Name = Manish kumar

Roll No = 001811001078

Class =IT 4th year 1st semester

Subject = Machine Learning

Question no 1

Import required header files

```
import pandas as pd

from sklearn.datasets import load_wine # import datasets
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
import seaborn as sns

from sklearn.svm import SVC #import SVM classifier

from sklearn.tree import DecisionTreeClassifier # import decision tree
classifier

from sklearn.ensemble import RandomForestClassifier # import random
forest classifier

from sklearn.naive_bayes import GaussianNB # import naive bayes
classifier
```

Load Wine Dataset

```
# load wine dataset
wine = load_wine()
dir(wine)
```

```
['DESCR', 'data', 'feature_names', 'frame', 'target', 'target_names']
```

```
wine.feature_names
```

```
['alcohol',
 'malic_acid',
```

```
 'ash',
 'alcalinity_of_ash',
 'magnesium',
 'total_phenols',
 'flavanoids',
 'nonflavanoid_phenols',
 'proanthocyanins',
 'color_intensity',
 'hue',
 'od280/od315_of_diluted_wines',
 'proline']
```

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

```
   alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols  \
0    14.23        1.71  2.43               15.6      127.0
2.80
1    13.20        1.78  2.14               11.2      100.0
2.65
2    13.16        2.36  2.67               18.6      101.0
2.80
3    14.37        1.95  2.50               16.8      113.0
3.85
4    13.24        2.59  2.87               21.0      118.0
2.80

   flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity
hue  \
0        3.06                  0.28             2.29             5.64
1.04
1        2.76                  0.26             1.28             4.38
1.05
2        3.24                  0.30             2.81             5.68
1.03
3        3.49                  0.24             2.18             7.80
0.86
4        2.69                  0.39             1.82             4.32
1.04

   od280/od315_of_diluted_wines  proline
0                          3.92   1065.0
1                          3.40   1050.0
2                          3.17   1185.0
3                          3.45   1480.0
4                          2.93    735.0
```

```
df['target'] = wine.target
df.head()
```

```
   alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  
total_phenols  \
0    14.23        1.71  2.43               15.6      127.0  
2.80   
1    13.20        1.78  2.14               11.2      100.0  
2.65   
2    13.16        2.36  2.67               18.6      101.0  
2.80   
3    14.37        1.95  2.50               16.8      113.0  
3.85   
4    13.24        2.59  2.87               21.0      118.0  
2.80   

   flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity  
hue  \
0        3.06                  0.28             2.29             5.64  
1.04   
1        2.76                  0.26             1.28             4.38  
1.05   
2        3.24                  0.30             2.81             5.68  
1.03   
3        3.49                  0.24             2.18             7.80  
0.86   
4        2.69                  0.39             1.82             4.32  
1.04   

   od280/od315_of_diluted_wines  proline  target  
0                          3.92   1065.0       0  
1                          3.40   1050.0       0  
2                          3.17   1185.0       0  
3                          3.45   1480.0       0  
4                          2.93    735.0       0  
```

```
wine.target_names
```

```
array(['class_0', 'class_1', 'class_2'], dtype='<U7')
```

```
df['target'].value_counts()
```

```
1    71
0    59
2    48
Name: target, dtype: int64
```

```
X = df.drop(['target'], axis='columns')
len(X)
```

```
178
```

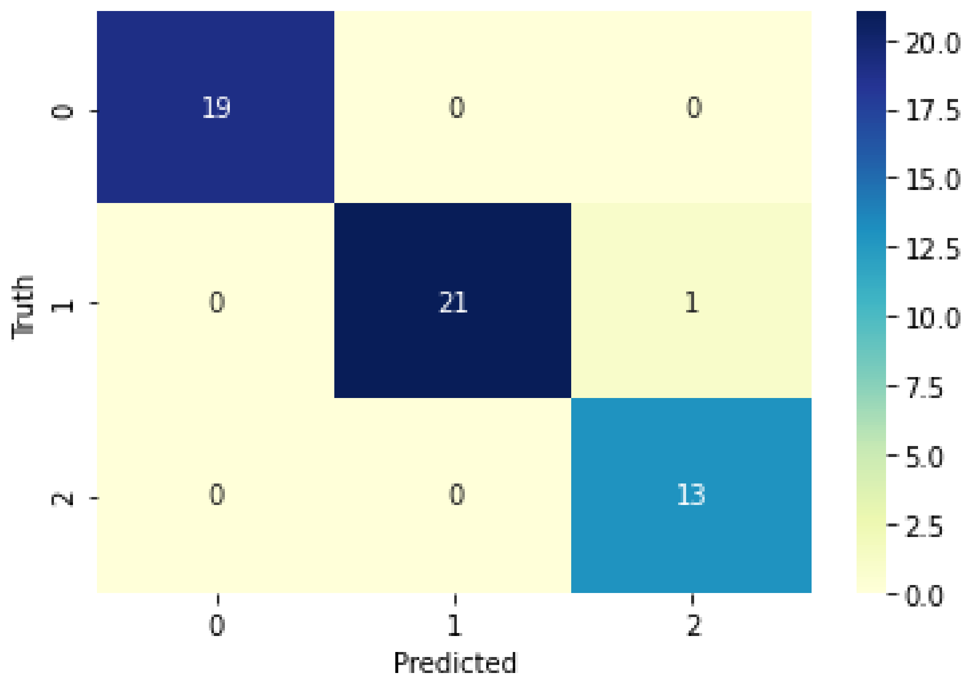
```
y = df.target
len(y)
```

training and test data split

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3,random_state=0)
```

Work for SVM classifier

```
model = SVC(kernel='linear')

model.fit(X_train, y_train)

SVC(kernel='linear')

model.score(X_test,y_test)

0.9814814814814815

y_pred = model.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))

Accuracy: 98.14814814814815%

Confusion Matrix:
[[19  0  0]
 [ 0 21  1]
 [ 0  0 13]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      0.95      0.98        22
           2       0.93      1.00      0.96        13

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


%matplotlib inline
sns.heatmap(cf_matrix,annot=True,cmap="YlGnBu")
plt.xlabel('Predicted')
plt.ylabel('Truth')

Text(33.0, 0.5, 'Truth')
```



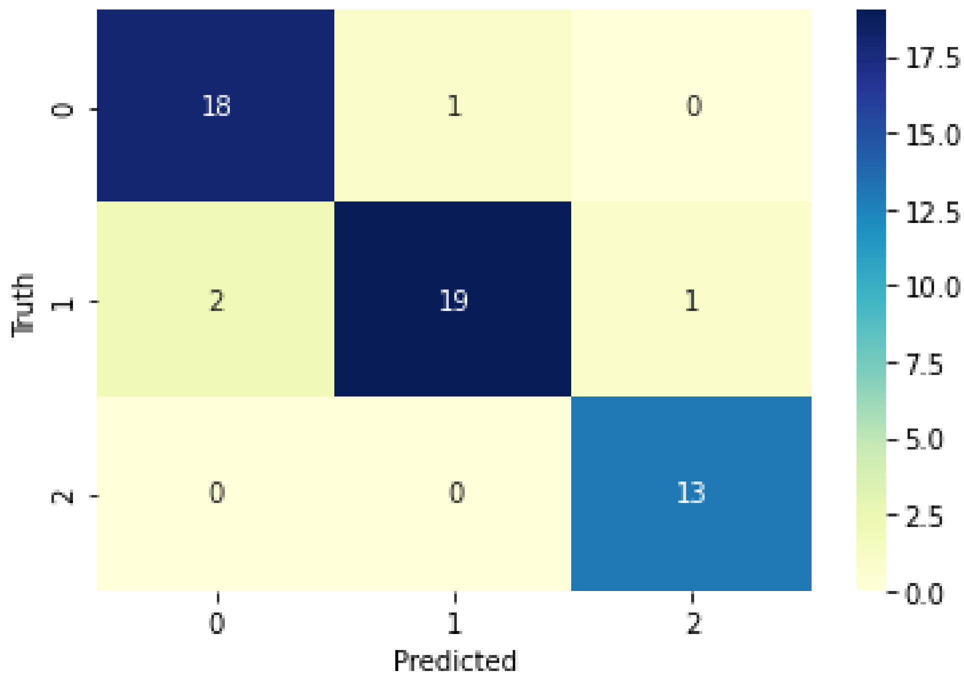
Work for Decision Tree classifier

```
model = DecisionTreeClassifier(criterion='entropy')

model.fit(X_train, y_train)

DecisionTreeClassifier(criterion='entropy')

model.score(X_test, y_test)

0.9259259259259259

y_pred = model.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))

Accuracy: 92.5925925925926%

Confusion Matrix:
[[18  1  0]
 [ 2 19  1]
 [ 0  0 13]]

Classification Report:

              precision    recall  f1-score   support
```

```
              0       0.90      0.95      0.92        19
              1       0.95      0.86      0.90        22
              2       0.93      1.00      0.96        13

       accuracy                           0.93        54
      macro avg       0.93      0.94      0.93        54
   weighted avg       0.93      0.93      0.93        54
```

```
%matplotlib inline
sns.heatmap(cf_matrix,annot=True,cmap="YlGnBu")
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(33.0, 0.5, 'Truth')
```



Work for Random forest classifier

