**Machine Learning Lab**

**Assignment 2**

**Name - Pritesh Kumar Sahani**

**Roll - 001811001005**

**Semester - 7**

**Year - 4**

**Department - Information Technology**

**GITHUB LINK:** [**https://github.com/stepupgithub/Machine-Learning-Assignments**](https://github.com/stepupgithub/Machine-Learning-Assignments)

**ENTIRE ASSIGNMENT LINK (GOOGLE COLLAB + COMPARISON TABLE):** [**https://drive.google.com/drive/folders/1aDSZ8XfYNx15FVNs4G\_foG5YHliWNpi1?usp=sharing**](https://drive.google.com/drive/folders/1aDSZ8XfYNx15FVNs4G_foG5YHliWNpi1?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:")**

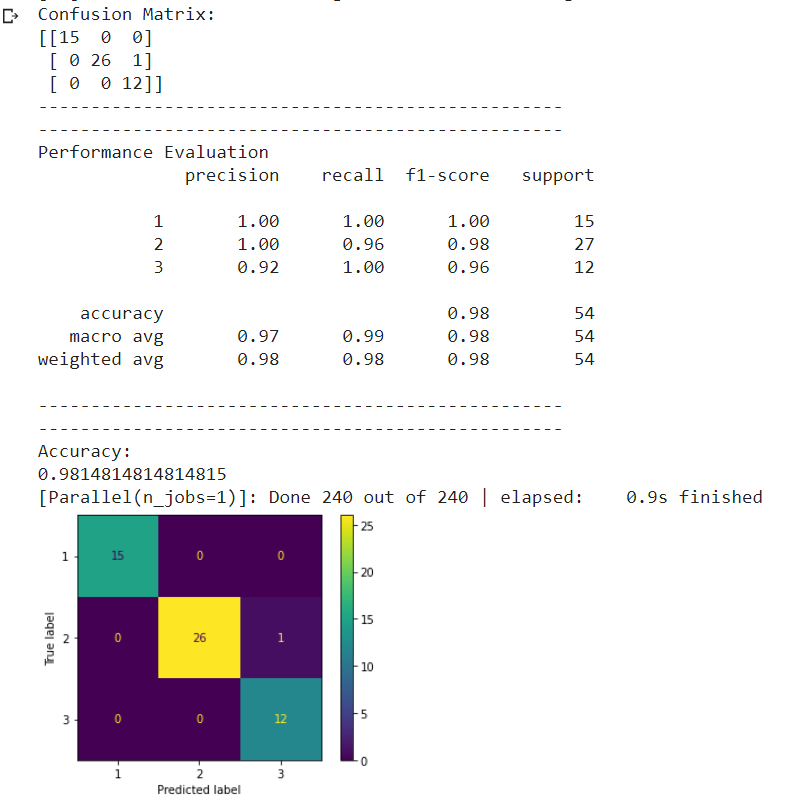
**print(accuracy\_score(y\_test, y\_pred))**

**import matplotlib.pyplot as plt**

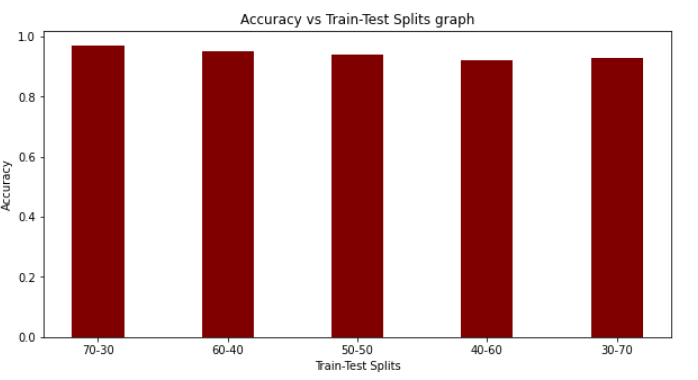
**from sklearn.metrics import plot\_confusion\_matrix**

