Name :	**Dipangshu Roy**
Roll No :	**001811001014**
Department :	**Information Technology**
Class :	**4th Year 1st Sem**
Subject :	**Machine Learning Lab**

Assignment 2

Construct a machine learning based model for classification using Python for the following UCI datasets:

UCI datasets (can be loaded from the package itself):

1. Iris plants dataset : https://archive.ics.uci.edu/ml/datasets/Iris/
2. Wine Dataset: https://archive.ics.uci.edu/ml/datasets/wine
3. Ionosphere Dataset: https://archive.ics.uci.edu/ml/datasets/Ionosphere
4. Wisconsin Breast Cancer Dataset : https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

> **Note :** I have done all the tasks of this assignment on Google Collab Platform. So, Here is the link of the Google Collab Notebook for further references :
>
> https://colab.research.google.com/drive/19VLWxzZBEzuy9t8KpU4kqqbxSNlY2Tfh?usp=sharing

Implement and compare the following ML classifiers for all the three datasets and show the classification results (Accuracy, Precision, Recall, F-score, confusion matrix) with and without parameter tuning:

1. **SVM classifier (Linear, Polynomial, Gaussian, & Sigmoid)**
2. **MLP classifier (Momentum term, Epoch size and learning rate)**
3. **Random Forest classifier**

In the output of classification results, Generated images (heat map) of the confusion matrix for every experimentation are already included.

Random Forest classifier

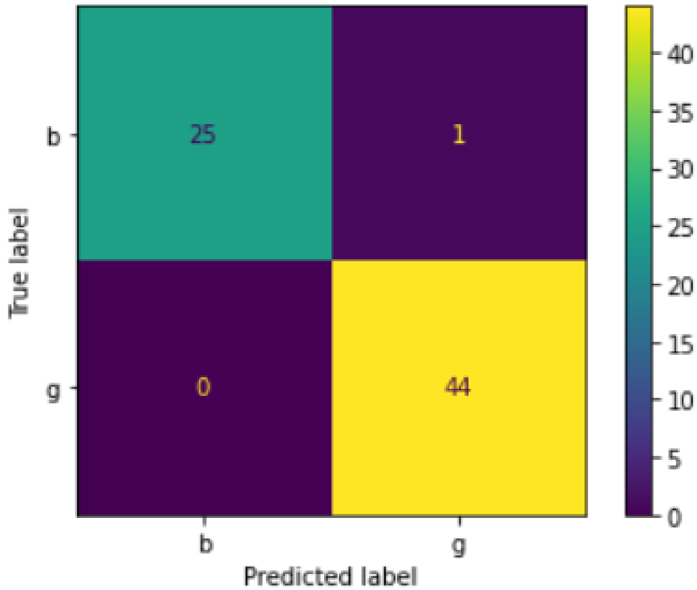
Code :

```
import pandas as pd
```

```python
# Dataset Preparation
dataset = pd.read_csv("drive/MyDrive/ML_As2/ionosphere.data");
dataset.columns = [ i for i in range(35) ]
X = dataset.drop(columns=[34])
y = dataset[34]

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=0)

# 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()

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

# Evaluation of Classifier Performance
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

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

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

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

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

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

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

Without parameter tuning :

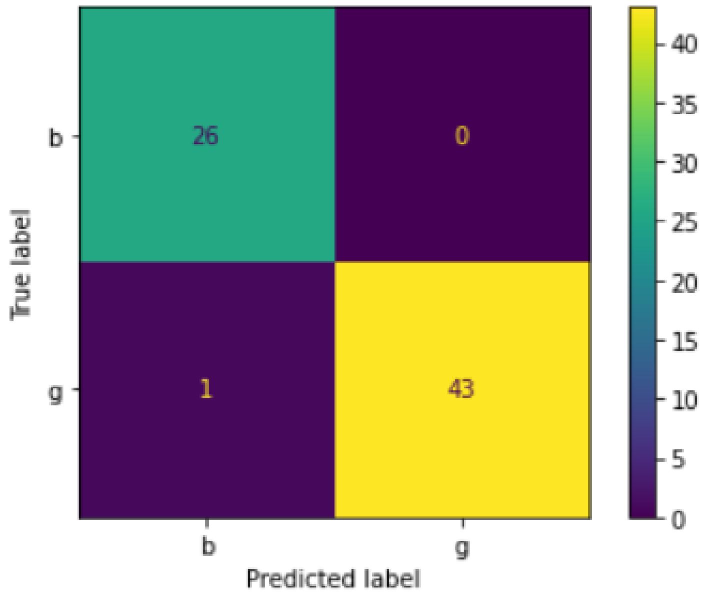
```
Confusion Matrix:
[[25  1]
 [ 0 44]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           b       1.00      0.96      0.98        26
           g       0.98      1.00      0.99        44

    accuracy                           0.99        70
   macro avg       0.99      0.98      0.98        70
weighted avg       0.99      0.99      0.99        70

------------------------------------------------------------
Accuracy:
0.9857142857142858
```



With parameter tuning :

```
# Classification
from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(criterion="entropy", n_estimators=20,random_state=0)

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

```
Confusion Matrix:
[[26  0]
 [ 1 43]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           b       0.96      1.00      0.98        26
           g       1.00      0.98      0.99        44

    accuracy                           0.99        70
   macro avg       0.98      0.99      0.98        70
weighted avg       0.99      0.99      0.99        70

------------------------------------------------------------
Accuracy:
0.9857142857142858
```



SVM classifier

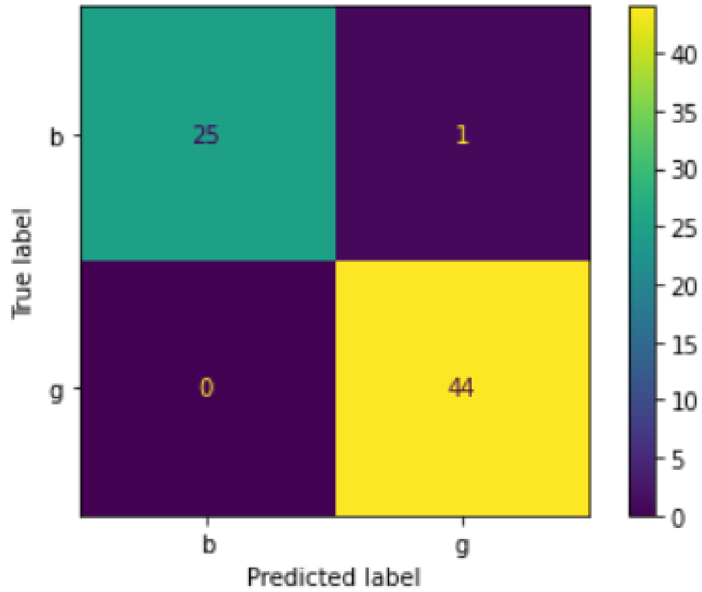
Code :

```
import pandas as pd

# Dataset Preparation
dataset = pd.read_csv("drive/MyDrive/ML_As2/ionosphere.data");
dataset.columns = [ i for i in range(35) ]
X = dataset.drop(columns=[34])
y = dataset[34]

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=0)

# Classification
from sklearn.svm import SVC
```

```python
classifier = SVC()

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

# Evaluation of Classifier Performance
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

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

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

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

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

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

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

Without parameter tuning :

