Machine Learning Lab
Assignment 2

Name - Md Faiz Ansari
Roll - 001811001045
Semester - 7
Year - 4
Department - Information Technology

Drive : - https://drive.google.com/drive/folders/1nL0VZxTM1XUf5NPpteQJk2PLK8DDAPL6?usp=sharing

1. WINE DATASET

1.1 SVM Classifier(With Tuning)

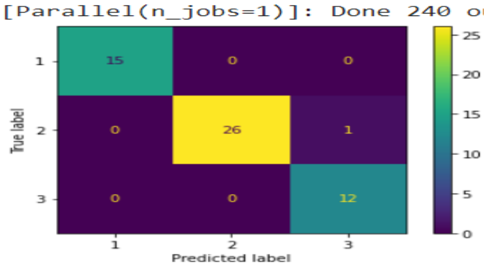
```
# WINE DATASET
# SVM(With Tuning)[70-30 split]


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.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']}
```

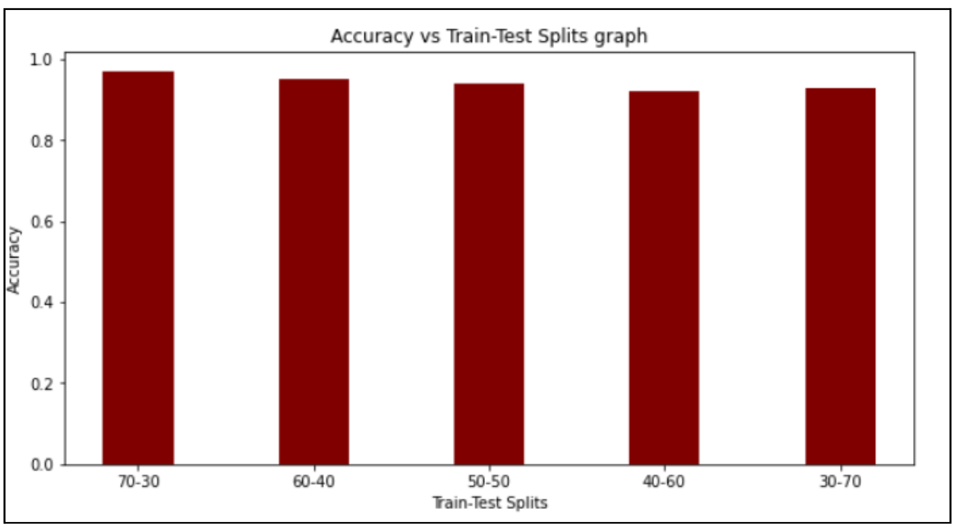
```
pprint(param_grid)

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

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()
```

```
Confusion Matrix:
[[15  0  0]
 [ 0 26  1]
 [ 0  0 12]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        15
           2       1.00      0.96      0.98        27
           3       0.92      1.00      0.96        12

    accuracy                           0.98        54
   macro avg       0.97      0.99      0.98        54
weighted avg       0.98      0.98      0.98        54

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9814814814814815
[Parallel(n_jobs=1)]: Done 240 out of 240 | elapsed:    0.9s finished
```



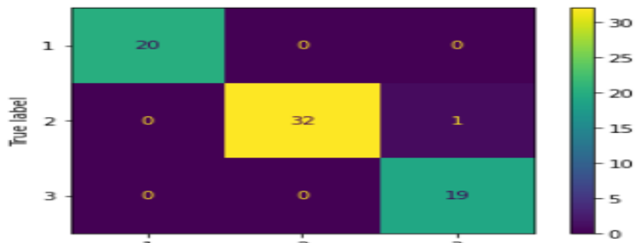
COMPARISON:



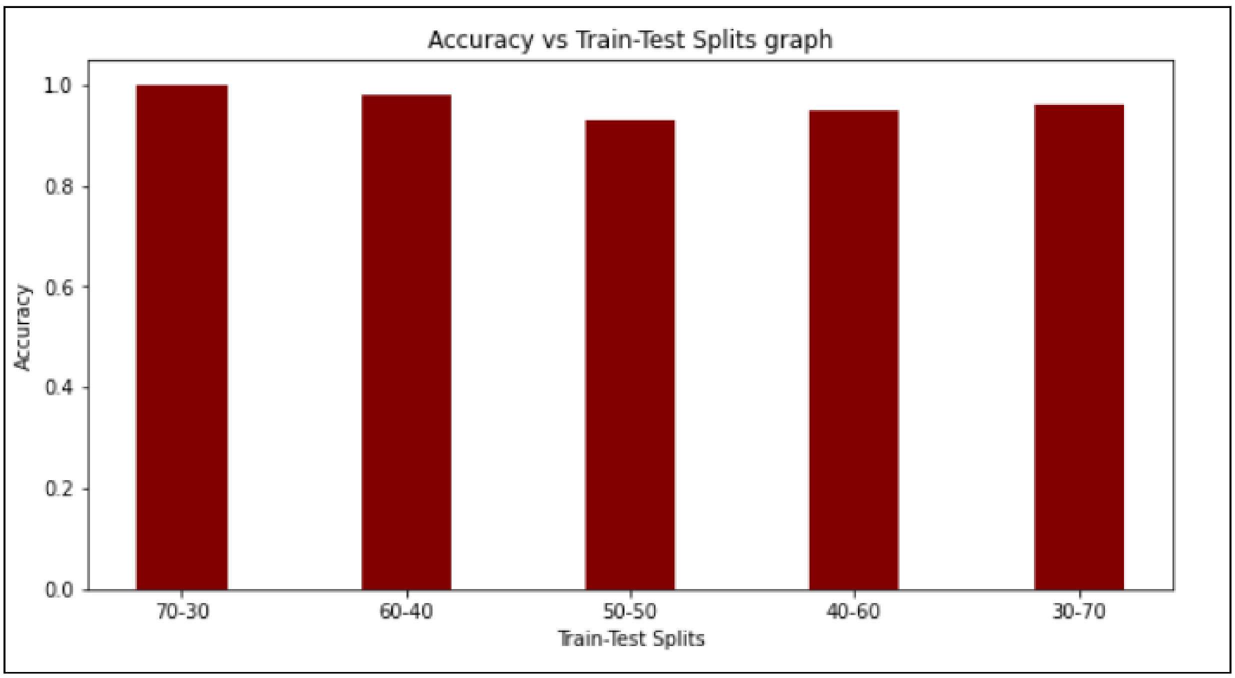
Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 70:30.

1.2 SVM Classifier(Without Tuning)

```
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
```



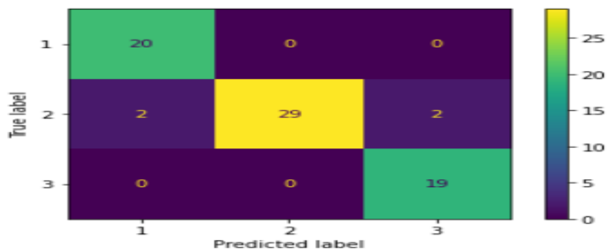
COMPARISON:



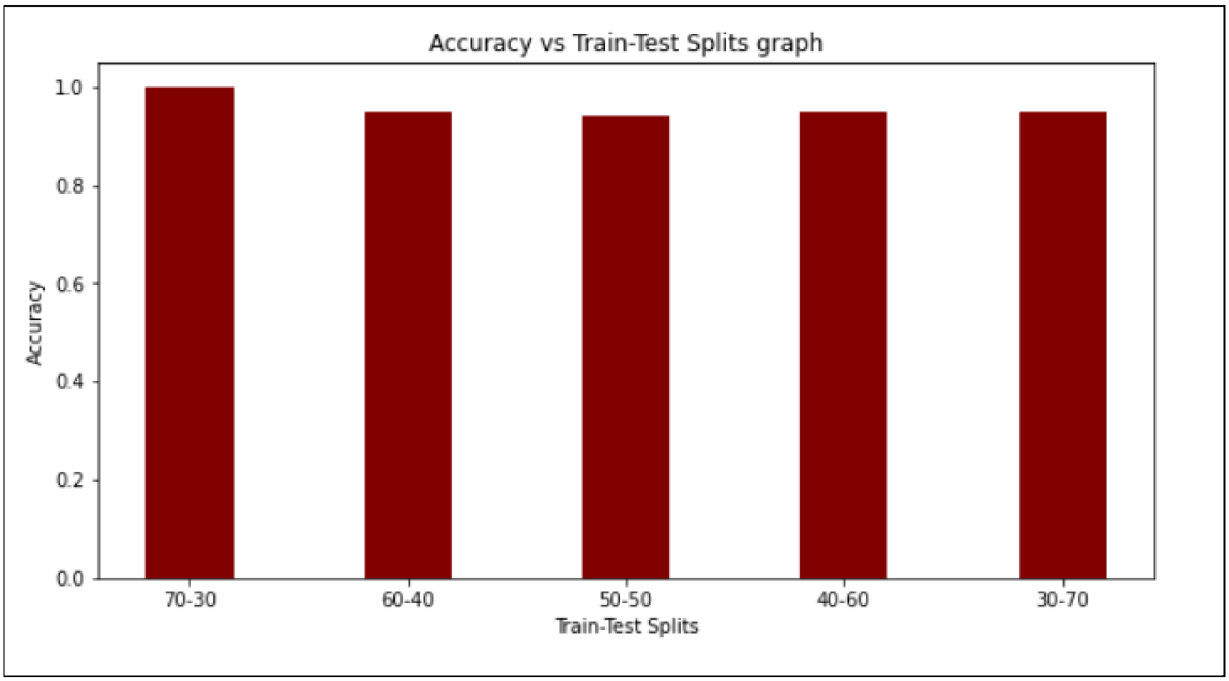
Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 70:30.

1.3 MLP Classifier(With Tuning)

```
Confusion Matrix:
[[20  0  0]
 [ 2 29  2]
 [ 0  0 19]]
------------------------------------------------
------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       0.91      1.00      0.95        20
           2       1.00      0.88      0.94        33
           3       0.90      1.00      0.95        19

    accuracy                           0.94        72
   macro avg       0.94      0.96      0.95        72
weighted avg       0.95      0.94      0.94        72

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



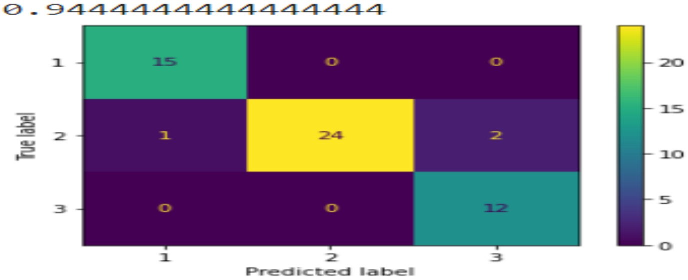
COMPARISON:



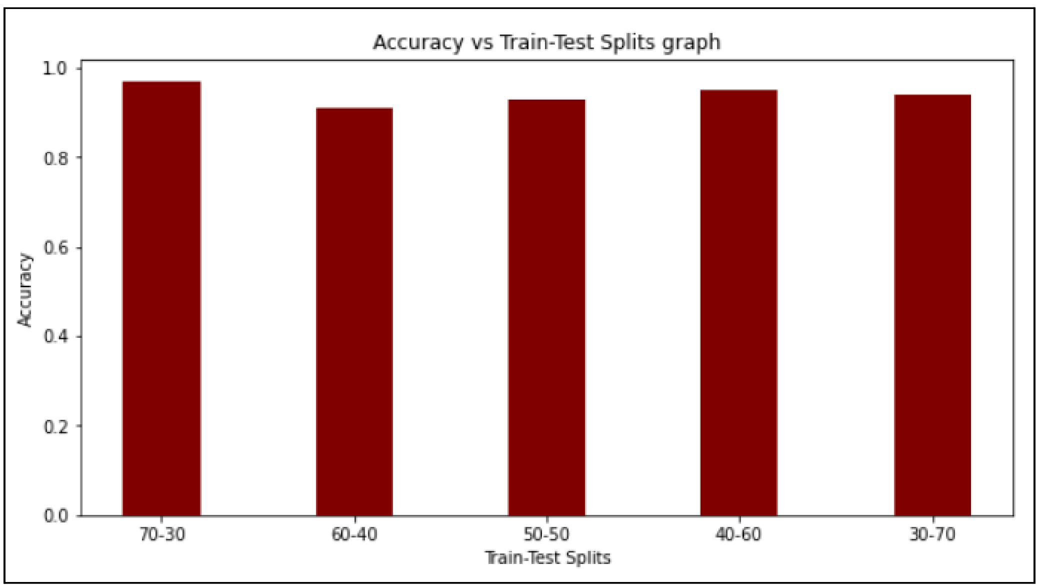
Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 70:30.

1.4 MLP Classifier(Without Tuning)

