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

Solutions::

Q1. Datasets used :

a. Wine Dataset: https://archive.ics.uci.edu/ml/datasets/wine
b. Ionosphere Dataset: https://archive.ics.uci.edu/ml/datasets/Ionosphere

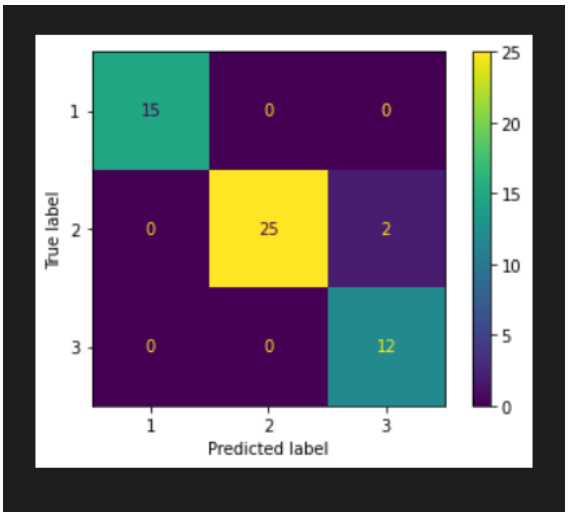
Q1. Answers::

```
# 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()
```

```
Confusion Matrix:
[[15  0  0]
 [ 0 25  2]
 [ 0  0 12]]
-------------------------------------------------
-------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        15
           2       1.00      0.93      0.96        27
           3       0.86      1.00      0.92        12

    accuracy                           0.96        54
   macro avg       0.95      0.98      0.96        54
weighted avg       0.97      0.96      0.96        54

-------------------------------------------------
-------------------------------------------------
Accuracy:
0.9629629629629629
```



WINE DATASET

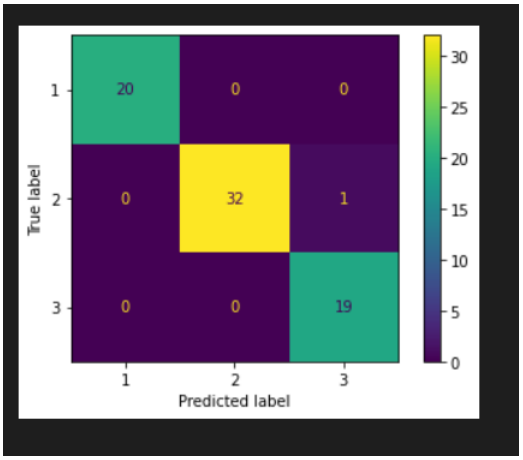
SVM(Without Tuning)[60-40 split]

```
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()
```

```
Confusion Matrix:
[[20  0  0]
 [ 0 32  1]
 [ 0  0 19]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        20
           2       1.00      0.97      0.98        33
           3       0.95      1.00      0.97        19

    accuracy                           0.99        72
   macro avg       0.98      0.99      0.99        72
weighted avg       0.99      0.99      0.99        72

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9861111111111112
```

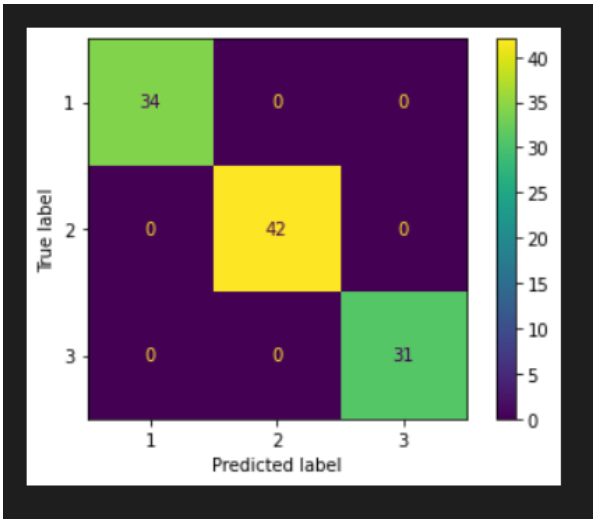



```
# WINE DATASET

# 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()
```

```
Confusion Matrix:
[[34  0  0]
 [ 0 42  0]
 [ 0  0 31]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        34
           2       1.00      1.00      1.00        42
           3       1.00      1.00      1.00        31

    accuracy                           1.00       107
   macro avg       1.00      1.00      1.00       107
weighted avg       1.00      1.00      1.00       107

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



WINE DATASET

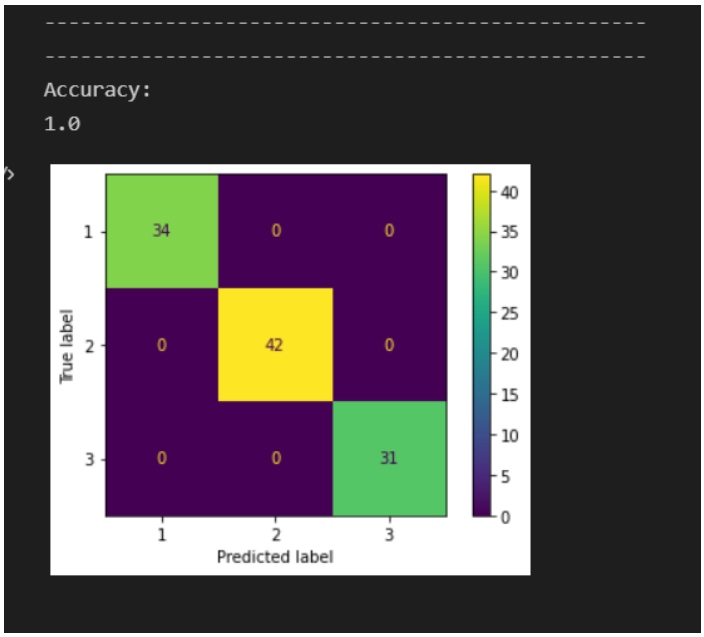
SVM(With Tuning)[40-60 split]

```
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()
```

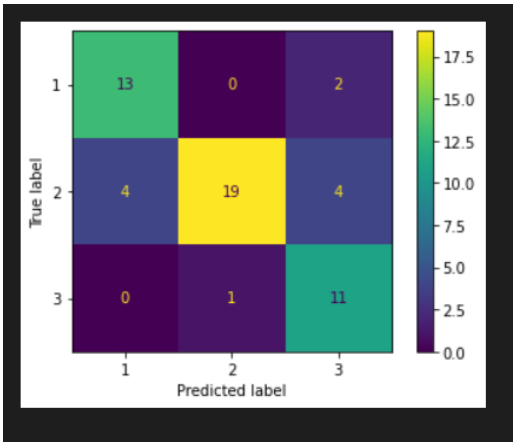



```
# WINE DATASET
# Decision Tree (Without Tuning)[70-30 split]


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)
```

```
Confusion Matrix:
[[13  0  2]
 [ 4 19  4]
 [ 0  1 11]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       0.76      0.87      0.81        15
           2       0.95      0.70      0.81        27
           3       0.65      0.92      0.76        12

    accuracy                           0.80        54
   macro avg       0.79      0.83      0.79        54
weighted avg       0.83      0.80      0.80        54

