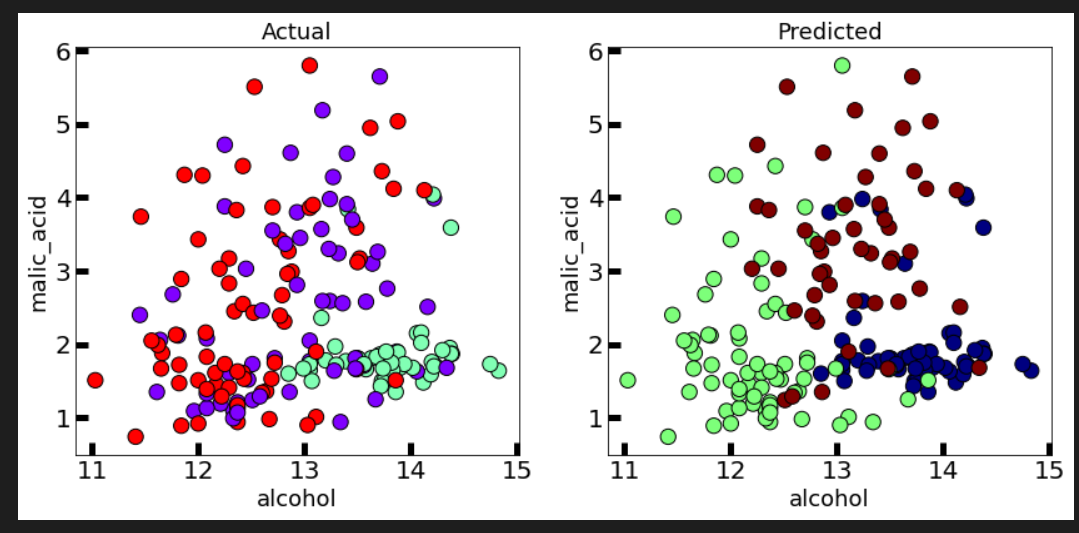
axes[1].set\_ylabel('malic\_acid', fontsize=18)

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)

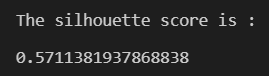
axes[1].set\_title('Predicted', fontsize=18)



from sklearn.metrics import silhouette\_score

print("The silhouette score is :")

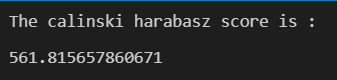
silhouette\_score(x, kmeans.labels\_)



from sklearn.metrics import calinski\_harabasz\_score

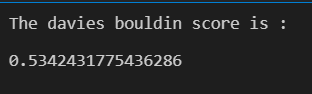
print("The calinski harabasz score is :")

calinski\_harabasz\_score(x, kmeans.labels\_)



from sklearn.metrics import davies\_bouldin\_score

print("The davies bouldin score is :")

davies\_bouldin\_score(x, kmeans.labels\_) 

#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn\_extra.cluster import KMedoids

from sklearn.datasets import load\_wine

wine=load\_wine() #loading iris dataset from sklearn.datasets

x=wine.data

df=pd.DataFrame(data=wine.data, columns=['alcohol',

'malic\_acid',

'ash',

'alcalinity\_of\_ash',

'magnesium',

'total\_phenols',

'flavanoids',

'nonflavanoid\_phenols',

'proanthocyanins',

'color\_intensity',

'hue',

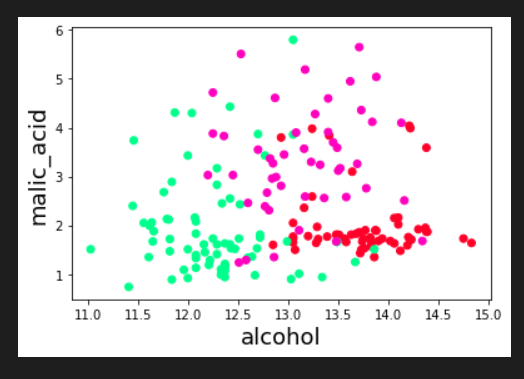
'od280/od315\_of\_diluted\_wines',

'proline'])

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=wine.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('alcohol', fontsize=18)

plt.ylabel('malic\_acid', fontsize=18)