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

**plt.show()**

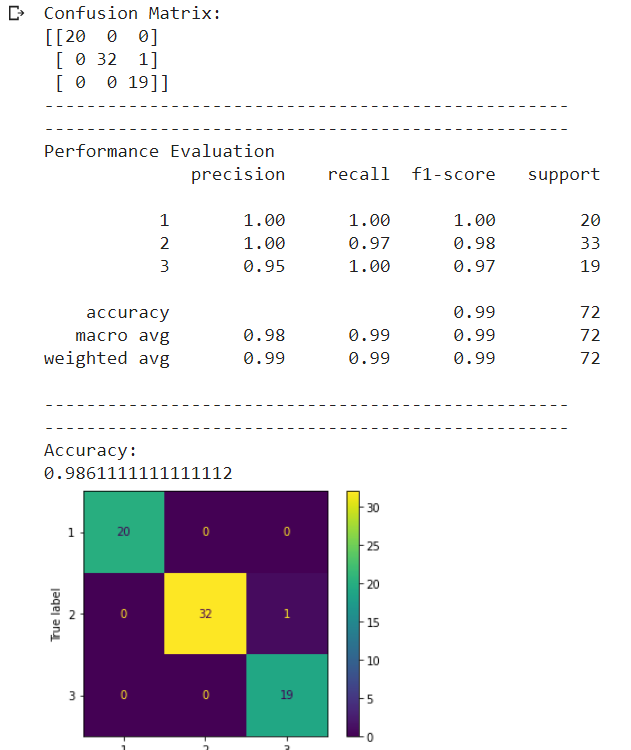
****

**COMPARISON:**

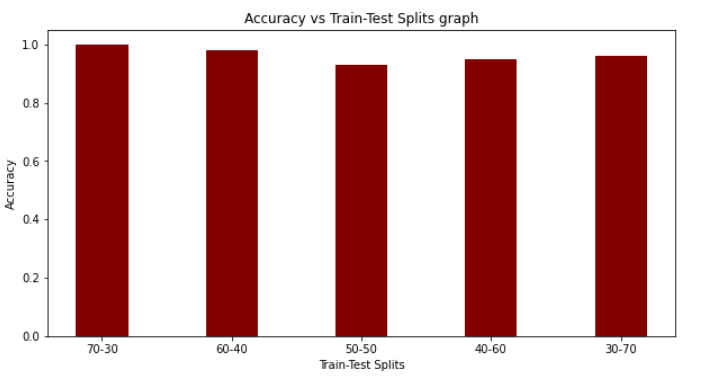
****

**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)**

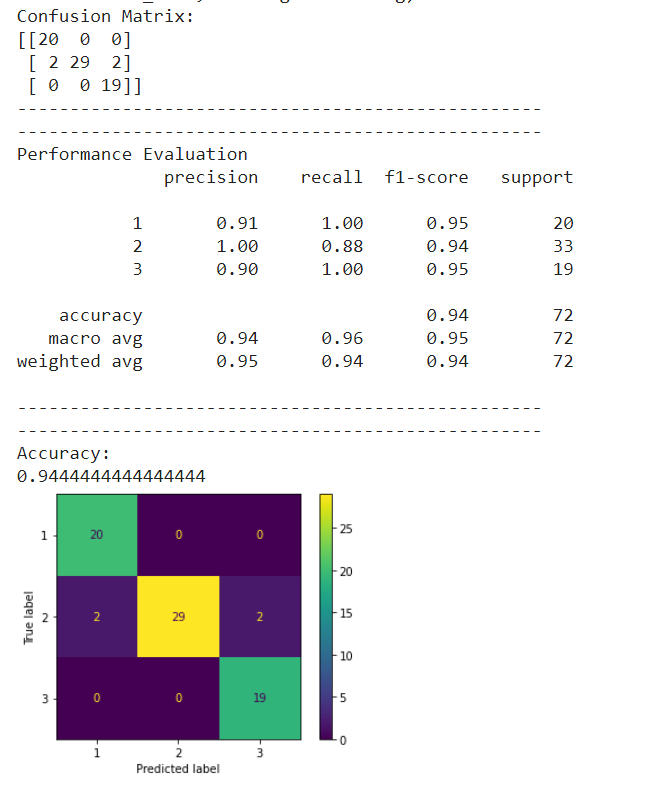
****

**COMPARISON:**

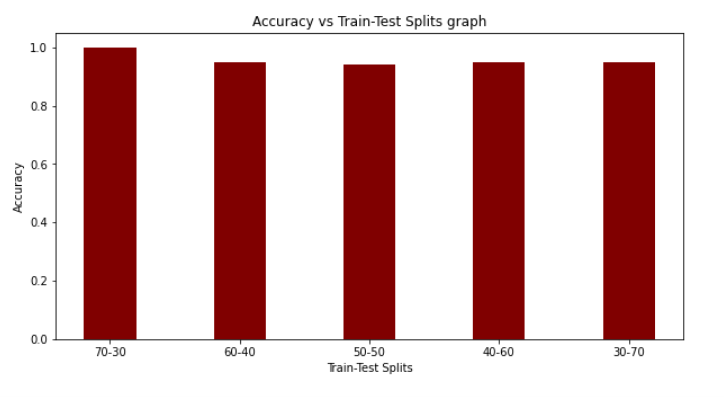
****

**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)**

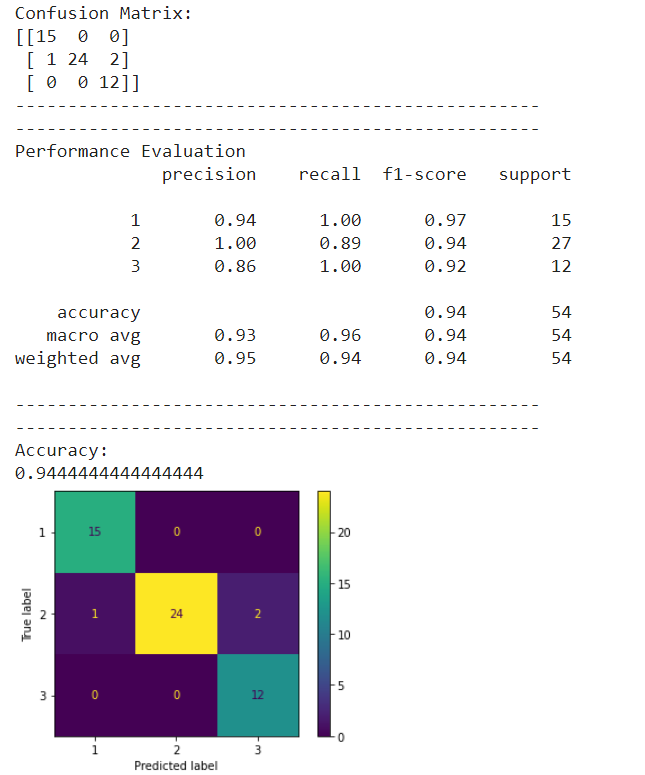