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.7962962962962963
```




```
|--- feature_12 <= 0.06
|   |--- feature_9 <= -0.47
|   |   |--- class: 2
|   |--- feature_9 >  -0.47
|   |   |--- feature_10 <= 0.17
|   |   |   |--- class: 3
|   |   |--- feature_10 >  0.17
|   |   |   |--- feature_11 <= 0.38
|   |   |   |   |--- class: 2
|   |   |   |--- feature_11 >  0.38
|   |   |   |   |--- class: 1
|--- feature_12 >  0.06
|   |--- feature_6 <= 0.16
|   |   |--- feature_7 <= -0.76
|   |   |   |--- class: 2
|   |   |--- feature_7 >  -0.76
|   |   |   |--- class: 3
|   |--- feature_6 >  0.16
|   |   |--- feature_3 <= 2.32
|   |   |   |--- class: 1
|   |   |--- feature_3 >  2.32
|   |   |   |--- class: 2
```

WINE DATASET

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

import pandas as pd

import numpy as np

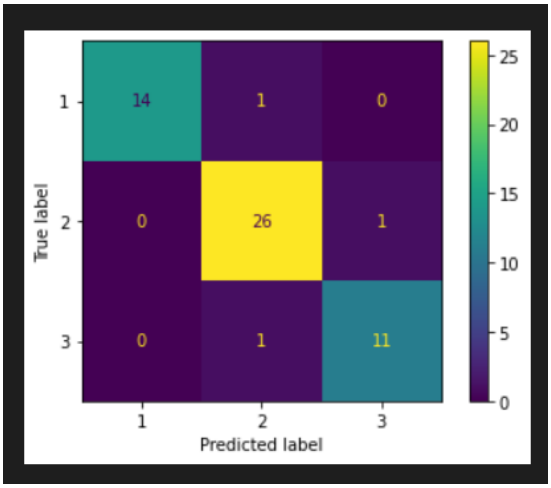
Dataset Preparation

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)
```

```
Confusion Matrix:
[[14  1  0]
 [ 0 26  1]
 [ 0  1 11]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       1.00      0.93      0.97        15
           2       0.93      0.96      0.95        27
           3       0.92      0.92      0.92        12

    accuracy                           0.94        54
   macro avg       0.95      0.94      0.94        54
weighted avg       0.95      0.94      0.94        54

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9444444444444444
```




```
|--- feature_6 <= -0.60
|   |--- feature_9 <= -0.61
|   |   |--- class: 2
|   |--- feature_9 >  -0.61
|   |   |--- class: 3
|--- feature_6 >  -0.60
|   |--- feature_12 <= -0.14
|   |   |--- class: 2
|   |--- feature_12 >  -0.14
|   |   |--- feature_9 <= -0.71
|   |   |   |--- class: 2
|   |   |--- feature_9 >  -0.71
|   |   |   |--- class: 1
```

WINE DATASET

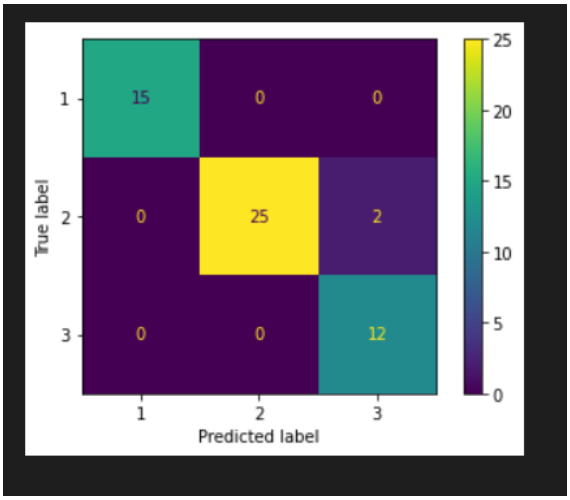
Random Forest Classifier(Without Tuning)[70-30 split]

```
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()
```

```
Confusion Matrix:
[[15  0  0]
 [ 0 25  2]
 [ 0  0 12]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        15
           2       1.00      0.93      0.96        27
           3       0.86      1.00      0.92        12

    accuracy                           0.96        54
   macro avg       0.95      0.98      0.96        54
weighted avg       0.97      0.96      0.96        54

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9629629629629629
```



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
```

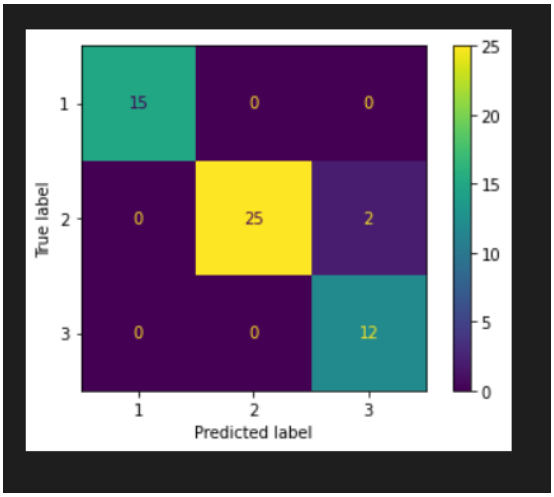
```
X = df.drop(['Class'], axis=1)
y = df['Class']

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()
```

```
{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
show more (open the raw output data in a text editor) ...

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9629629629629629
```



WINE DATASET

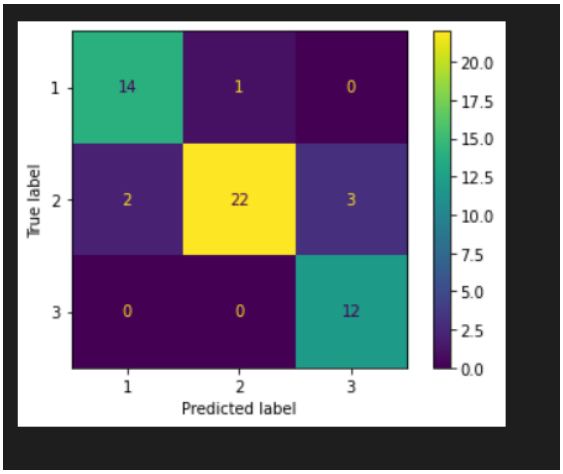
Gaussian Naive Bayes (Without Tuning)[70-30 split]

```
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()
```

```
Confusion Matrix:
[[14  1  0]
 [ 2 22  3]
 [ 0  0 12]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       0.88      0.93      0.90        15
           2       0.96      0.81      0.88        27
           3       0.80      1.00      0.89        12

    accuracy                           0.89        54
   macro avg       0.88      0.92      0.89        54
weighted avg       0.90      0.89      0.89        54

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.8888888888888888
```



WINE DATASET

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

import pandas as pd

import numpy as np

Dataset Preparation

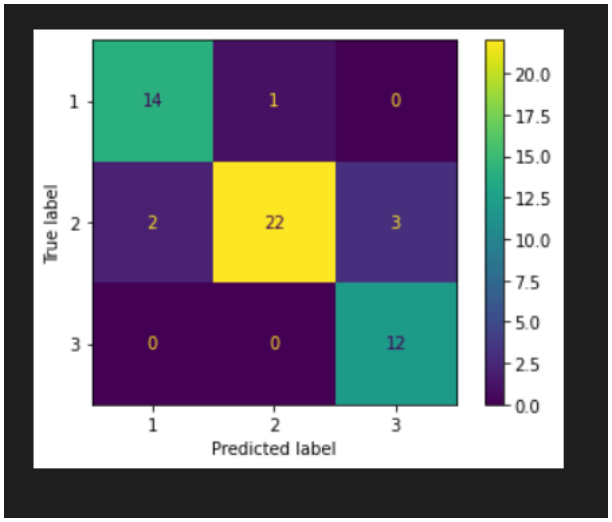
df = pd.read_csv("wine.data",header=None)

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()
```

```
Confusion Matrix:
[[14  1  0]
 [ 2 22  3]
 [ 0  0 12]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       0.88      0.93      0.90        15
           2       0.96      0.81      0.88        27
           3       0.80      1.00      0.89        12

    accuracy                           0.89        54
   macro avg       0.88      0.92      0.89        54
weighted avg       0.90      0.89      0.89        54

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.8888888888888888
```