```
Confusion Matrix:
[[15  0  0]
 [ 1 24  2]
 [ 0  0 12]]
-------------------------------------------------------------
-------------------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       0.94      1.00      0.97        15
           2       1.00      0.89      0.94        27
           3       0.86      1.00      0.92        12

    accuracy                           0.94        54
   macro avg       0.93      0.96      0.94        54
weighted avg       0.95      0.94      0.94        54

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



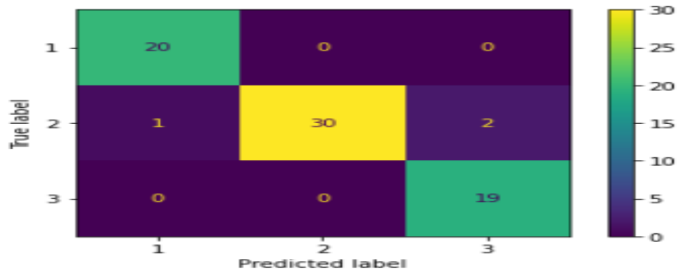
COMPARISON:



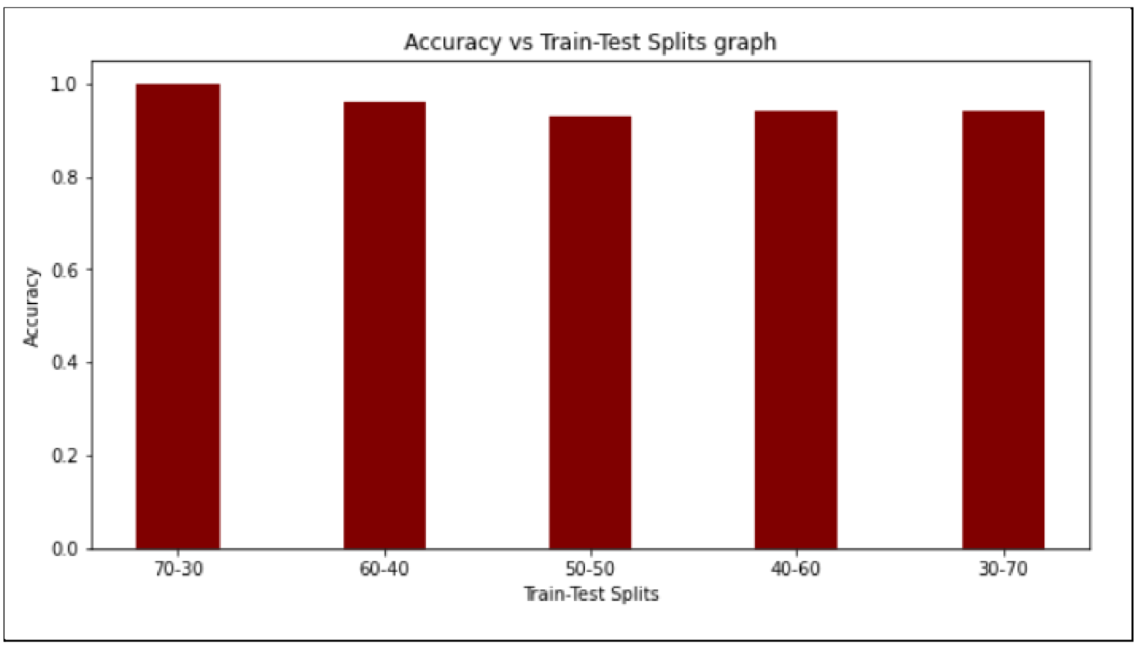
Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 70:30.

1.5 Random Forest Classifier(With Tuning)

```
Confusion Matrix:
[[20  0  0]
 [ 1 30  2]
 [ 0  0 19]]
---------------------------------------------------
---------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       0.95      1.00      0.98        20
           2       1.00      0.91      0.95        33
           3       0.90      1.00      0.95        19

    accuracy                           0.96        72
   macro avg       0.95      0.97      0.96        72
weighted avg       0.96      0.96      0.96        72

---------------------------------------------------
---------------------------------------------------
Accuracy:
0.9583333333333334
```



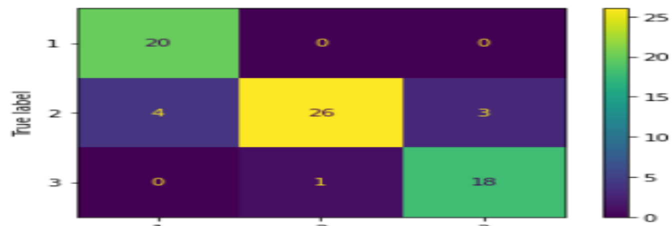
COMPARISON:



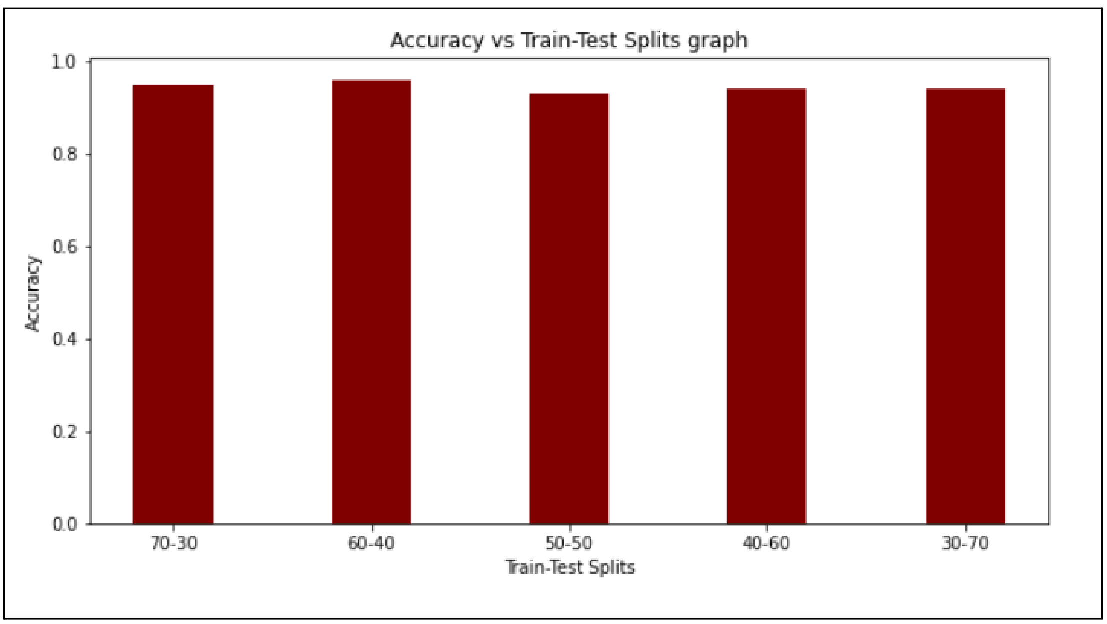
Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 70:30.

1.6 Random Forest Classifier(Without Tuning)

```
Confusion Matrix:
[[20  0  0]
 [ 4 26  3]
 [ 0  1 18]]
-----------------------------------------------------
-----------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       0.83      1.00      0.91        20
           2       0.96      0.79      0.87        33
           3       0.86      0.95      0.90        19

    accuracy                           0.89        72
   macro avg       0.88      0.91      0.89        72
weighted avg       0.90      0.89      0.89        72

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



COMPARISON:



Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 60:40.

2. IRIS PLANT DATASET

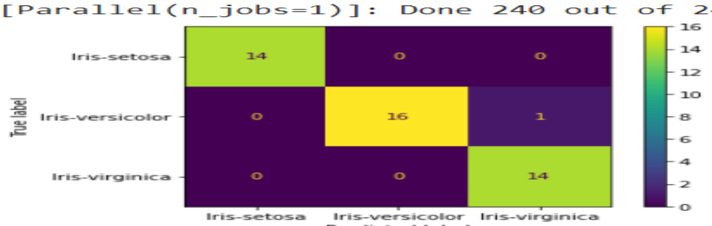
2.1 SVM Classifier(With Tuning)

```
# IRIS PLANT DATASET
# SVM(With Tuning)[70-30 split]


import pandas as pd
import numpy as np

# Dataset Preparation
df = pd.read_csv("iris.data",header=None)

col_name = ['Sepal Length','Sepal Width','Petal Length','Petal Width','Class']


df.columns = col_name


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

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =
train_test_split(X,y,train_size=0.7,test_size=0.3,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()
```

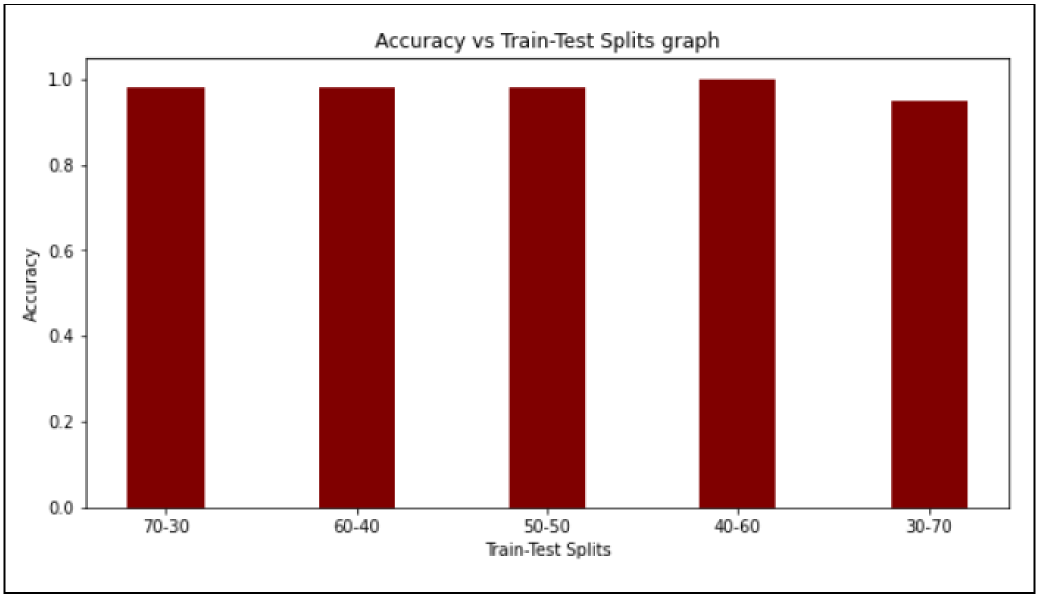
```python
############################################################################
# Showing all the parameters

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("--------------------------------------------------")
```

```python
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()
```

```
Confusion Matrix:
[[14  0  0]
 [ 0 16  1]
 [ 0  0 14]]
-----------------------------------------------------
-----------------------------------------------------
Performance Evaluation
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       1.00      0.94      0.97        17
 Iris-virginica       0.93      1.00      0.97        14

       accuracy                           0.98        45
      macro avg       0.98      0.98      0.98        45
   weighted avg       0.98      0.98      0.98        45