****

**COMPARISON:**

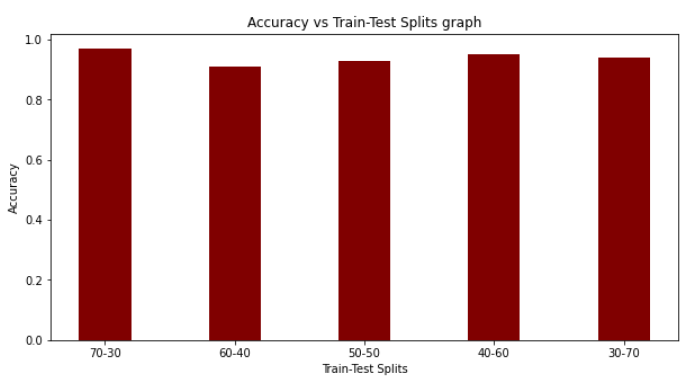
****

**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)**

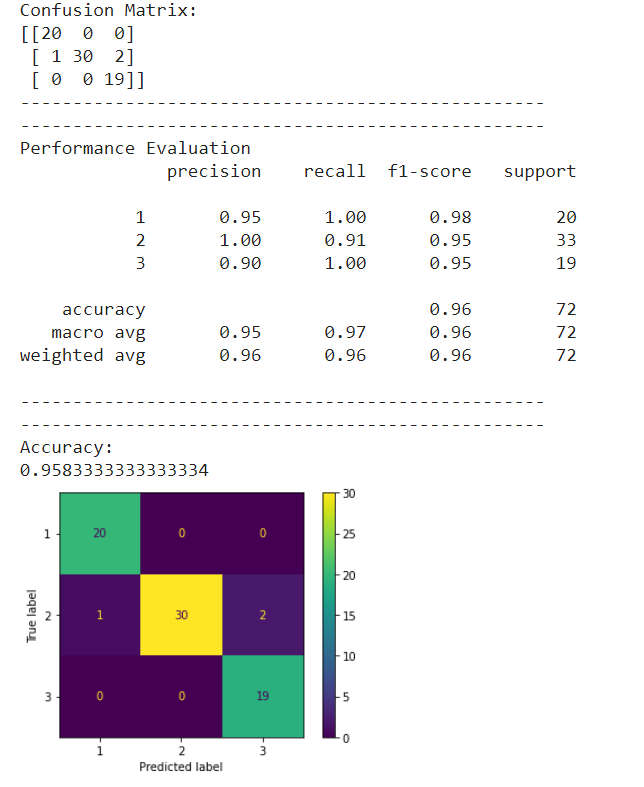
****

**COMPARISON:**

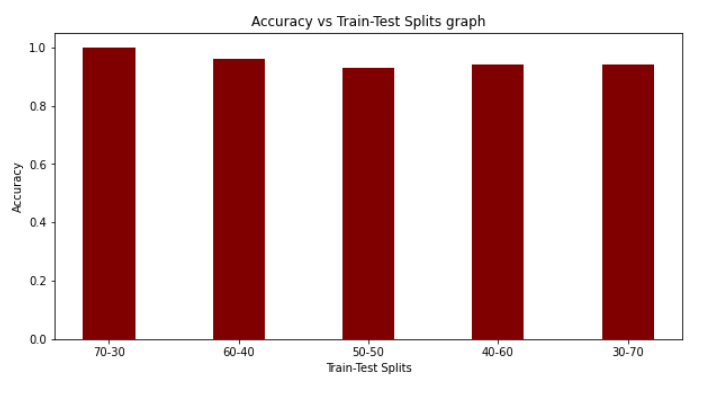
****

**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)**

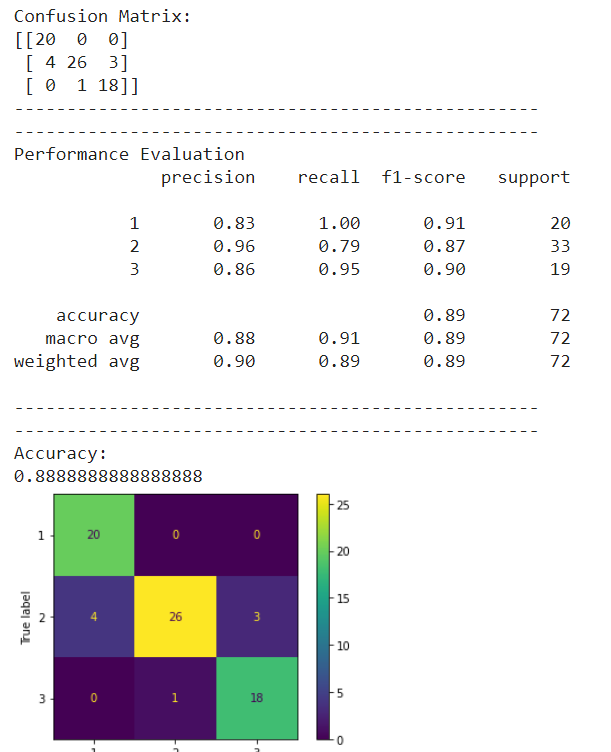
****

**COMPARISON:**

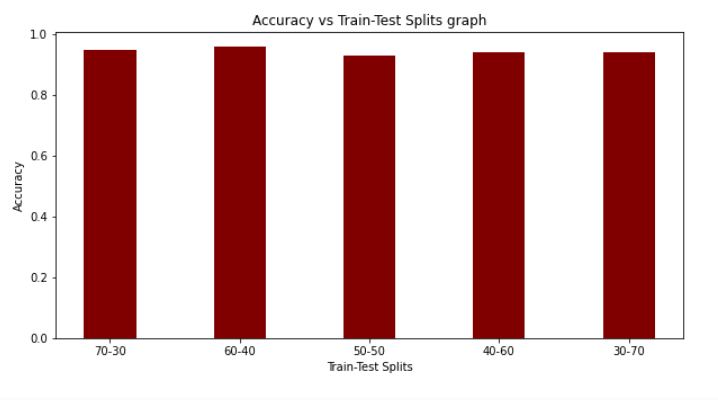
****

**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)**

****

**COMPARISON:**

****

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