IONOSPHERE DATASET

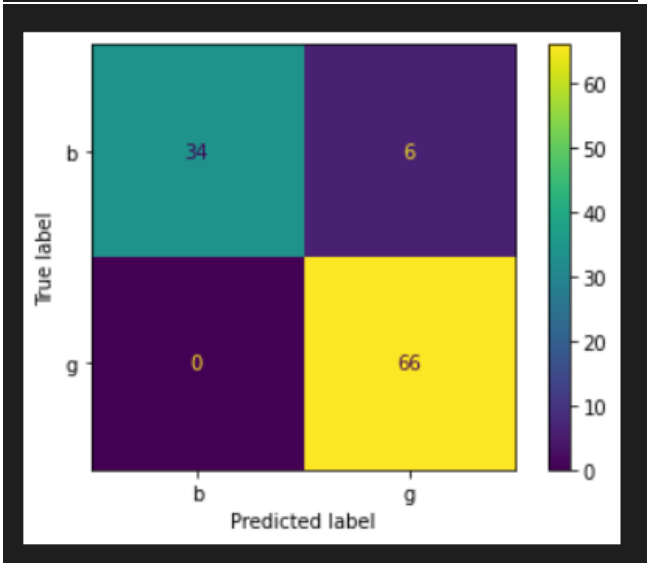
SVM(Without Tuning)[70-30 split]

```
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()
```

```
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
```



IONOSPHERE DATASET

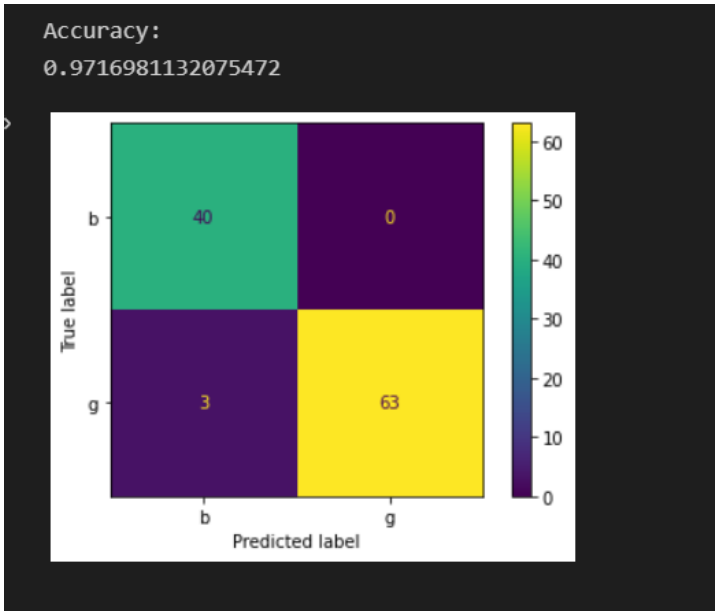
SVM(With Tuning)[70-30 split]

```
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]

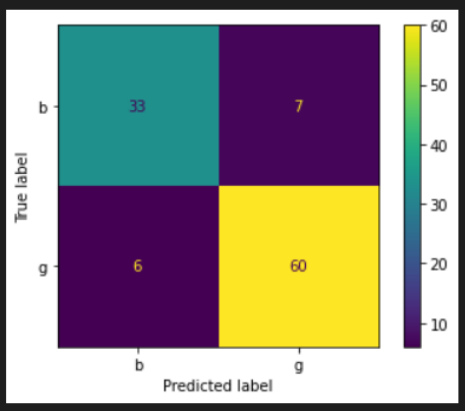
```
import pandas as pd
import numpy as np

# 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)
```

```
Confusion Matrix:
[[33  7]
 [ 6 60]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.85      0.82      0.84        40
           g       0.90      0.91      0.90        66

    accuracy                           0.88       106
   macro avg       0.87      0.87      0.87       106
weighted avg       0.88      0.88      0.88       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.8773584905660378
```

```
Accuracy:
0.8773584905660378
```




```
|--- feature_4 <= -0.70
|   |--- feature_4 <= -1.07
|   |   |--- class: b
|   |--- feature_4 >  -1.07
|   |   |--- feature_2 <= -1.13
|   |   |   |--- class: b
|   |   |--- feature_2 >  -1.13
|   |   |   |--- feature_7 <= 1.52
|   |   |   |   |--- class: g
|   |   |   |--- feature_7 >  1.52
|   |   |   |   |--- class: b
|--- feature_4 >  -0.70
|   |--- feature_26 <= 0.91
|   |   |--- feature_9 <= -1.93
```

Ionosphere DATASET

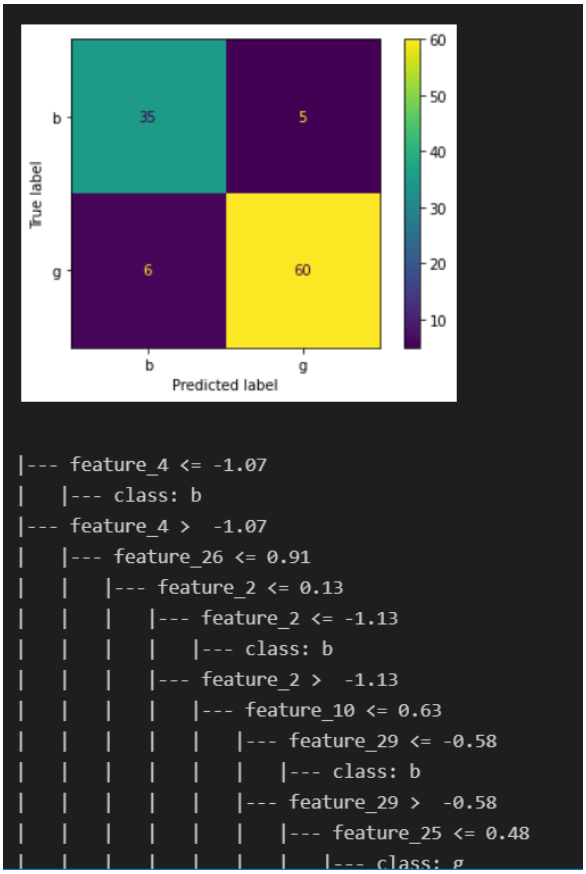
Decision Tree (With Tuning)[70-30 split]

```
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)
```

```
Confusion Matrix:
[[35  5]
 [ 6 60]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.85      0.88      0.86        40
           g       0.92      0.91      0.92        66

    accuracy                           0.90       106
   macro avg       0.89      0.89      0.89       106
weighted avg       0.90      0.90      0.90       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.8962264150943396
```




```
|--- feature_4 <= -1.07
|   |--- class: b
|--- feature_4 >  -1.07
|   |--- feature_26 <= 0.91
|   |   |--- feature_2 <= 0.13
|   |   |   |--- feature_2 <= -1.13
|   |   |   |   |--- class: b
|   |   |   |--- feature_2 >  -1.13
|   |   |   |   |--- feature_10 <= 0.63
|   |   |   |   |   |--- feature_29 <= -0.58
|   |   |   |   |   |   |--- class: b
|   |   |   |   |   |--- feature_29 >  -0.58
|   |   |   |   |   |   |--- feature_25 <= 0.48
|   |   |   |   |   |   |   |--- class: g
```

IONOSPHERE DATASET

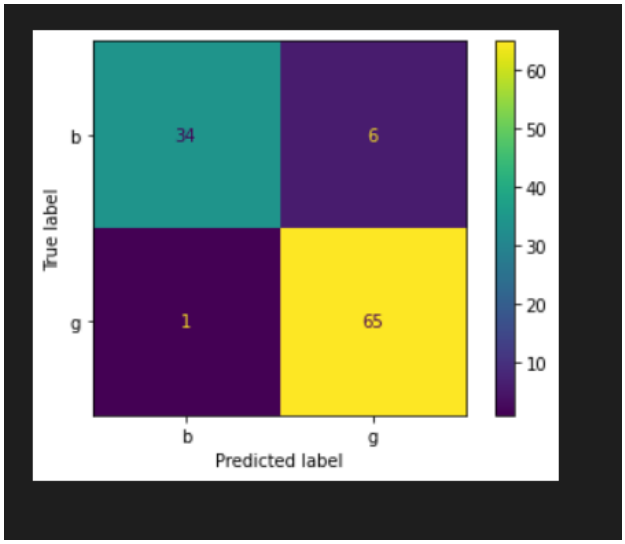
Random Forest Classifier(Without Tuning)[70-30 split]