-----------------------------------------------------
-----------------------------------------------------
Accuracy:
0.9777777777777777
[Parallel(n_jobs=1)]: Done 240 out of 240 | elapsed:    0.9s finished
```



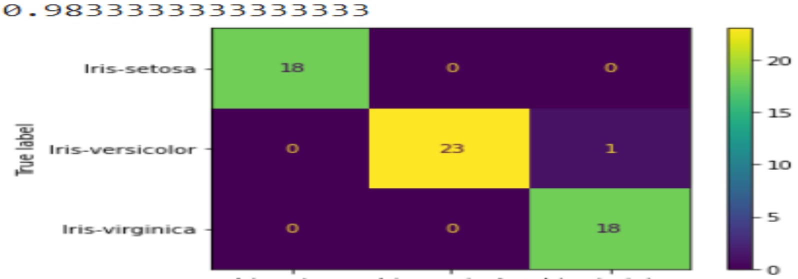
COMPARISON:



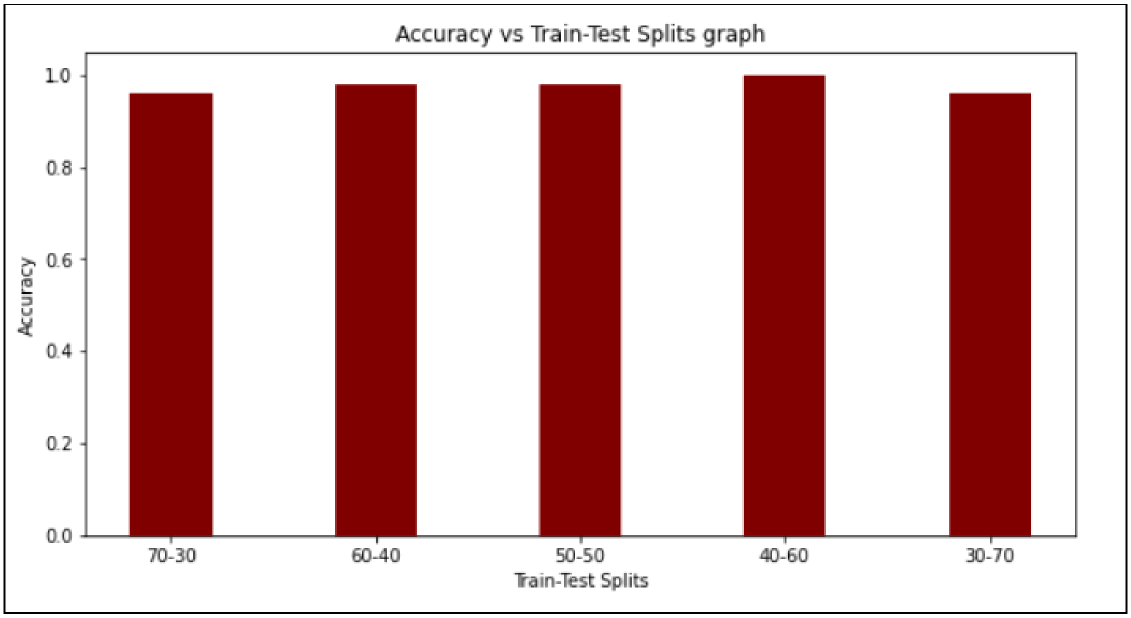
Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 40:60.

2.2 SVM Classifier(Without Tuning)

```
Confusion Matrix:
[[18  0  0]
 [ 0 23  1]
 [ 0  0 18]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        18
Iris-versicolor       1.00      0.96      0.98        24
 Iris-virginica       0.95      1.00      0.97        18

       accuracy                           0.98        60
      macro avg       0.98      0.99      0.98        60
   weighted avg       0.98      0.98      0.98        60

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9833333333333333
```



COMPARISON:

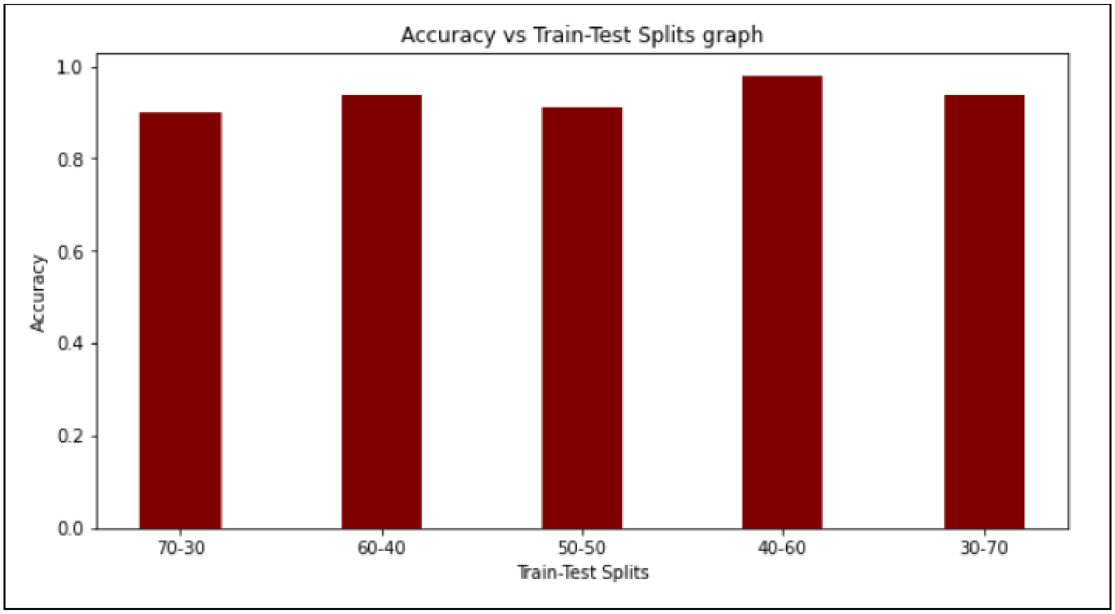


Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 40:60.

2.3 MLP Classifier(With Tuning)

```
Confusion Matrix:
[[28  0  0]
 [ 0 28  4]
 [ 0  0 30]]
-----------------------------------------------------
-----------------------------------------------------
Performance Evaluation
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        28
Iris-versicolor       1.00      0.88      0.93        32
 Iris-virginica       0.88      1.00      0.94        30

       accuracy                           0.96        90
      macro avg       0.96      0.96      0.96        90
   weighted avg       0.96      0.96      0.96        90

-----------------------------------------------------
-----------------------------------------------------
Accuracy:
0.9555555555555556
```

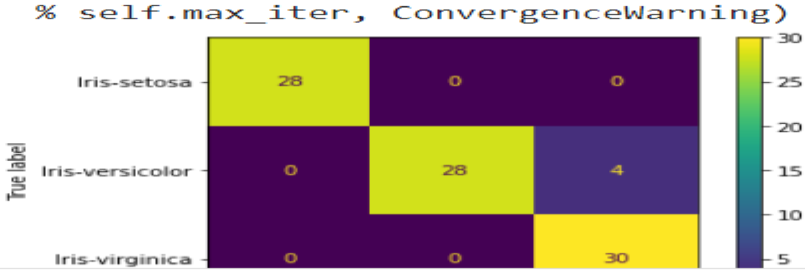
COMPARISON:



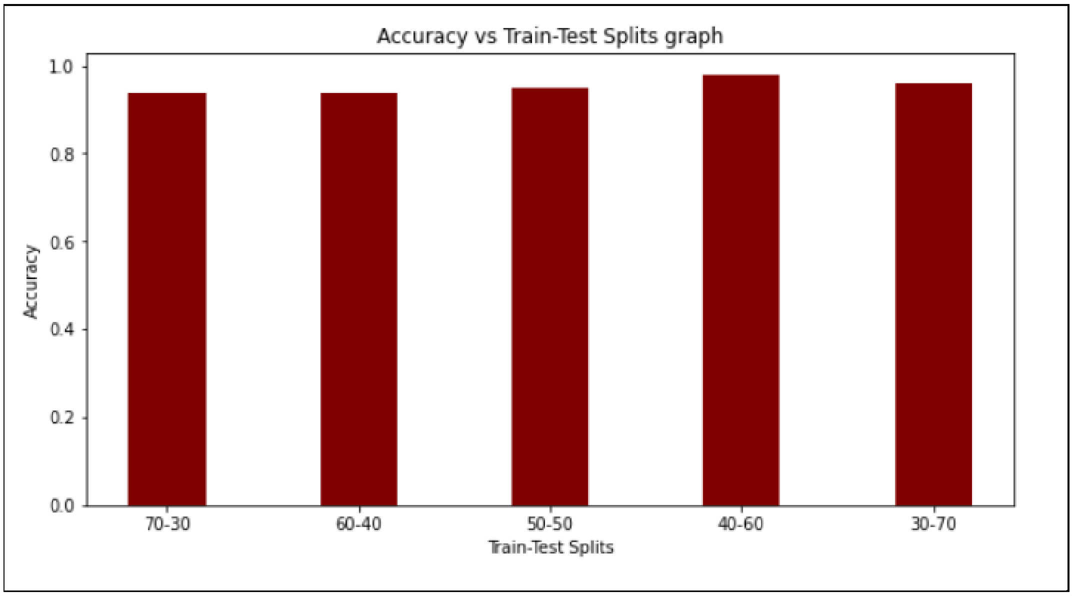
Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 40:60.

2.4 MLP Classifier(Without Tuning)

```
Confusion Matrix:
[[28  0  0]
 [ 0 28  4]
 [ 0  0 30]]
------------------------------------------------------------
------------------------------------------------------------
Performance Evaluation
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        28
Iris-versicolor       1.00      0.88      0.93        32
 Iris-virginica       0.88      1.00      0.94        30

       accuracy                           0.96        90
      macro avg       0.96      0.96      0.96        90
   weighted avg       0.96      0.96      0.96        90