1. **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()**

**############################################################################**

**# 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:")**

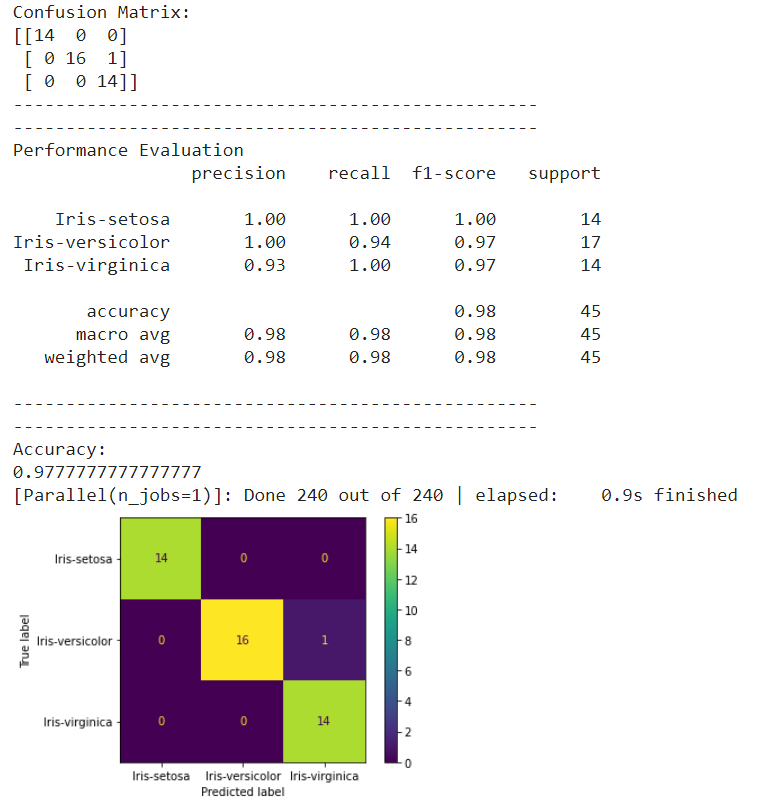
**print(accuracy\_score(y\_test, y\_pred))**

**import matplotlib.pyplot as plt**

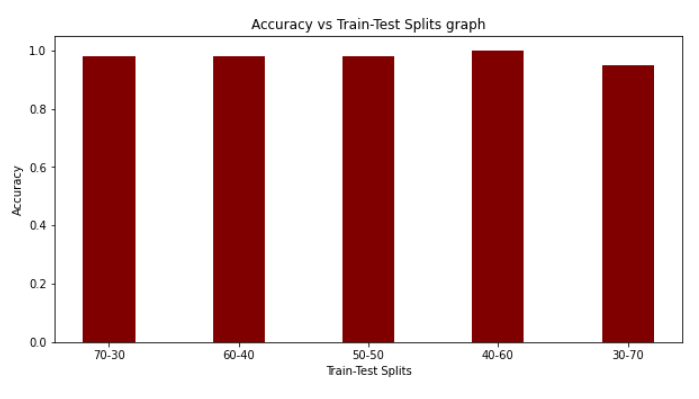
**from sklearn.metrics import plot\_confusion\_matrix**