```
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()
```

```
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
```



IONOSPHERE DATASET

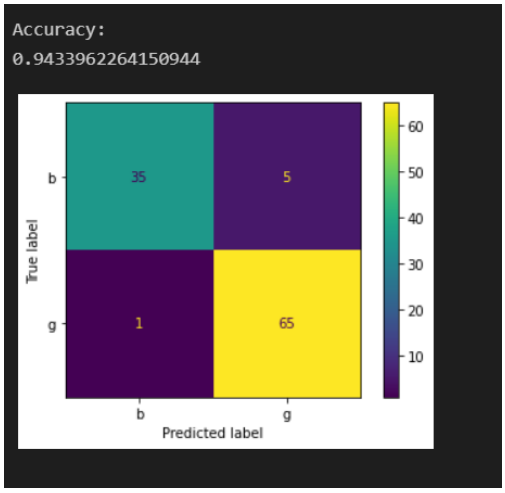
Random Forest Classifier(With Tuning)[70-30 split]

```
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)

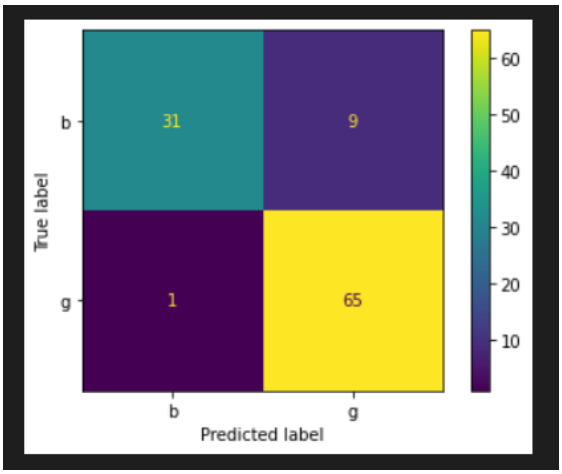
y = df['Class']

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()
```

```
Confusion Matrix:
[[31  9]
 [ 1 65]]
-------------------------------------------------
-------------------------------------------------
Performance Evaluation
              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

-------------------------------------------------
-------------------------------------------------
Accuracy:
0.9056603773584906
```



Q3. Used datasets

a. Iris plants dataset: https://archive.ics.uci.edu/ml/datasets/Iris/
b. Diabetes dataset: https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html

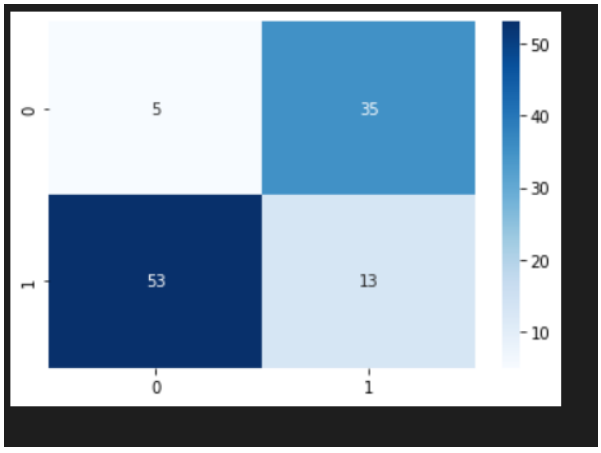
```
# IONOSPHERE DATASET
# GaussianHMM(Without Tuning)[70-30 split]

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()
```

```
Split: 70-30
Confusion Matrix:
[[ 5 35]
 [53 13]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.09      0.12      0.10        40
           g       0.27      0.20      0.23        66

    accuracy                           0.17       106
   macro avg       0.18      0.16      0.17       106
weighted avg       0.20      0.17      0.18       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.16981132075471697
```

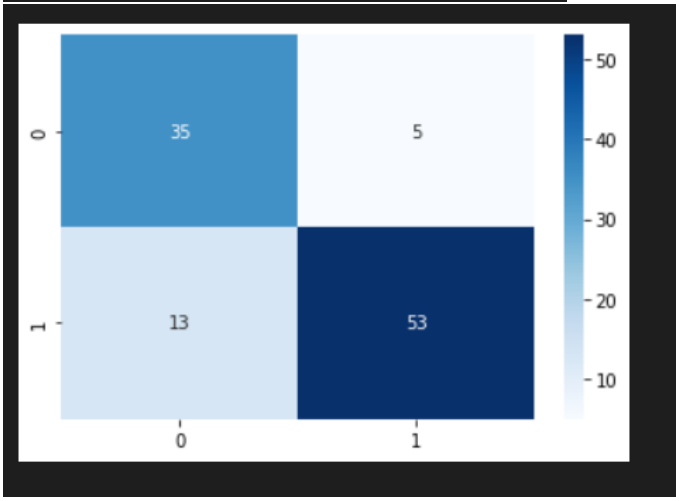


```
# IONOSPHERE DATASET
# GaussianHMM(With Tuning)[70-30 split]


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()
```

```
Confusion Matrix:
[[35  5]
 [13 53]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.73      0.88      0.80        40
           g       0.91      0.80      0.85        66

    accuracy                           0.83       106
   macro avg       0.82      0.84      0.83       106
weighted avg       0.84      0.83      0.83       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.8301886792452831
```



IONOSPHERE DATASET

GMMHMM(Without Tuning)[60-40 split]

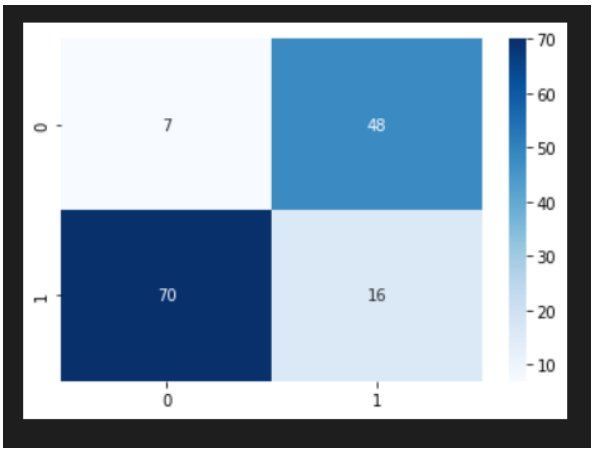
```
print("Split: 60-40")
import pandas as pd
import numpy as np

# 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()
```

```
Split: 60-40
Confusion Matrix:
[[ 7 48]
 [70 16]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.09      0.13      0.11        55
           g       0.25      0.19      0.21        86

    accuracy                           0.16       141
   macro avg       0.17      0.16      0.16       141
weighted avg       0.19      0.16      0.17       141

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.16312056737588654
```



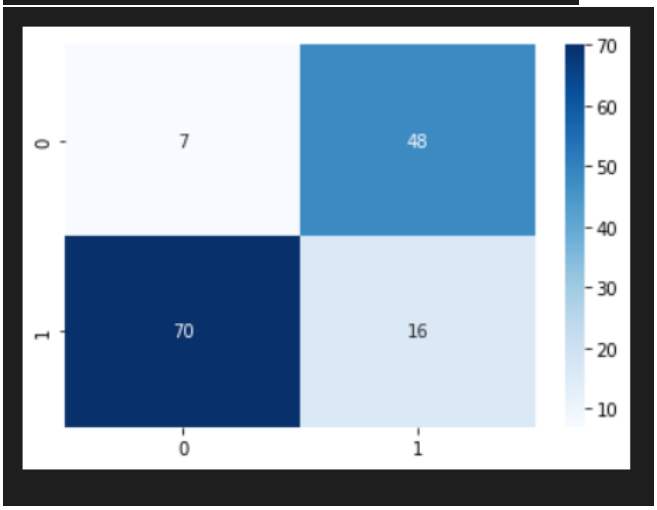