```
Confusion Matrix:
[[25  1]
 [ 0 44]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           b       1.00      0.96      0.98        26
           g       0.98      1.00      0.99        44

    accuracy                           0.99        70
   macro avg       0.99      0.98      0.98        70
weighted avg       0.99      0.99      0.99        70

------------------------------------------------------------
Accuracy:
0.9857142857142858
```

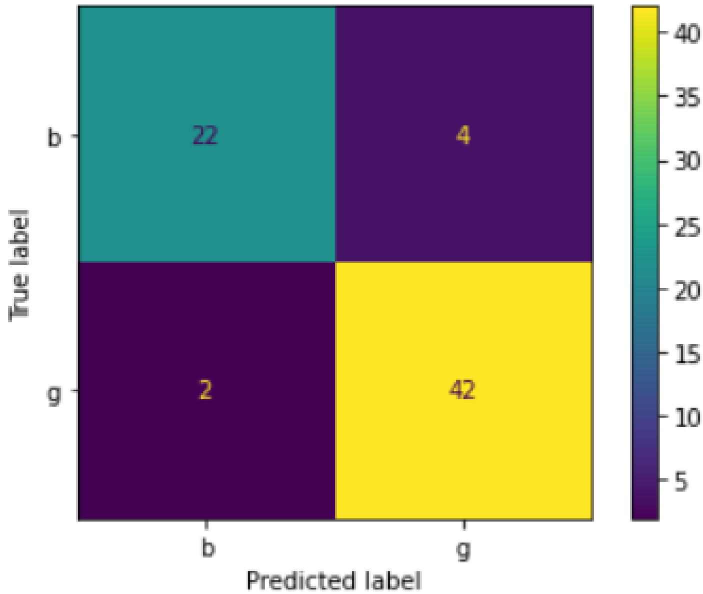


With parameter tuning :

> **Linear :**

```
# Classification
from sklearn.svm import SVC

classifier = SVC(kernel='linear')

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

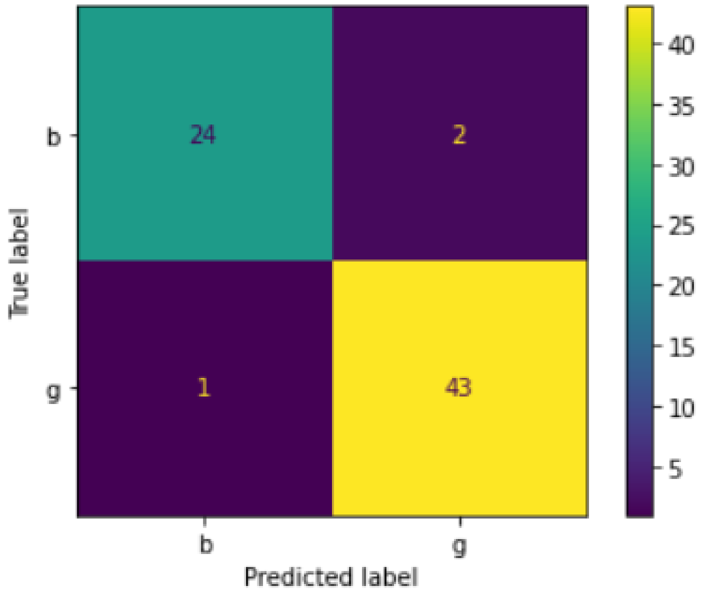
```
Confusion Matrix:
[[22  4]
 [ 2 42]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           b       0.92      0.85      0.88        26
           g       0.91      0.95      0.93        44

    accuracy                           0.91        70
   macro avg       0.91      0.90      0.91        70
weighted avg       0.91      0.91      0.91        70

------------------------------------------------------------
Accuracy:
0.9142857142857143
```



Polynomial :

```
# Classification
from sklearn.svm import SVC

classifier = SVC(kernel='poly',degree=2)

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

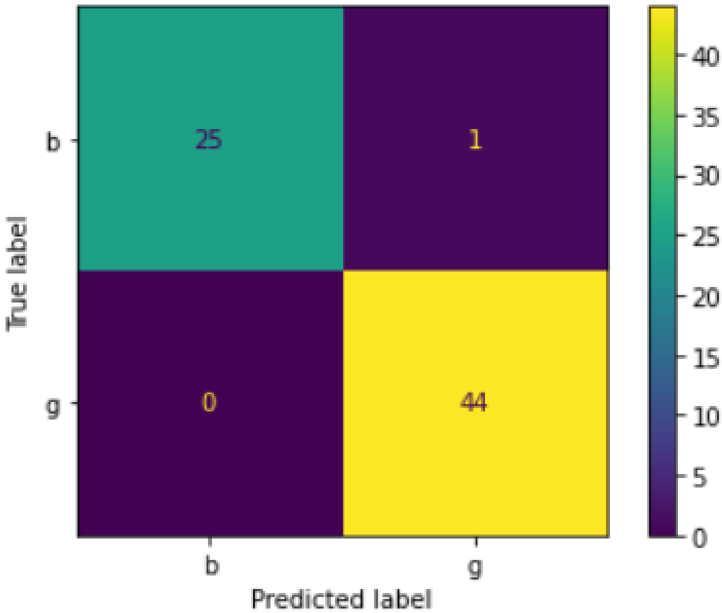
```
Confusion Matrix:
[[24  2]
 [ 1 43]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           b       0.96      0.92      0.94        26
           g       0.96      0.98      0.97        44

    accuracy                           0.96        70
   macro avg       0.96      0.95      0.95        70
weighted avg       0.96      0.96      0.96        70

------------------------------------------------------------
Accuracy:
0.9571428571428572
```



Gaussian :

```
# Classification
from sklearn.svm import SVC

classifier = SVC(kernel='rbf') # Gaussian Kernel

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

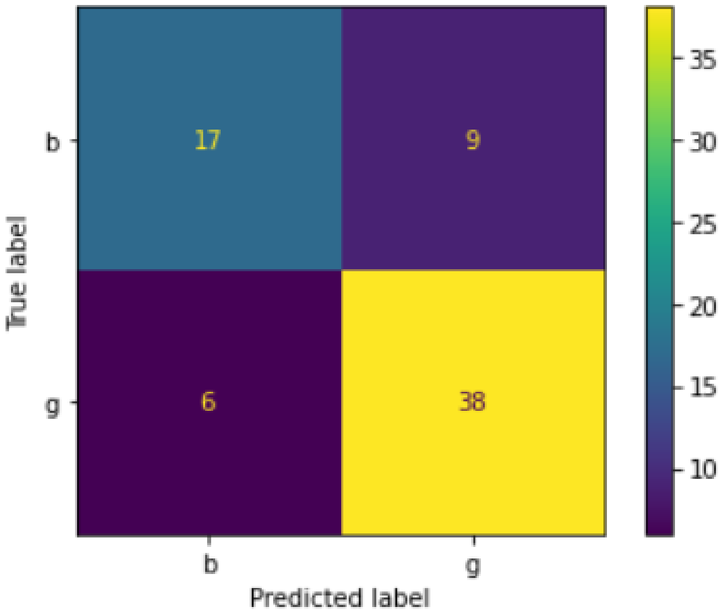
```
Confusion Matrix:
[[25  1]
 [ 0 44]]
-------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           b       1.00      0.96      0.98        26
           g       0.98      1.00      0.99        44

    accuracy                           0.99        70
   macro avg       0.99      0.98      0.98        70
weighted avg       0.99      0.99      0.99        70

-------------------------------------------------------------
Accuracy:
0.9857142857142858
```