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

**plt.show()**

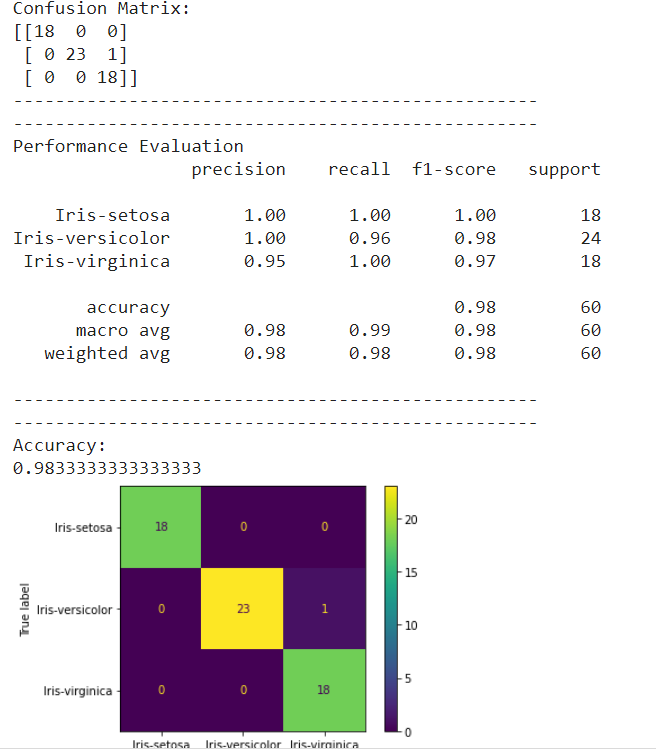
****

**COMPARISON:**

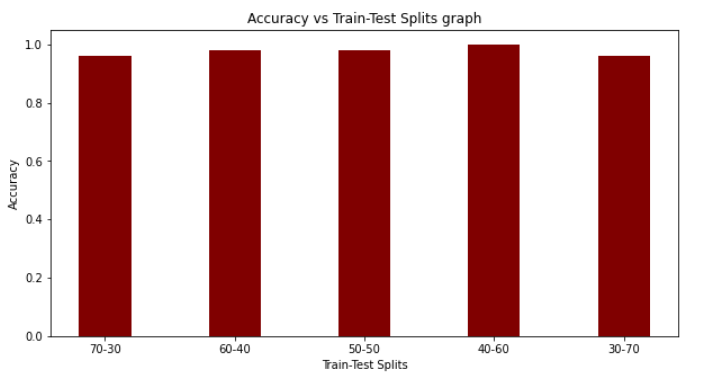
****

**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)**

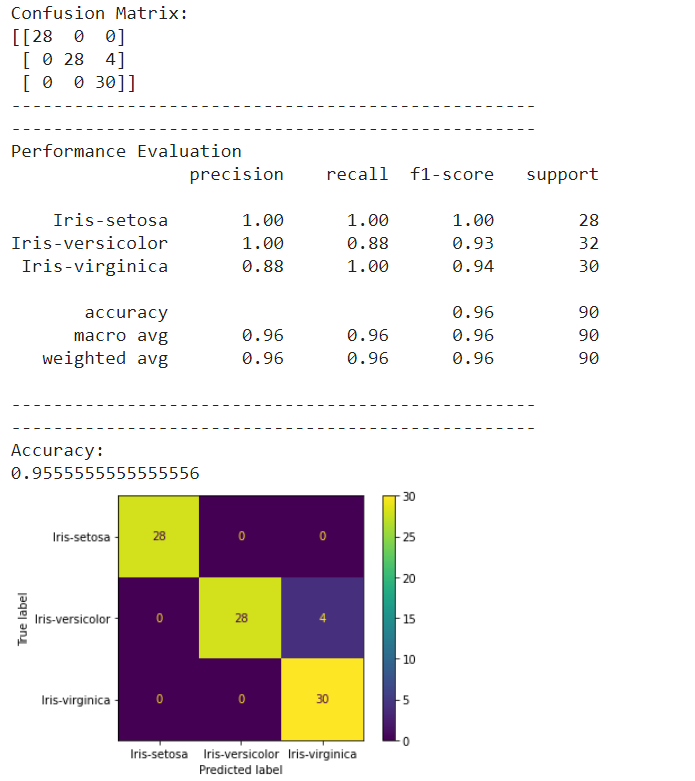
****

**COMPARISON:**

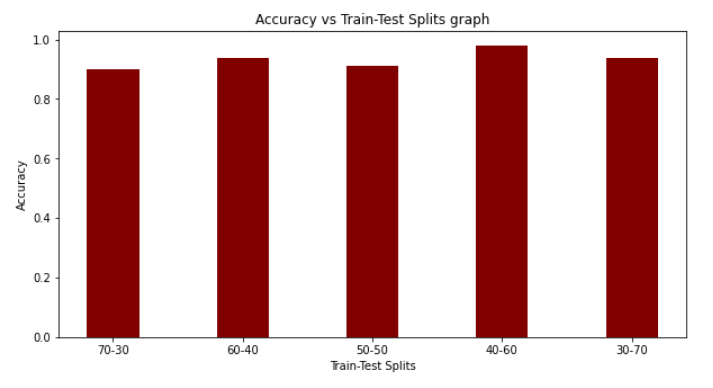
****

**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)**

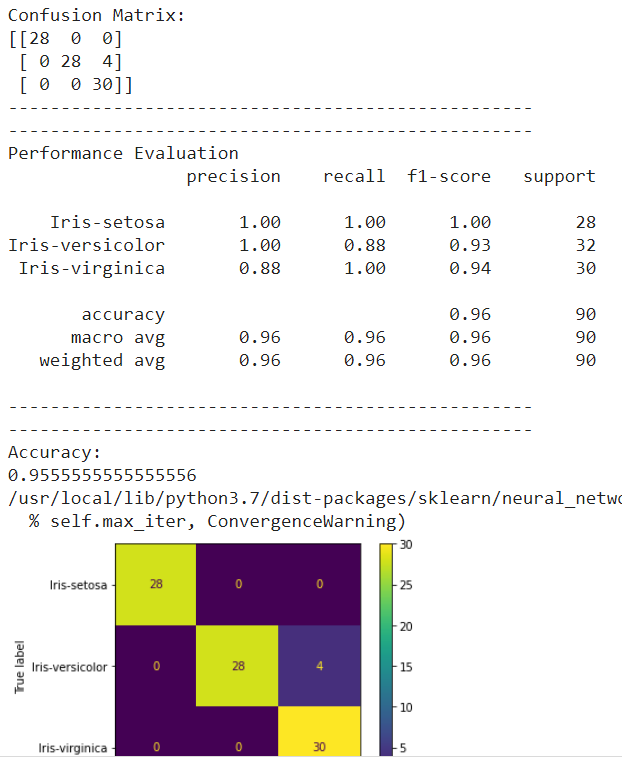
****

**COMPARISON:**

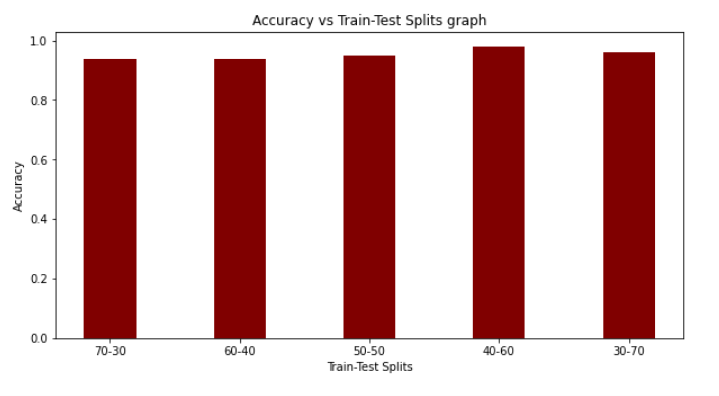
****

**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)**

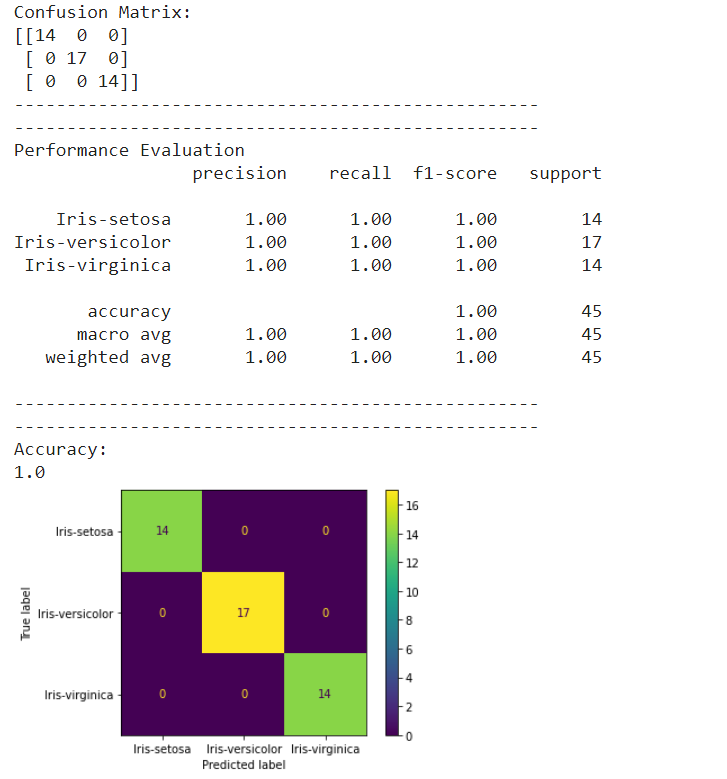
****

**COMPARISON:**

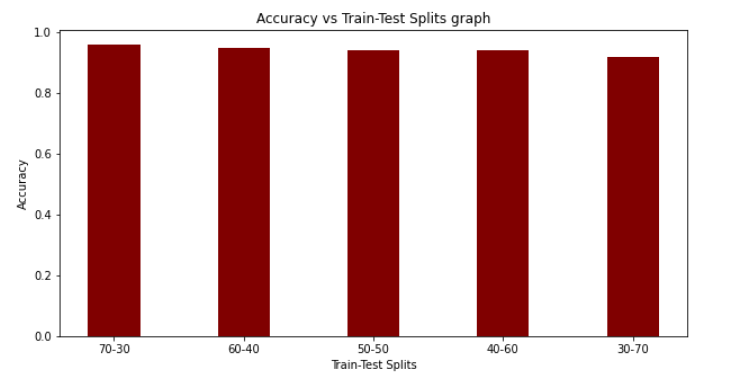
****

**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)**

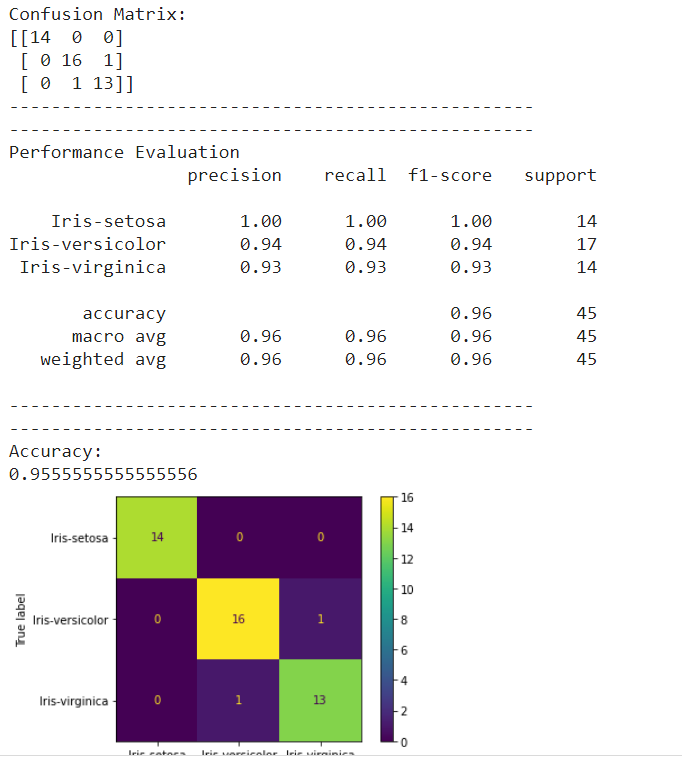
****

**COMPARISON:**

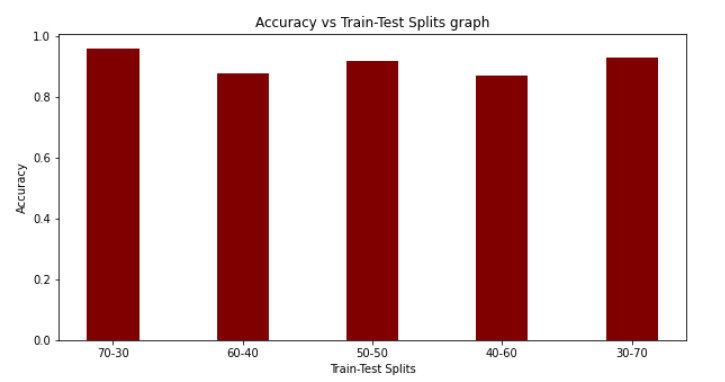
****

**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)**

****

**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. **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','Class']**

**df.columns = col\_name**

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