------------------------------------------------------------
------------------------------------------------------------
Accuracy:
0.9555555555555556
/usr/local/lib/python3.7/dist-packages/sklearn/neural_netwo
  % self.max_iter, ConvergenceWarning)
```



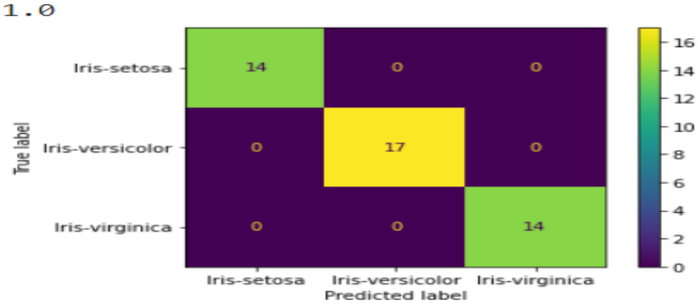
COMPARISON:



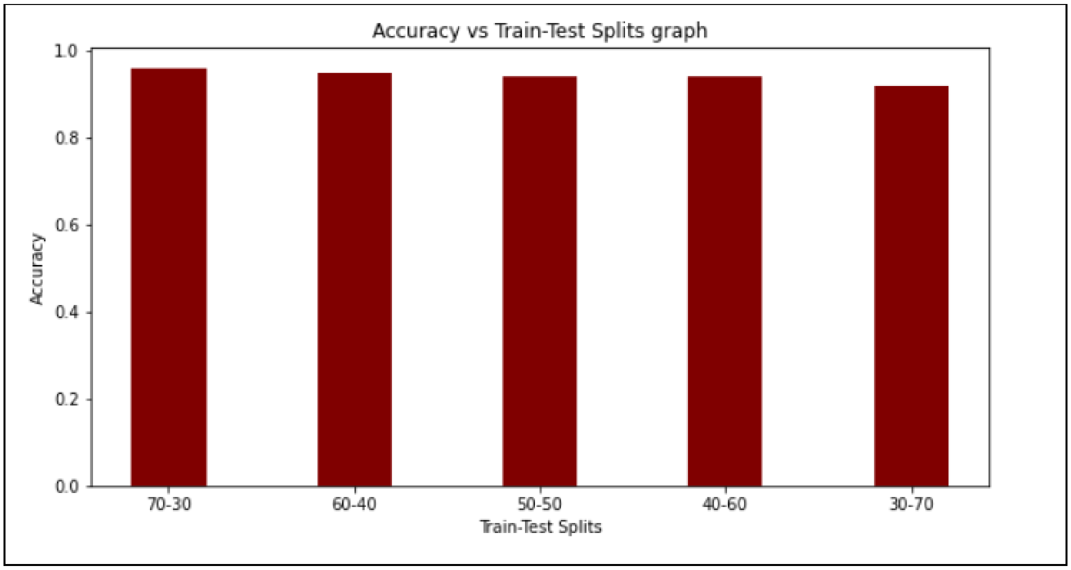
Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 40:60.

2.5 Random Forest Classifier(With Tuning)

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

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       1.00      1.00      1.00        17
 Iris-virginica       1.00      1.00      1.00        14

       accuracy                           1.00        45
      macro avg       1.00      1.00      1.00        45
   weighted avg       1.00      1.00      1.00        45

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



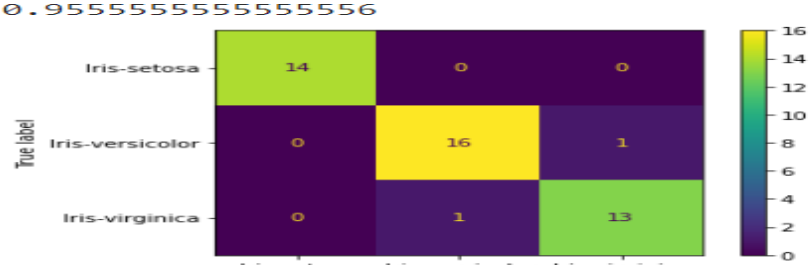
COMPARISON:



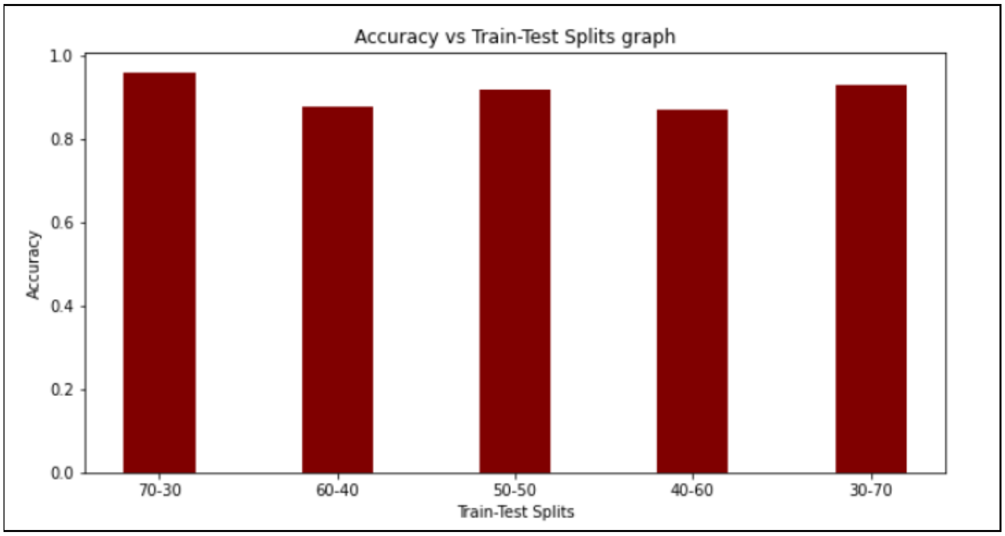
Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 70:30.

2.6 Random Forest Classifier(Without Tuning)

```
Confusion Matrix:
[[14  0  0]
 [ 0 16  1]
 [ 0  1 13]]
-------------------------------------------------------
-------------------------------------------------------
Performance Evaluation
                  precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       0.94      0.94      0.94        17
 Iris-virginica       0.93      0.93      0.93        14

       accuracy                           0.96        45
      macro avg       0.96      0.96      0.96        45
   weighted avg       0.96      0.96      0.96        45

-------------------------------------------------------
-------------------------------------------------------
Accuracy:
0.9555555555555556
```



COMPARISON:



Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 70:30.

3. IONOSPHERE DATASET

3.1 SVM Classifier(With Tuning)

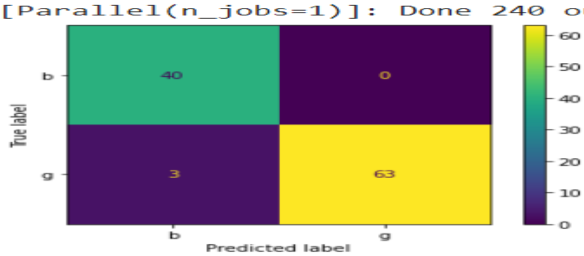
```
# IONOSPHERE DATASET
# SVM(With Tuning)[70-30 split]


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','Cla
ss']
```

```
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)
```

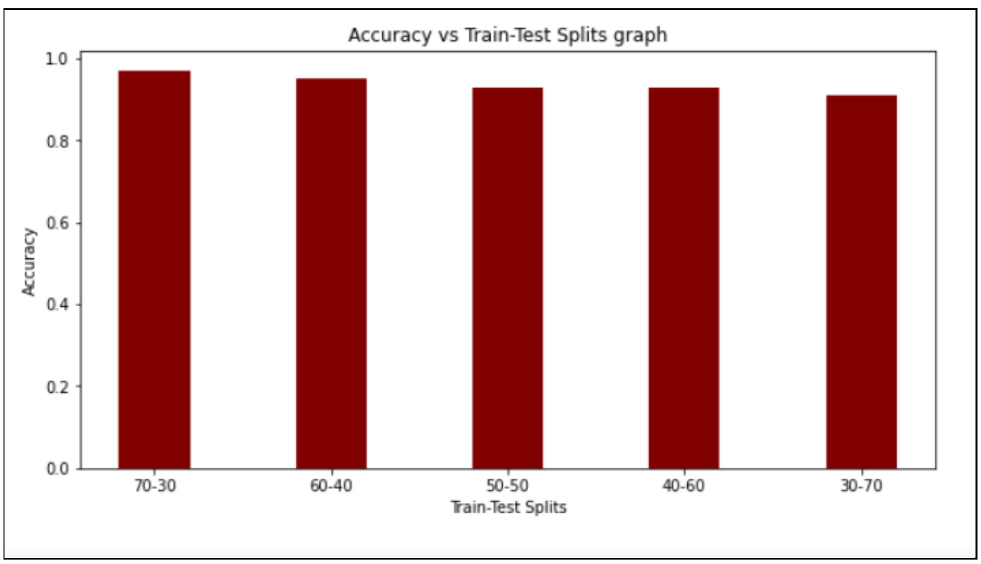
```python
############################################################################

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()
```

```
Confusion Matrix:
[[40  0]
 [ 3 63]]
-------------------------------------------------
-------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.93      1.00      0.96        40
           g       1.00      0.95      0.98        66

    accuracy                           0.97       106
   macro avg       0.97      0.98      0.97       106
weighted avg       0.97      0.97      0.97       106

-------------------------------------------------
-------------------------------------------------
Accuracy:
0.9716981132075472
[Parallel(n_jobs=1)]: Done 240 out of 240 | elapsed:    1.3s finished
```



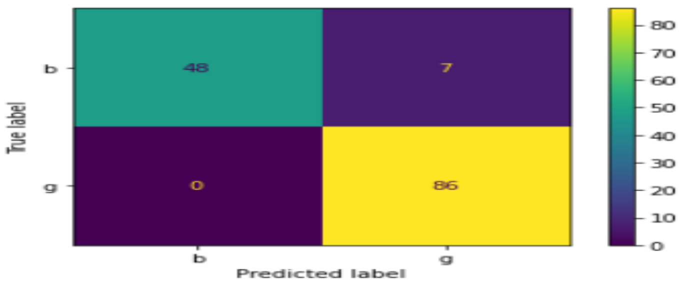
COMPARISON:



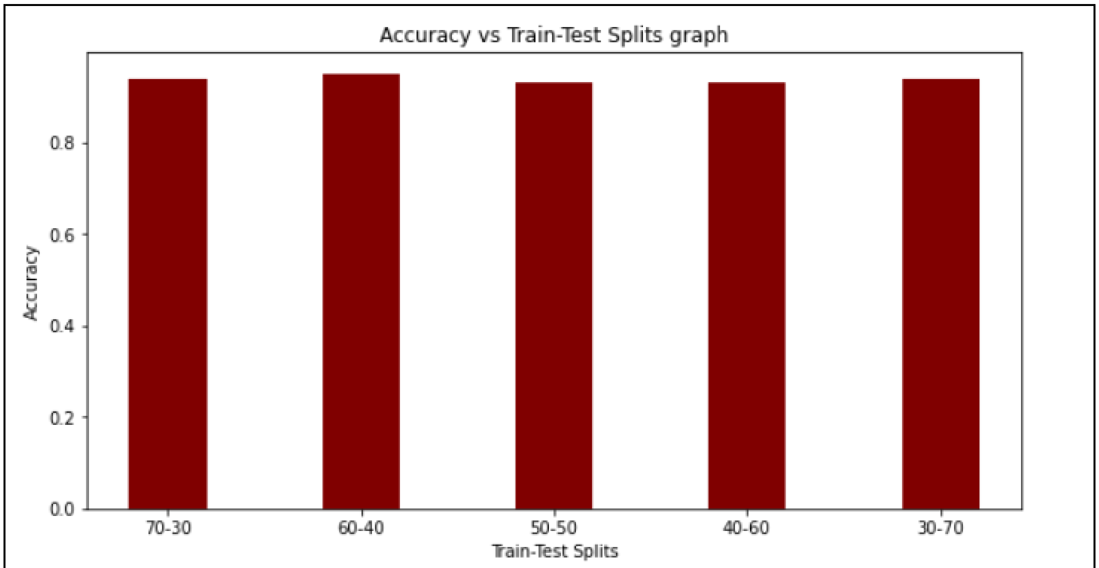
Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 70:30.

3.2 SVM Classifier(Without Tuning)

```
Confusion Matrix:
[[48  7]
 [ 0 86]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       1.00      0.87      0.93        55
           g       0.92      1.00      0.96        86

    accuracy                           0.95       141
   macro avg       0.96      0.94      0.95       141
weighted avg       0.95      0.95      0.95       141

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.950354609929078
```



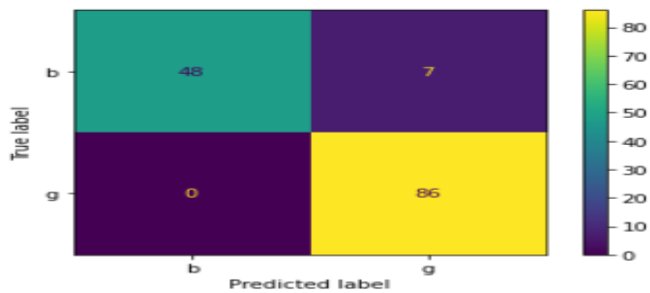
COMPARISON:



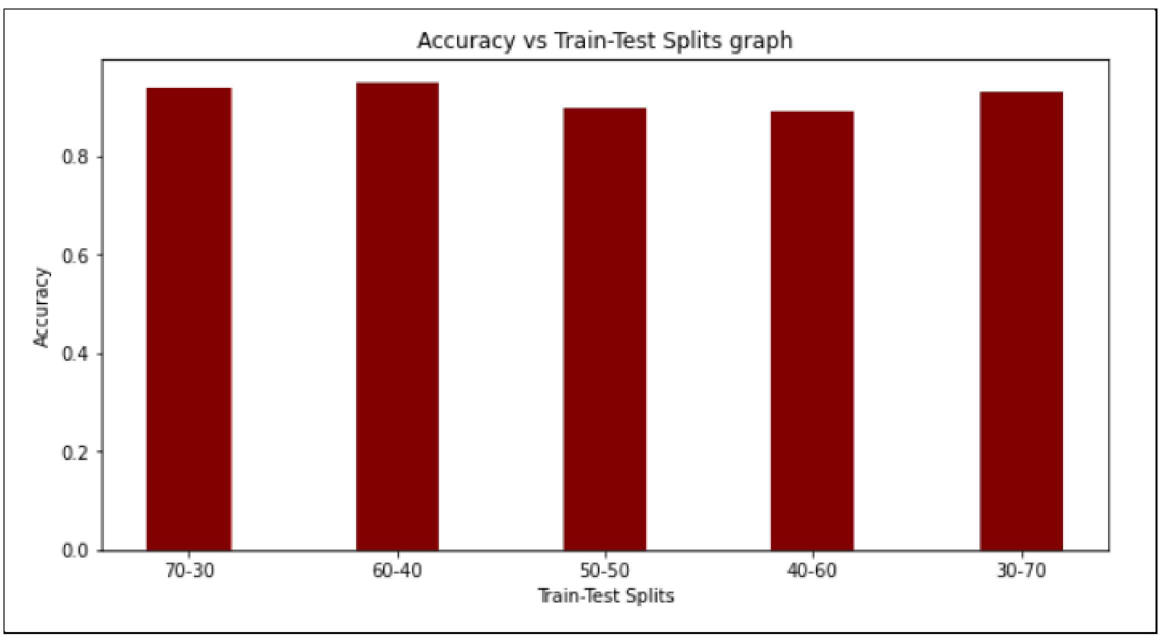
Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 60:40.

3.3 MLP Classifier(With Tuning)