Sigmoid :

```python
# Classification
from sklearn.svm import SVC

classifier = SVC(kernel='sigmoid')

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

```
Confusion Matrix:
[[17  9]
 [ 6 38]]
----------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           b       0.74      0.65      0.69        26
           g       0.81      0.86      0.84        44

    accuracy                           0.79        70
   macro avg       0.77      0.76      0.76        70
weighted avg       0.78      0.79      0.78        70

----------------------------------------------------------------
Accuracy:
0.7857142857142857
```



MLP classifier

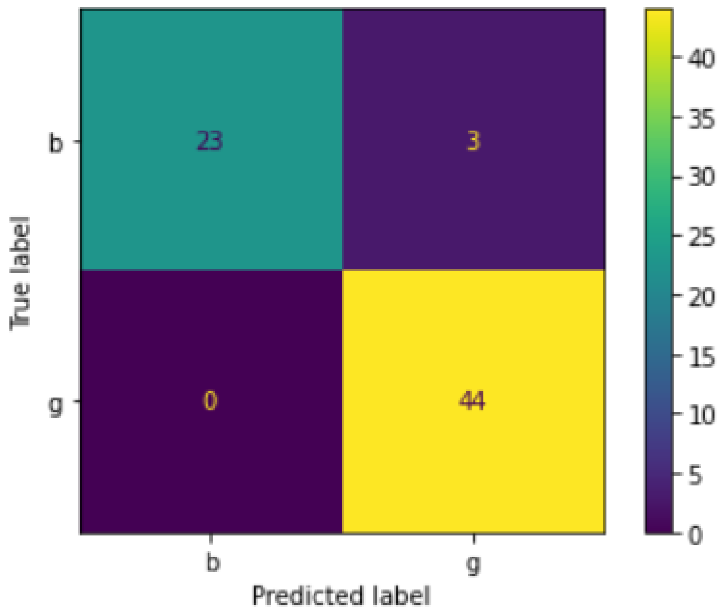
Code :

```
import pandas as pd

# Dataset Preparation
dataset = pd.read_csv("drive/MyDrive/ML_As2/ionosphere.data");
dataset.columns = [ i for i in range(35) ]
X = dataset.drop(columns=[34])
y = dataset[34]

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=0)
```

```python
# Classification
from sklearn.neural_network import MLPClassifier

classifier = MLPClassifier()

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

# Evaluation of Classifier Performance
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

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

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

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

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

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

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

Without parameter tuning :

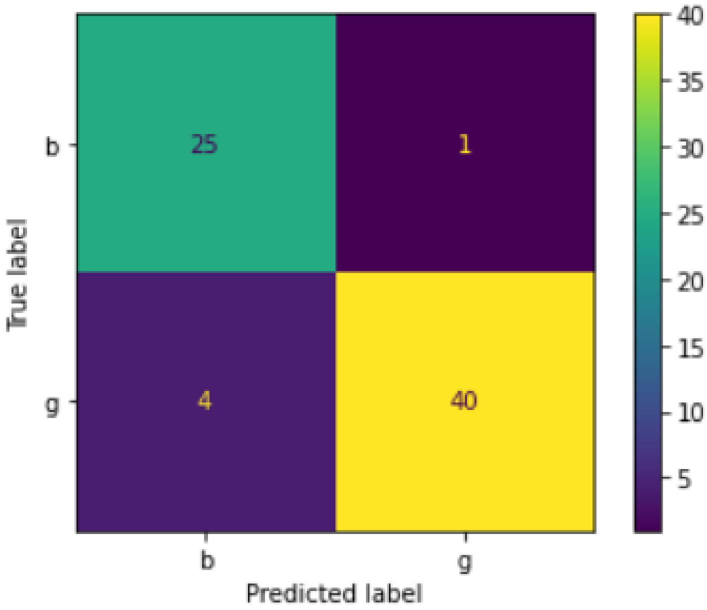
```
Confusion Matrix:
[[23  3]
 [ 0 44]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           b       1.00      0.88      0.94        26
           g       0.94      1.00      0.97        44

    accuracy                           0.96        70
   macro avg       0.97      0.94      0.95        70
weighted avg       0.96      0.96      0.96        70

------------------------------------------------------------
Accuracy:
0.9571428571428572
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network/_
  % self.max_iter, ConvergenceWarning)
```



With parameter tuning :

```python
# Classification
from sklearn.neural_network import MLPClassifier

classifier = MLPClassifier(hidden_layer_sizes=(10,10,10), max_iter=1000)

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

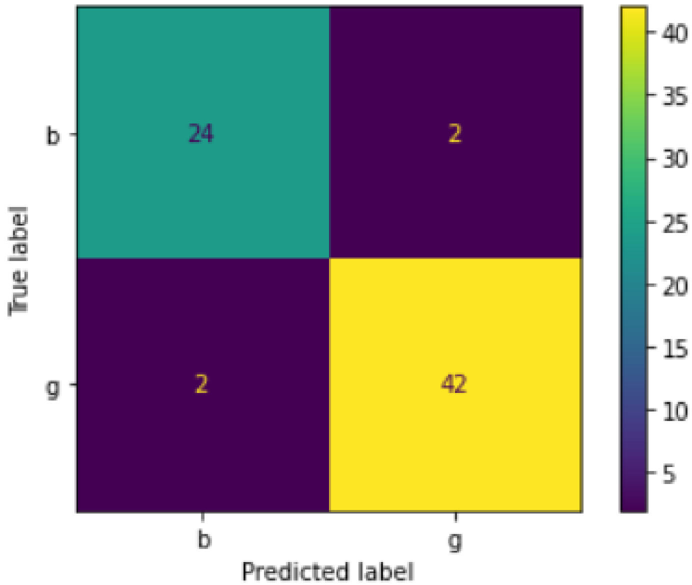
```
Confusion Matrix:
[[25  1]
 [ 4 40]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           b       0.86      0.96      0.91        26
           g       0.98      0.91      0.94        44

    accuracy                           0.93        70
   macro avg       0.92      0.94      0.93        70
weighted avg       0.93      0.93      0.93        70

------------------------------------------------------------
Accuracy:
0.9285714285714286
```



Momentum :