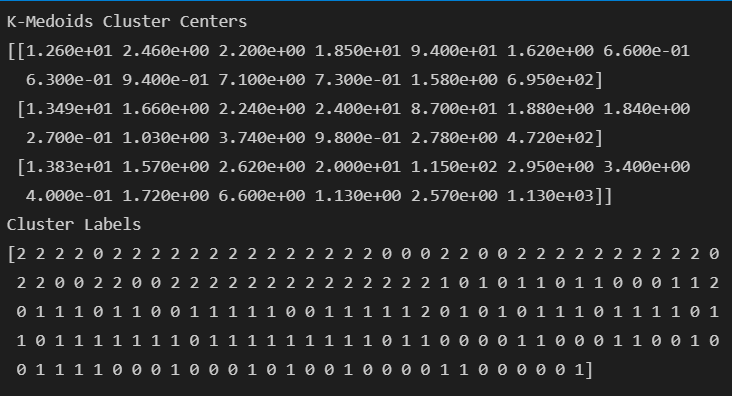
kmedoid = KMedoids(init="heuristic", n\_clusters=3, max\_iter=300, random\_state=42)

y = kmedoid.fit\_predict(x)

print("K-Medoids Cluster Centers")

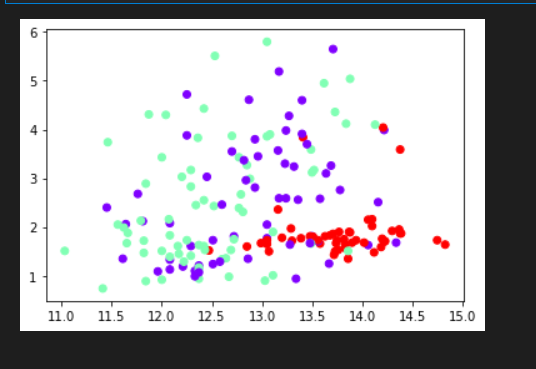
print(kmedoid.cluster\_centers\_)

print("Cluster Labels")

print(kmedoid.labels\_) 

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=kmedoid.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()



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)

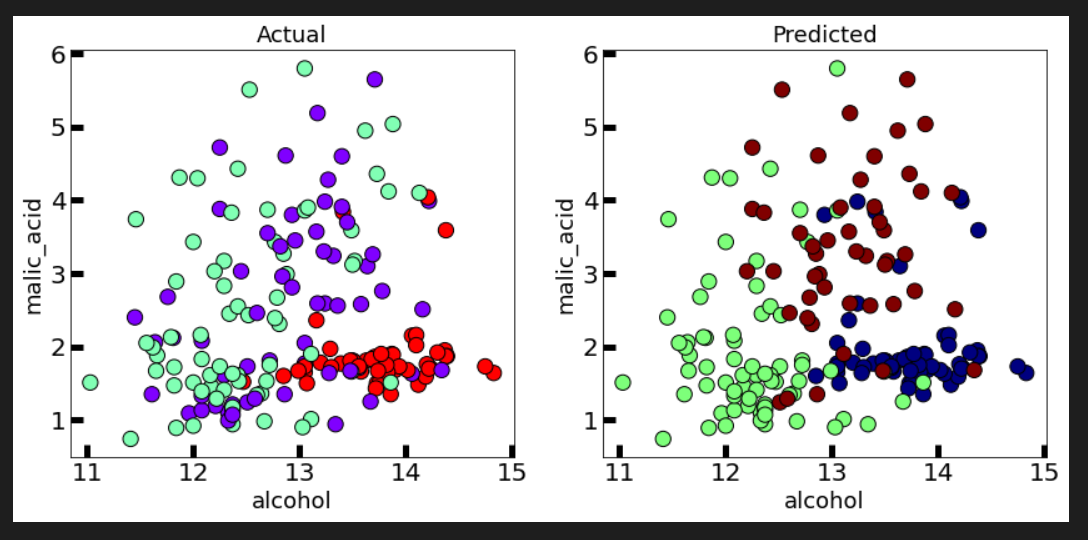
axes[1].set\_ylabel('malic\_acid', fontsize=18)

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)

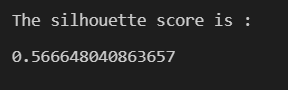
axes[1].set\_title('Predicted', fontsize=18)



from sklearn.metrics import silhouette\_score

print("The silhouette score is :")