```
Confusion Matrix:
[[48  7]
 [ 0 86]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       1.00      0.87      0.93        55
           g       0.92      1.00      0.96        86

    accuracy                           0.95       141
   macro avg       0.96      0.94      0.95       141
weighted avg       0.95      0.95      0.95       141

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.950354609929078
```



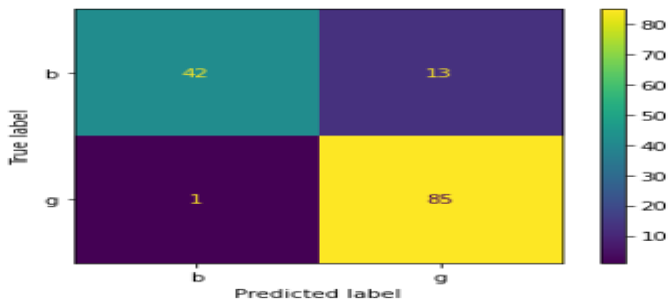
COMPARISON:



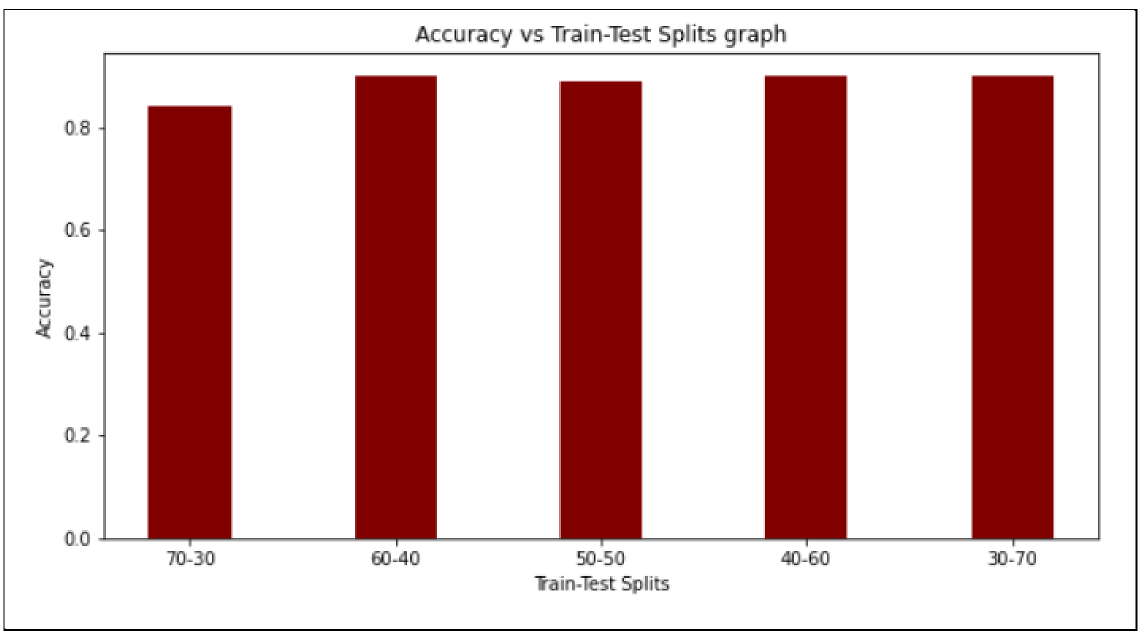
Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 60:40.

3.4 MLP Classifier(Without Tuning)

```
Confusion Matrix:
[[42 13]
 [ 1 85]]
----------------------------------------------------------
----------------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.98      0.76      0.86        55
           g       0.87      0.99      0.92        86

    accuracy                           0.90       141
   macro avg       0.92      0.88      0.89       141
weighted avg       0.91      0.90      0.90       141

----------------------------------------------------------
----------------------------------------------------------
Accuracy:
0.900709219858156
```



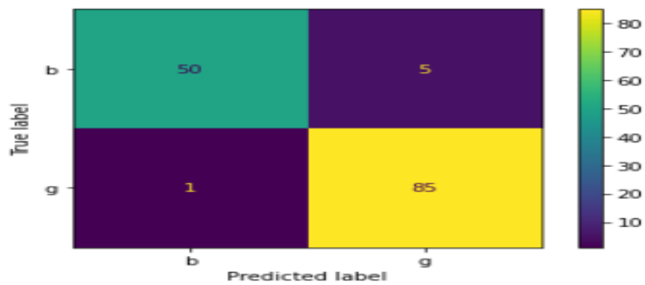
COMPARISON:



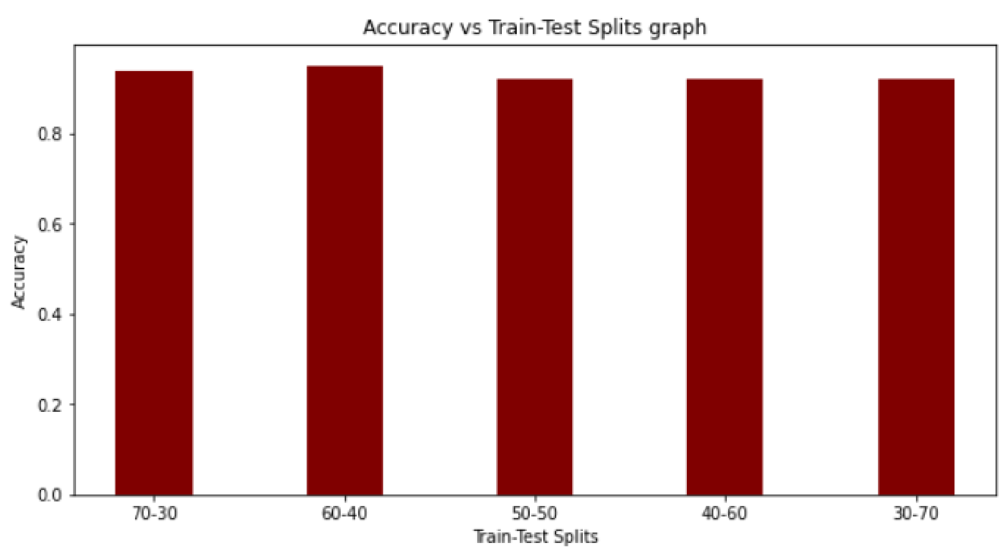
Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 60:40.

3.5 Random Forest Classifier(With Tuning)

```
Confusion Matrix:
[[50  5]
 [ 1 85]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.98      0.91      0.94        55
           g       0.94      0.99      0.97        86

    accuracy                           0.96       141
   macro avg       0.96      0.95      0.95       141
weighted avg       0.96      0.96      0.96       141

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9574468085106383
```



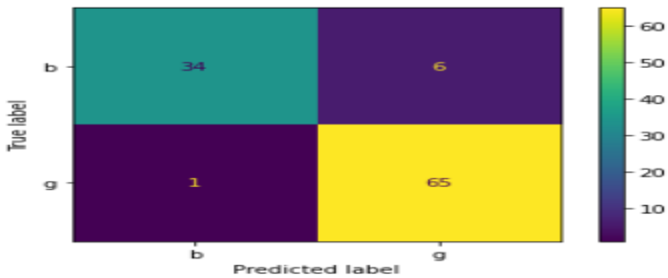
COMPARISON:



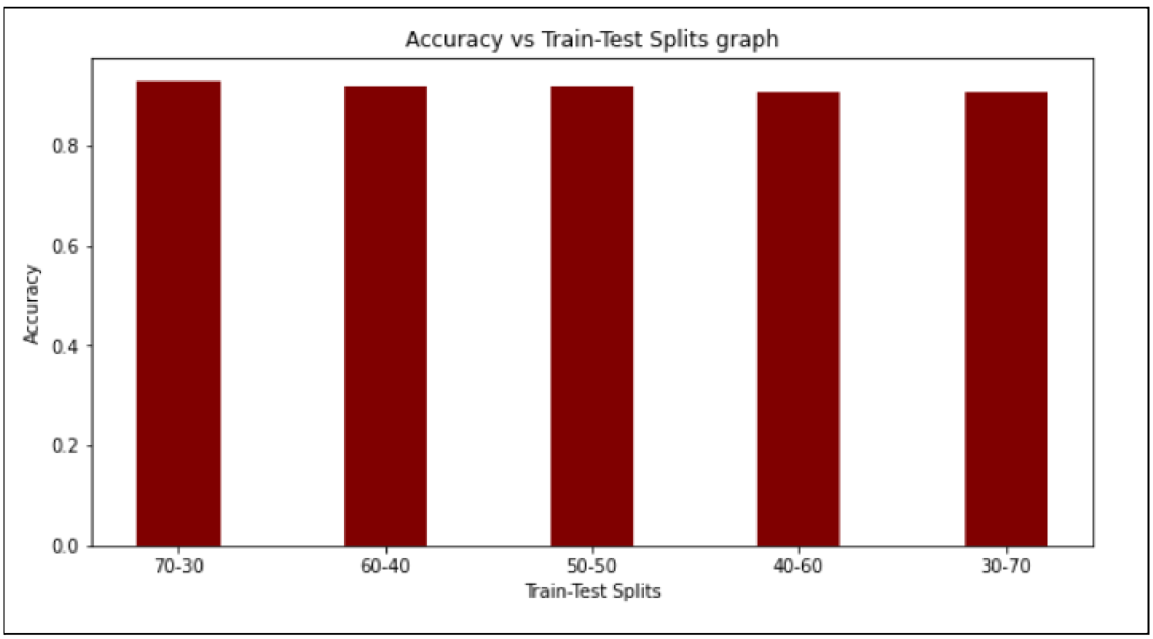
Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 60:40.

3.6 Random Forest Classifier(Without Tuning)

```
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
```



COMPARISON:



Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 70:30.

4. BREAST CANCER DATASET

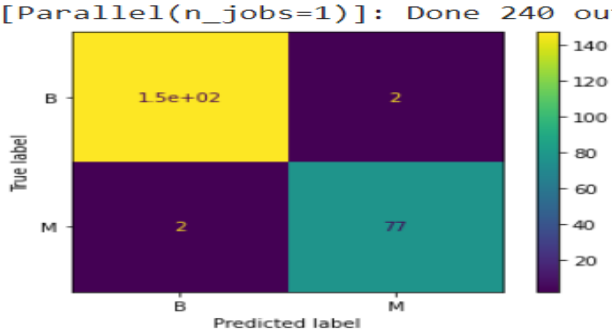
4.1 SVM Classifier(With Tuning)