```
# Classification
from sklearn.neural_network import MLPClassifier

classifier = MLPClassifier(momentum=0.5, hidden_layer_sizes=(10,10,10), max_iter=1000)

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

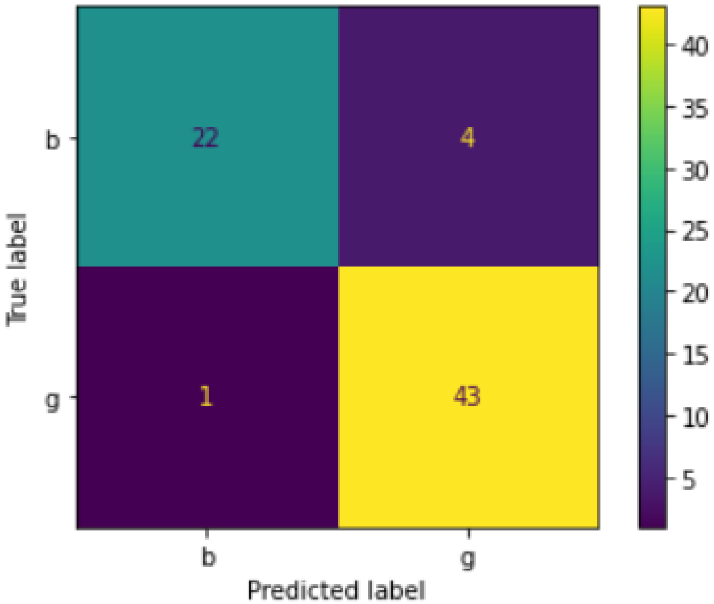
```
Confusion Matrix:
[[24  2]
 [ 2 42]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           b       0.92      0.92      0.92        26
           g       0.95      0.95      0.95        44

    accuracy                           0.94        70
   macro avg       0.94      0.94      0.94        70
weighted avg       0.94      0.94      0.94        70

------------------------------------------------------------
Accuracy:
0.9428571428571428
```



Learning Rate :

```
# Classification
from sklearn.neural_network import MLPClassifier

classifier = MLPClassifier(learning_rate='adaptive', hidden_layer_sizes=(10,10,10), max_iter=1000)

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

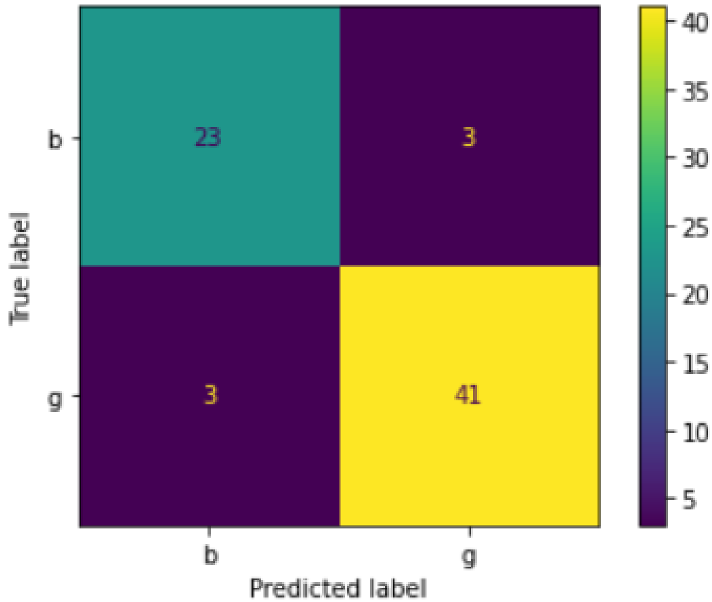
```
Confusion Matrix:
[[22  4]
 [ 1 43]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           b       0.96      0.85      0.90        26
           g       0.91      0.98      0.95        44

    accuracy                           0.93        70
   macro avg       0.94      0.91      0.92        70
weighted avg       0.93      0.93      0.93        70

------------------------------------------------------------
Accuracy:
0.9285714285714286
```



Activation :

```
# Classification
from sklearn.neural_network import MLPClassifier

classifier = MLPClassifier(activation='identity' , hidden_layer_sizes=(10,10,10), max_iter=1000)

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

```
Confusion Matrix:
[[23  3]
 [ 3 41]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           b       0.88      0.88      0.88        26
           g       0.93      0.93      0.93        44

    accuracy                           0.91        70
   macro avg       0.91      0.91      0.91        70
weighted avg       0.91      0.91      0.91        70

------------------------------------------------------------
Accuracy:
0.9142857142857143
```



Apply different values of train-test set splits (70:30, 60:40, 50:50, 40:60 and 30:70) and report the corresponding results for both the classifiers.

Here, I am implementing all the three classifiers for the given four datasets with following parameters :

Random Forest Classifier Parameters :

```
# 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)
```

SVM Classifier Parameters :

```
# Classification
from sklearn.svm import SVC

classifier = SVC(kernel='linear')

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

MLP Classifier Parameters :

```
# Classification
from sklearn.neural_network import MLPClassifier

classifier = MLPClassifier(hidden_layer_sizes=(10,10,10), max_iter=1000)

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

Here, is the Table for the accuracy of each datasets on implementing each of the classifiers with different values of train-test set splits (70:30, 60:40, 50:50, 40:60 and 30:70).

> **Accuracy Table :**

```
+---------------+---------------+----------------+------------+
| Dataset       | Classifier    | Train : Test   |   Accuracy |
+===============+===============+================+============+
| Iris Plants   | Random Forest | 70 : 30        |   0.888889 |
+---------------+---------------+----------------+------------+
| Iris Plants   | Random Forest | 60 : 40        |   0.933333 |
+---------------+---------------+----------------+------------+
| Iris Plants   | Random Forest | 50 : 50        |   0.933333 |
+---------------+---------------+----------------+------------+
| Iris Plants   | Random Forest | 40 : 60        |   0.933333 |
+---------------+---------------+----------------+------------+
| Iris Plants   | Random Forest | 30 : 70        |   0.942857 |
```

Iris Plants	SVM	70 : 30	0.911111
Iris Plants	SVM	60 : 40	0.966667
Iris Plants	SVM	50 : 50	0.96
Iris Plants	SVM	40 : 60	0.955556
Iris Plants	SVM	30 : 70	0.952381
Iris Plants	MLP	70 : 30	0.955556
Iris Plants	MLP	60 : 40	0.966667
Iris Plants	MLP	50 : 50	0.973333
Iris Plants	MLP	40 : 60	0.955556
Iris Plants	MLP	30 : 70	0.952381
Wine	Random Forest	70 : 30	1
Wine	Random Forest	60 : 40	0.957746
Wine	Random Forest	50 : 50	0.988764
Wine	Random Forest	40 : 60	0.953271
Wine	Random Forest	30 : 70	0.983871
Wine	SVM	70 : 30	0.925926
Wine	SVM	60 : 40	0.929577
Wine	SVM	50 : 50	0.94382
Wine	SVM	40 : 60	0.953271
Wine	SVM	30 : 70	0.959677
Wine	MLP	70 : 30	0.351852
Wine	MLP	60 : 40	0.971831
Wine	MLP	50 : 50	0.932584
Wine	MLP	40 : 60	0.560748
Wine	MLP	30 : 70	0.951613
Ionosphere	Random Forest	70 : 30	0.942857
Ionosphere	Random Forest	60 : 40	0.95
Ionosphere	Random Forest	50 : 50	0.908571
Ionosphere	Random Forest	40 : 60	0.933333
Ionosphere	Random Forest	30 : 70	0.897959