```
model = RandomForestClassifier()
```

```
model.fit(X_train,y_train)
```

```
RandomForestClassifier()
```

```
model.score(X_test,y_test)
```

```
0.9814814814814815
```

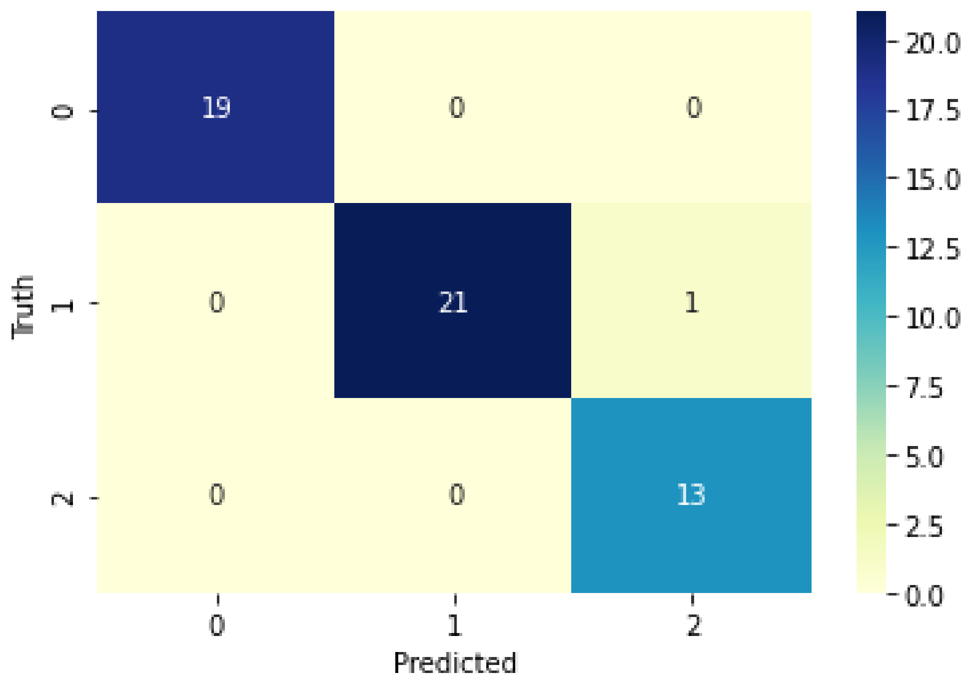
```
y_pred = model.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
```

```
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 98.14814814814815%

Confusion Matrix:
[[19  0  0]
 [ 0 21  1]
 [ 0  0 13]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      0.95      0.98        22
           2       0.93      1.00      0.96        13

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

```
%matplotlib inline
sns.heatmap(cf_matrix,annot=True,cmap="YlGnBu")
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(33.0, 0.5, 'Truth')
```



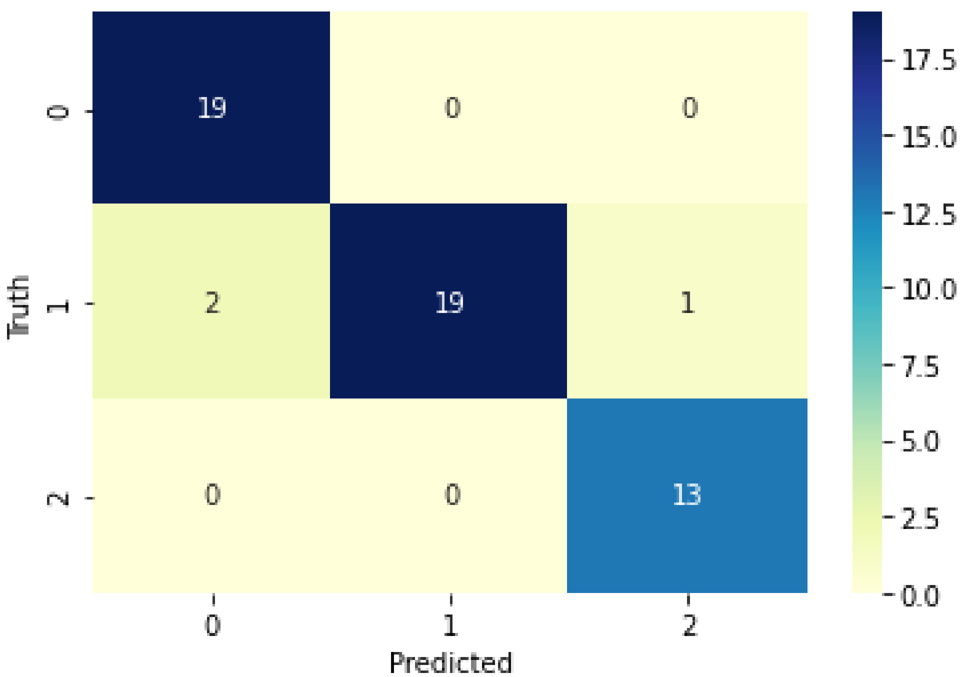
Work for Naive Bayes Classifier

```
model = GaussianNB()

model.fit(X_train,y_train)

GaussianNB()

model.score(X_test, y_test)

0.9444444444444444

y_pred = model.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))

Accuracy: 94.44444444444444%

Confusion Matrix:
[[19  0  0]
 [ 2 19  1]
 [ 0  0 13]]

Classification Report:

              precision    recall  f1-score   support
```

```
              0       0.90      1.00      0.95        19
              1       1.00      0.86      0.93        22
              2       0.93      1.00      0.96        13

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

```
%matplotlib inline
sns.heatmap(cf_matrix,annot=True,cmap="YlGnBu")
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(33.0, 0.5, 'Truth')
```



Ionosphere Dataset

```
# load ionosphere dataset
df = pd.read_csv('ionosphere_data.csv')
df.head()
```

```
   column_a  column_b  column_c  column_d  column_e  column_f
column_g  \
0      True     False   0.99539  -0.05889   0.85243   0.02306
0.83398
1      True     False   1.00000  -0.18829   0.93035  -0.36156  -
0.10868
2      True     False   1.00000  -0.03365   1.00000   0.00485
1.00000
```

```
3      True     False   1.00000  -0.45161   1.00000   1.00000
0.71216
4      True     False   1.00000  -0.02401   0.94140   0.06531
0.92106

   column_h  column_i  column_j  ...  column_z  column_aa
column_ab  \
0  -0.37708   1.00000   0.03760  ...  -0.51171    0.41078   -0.46168

1  -0.93597   1.00000  -0.04549  ...  -0.26569   -0.20468   -0.18401

2  -0.12062   0.88965   0.01198  ...  -0.40220    0.58984   -0.22145

3  -1.00000   0.00000   0.00000  ...   0.90695    0.51613    1.00000

4  -0.23255   0.77152  -0.16399  ...  -0.65158    0.13290   -0.53206


   column_ac  column_ad  column_ae  column_af  column_ag  column_ah
column_ai
0    0.21266   -0.34090    0.42267   -0.54487    0.18641   -0.45300
g
1   -0.19040   -0.11593   -0.16626   -0.06288   -0.13738   -0.02447
b
2    0.43100   -0.17365    0.60436   -0.24180    0.56045   -0.38238
g
3    1.00000   -0.20099    0.25682    1.00000   -0.32382    1.00000
b
4    0.02431   -0.62197   -0.05707   -0.59573   -0.04608   -0.65697
g

[5 rows x 35 columns]
```

```
X = df.drop(['column_ai'],axis='columns')
len(X)
```

```
351
```

```
y = df['column_ai']
len(y)
```

```
351
```

training and test split

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3,random_state=0)
```

Work for SVM classifier

```
model = SVC(kernel='linear')

model.fit(X_train,y_train)
```

```
SVC(kernel='linear')
```

```
model.score(X_test,y_test)
```

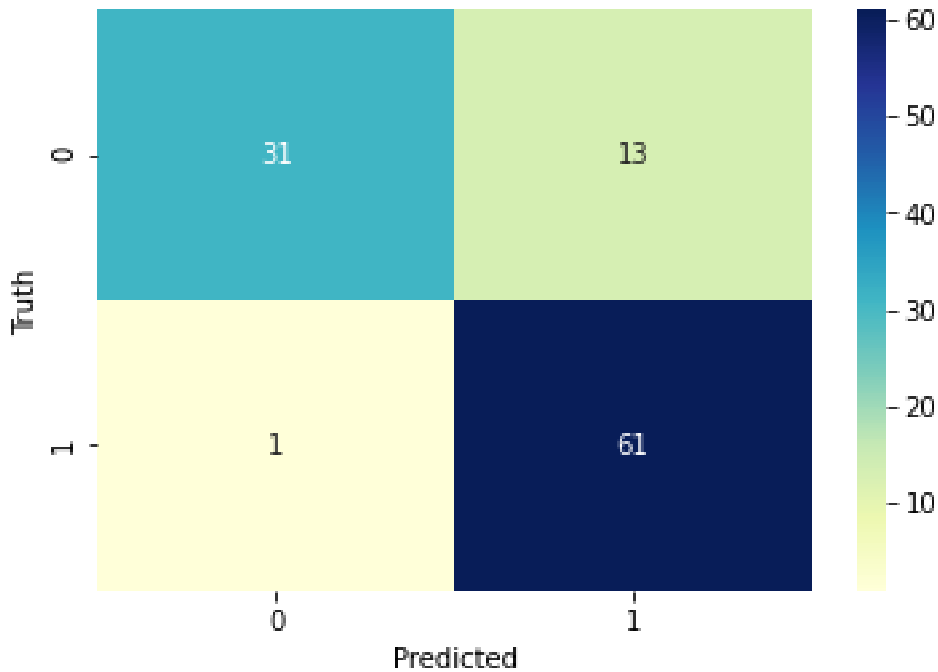
```
0.8679245283018868
```

```
y_pred = model.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 86.79245283018868%

Confusion Matrix:
[[31 13]
 [ 1 61]]

Classification Report:

              precision    recall  f1-score   support

           b       0.97      0.70      0.82        44
           g       0.82      0.98      0.90        62

    accuracy                           0.87       106
   macro avg       0.90      0.84      0.86       106
weighted avg       0.88      0.87      0.86       106
```

```
%matplotlib inline
sns.heatmap(cf_matrix,annot=True,cmap="YlGnBu")
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(33.0, 0.5, 'Truth')
```