```
# BREAST CANCER DATASET
# SVM(With Tuning)[60-40 split]


import pandas as pd
import numpy as np

# Dataset Preparation
df = pd.read_csv("wdbc.data",header=None)

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


df.columns = col_name


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

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =
train_test_split(X,y,train_size=0.6,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
```

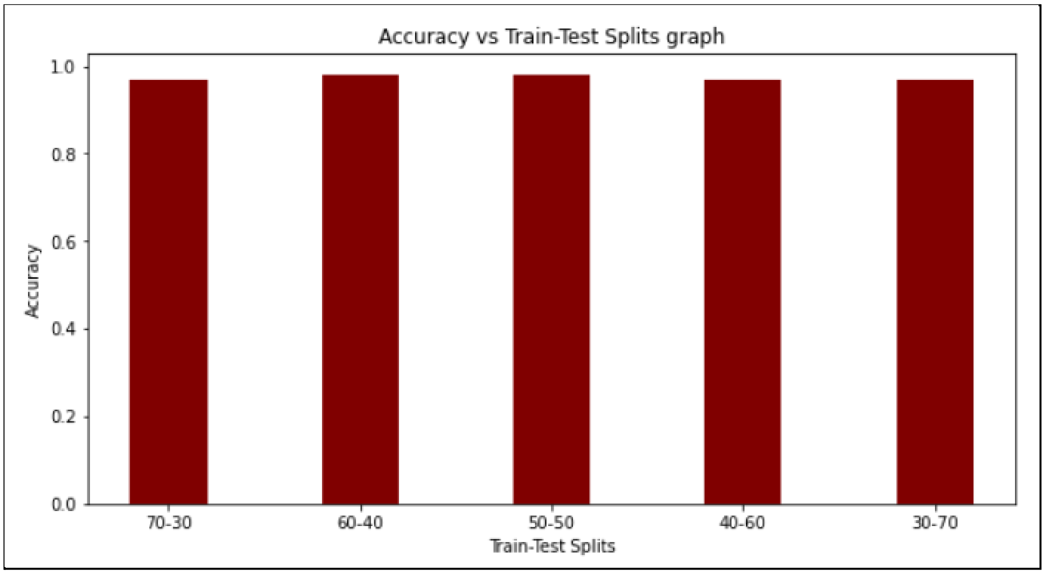
```
classifier = SVC()

##########################################################################
# 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))
```

```python
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()
```

```
Confusion Matrix:
[[147   2]
 [  2  77]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       0.99      0.99      0.99       149
           M       0.97      0.97      0.97        79

    accuracy                           0.98       228
   macro avg       0.98      0.98      0.98       228
weighted avg       0.98      0.98      0.98       228

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9824561403508771
[Parallel(n_jobs=1)]: Done 240 out of 240 | elapsed:    1.4s finished
```



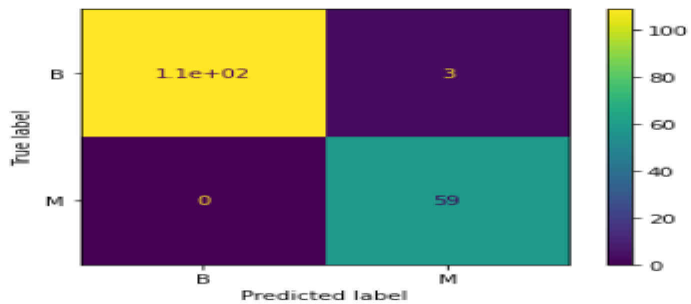
COMPARISON:



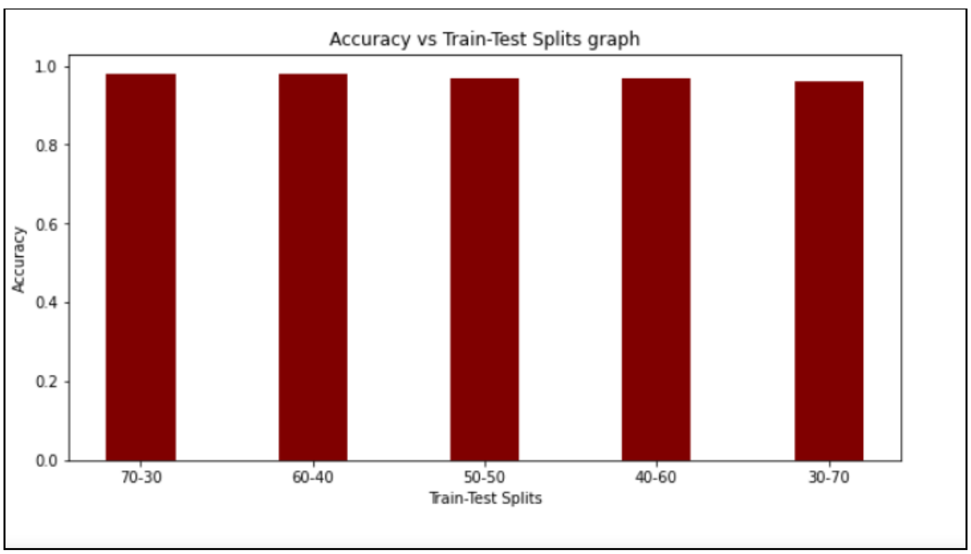
Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 60:40.

4.2 SVM Classifier(Without Tuning)

```
Confusion Matrix:
[[109   3]
 [  0  59]]
-------------------------------------------------
-------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       1.00      0.97      0.99       112
           M       0.95      1.00      0.98        59

    accuracy                           0.98       171
   macro avg       0.98      0.99      0.98       171
weighted avg       0.98      0.98      0.98       171

-------------------------------------------------
-------------------------------------------------
Accuracy:
0.9824561403508771
```



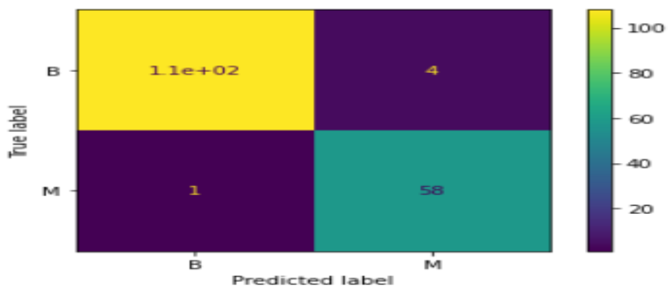
COMPARISON:



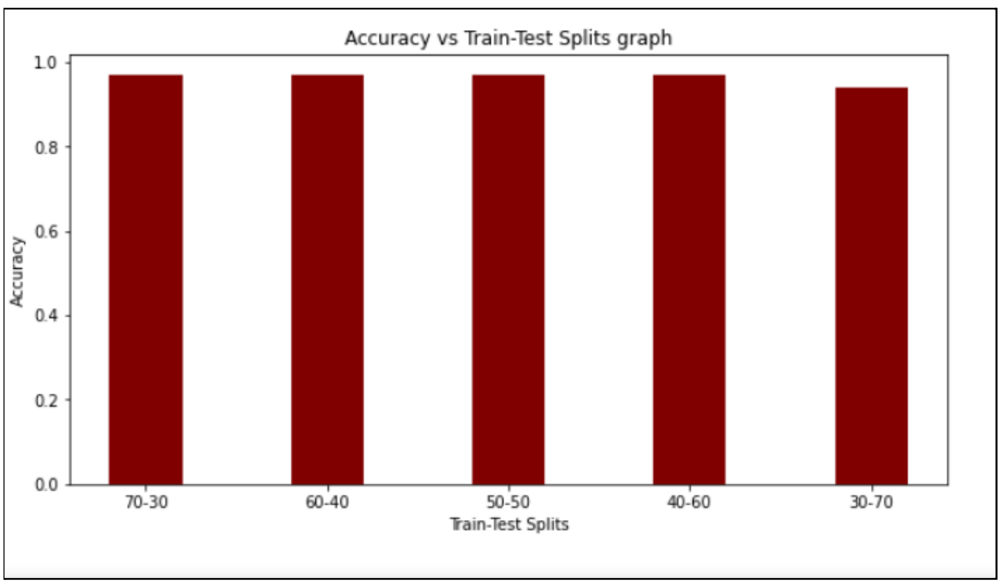
Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 70:30.

4.3 MLP Classifier(With Tuning)

```
Confusion Matrix:
[[108   4]
 [  1  58]]
---------------------------------------------
---------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       0.99      0.96      0.98       112
           M       0.94      0.98      0.96        59

    accuracy                           0.97       171
   macro avg       0.96      0.97      0.97       171
weighted avg       0.97      0.97      0.97       171

---------------------------------------------
---------------------------------------------
Accuracy:
0.9707602339181286
```



COMPARISON:

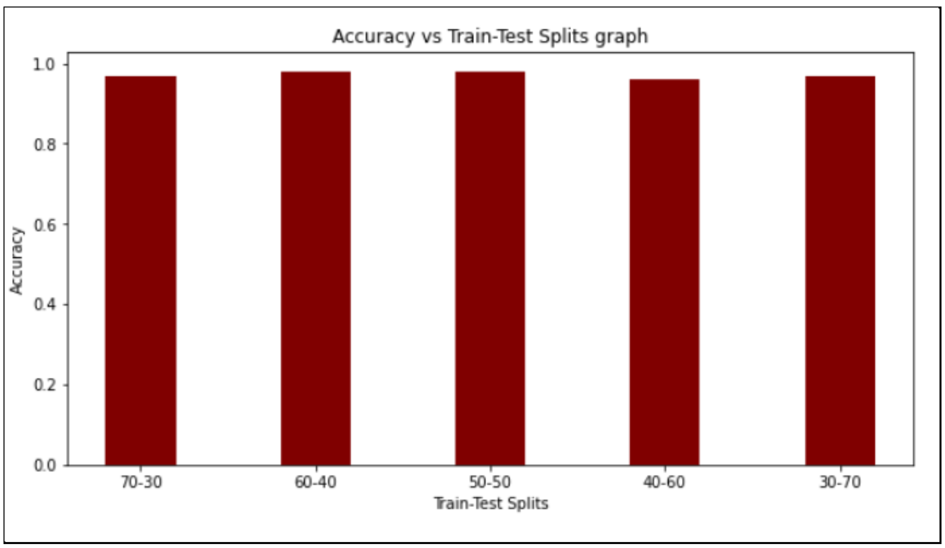


Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 70:30.

4.4 MLP Classifier(Without Tuning)

```
Confusion Matrix:
[[147    2]
 [  2   77]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       0.99      0.99      0.99       149
           M       0.97      0.97      0.97        79

    accuracy                           0.98       228
   macro avg       0.98      0.98      0.98       228
weighted avg       0.98      0.98      0.98       228

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9824561403508771
```

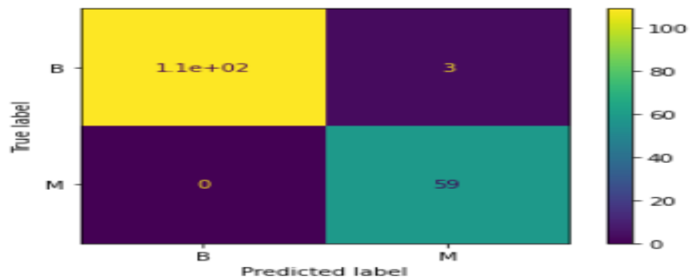
COMPARISON:



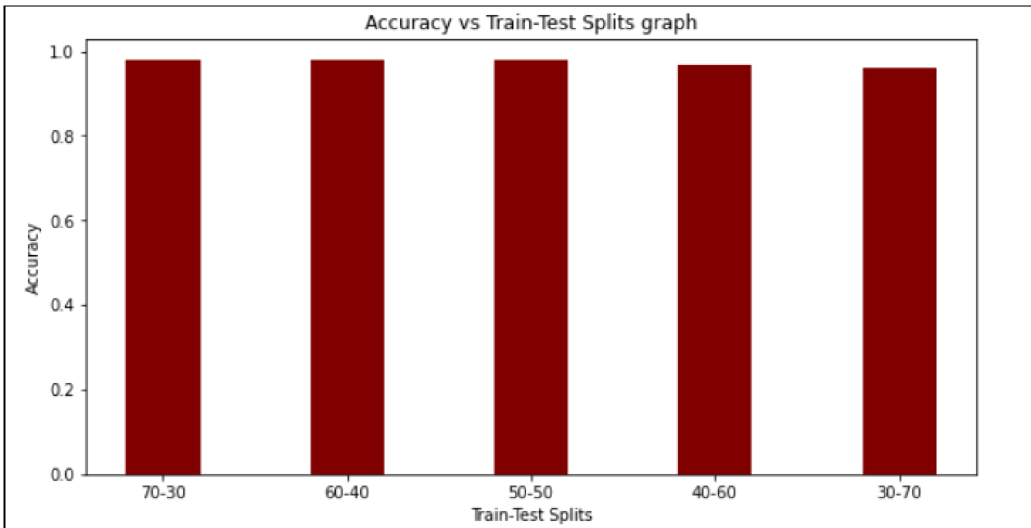
Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 60:40.

4.5 Random Forest Classifier(With Tuning)