```
# IONOSPHERE DATASET
# GMMHMM(With Tuning)[60-40 split]

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()
```

```
Split size: 60-40
Confusion Matrix:
[[ 7 48]
 [70 16]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.09      0.13      0.11        55
           g       0.25      0.19      0.21        86

    accuracy                           0.16       141
   macro avg       0.17      0.16      0.16       141
weighted avg       0.19      0.16      0.17       141

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.16312056737588654
```




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


print("Split: 40-60")
import pandas as pd
import numpy as np


# Dataset Preparation
```

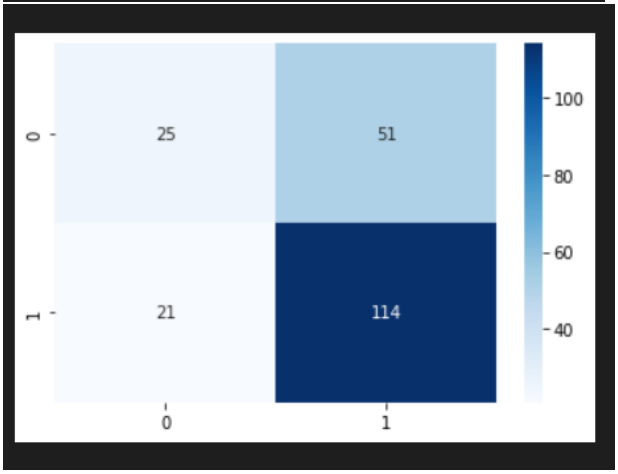
```
col_name = ['1','2','3','4','5','6','7','8','9','10','11','12','13','14','15','16','17','18','19'
        ,'20','21','22','23','24','25','26','27','28','29','30','31','32','33','34','Class']


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()
```

```
Split: 40-60
Confusion Matrix:
[[ 25  51]
 [ 21 114]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.54      0.33      0.41        76
           g       0.69      0.84      0.76       135

    accuracy                           0.66       211
   macro avg       0.62      0.59      0.58       211
weighted avg       0.64      0.66      0.63       211

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.6587677725118484
```



Irish DATASET

GaussianHMM(Without Tuning)[30-70 split]

print("Split: 30-70")

import pandas as pd

import numpy as np

import sklearn

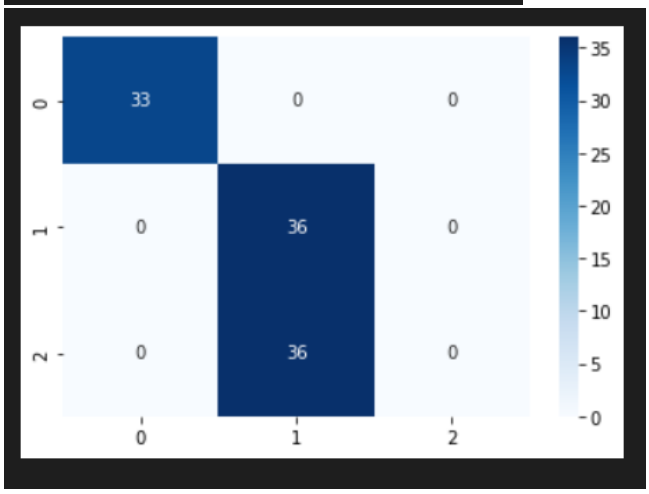
from sklearn import datasets

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()
```

```
Split: 30-70
Confusion Matrix:
[[33  0  0]
 [ 0 36  0]
 [ 0 36  0]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       0.50      1.00      0.67        36
           2       0.00      0.00      0.00        36

    accuracy                           0.66       105
   macro avg       0.50      0.67      0.56       105
weighted avg       0.49      0.66      0.54       105

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.6571428571428571
```



Irish DATASET

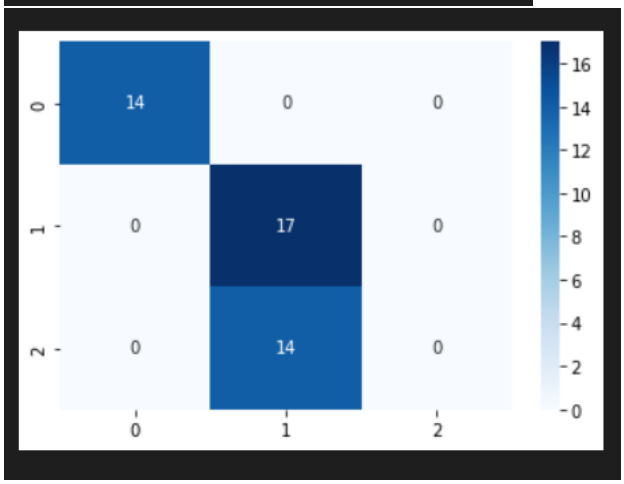
GaussianHMM(With Tuning)[70-30 split]

```
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()
```

```
Split: 70-30
Confusion Matrix:
[[14  0  0]
 [ 0 17  0]
 [ 0 14  0]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       0.55      1.00      0.71        17
           2       0.00      0.00      0.00        14

    accuracy                           0.69        45
   macro avg       0.52      0.67      0.57        45
weighted avg       0.52      0.69      0.58        45

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.6888888888888889
```



Irish DATASET

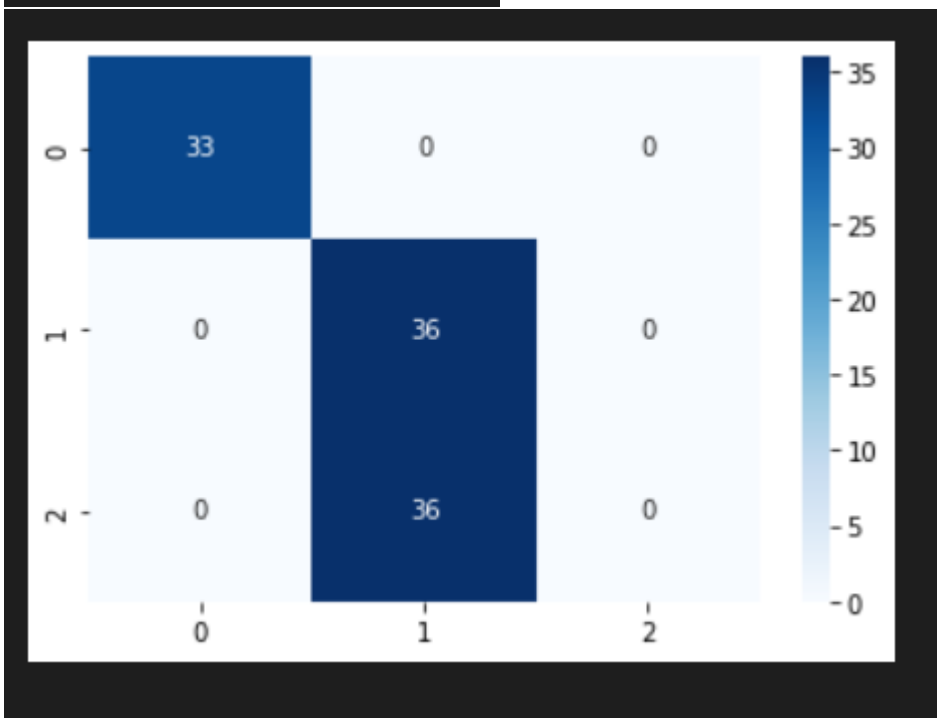
GMMHMM(With Tuning)[30-70 split]