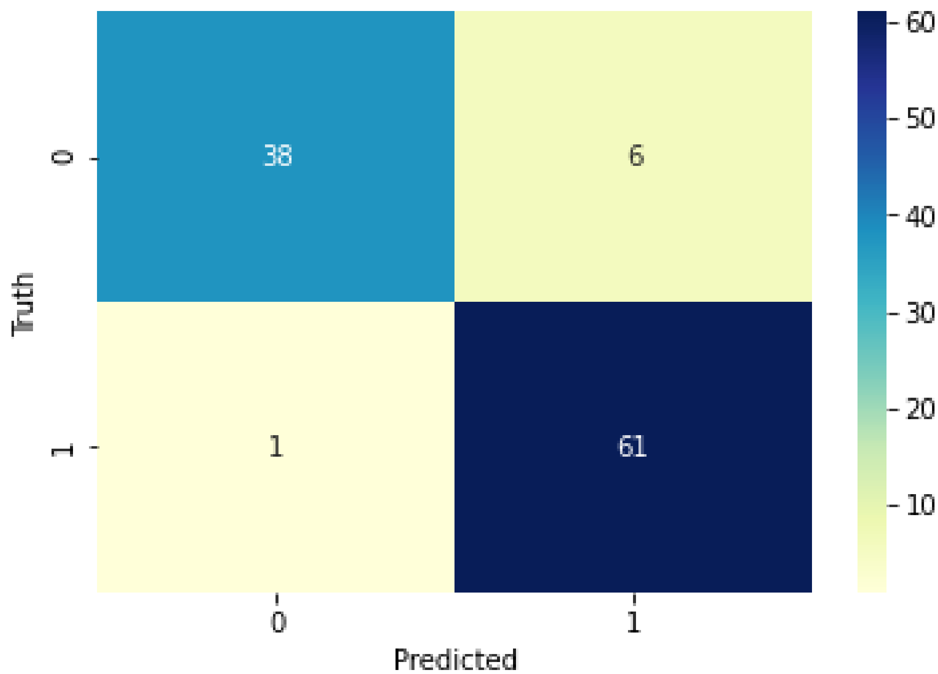
Work for Decision Tree classifier

```
model = DecisionTreeClassifier(criterion='entropy')

model.fit(X_train,y_train)

DecisionTreeClassifier(criterion='entropy')

model.score(X_test,y_test)

0.9339622641509434

y_pred = model.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))

Accuracy: 93.39622641509435%

Confusion Matrix:
[[38  6]
 [ 1 61]]

Classification Report:

              precision    recall  f1-score   support

           b       0.97      0.86      0.92        44
```

```
           g       0.91      0.98      0.95        62

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

```
%matplotlib inline
sns.heatmap(cf_matrix,annot=True,cmap="YlGnBu")
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(33.0, 0.5, 'Truth')
```



Work for Random forest classifier

```
model = RandomForestClassifier()
```

```
model.fit(X_train,y_train)
```

```
RandomForestClassifier()
```

```
model.score(X_test,y_test)
```

```
0.9339622641509434
```

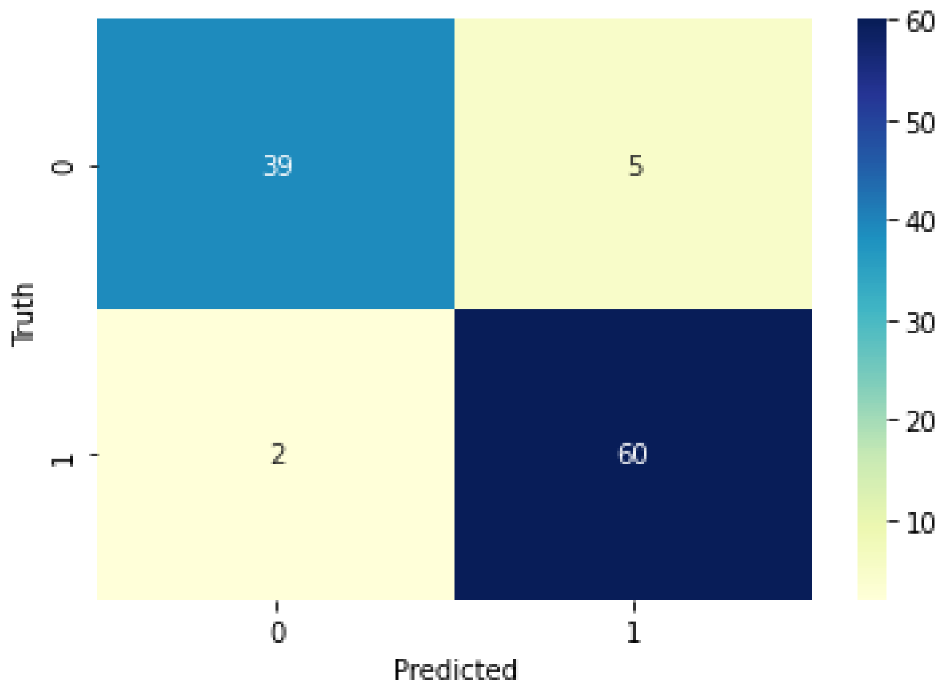
```
y_pred = model.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
```

```
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 93.39622641509435%

Confusion Matrix:
[[39  5]
 [ 2 60]]

Classification Report:

              precision    recall  f1-score   support

           b       0.95      0.89      0.92        44
           g       0.92      0.97      0.94        62

    accuracy                           0.93       106
   macro avg       0.94      0.93      0.93       106
weighted avg       0.93      0.93      0.93       106
```

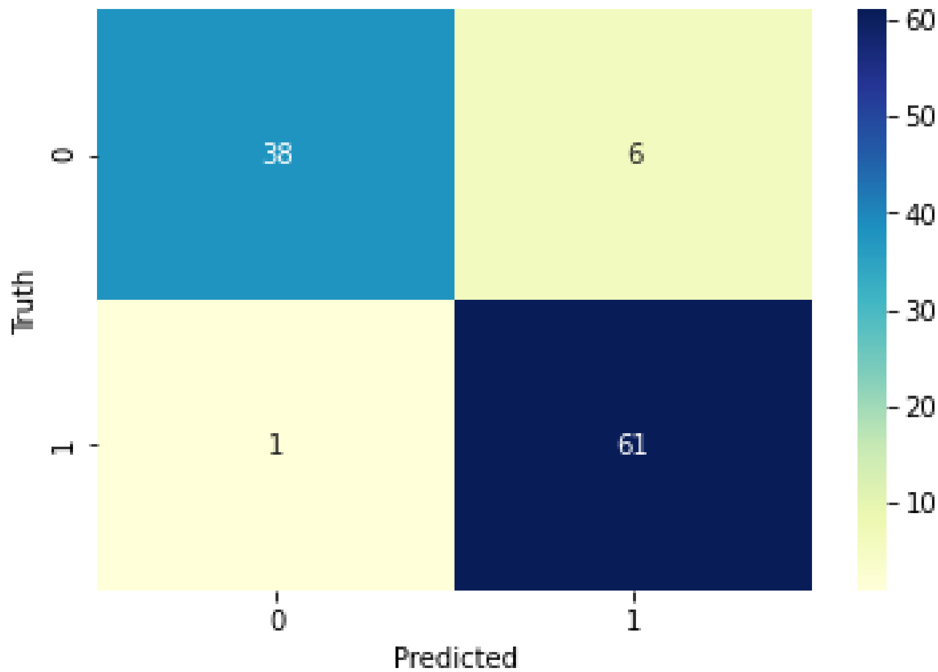
```
%matplotlib inline
sns.heatmap(cf_matrix,annot=True,cmap="YlGnBu")
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(33.0, 0.5, 'Truth')
```



Work for Nayes Bayes classifier

```
model = GaussianNB()

model.fit(X_train,y_train)

GaussianNB()

model.score(X_test,y_test)

0.9339622641509434

y_pred = model.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))

Accuracy: 93.39622641509435%

Confusion Matrix:
[[38  6]
 [ 1 61]]

Classification Report:

              precision    recall  f1-score   support

           b       0.97      0.86      0.92        44
           g       0.91      0.98      0.95        62

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


%matplotlib inline
sns.heatmap(cf_matrix,annot=True,cmap="YlGnBu")
plt.xlabel('Predicted')
plt.ylabel('Truth')

Text(33.0, 0.5, 'Truth')
```



38
6
1
61
0
1
Truth
0
1
Predicted
60
50
40
30
20
10

Name = Manish kumar

Roll No = 001811001078

Class =IT 4th year 1st semester

Subject = Machine Learning

Question no 2

Import required modules

```
import pandas as pd

from sklearn.datasets import load_diabetes, load_iris,
load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
import seaborn as sns
from matplotlib import pyplot as plt

from sklearn.preprocessing import StandardScaler # for feature scaling
from sklearn.neural_network import MLPClassifier # import ANN
classifier
```

Iris Dataset

```
iris = load_iris()
df = pd.DataFrame(iris.data,columns=iris.feature_names)
df['target'] = iris.target
df.head()
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width
(cm)  \
0                5.1               3.5                1.4
0.2
1                4.9               3.0                1.4
0.2
2                4.7               3.2                1.3
0.2
3                4.6               3.1                1.5
0.2
4                5.0               3.6                1.4
```

```
0.2
```

```
   target
0       0
1       0
2       0
3       0
4       0
```

```
X = df.drop(['target'],axis="columns")
len(X)
```

```
150
```

```
y = df['target']
len(y)
```

```
150
```

training and test set split

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

feature scaling

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

load MLP classifier

```
model = MLPClassifier()
```

```
model.fit(X_train,y_train)
```

```
C:\Users\thisa\anaconda3\lib\site-packages\sklearn\neural_network\
_multilayer_perceptron.py:614: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (200) reached and the optimization
hasn't converged yet.
  warnings.warn(
```

```
MLPClassifier()
```

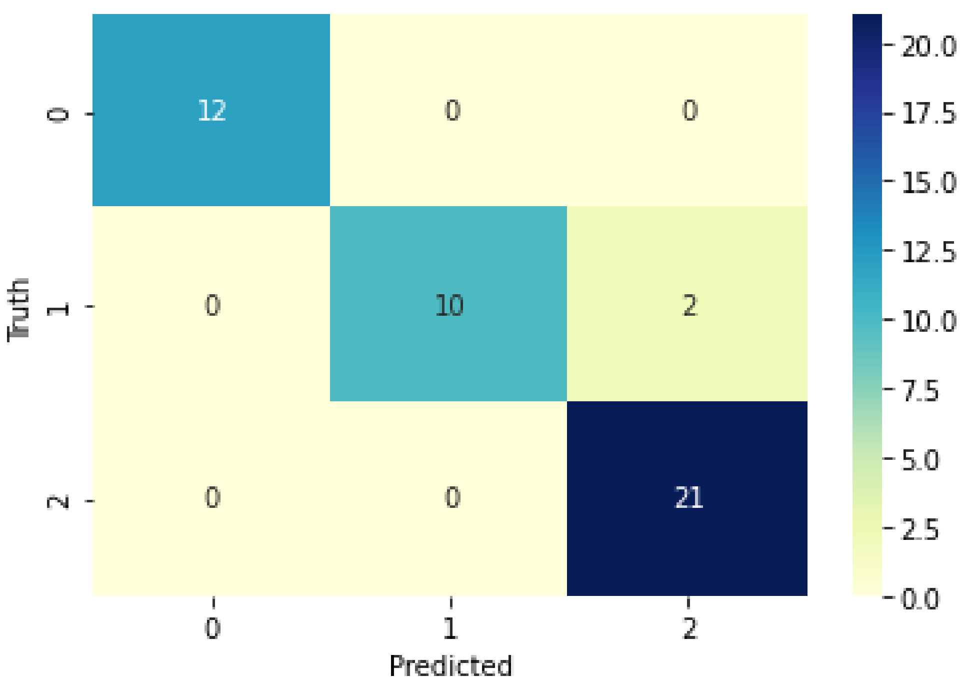
```
y_pred = model.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 95.55555555555556%