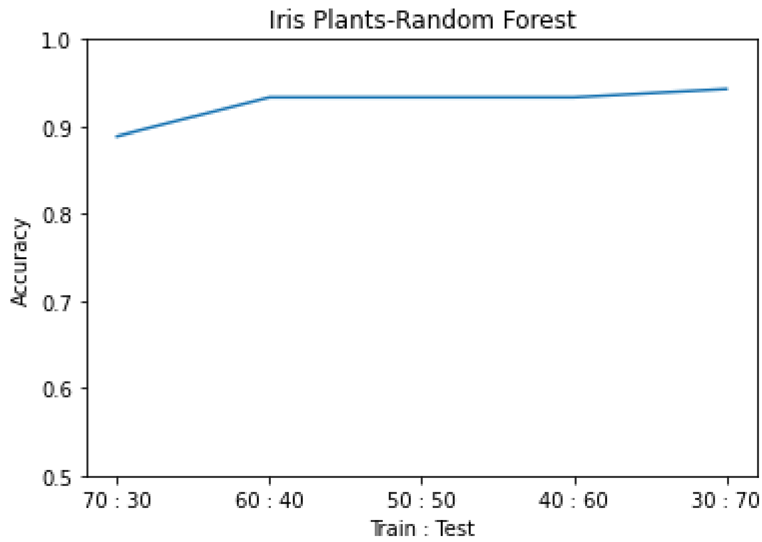
Ionosphere	SVM	70 : 30	0.87619
Ionosphere	SVM	60 : 40	0.857143
Ionosphere	SVM	50 : 50	0.845714
Ionosphere	SVM	40 : 60	0.861905
Ionosphere	SVM	30 : 70	0.869388
Ionosphere	MLP	70 : 30	0.885714
Ionosphere	MLP	60 : 40	0.95
Ionosphere	MLP	50 : 50	0.874286
Ionosphere	MLP	40 : 60	0.87619
Ionosphere	MLP	30 : 70	0.877551
Breast Cancer	Random Forest	70 : 30	0.980488
Breast Cancer	Random Forest	60 : 40	0.970696
Breast Cancer	Random Forest	50 : 50	0.970674
Breast Cancer	Random Forest	40 : 60	0.958537
Breast Cancer	Random Forest	30 : 70	0.958159
Breast Cancer	SVM	70 : 30	0.97561
Breast Cancer	SVM	60 : 40	0.974359
Breast Cancer	SVM	50 : 50	0.958944
Breast Cancer	SVM	40 : 60	0.958537
Breast Cancer	SVM	30 : 70	0.958159
Breast Cancer	MLP	70 : 30	0.965854
Breast Cancer	MLP	60 : 40	0.952381
Breast Cancer	MLP	50 : 50	0.947214
Breast Cancer	MLP	40 : 60	0.929268
Breast Cancer	MLP	30 : 70	0.92887

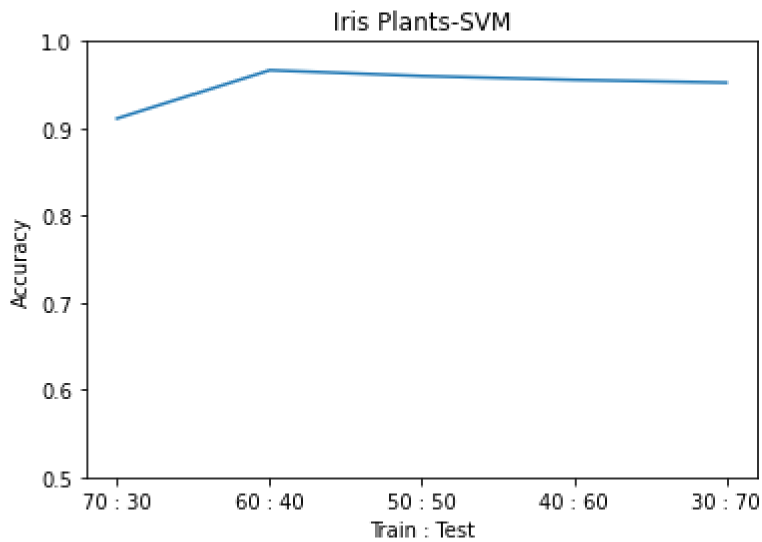
Graph of Accuracy vs Train : Test splits for each Datasets

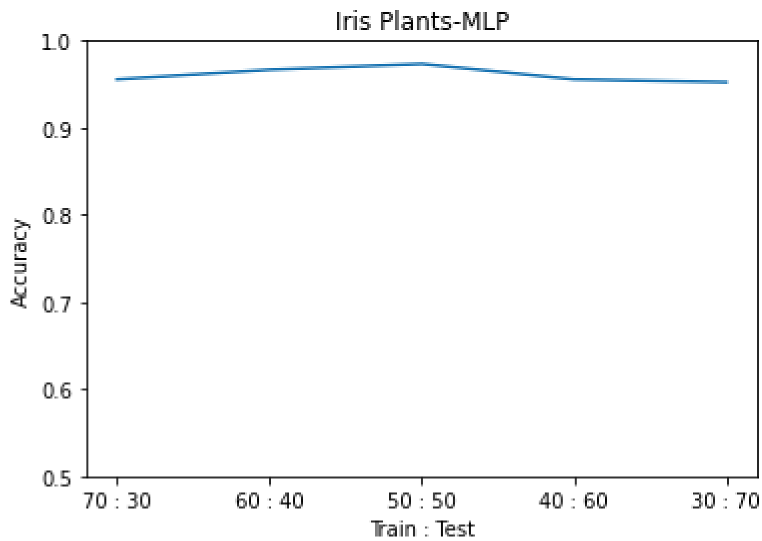
Here, the data of the above mentioned table is plotted in graph for visualization.

Each graph of the datasets are plotted with respect to a particular ML classifier and on X-axis, there are the ratio of train test splits & on Y-axis the accuracy is given..