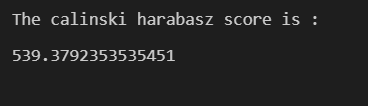
silhouette\_score(x, kmedoid.labels\_)



from sklearn.metrics import calinski\_harabasz\_score

print("The calinski harabasz score is :")

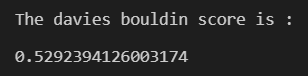
calinski\_harabasz\_score(x, kmedoid.labels\_)



from sklearn.metrics import davies\_bouldin\_score

print("The davies bouldin score is :")

davies\_bouldin\_score(x, kmedoid.labels\_)



#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.cluster import KMeans

from sklearn.datasets import load\_wine

wine=load\_wine()   #loading iris dataset from sklearn.datasets

x=wine.data

df=pd.DataFrame(data=wine.data, columns=['alcohol',

  'malic\_acid',

  'ash',

  'alcalinity\_of\_ash',

  'magnesium',

  'total\_phenols',

  'flavanoids',

  'nonflavanoid\_phenols',

  'proanthocyanins',

  'color\_intensity',

  'hue',

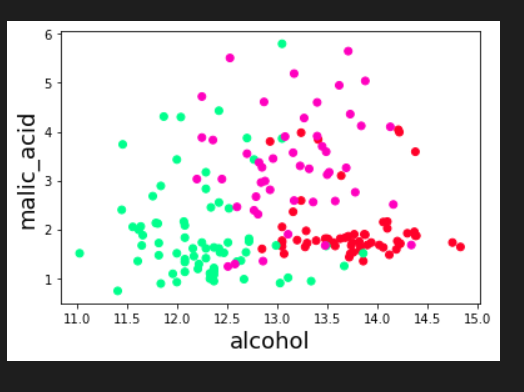
  'od280/od315\_of\_diluted\_wines',

  'proline'])

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=wine.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('alcohol', fontsize=18)

plt.ylabel('malic\_acid', fontsize=18)



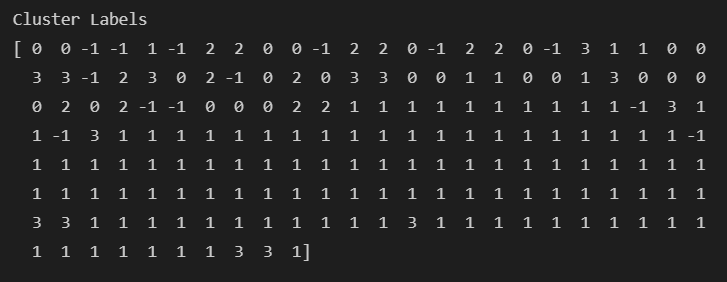
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=35, algorithm='auto', metric='euclidean')

y = dbscan.fit\_predict(x)

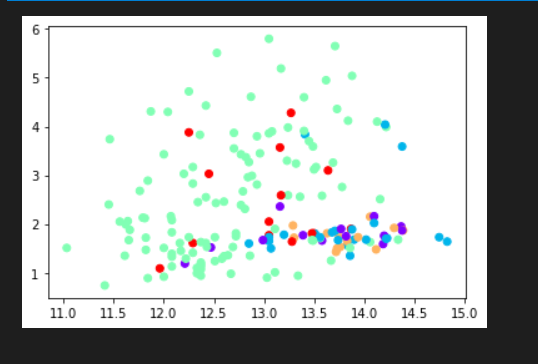
print("Cluster Labels")

print(dbscan.labels\_)



plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=dbscan.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()



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)