Confusion Matrix:
[[12  0  0]
```

```
 [ 0 10  2]
 [ 0  0 21]]

Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       1.00      0.83      0.91        12
           2       0.91      1.00      0.95        21

    accuracy                           0.96        45
   macro avg       0.97      0.94      0.95        45
weighted avg       0.96      0.96      0.95        45
```

```
%matplotlib inline
sns.heatmap(cf_matrix,annot=True,cmap="YlGnBu")
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(33.0, 0.5, 'Truth')
```



Diabetes Dataset

```
df = pd.read_csv('diabetes.tab.txt',delimiter="\t")
df.head()
```

```
   AGE  SEX   BMI     BP   S1     S2    S3   S4      S5  S6    Y
0   59    2  32.1  101.0  157   93.2  38.0  4.0  4.8598  87  151
```

```
1   48    1  21.6   87.0  183  103.2  70.0  3.0  3.8918  69   75
2   72    2  30.5   93.0  156   93.6  41.0  4.0  4.6728  85  141
3   24    1  25.3   84.0  198  131.4  40.0  5.0  4.8903  89  206
4   50    1  23.0  101.0  192  125.4  52.0  4.0  4.2905  80  135
```

```
X = df.drop(['SEX'],axis="columns")
len(X)
```

```
442
```

```
y = df['SEX']
len(y)
```

```
442
```

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

training and test set split

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

load MLP classifier

```
model = MLPClassifier()
```

```
model.fit(X_train,y_train)
```

```
MLPClassifier()
```

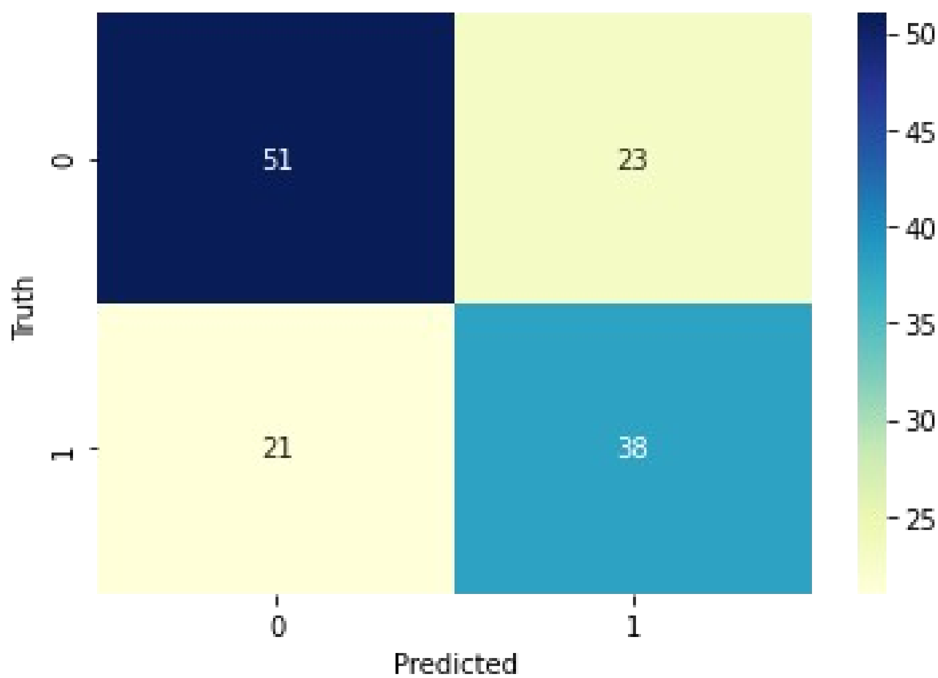
```
y_pred = model.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 66.9172932330827%

Confusion Matrix:
[[51 23]
 [21 38]]

Classification Report:

              precision    recall  f1-score   support

           1       0.71      0.69      0.70        74
           2       0.62      0.64      0.63        59

    accuracy                           0.67       133
   macro avg       0.67      0.67      0.67       133
```

```
weighted avg       0.67      0.67      0.67       133
```

```
%matplotlib inline
sns.heatmap(cf_matrix,annot=True,cmap="YlGnBu")
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(33.0, 0.5, 'Truth')
```



Wisconsin Breast Cancer Dataset

```
breast_cancer = load_breast_cancer()
df =
pd.DataFrame(breast_cancer.data,columns=breast_cancer.feature_names)
df['target'] = breast_cancer.target
df.head()
```

```
   mean radius  mean texture  mean perimeter  mean area  mean
smoothness  \
0        17.99         10.38          122.80     1001.0
0.11840
1        20.57         17.77          132.90     1326.0
0.08474
2        19.69         21.25          130.00     1203.0
0.10960
3        11.42         20.38           77.58      386.1
0.14250
4        20.29         14.34          135.10     1297.0
0.10030
```

```
   mean compactness  mean concavity  mean concave points  mean
symmetry  \
0           0.27760          0.3001              0.14710
0.2419
1           0.07864          0.0869              0.07017
0.1812
2           0.15990          0.1974              0.12790
0.2069
3           0.28390          0.2414              0.10520
0.2597
4           0.13280          0.1980              0.10430
0.1809

   mean fractal dimension  ...  worst texture  worst perimeter  worst
area  \
0                 0.07871  ...          17.33           184.60
2019.0
1                 0.05667  ...          23.41           158.80
1956.0
2                 0.05999  ...          25.53           152.50
1709.0
3                 0.09744  ...          26.50            98.87
567.7
4                 0.05883  ...          16.67           152.20
1575.0

   worst smoothness  worst compactness  worst concavity  worst concave
points  \
0            0.1622             0.6656           0.7119
0.2654
1            0.1238             0.1866           0.2416
0.1860
2            0.1444             0.4245           0.4504
0.2430
3            0.2098             0.8663           0.6869
0.2575
4            0.1374             0.2050           0.4000
0.1625

   worst symmetry  worst fractal dimension  target
0          0.4601                  0.11890       0
1          0.2750                  0.08902       0
2          0.3613                  0.08758       0
3          0.6638                  0.17300       0
4          0.2364                  0.07678       0

[5 rows x 31 columns]
```

```
X = df.drop(['target'],axis="columns")
len(X)
```

```
569
```

```
y = df['target']
len(y)
```

```
569
```

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

training and test set split

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

load MLP classifier

```
model = MLPClassifier()
```

```
model.fit(X_train,y_train)
```

```
C:\Users\thisa\anaconda3\lib\site-packages\sklearn\neural_network\
_multilayer_perceptron.py:614: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (200) reached and the optimization
hasn't converged yet.
  warnings.warn(
```

```
MLPClassifier()
```

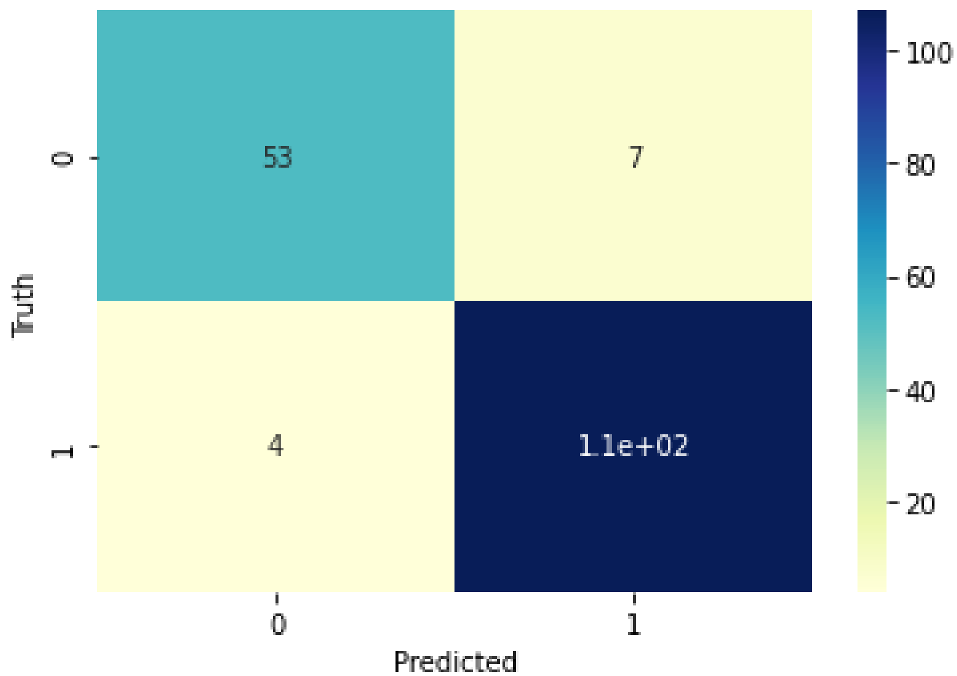
```
y_pred = model.predict(X_test)
print(f"Accuracy: {100 * accuracy_score(y_test,y_pred)}%\n")
cf_matrix = confusion_matrix(y_test,y_pred)
print("Confusion Matrix:")
print(cf_matrix)
print("\nClassification Report:\n")
print(classification_report(y_test,y_pred))
```

```
Accuracy: 93.56725146198829%

Confusion Matrix:
[[ 53   7]
 [  4 107]]