```
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()
```

```
Split: 30-70
Confusion Matrix:
[[33  0  0]
 [ 0 36  0]
 [ 0 36  0]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       0.50      1.00      0.67        36
           2       0.00      0.00      0.00        36

    accuracy                           0.66       105
   macro avg       0.50      0.67      0.56       105
weighted avg       0.49      0.66      0.54       105

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.6571428571428571
```



Q5. Used Dataset

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

Kmeans vs Kmedoids

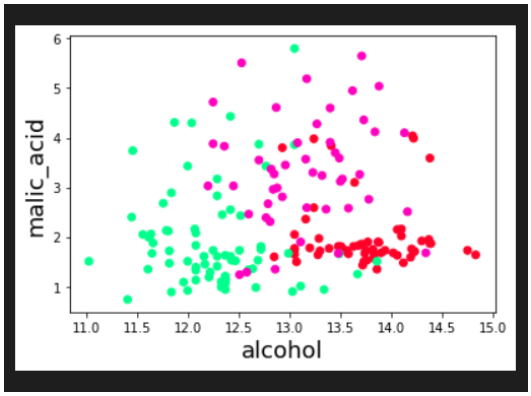
#importing libraries

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_)

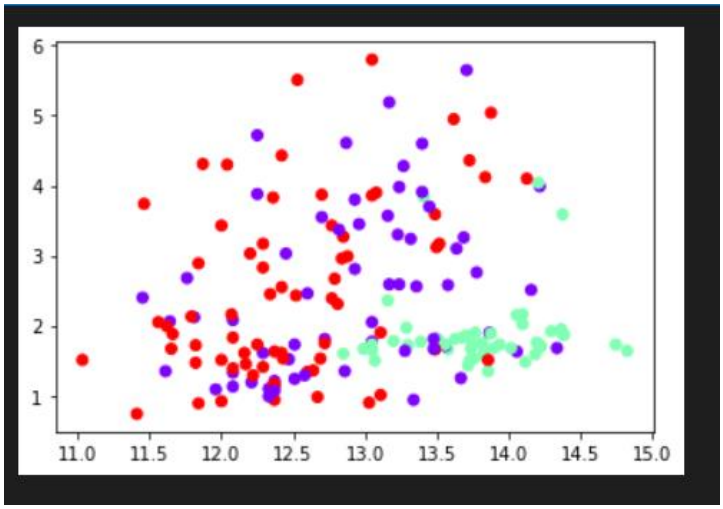
print("Cluster Labels")

print(kmeans.labels_)

```
K-Means Cluster Centers
[[1.29298387e+01 2.50403226e+00 2.40806452e+00 1.98903226e+01
  1.03596774e+02 2.11112903e+00 1.58403226e+00 3.88387097e-01
  1.50338710e+00 5.65032258e+00 8.83967742e-01 2.36548387e+00
  7.28338710e+02]
 [1.38044681e+01 1.88340426e+00 2.42617021e+00 1.70234043e+01
  1.05510638e+02 2.86723404e+00 3.01425532e+00 2.85319149e-01
  1.91042553e+00 5.70255319e+00 1.07829787e+00 3.11404255e+00
  1.19514894e+03]
 [1.25166667e+01 2.49420290e+00 2.28855072e+00 2.08231884e+01
  9.23478261e+01 2.07072464e+00 1.75840580e+00 3.90144928e-01
  1.45188406e+00 4.08695651e+00 9.41159420e-01 2.49072464e+00
  4.58231884e+02]]
Cluster Labels
[1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 1 1 0 1 1 1 1 1 1 0 0
 1 1 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 0 2 0 2 2 0 2 2 0 0 0 2 2 1
 0 2 2 2 0 2 2 0 0 2 2 2 2 2 2 0 0 2 2 2 2 2 0 0 2 0 2 0 2 2 2 0 2 2 2 2 0 2
 2 0 2 2 2 2 2 2 2 0 2 2 2 2 2 2 2 2 2 0 2 2 0 0 0 0 2 2 2 0 0 2 2 0 0 2 0
 0 2 2 2 2 0 0 0 2 0 0 0 2 0 2 0 0 2 0 0 0 0 2 2 0 0 0 0 0 2]
```

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)

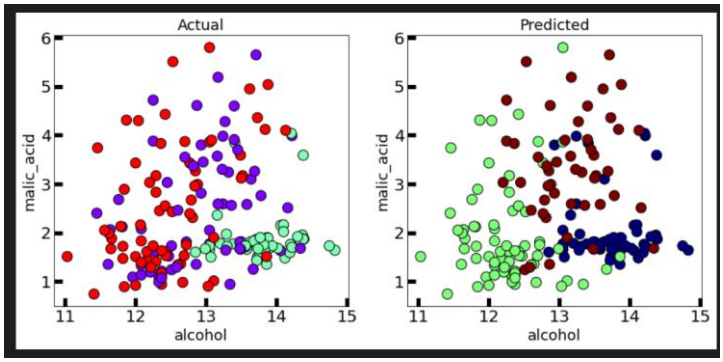
axes[1].set_ylabel('malic_acid', fontsize=18)

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_)
```

```
The silhouette score is :
0.5711381937868838
```

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

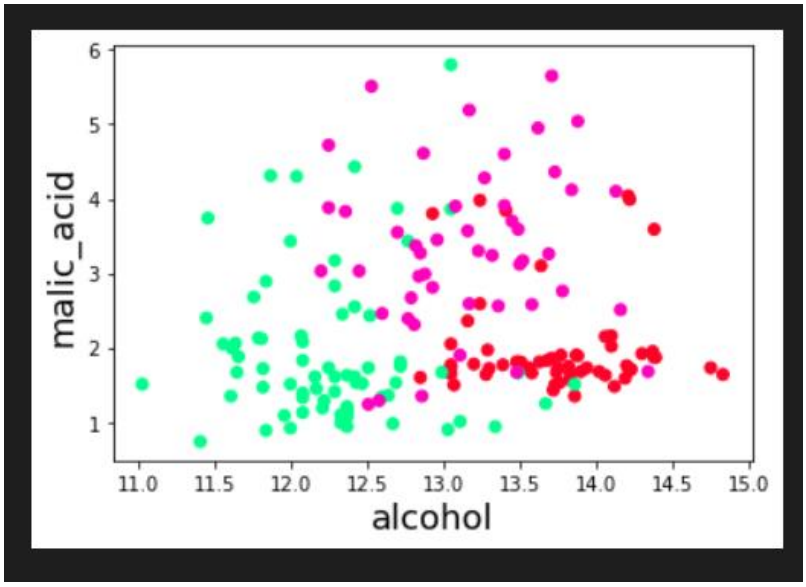
```
The calinski harabasz score is :
561.815657860671
```

```
from sklearn.metrics import davies_bouldin_score
print("The davies bouldin score is :")
davies_bouldin_score(x, kmeans.labels_)
```

```
The davies bouldin score is :
0.5342431775436286
```

```
#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
```