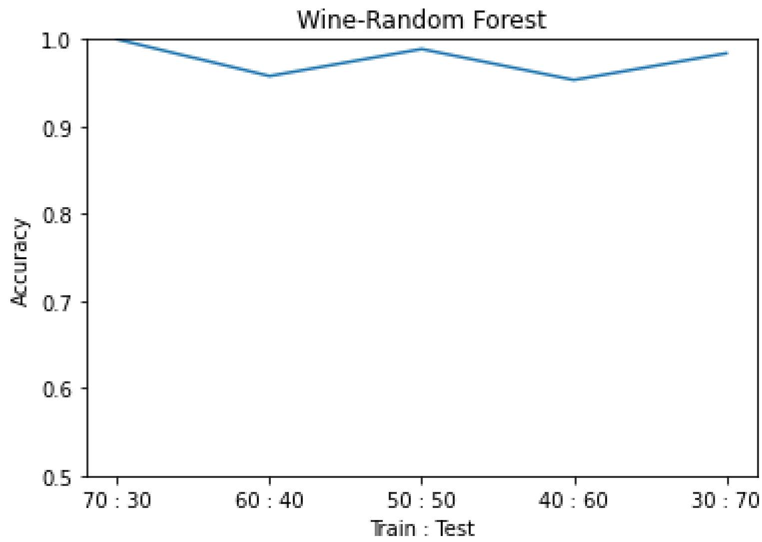
Iris Plants Dataset :

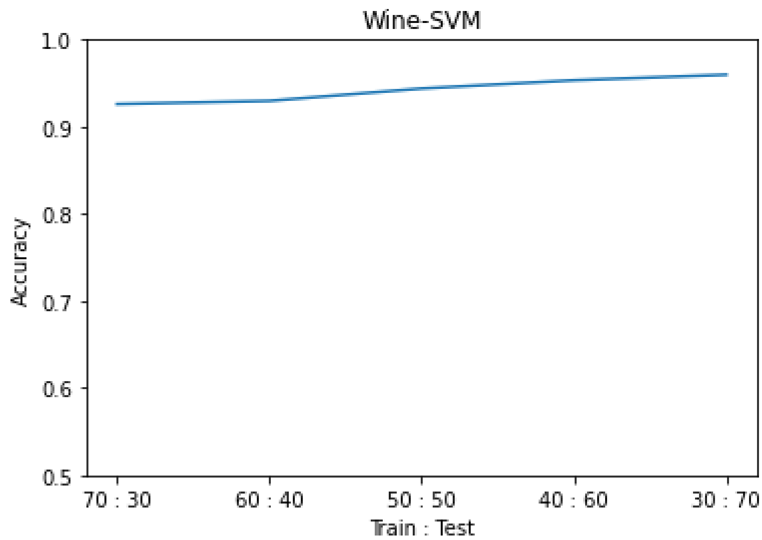


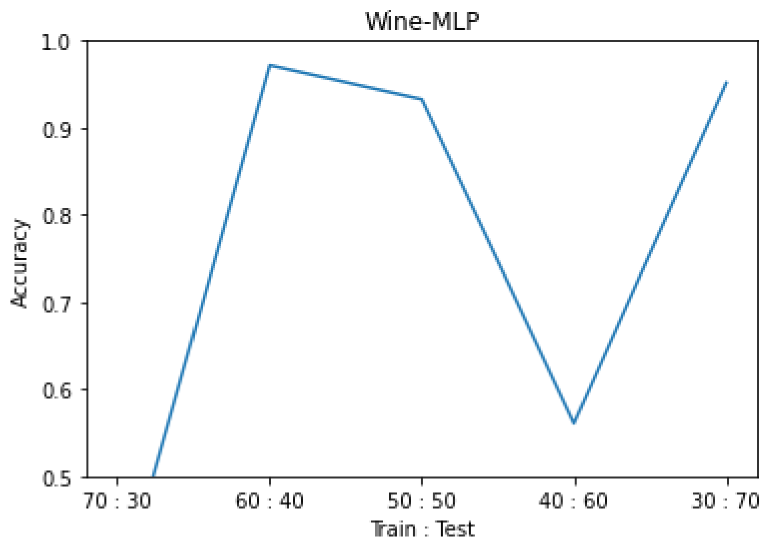




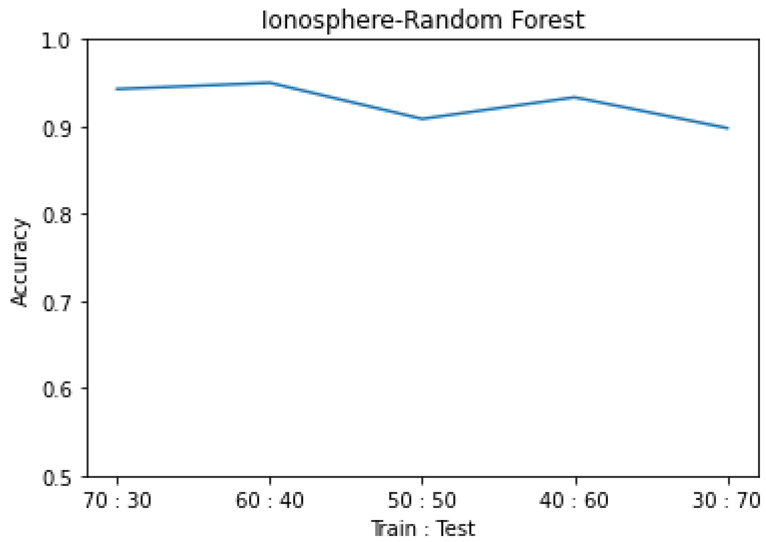
Wine Dataset :