Classification Report:

              precision    recall  f1-score   support

           0       0.93      0.88      0.91        60
           1       0.94      0.96      0.95       111

    accuracy                           0.94       171
```

```
   macro avg       0.93      0.92      0.93       171
weighted avg       0.94      0.94      0.94       171
```

```
%matplotlib inline
sns.heatmap(cf_matrix,annot=True,cmap="YlGnBu")
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(33.0, 0.5, 'Truth')
```



Name = Manish kumar

Roll No = 001811001078

Class =IT 4th year 1st semester

Subject = Machine Learning

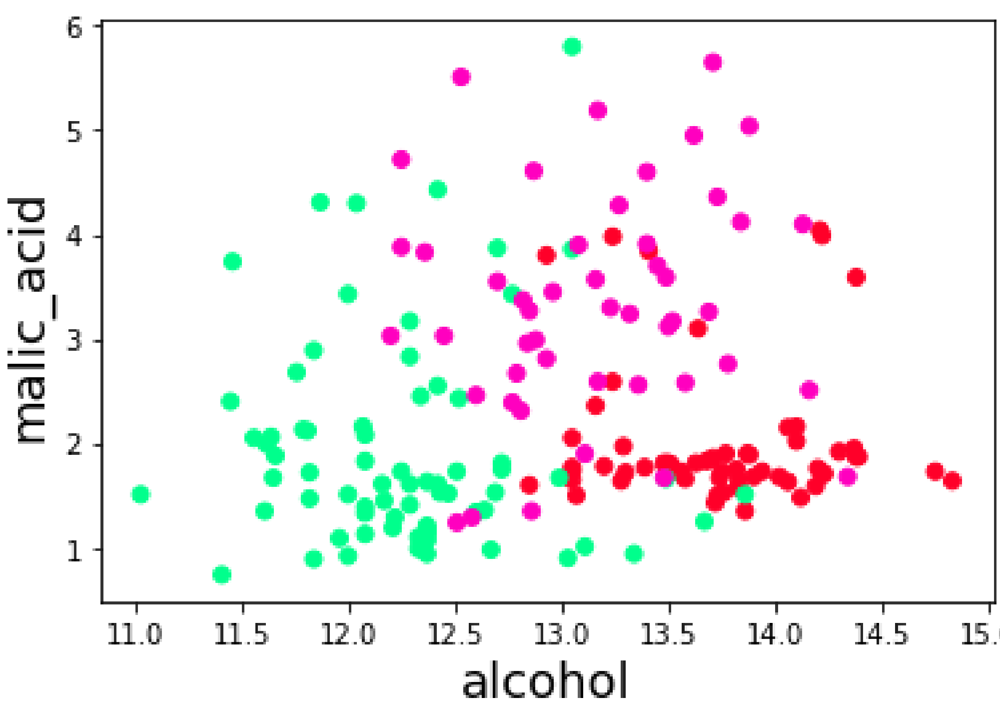
Question no 5

```
import numpy as np
import pandas as pd
import sklearn as sk
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.cluster import KMeans
from sklearn_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=wine.feature_names)
df.head()

   alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols  \
0    14.23        1.71  2.43               15.6      127.0
2.80
1    13.20        1.78  2.14               11.2      100.0
2.65
2    13.16        2.36  2.67               18.6      101.0
2.80
3    14.37        1.95  2.50               16.8      113.0
3.85
4    13.24        2.59  2.87               21.0      118.0
2.80

   flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity
hue  \
0        3.06                  0.28             2.29             5.64
1.04
```

```
1        2.76                     0.26                1.28                4.38
1.05
2        3.24                     0.30                2.81                5.68
1.03
3        3.49                     0.24                2.18                7.80
0.86
4        2.69                     0.39                1.82                4.32
1.04

   od280/od315_of_diluted_wines  proline
0                          3.92   1065.0
1                          3.40   1050.0
2                          3.17   1185.0
3                          3.45   1480.0
4                          2.93    735.0
```

```
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')
```

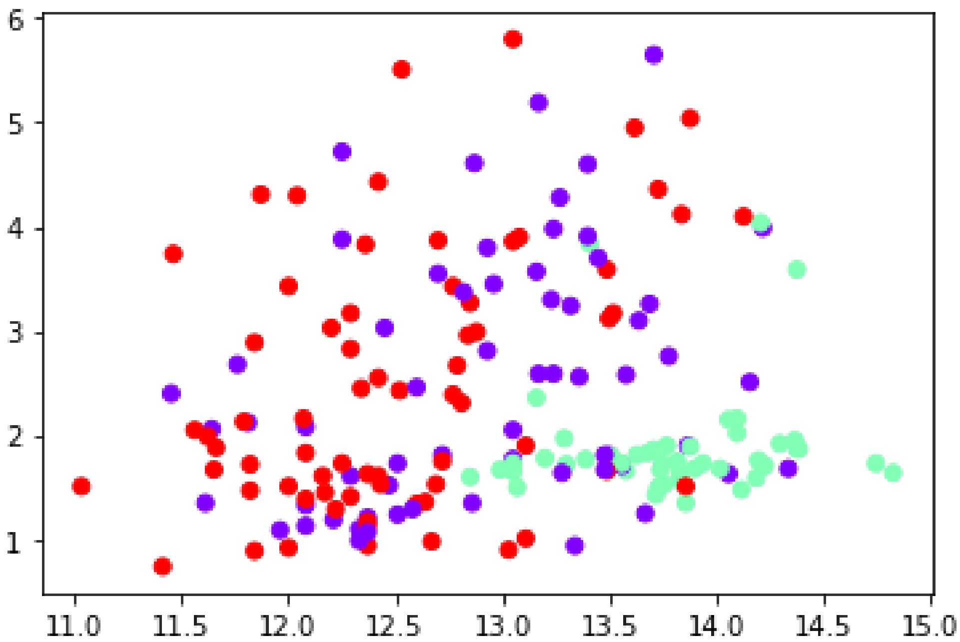



```
kmeans = KMeans(init="random", n_clusters=3, n_init=10, max_iter=300,
random_state=42)
y = kmeans.fit_predict(x)
```

```
print("K-Means Cluster Centers")
print(kmeans.cluster_centers_)
print("Cluster Labels")
print(kmeans.labels_)
```

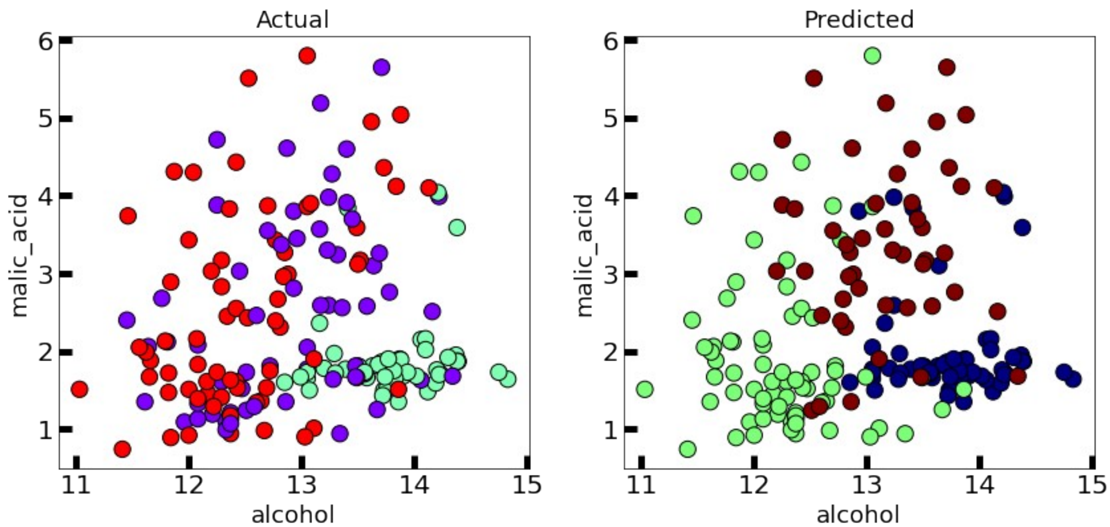
```
K-Means Cluster Centers
[[1.29298387e+01 2.50403226e+00 2.40806452e+00 1.98903226e+01
  1.03596774e+02 2.11112903e+00 1.58403226e+00 3.88387097e-01
  1.50338710e+00 5.65032258e+00 8.83967742e-01 2.36548387e+00
  7.28338710e+02]
 [1.38044681e+01 1.88340426e+00 2.42617021e+00 1.70234043e+01
  1.05510638e+02 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 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 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 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)
```

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



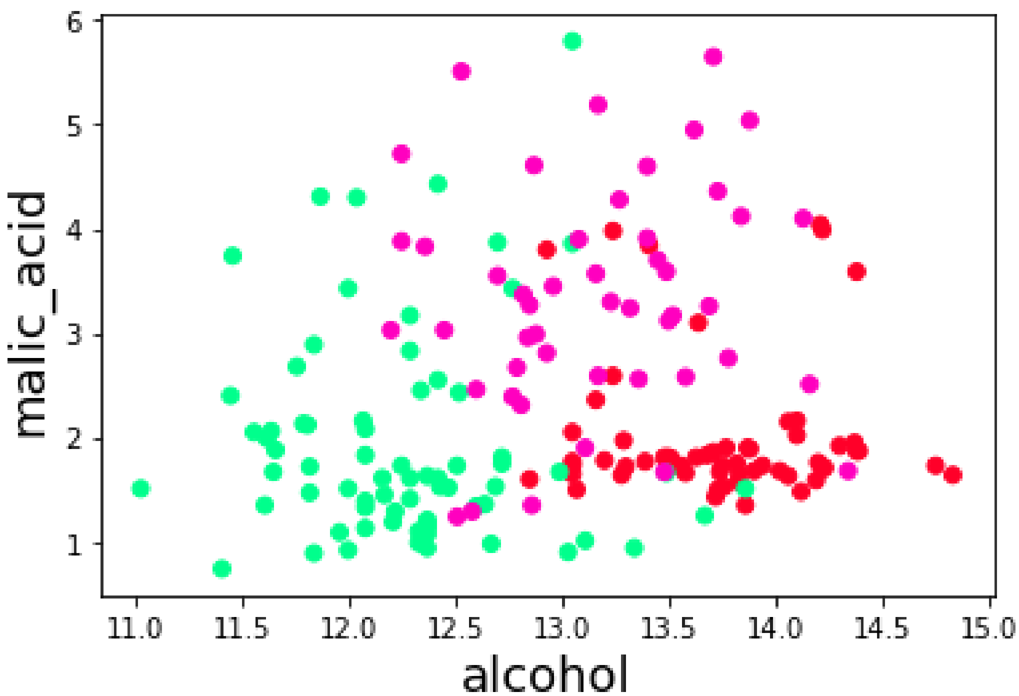