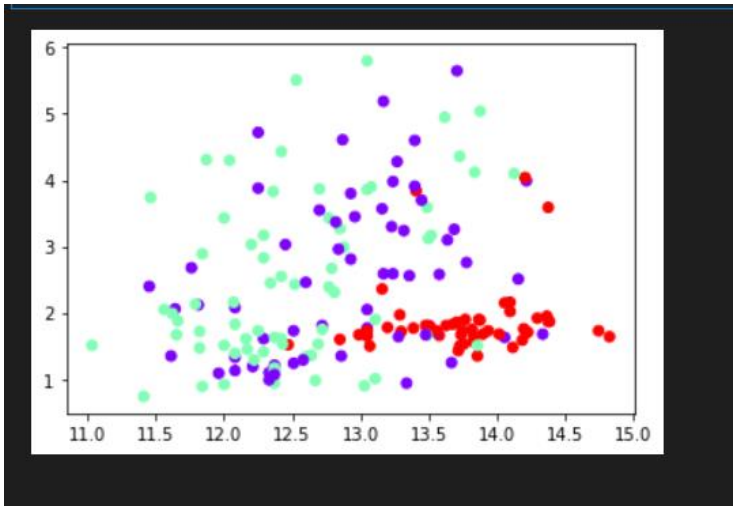
```
x=wine.data
df=pd.DataFrame(data=wine.data, columns=['alcohol',
  'malic_acid',
  'ash',
  'alcalinity_of_ash',
  'magnesium',
  'total_phenols',
  'flavanoids',
  'nonflavanoid_phenols',
  'proanthocyanins',
  'color_intensity',
  'hue',
  'od280/od315_of_diluted_wines',
  'proline'])


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_)
```

```
K-Medoids Cluster Centers
[[1.260e+01 2.460e+00 2.200e+00 1.850e+01 9.400e+01 1.620e+00 6.600e-01
  6.300e-01 9.400e-01 7.100e+00 7.300e-01 1.580e+00 6.950e+02]
 [1.349e+01 1.660e+00 2.240e+00 2.400e+01 8.700e+01 1.880e+00 1.840e+00
  2.700e-01 1.030e+00 3.740e+00 9.800e-01 2.780e+00 4.720e+02]
 [1.383e+01 1.570e+00 2.620e+00 2.000e+01 1.150e+02 2.950e+00 3.400e+00
  4.000e-01 1.720e+00 6.600e+00 1.130e+00 2.570e+00 1.130e+03]]
Cluster Labels
[2 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 2 2 0 0 2 2 2 2 2 2 2 2 2 2 0
 2 2 0 0 2 2 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 1 0 1 0 1 1 0 1 1 0 0 0 1 1 2
 0 1 1 1 0 1 1 0 0 1 1 1 1 1 0 0 1 1 1 1 1 2 0 1 0 1 0 1 1 1 0 1 1 1 1 0 1
 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 0 0 0 1 1 0 0 1 0
 0 1 1 1 1 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 1 1 0 0 0 0 0 1]
```

```
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)

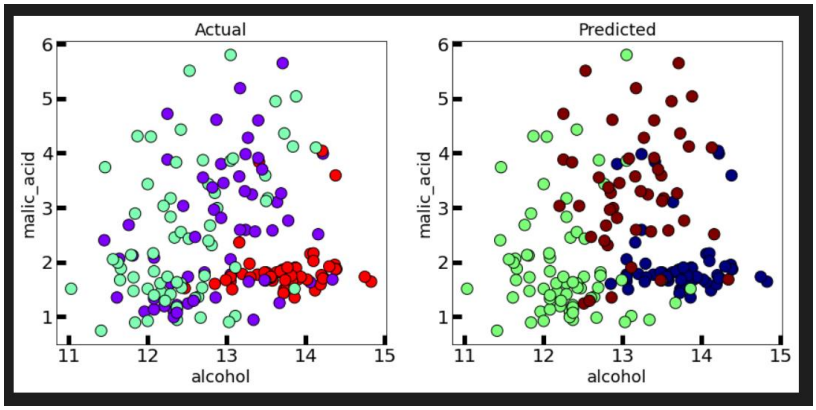
axes[1].set_ylabel('malic_acid', fontsize=18)

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_)
```

```
The silhouette score is :

0.566648040863657
```

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

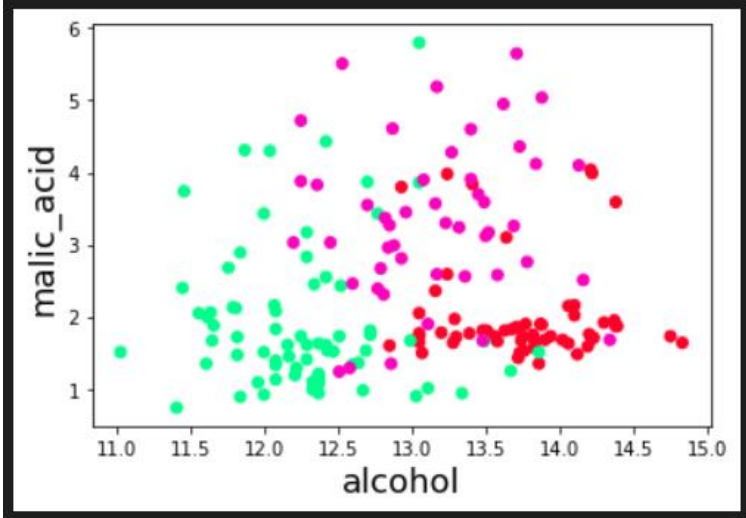
```
The calinski harabasz score is :

539.3792353535451
```

```
from sklearn.metrics import davies_bouldin_score
print("The davies bouldin score is :")
davies_bouldin_score(x, kmedoid.labels_)
```

```
The davies bouldin score is :

0.5292394126003174
```

```
#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',
```

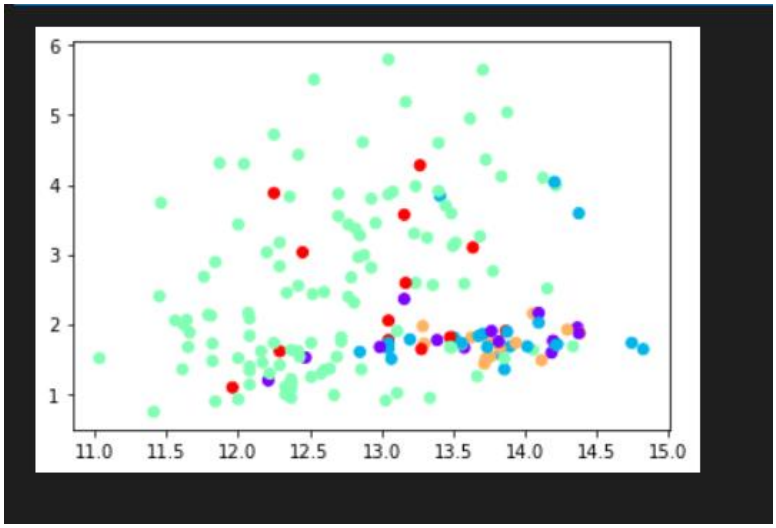
```
  'proline'])


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)
```



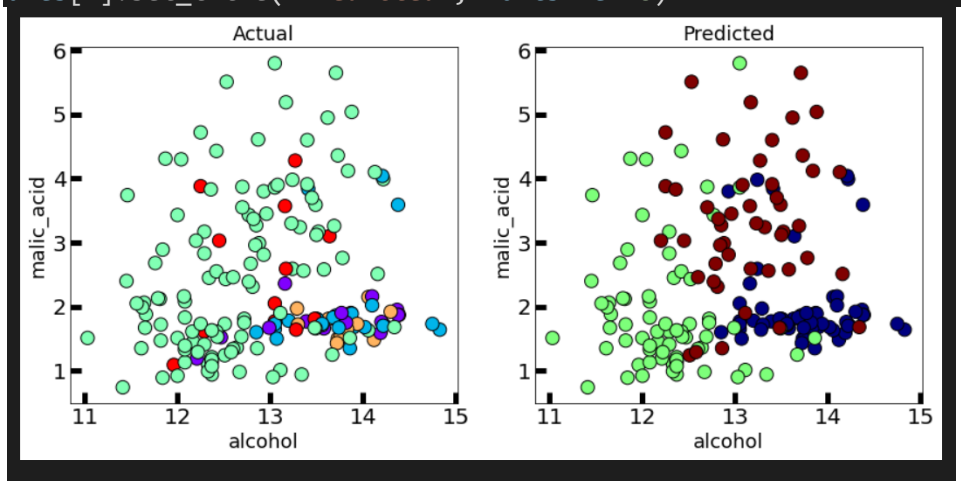

```
from sklearn.cluster import DBSCAN

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

print("Cluster Labels")
print(dbscan.labels_)
```

```
Cluster Labels
[ 0  0 -1 -1  1 -1  2  2  0  0 -1  2  2  0 -1  2  2  0 -1  3  1  1  0  0
  3  3 -1  2  3  0  2 -1  0  2  0  3  3  0  0  1  1  0  0  1  3  0  0  0
  0  2  0  2 -1 -1  0  0  0  2  2  1  1  1  1  1  1  1  1  1 -1  3  1
  1 -1  3  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1 -1
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  3  3  1  1  1  1  1  1  1  1  1  1  1  3  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1  1  3  3  1]
```

```
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_)
```

```
The silhouette score is :

0.4413295944891921
```

```
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
```