**y = df['Class']**

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

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,train\_size=0.7,test\_size=0.3,random\_state=10)**

**# Feature Scaling**

**from sklearn.preprocessing import StandardScaler**

**sc = StandardScaler()**

**X\_train = sc.fit\_transform(X\_train)**

**X\_test = sc.transform(X\_test)**

**# Classification**

**from sklearn.svm import SVC**

**classifier = SVC()**

**############################################################################**

**# Showing all the parameters**

**from pprint import pprint**

**# Look at parameters used by our current forest**

**print('Parameters currently in use:\n')**

**pprint(classifier.get\_params())**

**############################################################################**

**# Creating a set of important sample features**

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

**pprint(param\_grid)**

**############################################################################**

**from sklearn.model\_selection import GridSearchCV**

**# Use the random grid to search for best hyperparameters**

**# First create the base model to tune**

**classifier = SVC()**

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

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

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

**rf\_random.fit(X\_train, y\_train)**

**y\_pred = rf\_random.predict(X\_test)**

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

**print("Confusion Matrix:")**

**print(confusion\_matrix(y\_test, y\_pred))**

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

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

**print("Performance Evaluation")**

**print(classification\_report(y\_test, y\_pred))**

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

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

**print("Accuracy:")**

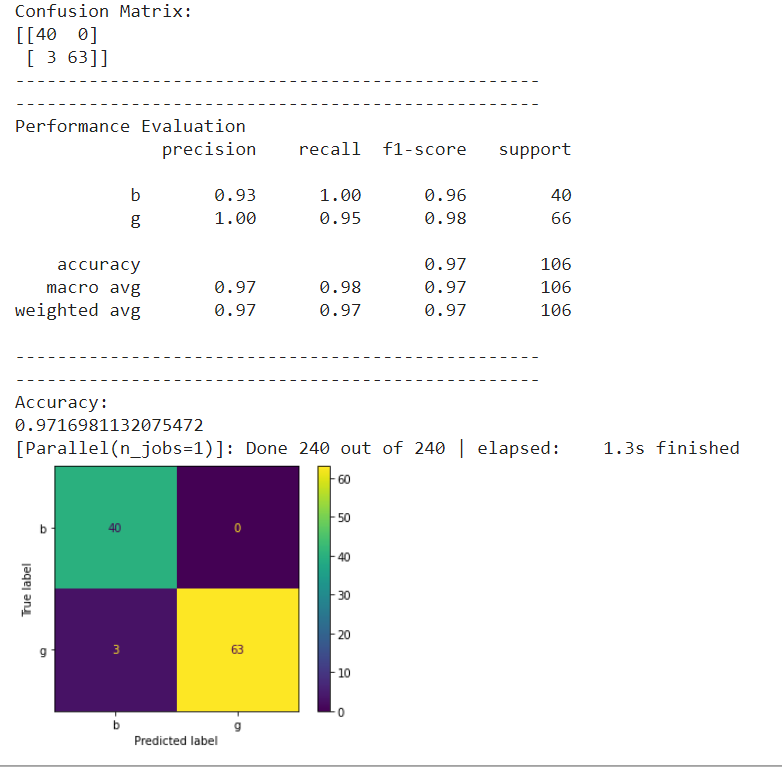
**print(accuracy\_score(y\_test, y\_pred))**

**import matplotlib.pyplot as plt**

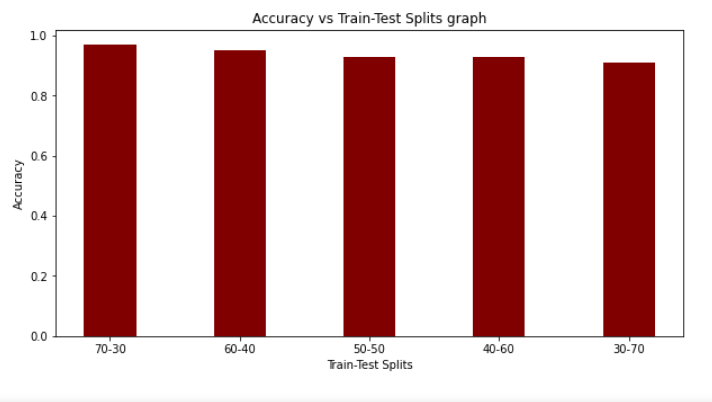
**from sklearn.metrics import plot\_confusion\_matrix**