```
from sklearn.metrics import silhouette_score
print("The silhouette score is :")
silhouette_score(x, kmeans.labels_)

The silhouette score is :

0.5711381937868844

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

The calinski harabasz score is :

561.815657860671

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

The davies bouldin score is :

0.5342431775436277

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

     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols  \
0      14.23        1.71  2.43               15.6      127.0
2.80
1      13.20        1.78  2.14               11.2      100.0
2.65
2      13.16        2.36  2.67               18.6      101.0
2.80
3      14.37        1.95  2.50               16.8      113.0
```

```
3.85
4      13.24        2.59  2.87              21.0      118.0
2.80
..       ...         ...   ...               ...        ...
...
173    13.71        5.65  2.45              20.5       95.0
1.68
174    13.40        3.91  2.48              23.0      102.0
1.80
175    13.27        4.28  2.26              20.0      120.0
1.59
176    13.17        2.59  2.37              20.0      120.0
1.65
177    14.13        4.10  2.74              24.5       96.0
2.05

     flavanoids  nonflavanoid_phenols  proanthocyanins
color_intensity   hue  \
0          3.06                  0.28             2.29
5.64  1.04
1          2.76                  0.26             1.28
4.38  1.05
2          3.24                  0.30             2.81
5.68  1.03
3          3.49                  0.24             2.18
7.80  0.86
4          2.69                  0.39             1.82
4.32  1.04
..          ...                   ...              ...               ..
.   ...
173        0.61                  0.52             1.06
7.70  0.64
174        0.75                  0.43             1.41
7.30  0.70
175        0.69                  0.43             1.35
10.20  0.59
176        0.68                  0.53             1.46
9.30  0.60
177        0.76                  0.56             1.35
9.20  0.61

     od280/od315_of_diluted_wines  proline
0                            3.92   1065.0
1                            3.40   1050.0
2                            3.17   1185.0
3                            3.45   1480.0
4                            2.93    735.0
..                            ...      ...
173                          1.74    740.0
174                          1.56    750.0
```

```
175                            1.56    835.0
176                            1.62    840.0
177                            1.60    560.0

[178 rows x 13 columns]
```

```
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')
```

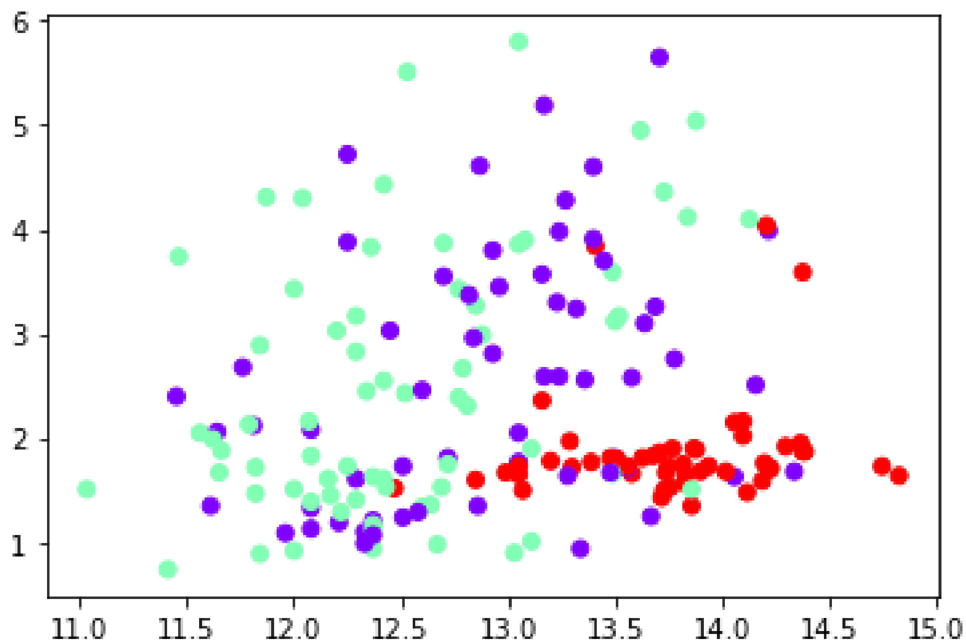



```
kmedoid = KMedoids(init="heuristic", n_clusters=3, max_iter=300,
random_state=42)
y = kmedoid.fit_predict(x)

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 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 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 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]
```

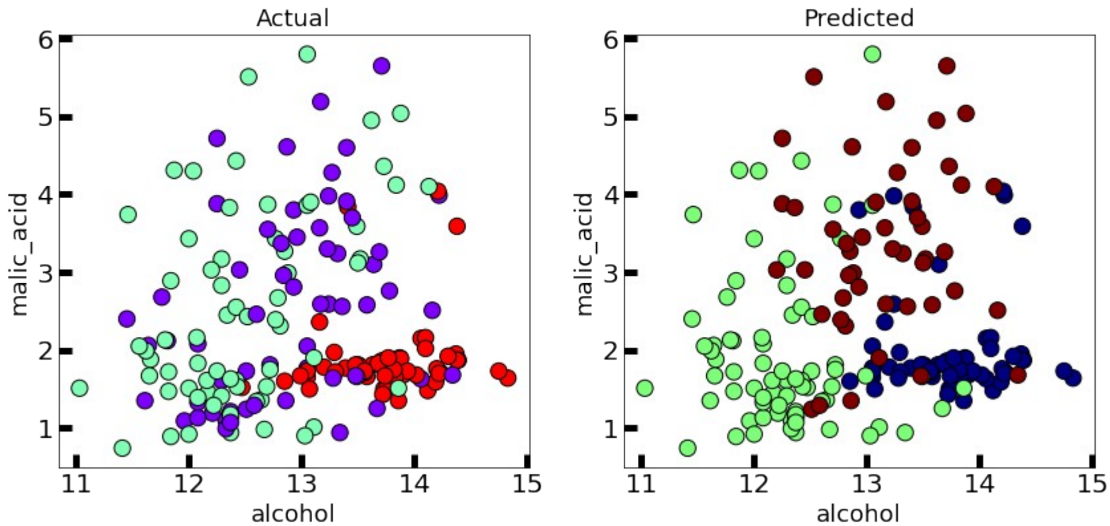
```
plt.scatter(x=df['alcohol'], y=df['malic_acid'] ,c=kmedoid.labels_,
cmap='rainbow') #try using cmap='rainbow'
plt.show()
```




```
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)
```

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




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

```
The silhouette score is :
```

```
0.5666480408636575
```

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

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

```
539.3792353535451
```

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

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

```
0.529239412600316
```

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

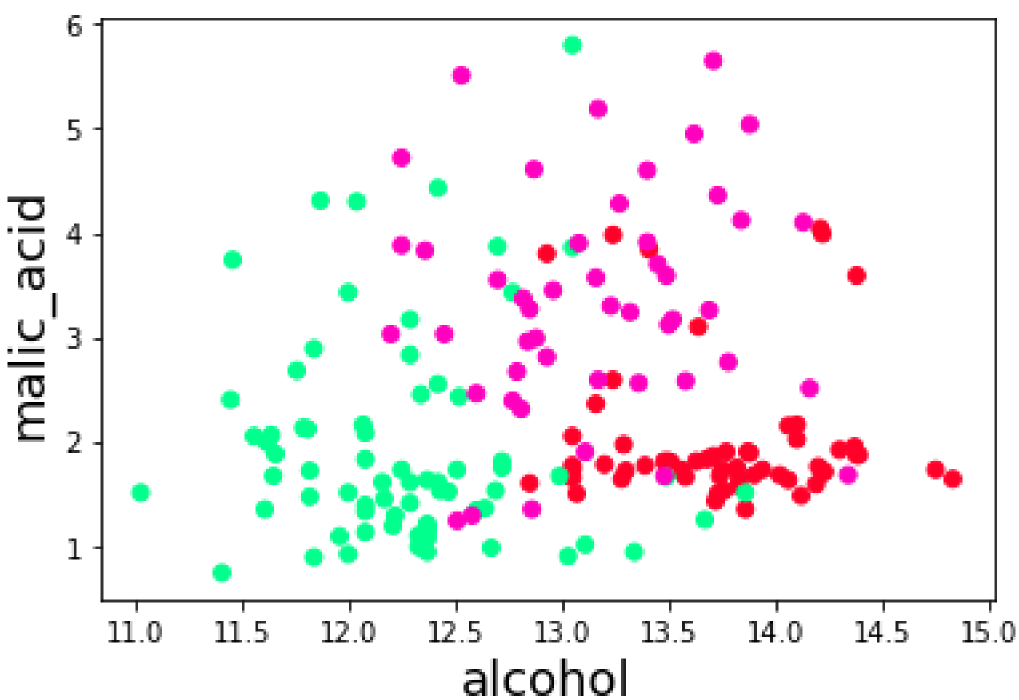
```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols  \
```