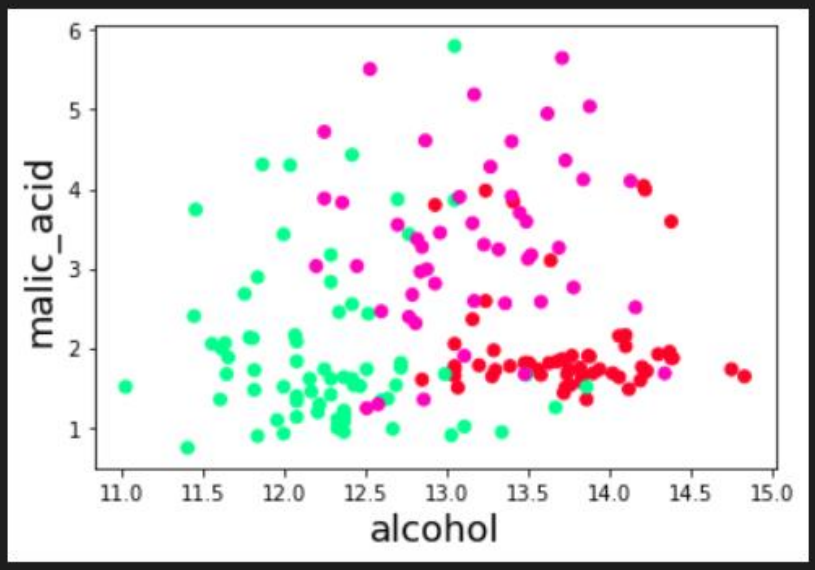
```
from sklearn.metrics import davies_bouldin_score
print("The davies bouldin score is :")
davies_bouldin_score(x, dbscan.labels_)
```

```
The davies bouldin score is :

7.812129203046089
```

```
#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)
```

```
Text(0, 0.5, 'malic_acid')
```

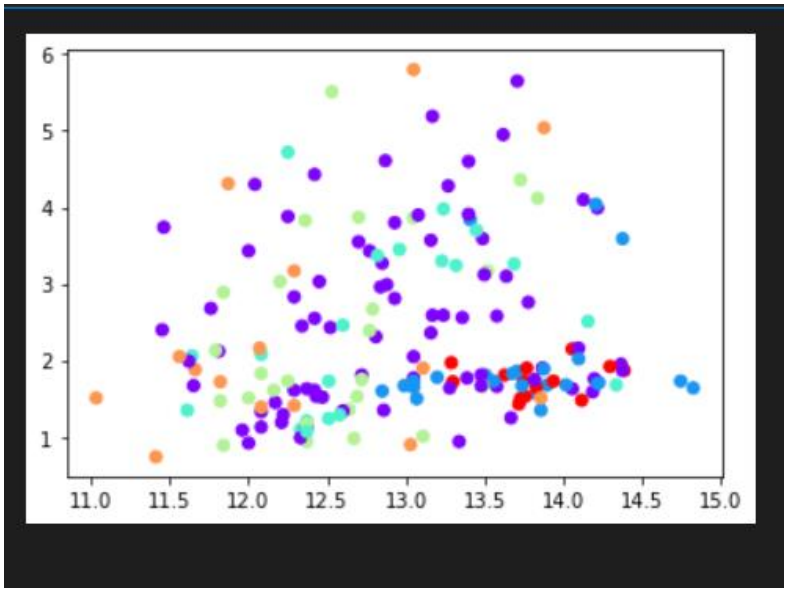



```
from sklearn.cluster import OPTICS

optics = OPTICS(min_samples=13)
y = optics.fit_predict(x)

print("Cluster Labels")
print(optics.labels_)
```

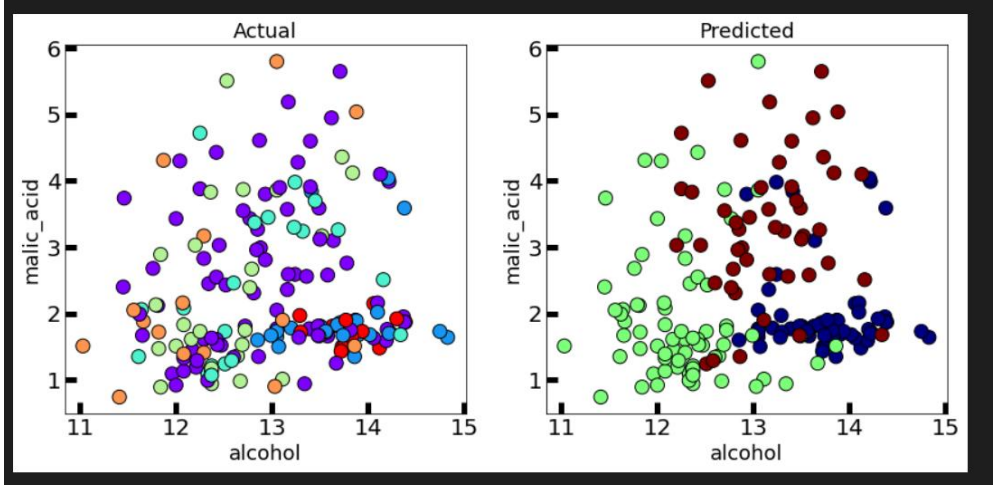
```
Cluster Labels
[ 0  0 -1 -1 -1 -1  4  4  0  0 -1  4  4  0 -1  4  4  0 -1 -1 -1 -1  0  0
 -1 -1 -1  4 -1  0  4 -1  0  4  0 -1 -1  0  0 -1 -1  0  0  1 -1  0  0  0
  0  4  0  4 -1  4  0  0  0  4  4  2  1  2 -1 -1 -1  1  2  2 -1 -1 -1  3
  2  0 -1  3  3  2 -1  2 -1 -1 -1  2  2  2  2 -1  1 -1  2  2  2 -1 -1 -1
 -1  3  1  3  1 -1 -1  3  1 -1  2  2 -1  1 -1 -1 -1  3  3  3  2 -1 -1 -1
 -1  3 -1  3  3  3 -1  2 -1 -1 -1 -1 -1 -1  1  1  1  2 -1 -1 -1 -1  2 -1
 -1 -1  3 -1  1 -1 -1  2  3  1  1 -1  2 -1  1 -1  2  1 -1  1 -1  2  1  1
 -1 -1  2  2  1 -1 -1 -1 -1 -1]
```

+ Code

```
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_)
```

```
The silhouette score is :

0.08708464008252535
```

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

```
The calinski harabasz score is :

44.955612959266325
```

```
from sklearn.metrics import davies_bouldin_score
print("The davies bouldin score is :")
davies_bouldin_score(x, optics.labels_)
```

```
The davies bouldin score is :

2.275571431762288
```