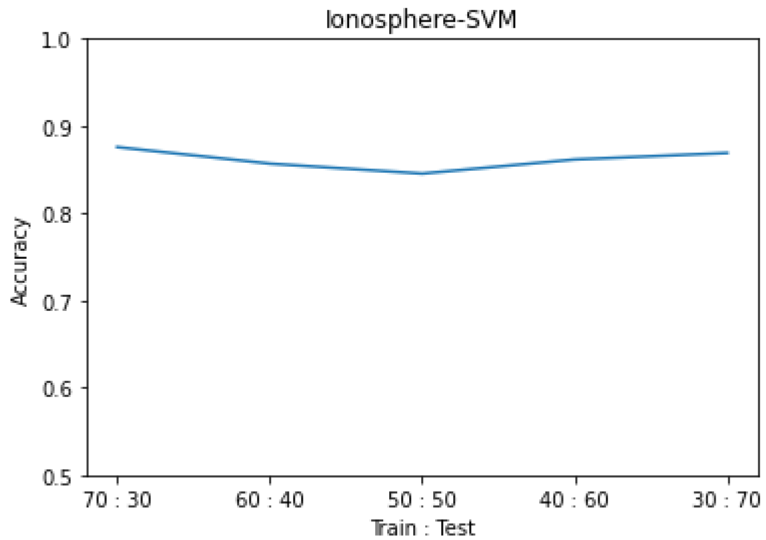


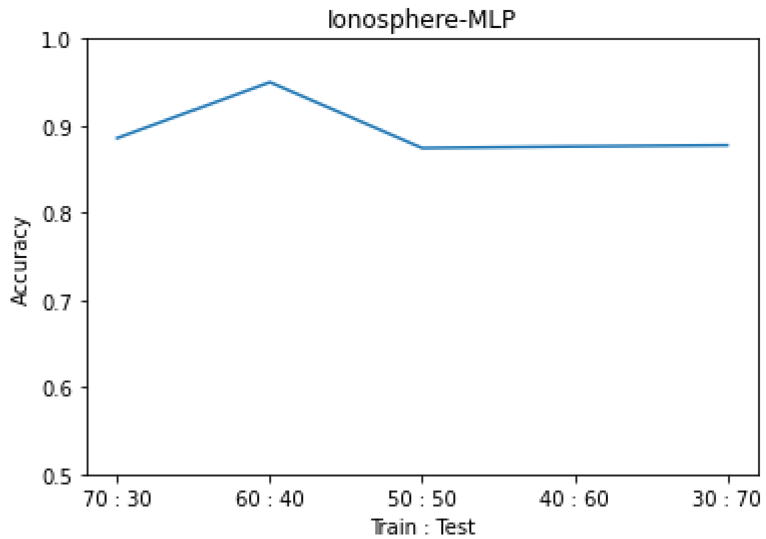




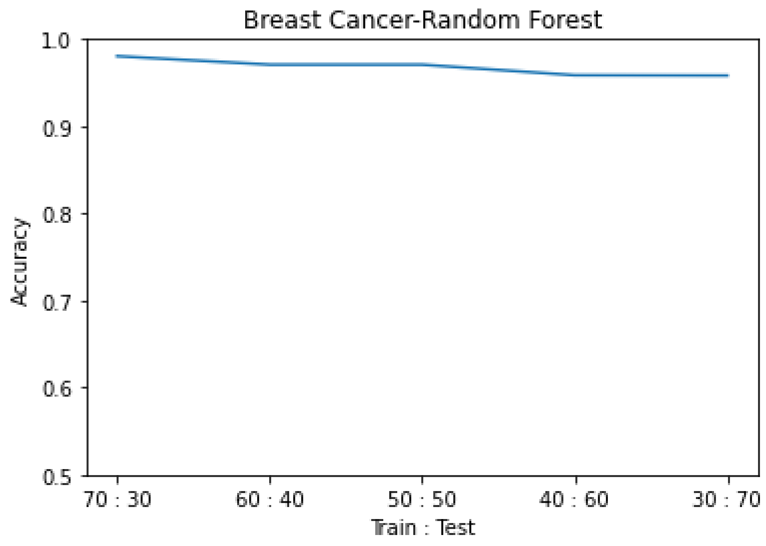
Ionosphere Dataset :

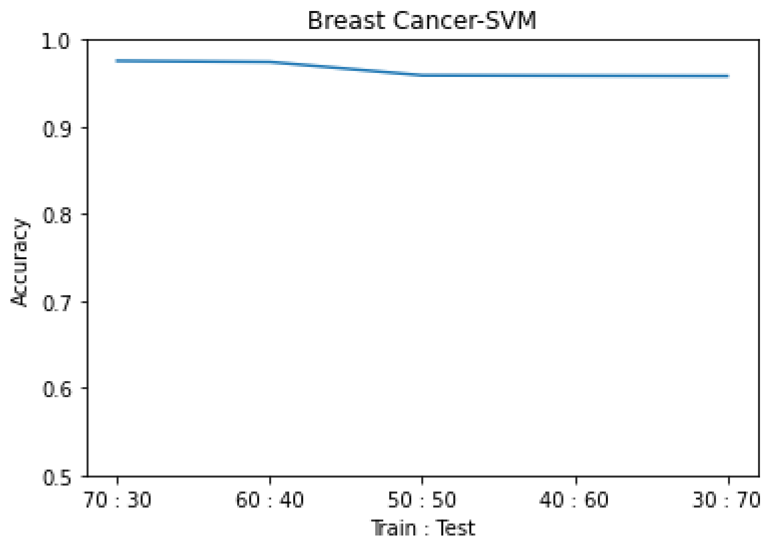


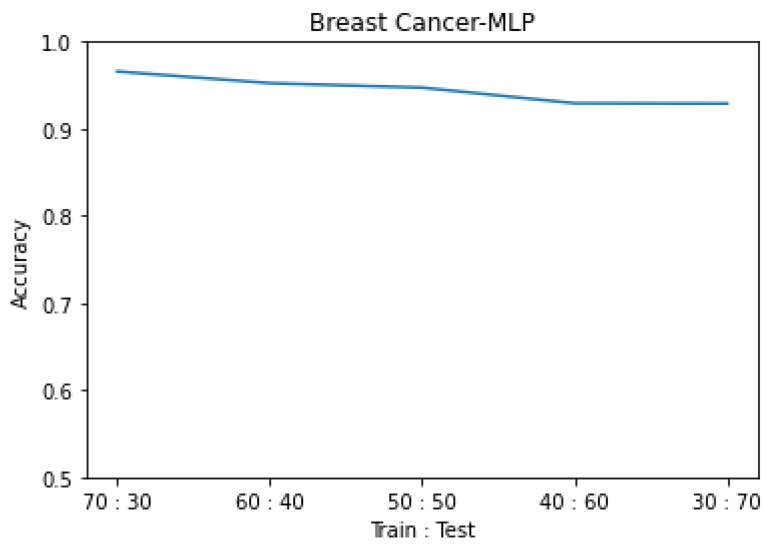




Breast Cancer Dataset :


Breast Cancer-Random Forest
Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
70 : 30
60 : 40
50 : 50
40 : 60
30 : 70
Train : Test


Breast Cancer-SVM
Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
70 : 30
60 : 40
50 : 50
40 : 60
30 : 70
Train : Test


Breast Cancer-MLP
Accuracy
1.0
0.9
0.8
0.7
0.6
0.5
70 : 30
60 : 40
50 : 50
40 : 60
30 : 70
Train : Test

Use Principal Component Analysis (PCA) for feature dimensionality reduction and again apply the above 3 ML classifiers on the reduced feature set. Show the classification results (Accuracy, Precision, Recall, F-score, confusion matrix).

Here, Principal Component Analysis (PCA) will be used before applying the ML classifiers to calculate classification results.