```
Confusion Matrix:
[[109   3]
 [  0  59]]
-------------------------------------------------------------
-------------------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       1.00      0.97      0.99       112
           M       0.95      1.00      0.98        59

    accuracy                           0.98       171
   macro avg       0.98      0.99      0.98       171
weighted avg       0.98      0.98      0.98       171

-------------------------------------------------------------
-------------------------------------------------------------
Accuracy:
0.9824561403508771
```



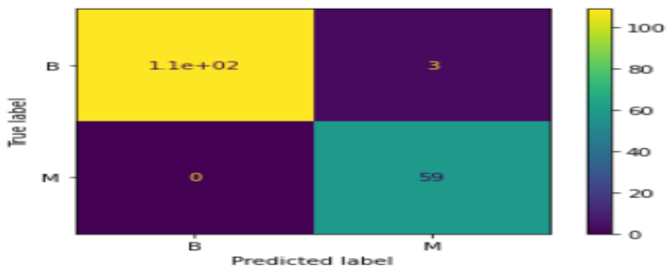
COMPARISON:



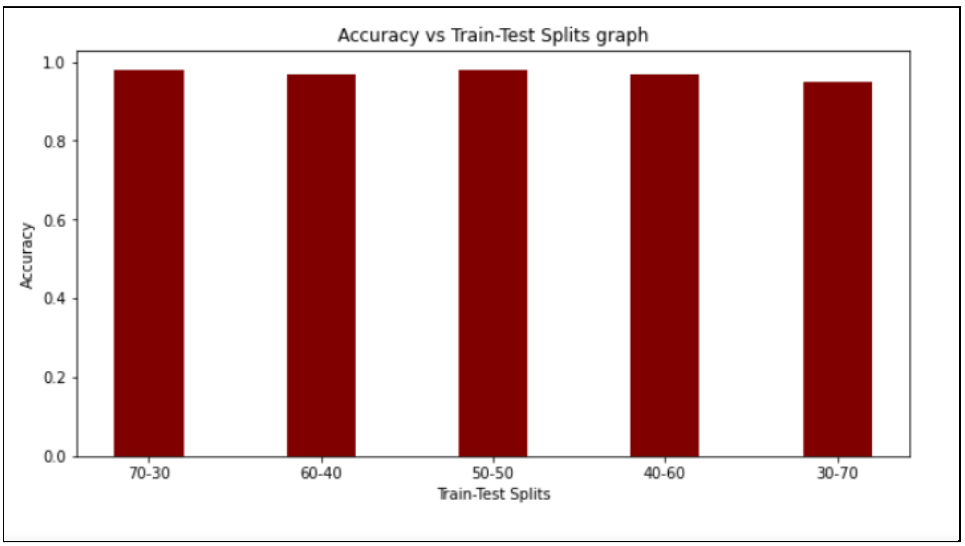
Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 70:30.

4.6 Random Forest Classifier(Without Tuning)

```
Confusion Matrix:
[[109   3]
 [  0  59]]
------------------------------------------------------------
------------------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       1.00      0.97      0.99       112
           M       0.95      1.00      0.98        59

    accuracy                           0.98       171
   macro avg       0.98      0.99      0.98       171
weighted avg       0.98      0.98      0.98       171

------------------------------------------------------------
------------------------------------------------------------
Accuracy:
0.9824561403508771
```



COMPARISON:



Here, we can see that the highest accuracy has been achieved when the Train-Test split is in the ratio of 70:30.

OVERALL RESULT:

In most of the cases, the highest accuracy is gained when the Train-Test split ratio is in the ratio of 70:30.

5.Using Principal Component Analysis:

5.1 Iris Plant Dataset

```
# IRIS PLANT DATASET
# SVM(With Tuning)[70-30 split]


import pandas as pd
import numpy as np

# Dataset Preparation
df = pd.read_csv("iris.data",header=None)

col_name = ['Sepal Length','Sepal Width','Petal Length','Petal
Width','Class']


df.columns = col_name


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

from sklearn.model_selection import train_test_split
```

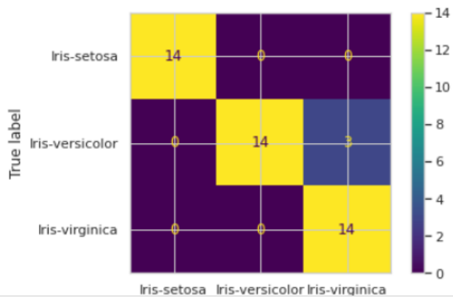
```
X_train, X_test, y_train, y_test =
train_test_split(X,y,train_size=0.7,test_size=0.3,random_state=10)


# Feature Scaling
from sklearn.preprocessing import StandardScaler

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


# Finding the important parameters that contribute to most of the variance
in the data.

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA

pca_test = PCA(n_components=4)
pca_test.fit(X_train)
sns.set(style='whitegrid')
plt.plot(np.cumsum(pca_test.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.axvline(linewidth=4, color='r', linestyle = '--', x=10, ymin=0,
ymax=1)
display(plt.show())

# So we can see that we have 10 important parameters

pca = PCA(n_components=2)
pca.fit(X_train)
X_train = pca.transform(X_train)
X_test = pca.transform(X_test)



# Classification
from sklearn.svm import SVC
```

```python
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
```

```python
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()
```

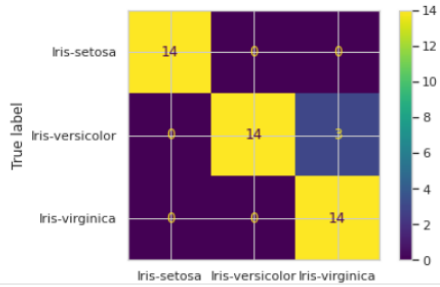
5.1.1 SVM Classifier(With Tuning)

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

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       1.00      0.82      0.90        17
 Iris-virginica       0.82      1.00      0.90        14

       accuracy                           0.93        45
      macro avg       0.94      0.94      0.94        45
   weighted avg       0.95      0.93      0.93        45

-------------------------------------------------
-------------------------------------------------
Accuracy:
0.9333333333333333
[Parallel(n_jobs=1)]: Done 240 out of 240 | elapsed:    0.7s finished
```



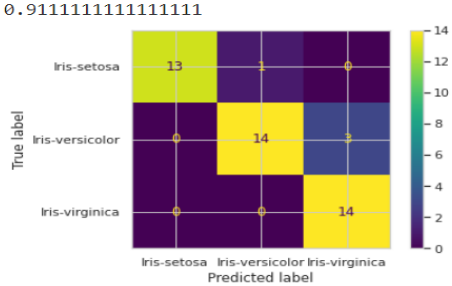
5.1.2 SVM Classifier(Without Tuning)

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

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       1.00      0.82      0.90        17
 Iris-virginica       0.82      1.00      0.90        14

       accuracy                           0.93        45
      macro avg       0.94      0.94      0.94        45
   weighted avg       0.95      0.93      0.93        45

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9333333333333333
```



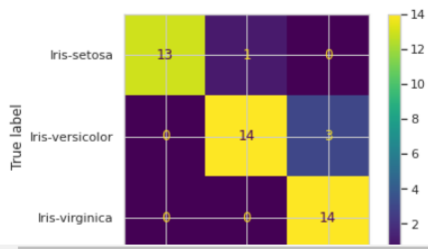
5.1.3 MLP Classifier(With Tuning)

```
Confusion Matrix:
[[13  1  0]
 [ 0 14  3]
 [ 0  0 14]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
                 precision    recall  f1-score   support

    Iris-setosa       1.00      0.93      0.96        14
Iris-versicolor       0.93      0.82      0.87        17
 Iris-virginica       0.82      1.00      0.90        14

       accuracy                           0.91        45
      macro avg       0.92      0.92      0.91        45
   weighted avg       0.92      0.91      0.91        45

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9111111111111111
```



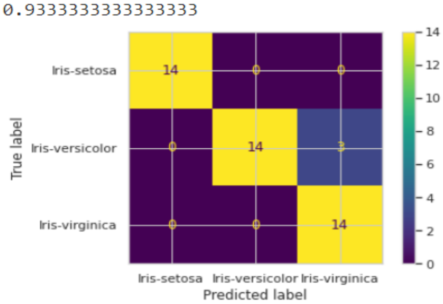
5.1.4 MLP Classifier(Without Tuning)

```
Confusion Matrix:
[[13  1  0]
 [ 0 14  3]
 [ 0  0 14]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
                 precision    recall  f1-score   support

    Iris-setosa       1.00      0.93      0.96        14
Iris-versicolor       0.93      0.82      0.87        17
 Iris-virginica       0.82      1.00      0.90        14

       accuracy                           0.91        45
      macro avg       0.92      0.92      0.91        45
   weighted avg       0.92      0.91      0.91        45

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9111111111111111
```



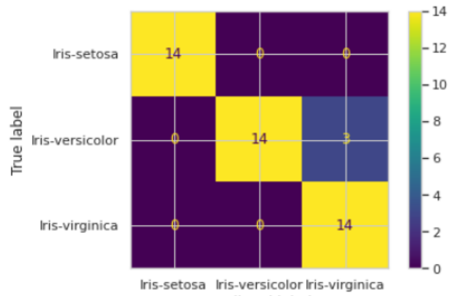
5.1.5 Random Forest Classifier(With Tuning)

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

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       1.00      0.82      0.90        17
 Iris-virginica       0.82      1.00      0.90        14

       accuracy                           0.93        45
      macro avg       0.94      0.94      0.94        45
   weighted avg       0.95      0.93      0.93        45

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9333333333333333
```



5.1.6 Random Forest Classifier(Without Tuning)

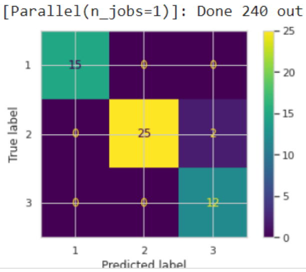
```
Confusion Matrix:
[[14  0  0]
 [ 0 14  3]
 [ 0  0 14]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        14
Iris-versicolor       1.00      0.82      0.90        17
 Iris-virginica       0.82      1.00      0.90        14

       accuracy                           0.93        45
      macro avg       0.94      0.94      0.94        45
   weighted avg       0.95      0.93      0.93        45

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9333333333333333
```



5.2 Wine Dataset

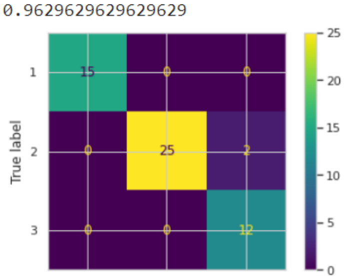
5.2.1 SVM Classifier(With Tuning)

```
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
[Parallel(n_jobs=1)]: Done 240 out of 240 | elapsed:    1.0s finished
```



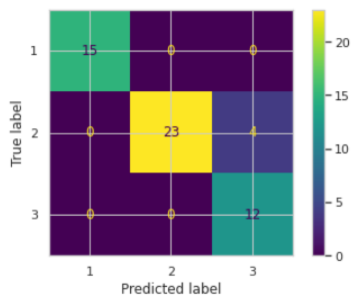
5.2.2 SVM Classifier(Without Tuning)

```
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
```