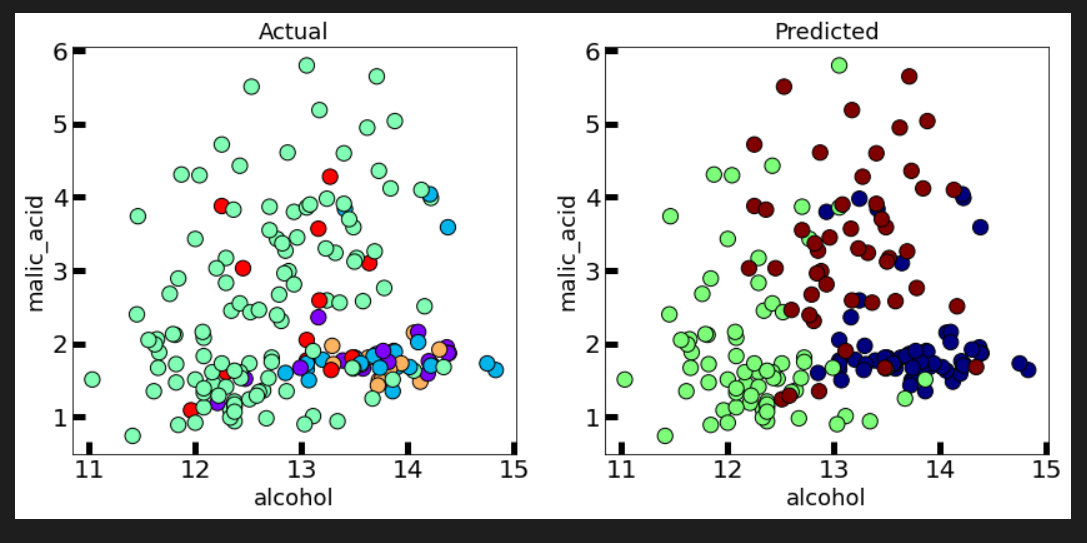
axes[1].set\_ylabel('malic\_acid', fontsize=18)

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)

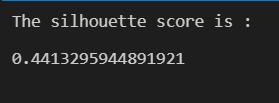
axes[1].set\_title('Predicted', fontsize=18)



from sklearn.metrics import silhouette\_score

print("The silhouette score is :")

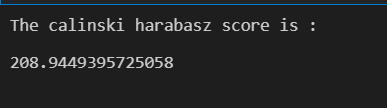
silhouette\_score(x, dbscan.labels\_)



from sklearn.metrics import calinski\_harabasz\_score

print("The calinski harabasz score is :")

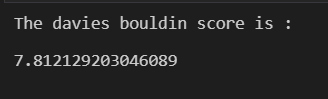
calinski\_harabasz\_score(x, dbscan.labels\_)



from sklearn.metrics import davies\_bouldin\_score

print("The davies bouldin score is :")

davies\_bouldin\_score(x, dbscan.labels\_)



#importing libraries

import numpy as np

import pandas as pd

import sklearn as sk

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn.cluster import KMeans

from sklearn.datasets import load\_wine

wine=load\_wine()   #loading iris dataset from sklearn.datasets

x=wine.data

df=pd.DataFrame(data=wine.data, columns=['alcohol',

  'malic\_acid',

  'ash',

  'alcalinity\_of\_ash',

  'magnesium',

  'total\_phenols',

  'flavanoids',

  'nonflavanoid\_phenols',

  'proanthocyanins',

  'color\_intensity',

  'hue',

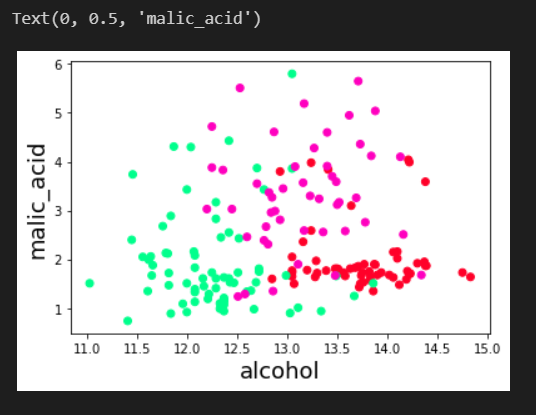
  'od280/od315\_of\_diluted\_wines',

  'proline'])

plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=wine.target, cmap='gist\_rainbow') #try using cmap='rainbow'

plt.xlabel('alcohol', fontsize=18)

plt.ylabel('malic\_acid', fontsize=18)