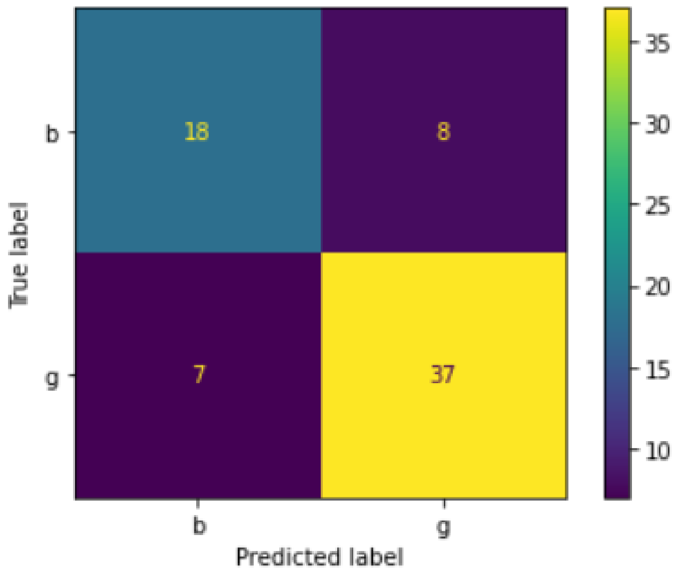
Random Forest Classifier :

```
import pandas as pd

# Dataset Preparation
dataset = pd.read_csv("drive/MyDrive/ML_As2/ionosphere.data");
dataset.columns = [ i for i in range(35) ]
X = dataset.drop(columns=[34])
y = dataset[34]

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler

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

print("Before dimensionality reduction the dimensions of X_train are :")
print(X_train.shape)

# Applying PCA function on training and testing set of X component
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)

X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)

explained_variance = pca.explained_variance_ratio_

print("After dimensionality reduction the dimensions of X_train are :")
print(X_train.shape)

# Classification
from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier( criterion="entropy",
n_estimators=20,random_state=0)

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

# Evaluation of Classifier Performance
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print("\nConfusion Matrix:")
```

```python
print(confusion_matrix(y_test, y_pred))

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

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

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

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

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

Classification Result :

```
Before dimensionality reduction the dimensions of X_train are :
(280, 34)
After dimensionality reduction the dimensions of X_train are :
(280, 2)

Confusion Matrix:
[[18  8]
 [ 7 37]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           b       0.72      0.69      0.71        26
           g       0.82      0.84      0.83        44

    accuracy                           0.79        70
   macro avg       0.77      0.77      0.77        70
weighted avg       0.78      0.79      0.78        70

------------------------------------------------------------
Accuracy:
0.7857142857142857
```



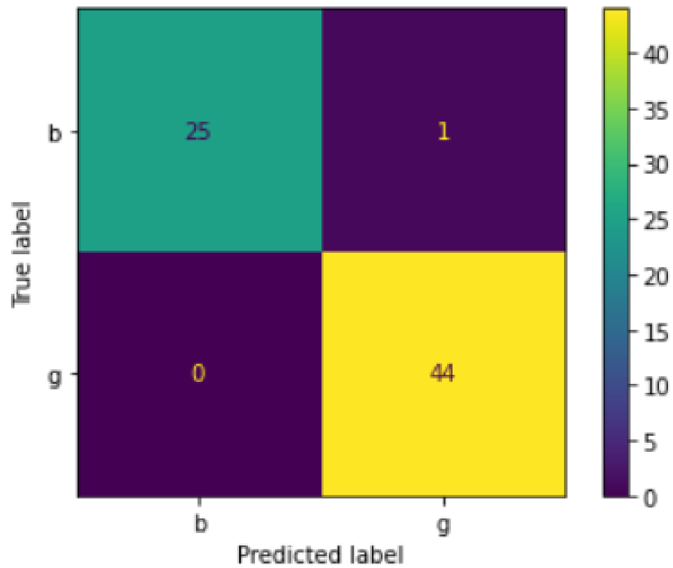
SVM Classifier :

```
import pandas as pd

# Dataset Preparation
dataset = pd.read_csv("drive/MyDrive/ML_As2/ionosphere.data");
dataset.columns = [ i for i in range(35) ]
X = dataset.drop(columns=[34])
y = dataset[34]

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=0)

print("Before dimensionality reduction the dimensions of X_train are :")
print(X_train.shape)
```

```python
# Applying PCA function on training and testing set of X component
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)

X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)

explained_variance = pca.explained_variance_ratio_

print("After dimensionality reduction the dimensions of X_train are :")
print(X_train.shape)

# Classification
from sklearn.svm import SVC

classifier = SVC()

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

# Evaluation of Classifier Performance
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

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

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

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

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

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

Classification Result :

```
Before dimensionality reduction the dimensions of X_train are :
(280, 34)
After dimensionality reduction the dimensions of X_train are :
(280, 34)

Confusion Matrix:
[[25  1]
 [ 0 44]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           b       1.00      0.96      0.98        26
           g       0.98      1.00      0.99        44

    accuracy                           0.99        70
   macro avg       0.99      0.98      0.98        70
weighted avg       0.99      0.99      0.99        70

------------------------------------------------------------
Accuracy:
0.9857142857142858
```



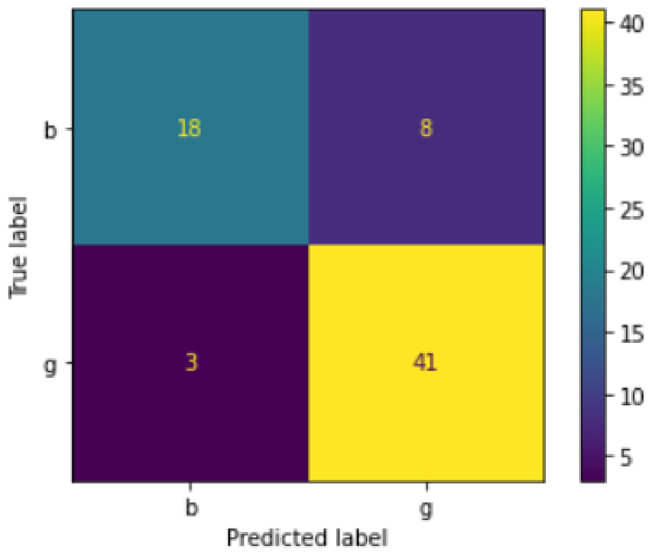
MLP Classifier :

```
import pandas as pd

# Dataset Preparation
dataset = pd.read_csv("drive/MyDrive/ML_As2/ionosphere.data");
dataset.columns = [ i for i in range(35) ]
X = dataset.drop(columns=[34])
y = dataset[34]

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=0)

print("Before dimensionality reduction the dimensions of X_train are :")
```

```python
print(X_train.shape)

# Applying PCA function on training and testing set of X component
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)

X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)

explained_variance = pca.explained_variance_ratio_

print("After dimensionality reduction the dimensions of X_train are :")
print(X_train.shape)

# Classification
from sklearn.neural_network import MLPClassifier

classifier = MLPClassifier( max_iter=1000)

classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

# Evaluation of Classifier Performance
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

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

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

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

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

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

Classification Result :

```
Before dimensionality reduction the dimensions of X_train are :
(280, 34)
After dimensionality reduction the dimensions of X_train are :
(280, 2)

Confusion Matrix:
[[18  8]
 [ 3 41]]
------------------------------------------------------------
Performance Evaluation:
              precision    recall  f1-score   support

           b       0.86      0.69      0.77        26
           g       0.84      0.93      0.88        44

    accuracy                           0.84        70
   macro avg       0.85      0.81      0.82        70
weighted avg       0.84      0.84      0.84        70

------------------------------------------------------------
Accuracy:
0.8428571428571429
```



Show the performance comparison among classifiers in a table.

Here, In this table the performance of the classifiers is compared while not using the PCA and while using PCA.

Classifier	Accuracy	Accuracy (PCA)
Random Forest	0.98	0.79
SVM	0.99	0.99
MLP	0.93	0.84