```
0      14.23        1.71  2.43               15.6      127.0
2.80
1      13.20        1.78  2.14               11.2      100.0
2.65
2      13.16        2.36  2.67               18.6      101.0
2.80
3      14.37        1.95  2.50               16.8      113.0
3.85
4      13.24        2.59  2.87               21.0      118.0
2.80
..       ...         ...   ...                ...        ...
...
173    13.71        5.65  2.45               20.5       95.0
1.68
174    13.40        3.91  2.48               23.0      102.0
1.80
175    13.27        4.28  2.26               20.0      120.0
1.59
176    13.17        2.59  2.37               20.0      120.0
1.65
177    14.13        4.10  2.74               24.5       96.0
2.05

     flavanoids  nonflavanoid_phenols  proanthocyanins
color_intensity   hue  \
0          3.06                  0.28             2.29
5.64  1.04
1          2.76                  0.26             1.28
4.38  1.05
2          3.24                  0.30             2.81
5.68  1.03
3          3.49                  0.24             2.18
7.80  0.86
4          2.69                  0.39             1.82
4.32  1.04
..          ...                   ...              ...               ..
.   ...
173        0.61                  0.52             1.06
7.70  0.64
174        0.75                  0.43             1.41
7.30  0.70
175        0.69                  0.43             1.35
10.20  0.59
176        0.68                  0.53             1.46
9.30  0.60
177        0.76                  0.56             1.35
9.20  0.61

     od280/od315_of_diluted_wines  proline
0                            3.92   1065.0
```

```
1                            3.40   1050.0
2                            3.17   1185.0
3                            3.45   1480.0
4                            2.93    735.0
..                            ...      ...
173                          1.74    740.0
174                          1.56    750.0
175                          1.56    835.0
176                          1.62    840.0
177                          1.60    560.0

[178 rows x 13 columns]
```

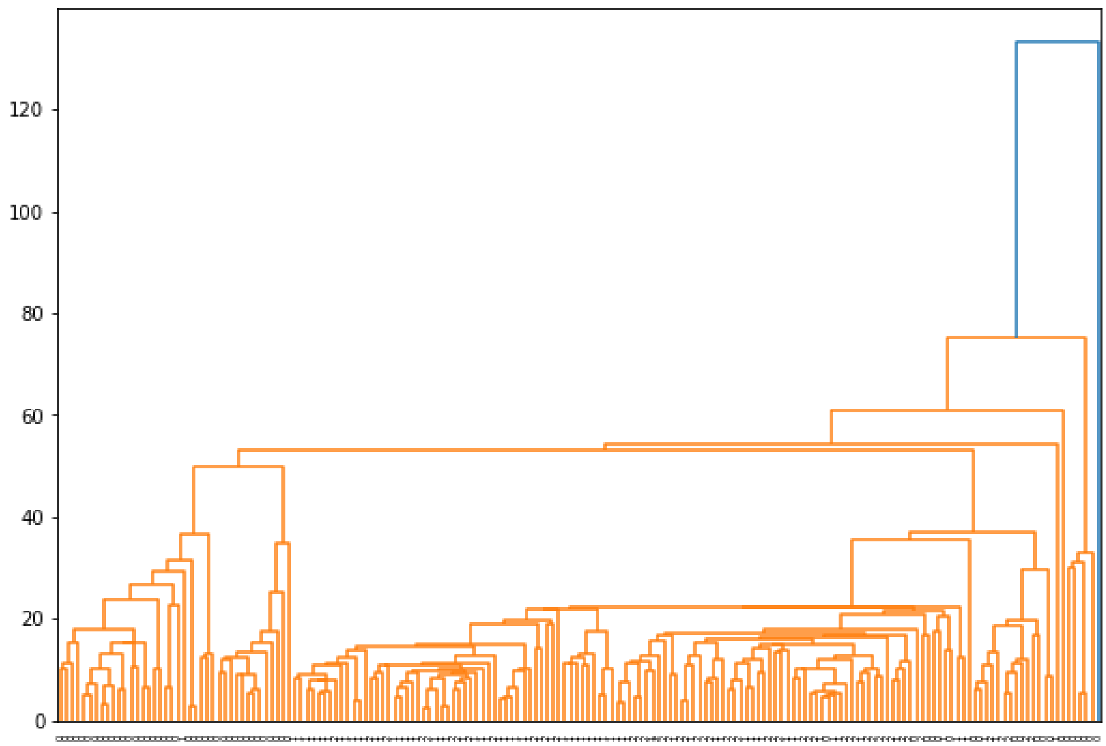
```
plt.scatter(x=df['alcohol'], y=df['malic_acid'] ,c=wine.target,
cmap='gist_rainbow') #try using cmap='rainbow'

plt.xlabel('alcohol', fontsize=18)
plt.ylabel('malic_acid', fontsize=18)
```

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




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

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

dendrogram(linked,
           orientation='top',
```

```
           labels=wine.target,
           distance_sort='descending',
           show_leaf_counts=True)

plt.show()
```



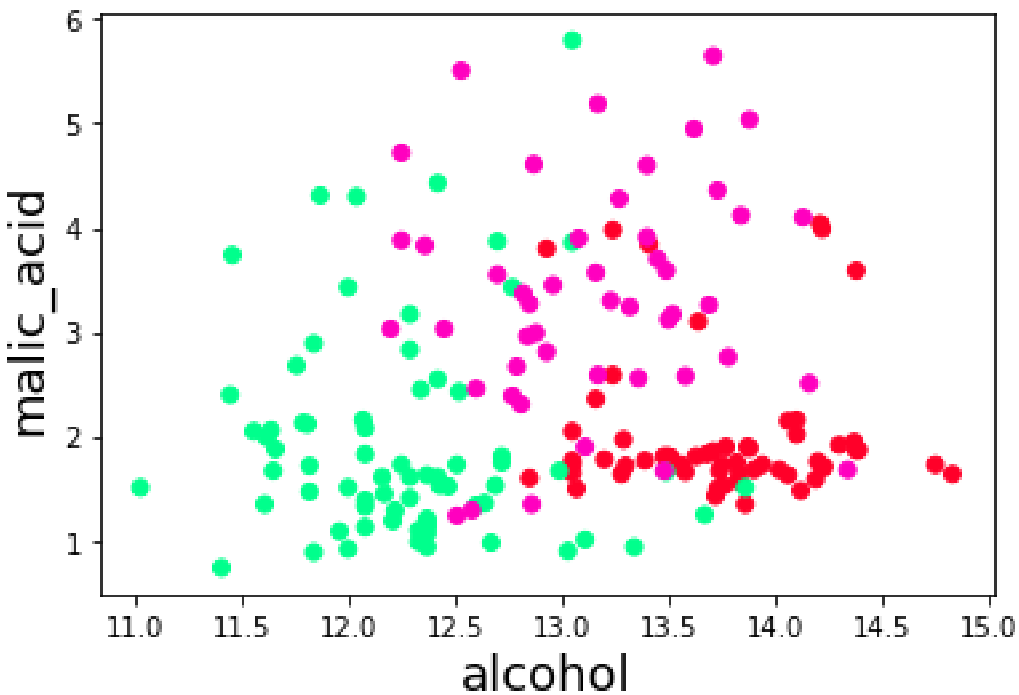

```
df=pd.DataFrame(data=wine.data, columns=wine.feature_names)
df

     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium
total_phenols  \
0      14.23        1.71  2.43               15.6      127.0
2.80
1      13.20        1.78  2.14               11.2      100.0
2.65
2      13.16        2.36  2.67               18.6      101.0
2.80
3      14.37        1.95  2.50               16.8      113.0
3.85
4      13.24        2.59  2.87               21.0      118.0
2.80
..       ...         ...   ...                ...        ...
...
173    13.71        5.65  2.45               20.5       95.0
1.68
174    13.40        3.91  2.48               23.0      102.0
```

```
1.80
175    13.27        4.28  2.26              20.0      120.0
1.59
176    13.17        2.59  2.37              20.0      120.0
1.65
177    14.13        4.10  2.74              24.5       96.0
2.05

     flavanoids  nonflavanoid_phenols  proanthocyanins
color_intensity   hue  \
0          3.06                  0.28             2.29
5.64  1.04
1          2.76                  0.26             1.28
4.38  1.05
2          3.24                  0.30             2.81
5.68  1.03
3          3.49                  0.24             2.18
7.80  0.86
4          2.69                  0.39             1.82
4.32  1.04
..          ...                   ...              ...              ..
.   ...
173        0.61                  0.52             1.06
7.70  0.64
174        0.75                  0.43             1.41
7.30  0.70
175        0.69                  0.43             1.35
10.20  0.59
176        0.68                  0.53             1.46
9.30  0.60
177        0.76                  0.56             1.35
9.20  0.61

     od280/od315_of_diluted_wines  proline
0                            3.92   1065.0
1                            3.40   1050.0
2                            3.17   1185.0
3                            3.45   1480.0
4                            2.93    735.0
..                            ...      ...
173                          1.74    740.0
174                          1.56    750.0
175                          1.56    835.0
176                          1.62    840.0
177                          1.60    560.0

[178 rows x 13 columns]
```

```
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)
```