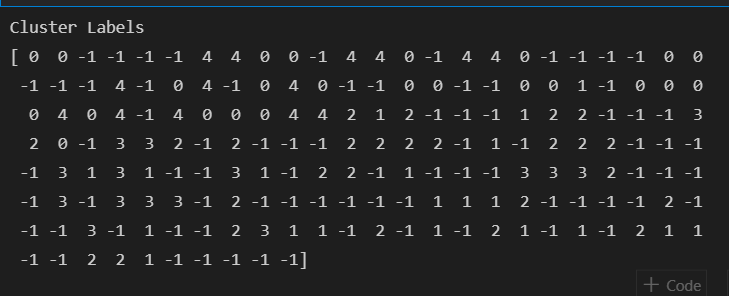
from sklearn.cluster import OPTICS

optics = OPTICS(min\_samples=13)

y = optics.fit\_predict(x)

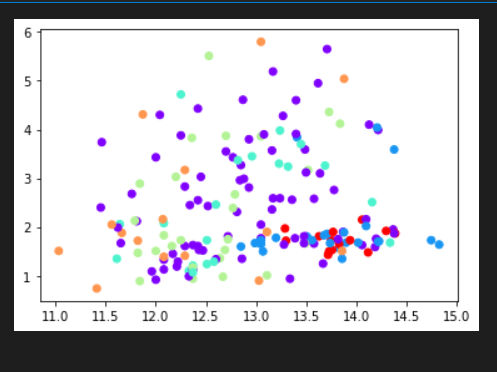
print("Cluster Labels")

print(optics.labels\_)



plt.scatter(x=df['alcohol'], y=df['malic\_acid'] ,c=optics.labels\_, cmap='rainbow') #try using cmap='rainbow'

plt.show()



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)

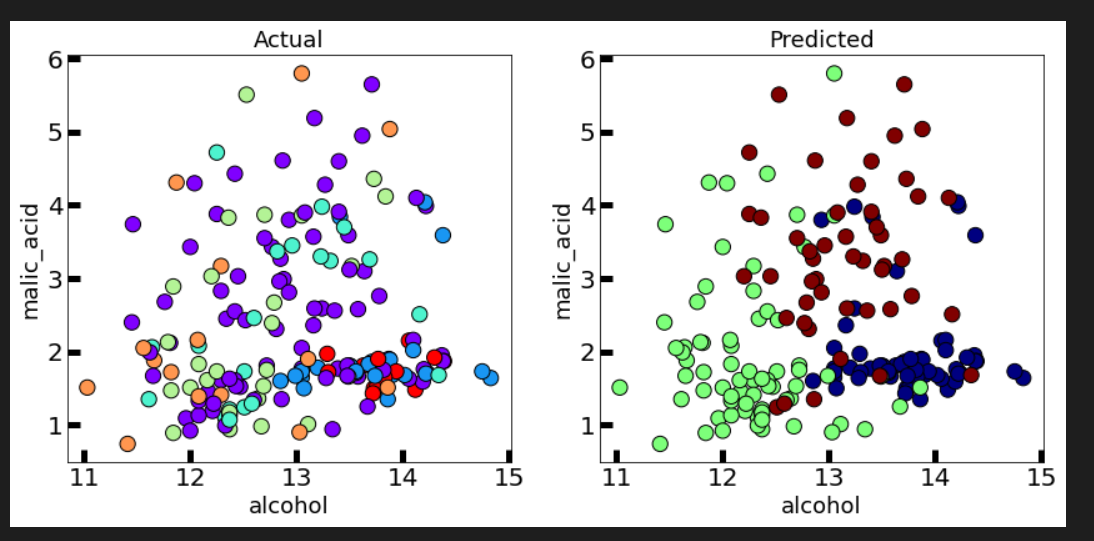
axes[1].set\_ylabel('malic\_acid', fontsize=18)

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)

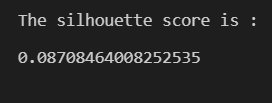
axes[1].set\_title('Predicted', fontsize=18)



from sklearn.metrics import silhouette\_score

print("The silhouette score is :")

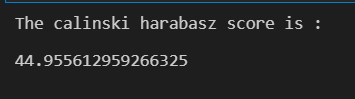
silhouette\_score(x, optics.labels\_)



from sklearn.metrics import calinski\_harabasz\_score

print("The calinski harabasz score is :")

calinski\_harabasz\_score(x, optics.labels\_)



from sklearn.metrics import davies\_bouldin\_score

print("The davies bouldin score is :")

davies\_bouldin\_score(x, optics.labels\_)