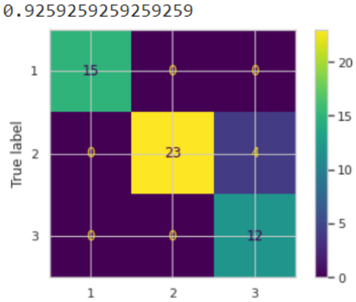
5.2.3 MLP Classifier(With Tuning)

```
Confusion Matrix:
[[15  0  0]
 [ 0 23  4]
 [ 0  0 12]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        15
           2       1.00      0.85      0.92        27
           3       0.75      1.00      0.86        12

    accuracy                           0.93        54
   macro avg       0.92      0.95      0.93        54
weighted avg       0.94      0.93      0.93        54

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9259259259259259
```



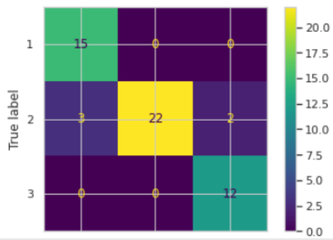
5.2.4 MLP Classifier(Without Tuning)

```
Confusion Matrix:
[[15  0  0]
 [ 0 23  4]
 [ 0  0 12]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        15
           2       1.00      0.85      0.92        27
           3       0.75      1.00      0.86        12

    accuracy                           0.93        54
   macro avg       0.92      0.95      0.93        54
weighted avg       0.94      0.93      0.93        54

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9259259259259259
```



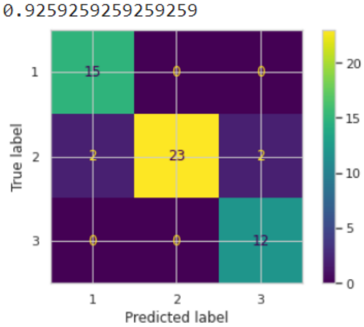
5.2.5 Random Forest Classifier(With Tuning)

```
Confusion Matrix:
[[15  0  0]
 [ 3 22  2]
 [ 0  0 12]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       0.83      1.00      0.91        15
           2       1.00      0.81      0.90        27
           3       0.86      1.00      0.92        12

    accuracy                           0.91        54
   macro avg       0.90      0.94      0.91        54
weighted avg       0.92      0.91      0.91        54

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9074074074074074
```



5.2.6 Random Forest Classifier(Without Tuning)

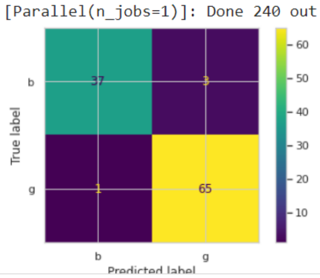
```
Confusion Matrix:
[[15  0  0]
 [ 2 23  2]
 [ 0  0 12]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           1       0.88      1.00      0.94        15
           2       1.00      0.85      0.92        27
           3       0.86      1.00      0.92        12

    accuracy                           0.93        54
   macro avg       0.91      0.95      0.93        54
weighted avg       0.94      0.93      0.93        54

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9259259259259259
```



5.3 Ionosphere Dataset

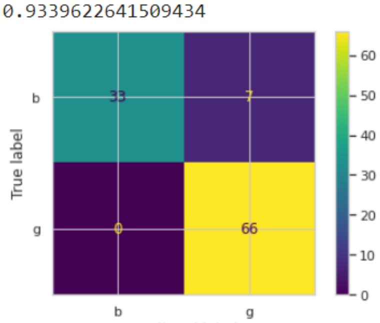
5.3.1 SVM Classifier(With Tuning)

```
Confusion Matrix:
[[37  3]
 [ 1 65]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.97      0.93      0.95        40
           g       0.96      0.98      0.97        66

    accuracy                           0.96       106
   macro avg       0.96      0.95      0.96       106
weighted avg       0.96      0.96      0.96       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9622641509433962
[Parallel(n_jobs=1)]: Done 240 out of 240 | elapsed:    0.9s finished
```



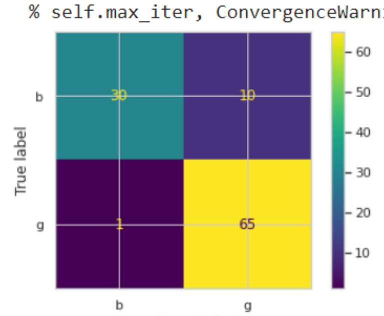
5.3.2 SVM Classifier(Without Tuning)

```
Confusion Matrix:
[[33  7]
 [ 0 66]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       1.00      0.82      0.90        40
           g       0.90      1.00      0.95        66

    accuracy                           0.93       106
   macro avg       0.95      0.91      0.93       106
weighted avg       0.94      0.93      0.93       106

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



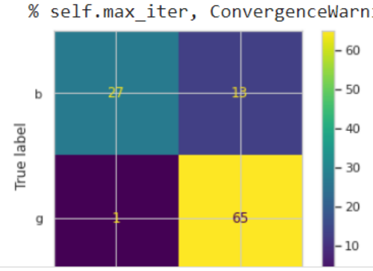
5.3.3 MLP Classifier(With Tuning)

```
Confusion Matrix:
[[30 10]
 [ 1 65]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.97      0.75      0.85        40
           g       0.87      0.98      0.92        66

    accuracy                           0.90       106
   macro avg       0.92      0.87      0.88       106
weighted avg       0.90      0.90      0.89       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.8962264150943396
/usr/local/lib/python3.7/dist-packages/sklearn/neural_networ
  % self.max_iter, ConvergenceWarning)
```



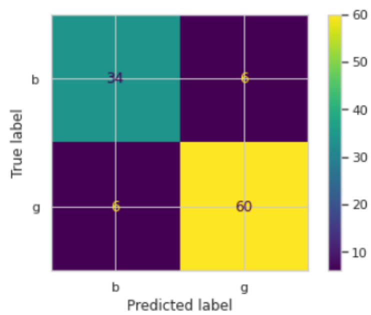
5.3.4 MLP Classifier(Without Tuning)

```
Confusion Matrix:
[[27 13]
 [ 1 65]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.96      0.68      0.79        40
           g       0.83      0.98      0.90        66

    accuracy                           0.87       106
   macro avg       0.90      0.83      0.85       106
weighted avg       0.88      0.87      0.86       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.8679245283018868
/usr/local/lib/python3.7/dist-packages/sklearn/neural_network
  % self.max_iter, ConvergenceWarning)
```



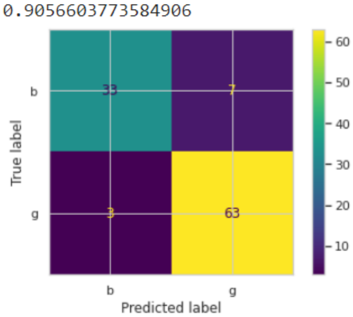
5.3.5 Random Forest Classifier(With Tuning)

```
Confusion Matrix:
[[34  6]
 [ 6 60]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.85      0.85      0.85        40
           g       0.91      0.91      0.91        66

    accuracy                           0.89       106
   macro avg       0.88      0.88      0.88       106
weighted avg       0.89      0.89      0.89       106

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.8867924528301887
```



5.3.6 Random Forest Classifier(Without Tuning)

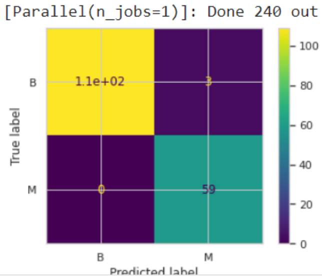
```
Confusion Matrix:
[[33  7]
 [ 3 63]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           b       0.92      0.82      0.87        40
           g       0.90      0.95      0.93        66

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

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



5.4 Iris Plant Dataset

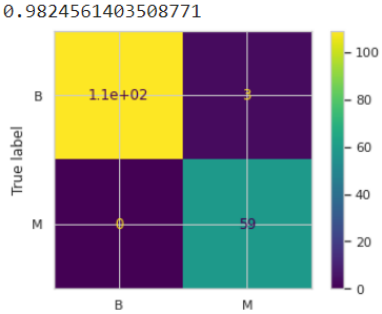
5.4.1 SVM Classifier(With Tuning)

```
Confusion Matrix:
[[109   3]
 [  0  59]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       1.00      0.97      0.99       112
           M       0.95      1.00      0.98        59

    accuracy                           0.98       171
   macro avg       0.98      0.99      0.98       171
weighted avg       0.98      0.98      0.98       171

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9824561403508771
[Parallel(n_jobs=1)]: Done 240 out of 240 | elapsed:    1.4s finished
```



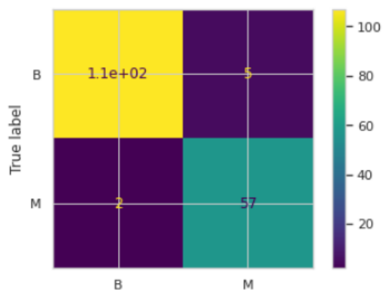
5.4.2 SVM Classifier(Without Tuning)

```
Confusion Matrix:
[[109   3]
 [  0  59]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       1.00      0.97      0.99       112
           M       0.95      1.00      0.98        59

    accuracy                           0.98       171
   macro avg       0.98      0.99      0.98       171
weighted avg       0.98      0.98      0.98       171

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9824561403508771
```



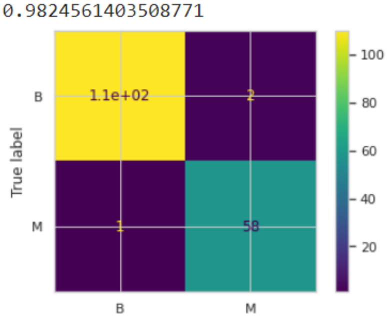
5.4.3 MLP Classifier(With Tuning)

```
Confusion Matrix:
[[107   5]
 [  2  57]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       0.98      0.96      0.97       112
           M       0.92      0.97      0.94        59

    accuracy                           0.96       171
   macro avg       0.95      0.96      0.96       171
weighted avg       0.96      0.96      0.96       171

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9590643274853801
```



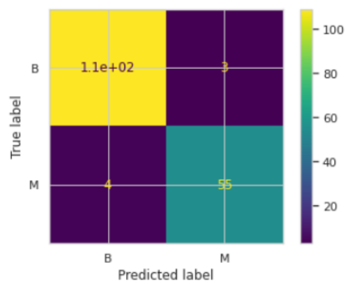
5.4.4 MLP Classifier(Without Tuning)

```
Confusion Matrix:
[[110   2]
 [  1  58]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       0.99      0.98      0.99       112
           M       0.97      0.98      0.97        59

    accuracy                           0.98       171
   macro avg       0.98      0.98      0.98       171
weighted avg       0.98      0.98      0.98       171

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9824561403508771
```



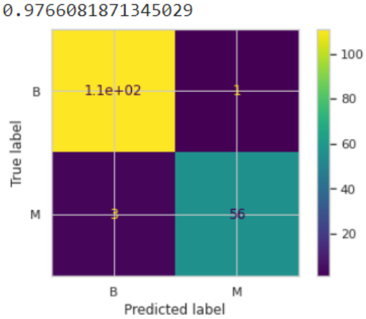
5.4.5 Random Forest Classifier(With Tuning)

```
Confusion Matrix:
[[109   3]
 [  4  55]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       0.96      0.97      0.97       112
           M       0.95      0.93      0.94        59

    accuracy                           0.96       171
   macro avg       0.96      0.95      0.95       171
weighted avg       0.96      0.96      0.96       171

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9590643274853801
```



5.4.6 Random Forest Classifier(Without Tuning)

```
Confusion Matrix:
[[111   1]
 [  3  56]]
--------------------------------------------------
--------------------------------------------------
Performance Evaluation
              precision    recall  f1-score   support

           B       0.97      0.99      0.98       112
           M       0.98      0.95      0.97        59

    accuracy                           0.98       171
   macro avg       0.98      0.97      0.97       171
weighted avg       0.98      0.98      0.98       171

--------------------------------------------------
--------------------------------------------------
Accuracy:
0.9766081871345029
```



CONCLUSION:

We can see that the overall accuracy in all the cases increases when we use Principal Component Analysis (PCA) in our dataset before applying the algorithms.