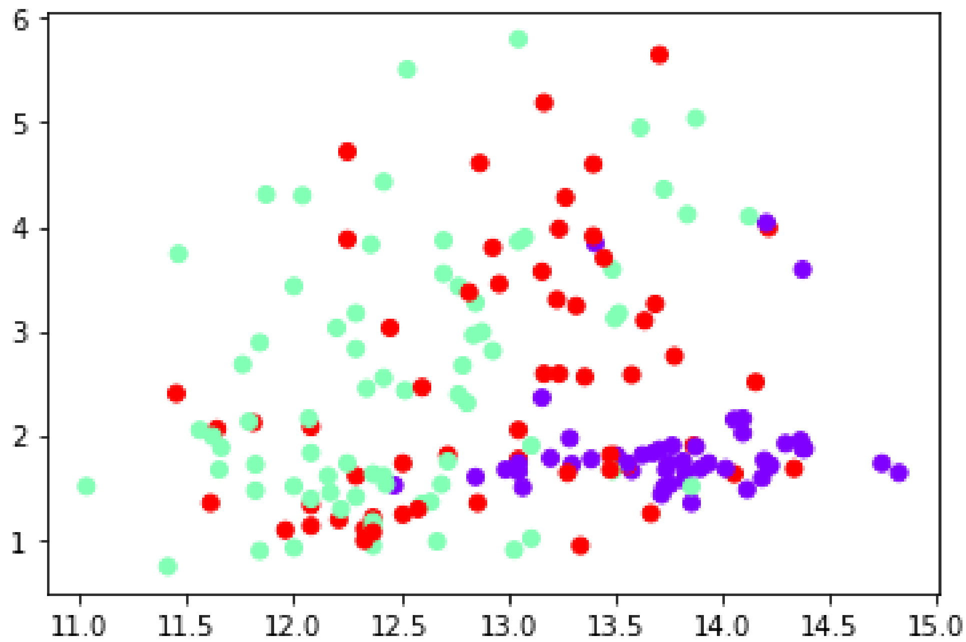
```
Text(0, 0.5, 'malic_acid')
```




```
from sklearn.cluster import AgglomerativeClustering

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

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

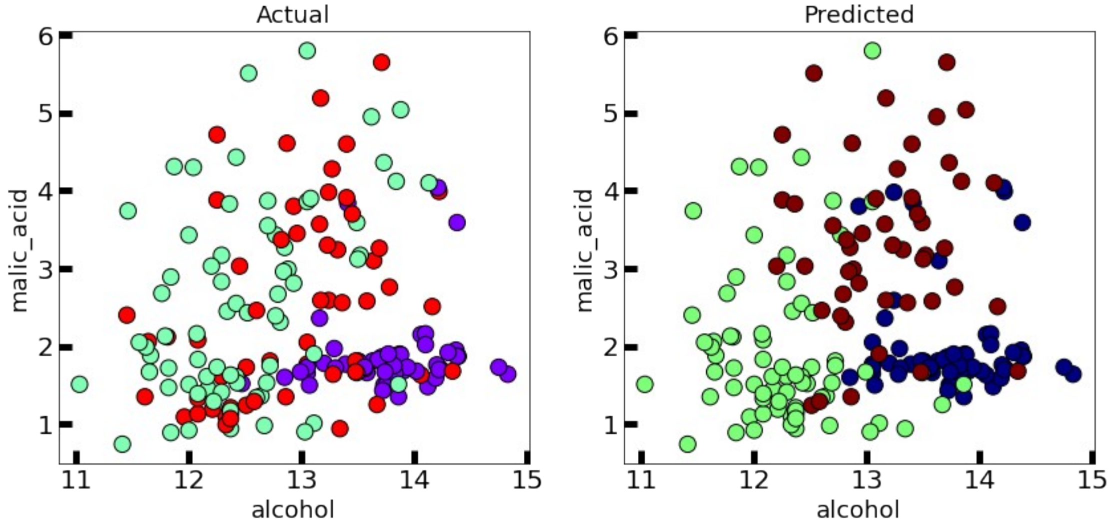
```
Cluster Labels
[0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 0 0 2 2 0 0 2 0 0 0 0 0 0
2 2
 0 0 2 2 0 0 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 1 2 1 1 2 1 1 2 2 2 1
1 0
 2 1 1 1 2 1 1 2 2 1 1 1 1 1 2 2 1 1 1 1 1 0 2 1 2 1 2 1 1 1 2 1 1 1 1
2 1
 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 2 2 2 1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

```
plt.scatter(x=df['alcohol'], y=df['malic_acid'] ,c=cluster.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)
```

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




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

```
The silhouette score is :
```

```
0.5644796401732074
```

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

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

```
552.851711505718
```

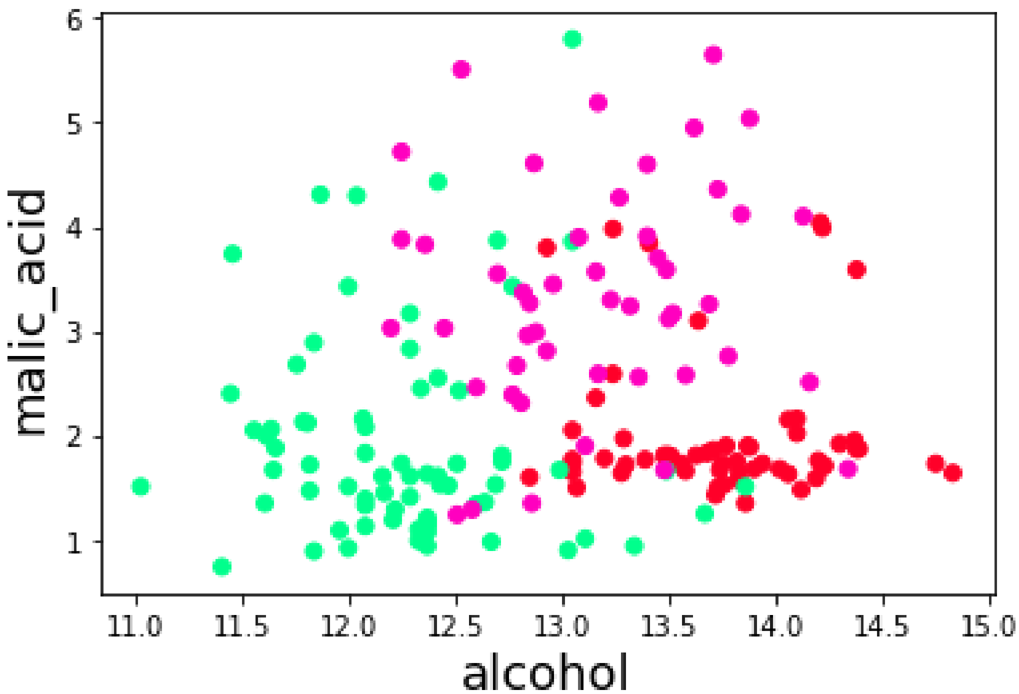
```
from sklearn.metrics import davies_bouldin_score
print("The davies bouldin score is :")
davies_bouldin_score(x, cluster.labels_)
```

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

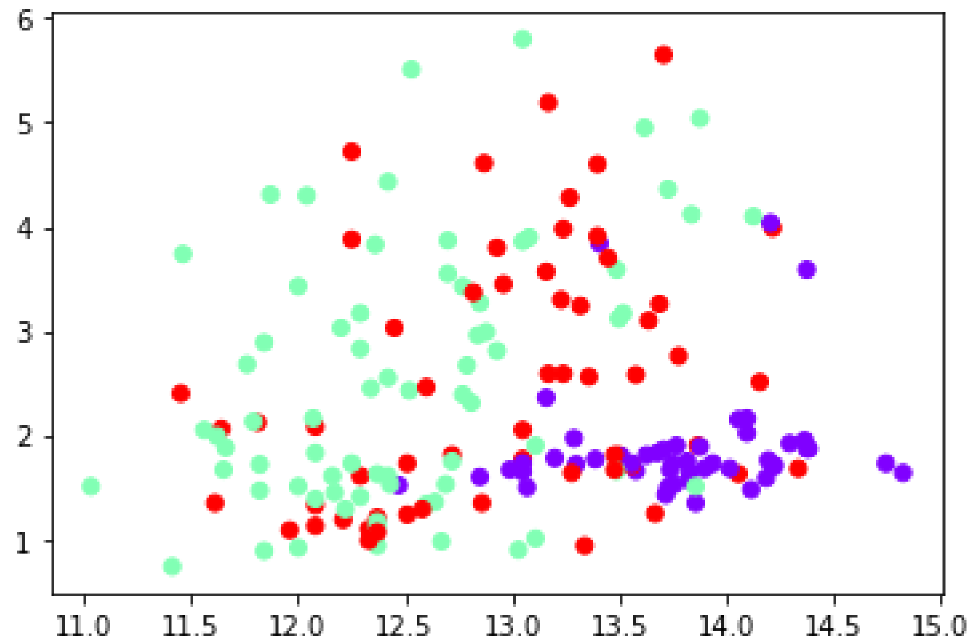
```
0.5357343073560216
```

```
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')
```

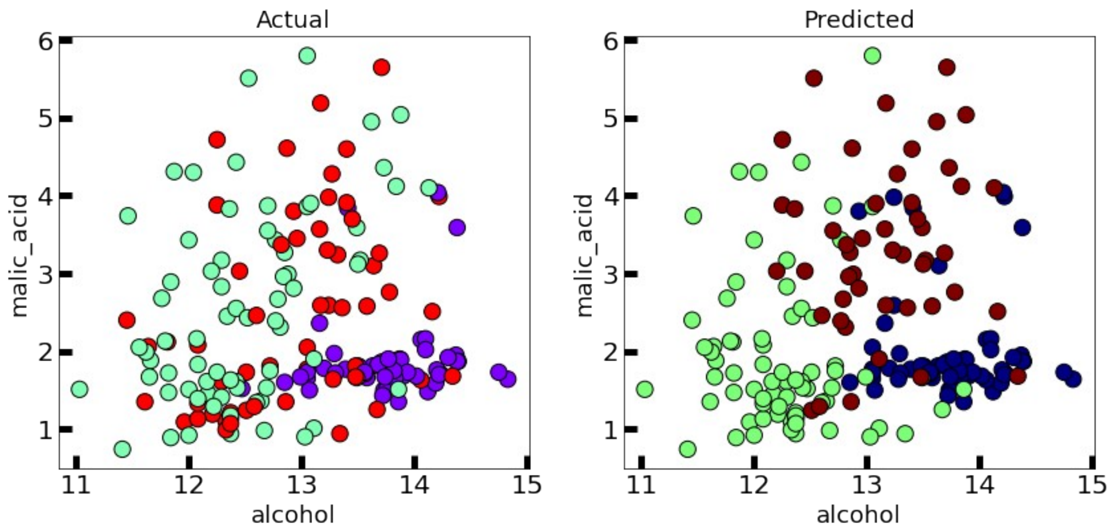



```
from sklearn.cluster import Birch

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

print("Cluster Labels")
print(cluster.labels_)

Cluster Labels
[0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 0 0 2 2 0 0 2 0 0 0 0 0 0
2 2
 0 0 2 2 0 0 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 1 2 1 1 2 1 1 2 2 2 1
1 0
 2 1 1 1 2 1 1 2 2 1 1 1 1 1 2 2 1 1 1 1 1 0 2 1 2 1 2 1 1 1 2 1 1 1 1
2 1
 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 2 2 2 1 1 1 1 2 1 1 2 2
1 2
 2 1 1 1 1 2 2 2 1 2 2 2 1 2 1 2 2 1 2 2 2 2 1 1 2 2 2 2 2 1]

plt.scatter(x=df['alcohol'], y=df['malic_acid'] ,c=birch.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)
```

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




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

```
The silhouette score is :
```

```
0.5644796401732074
```

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

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

```
552.851711505718
```