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

**plt.show()**

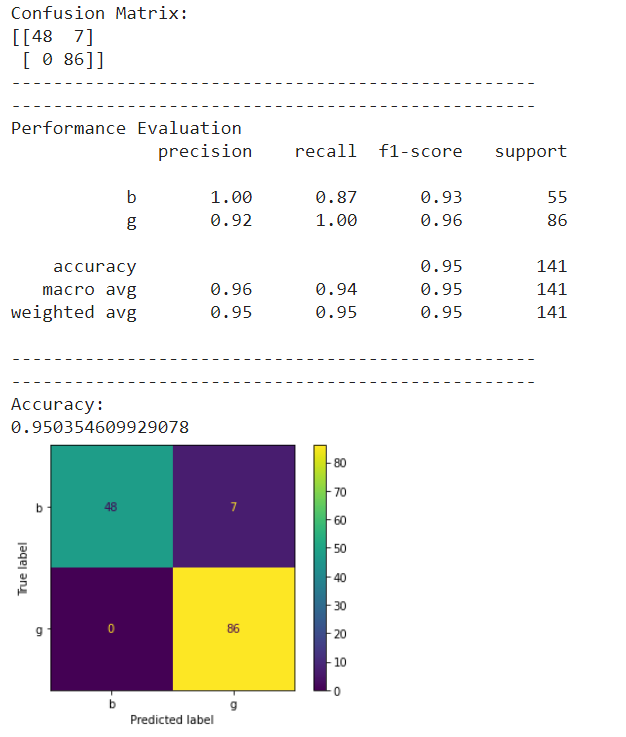
****

**COMPARISON:**

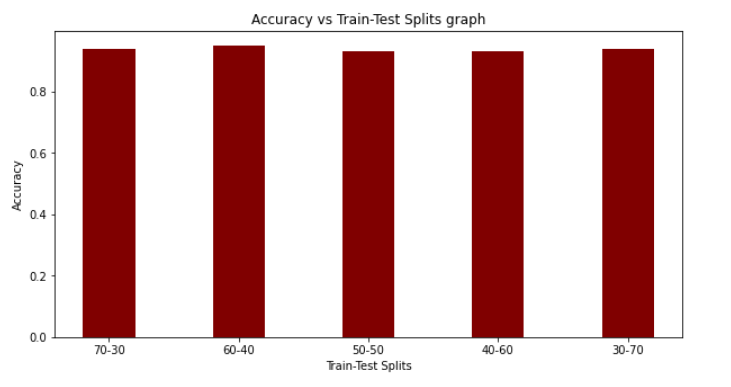
****

**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)**

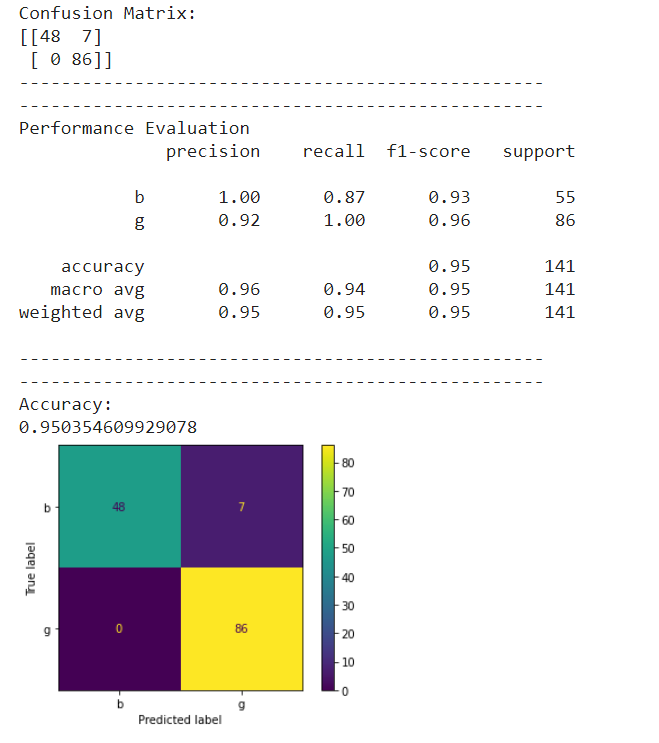
****

**COMPARISON:**

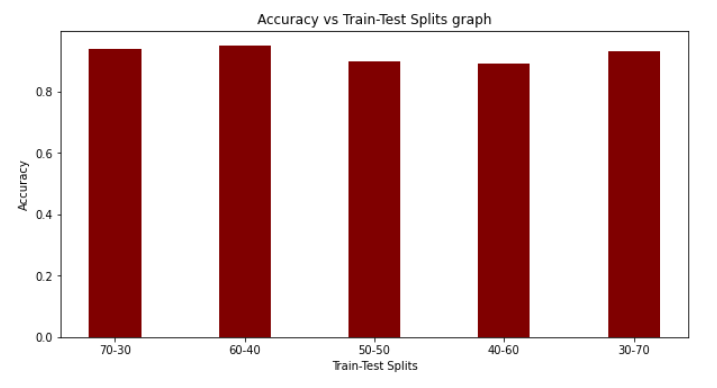
****

**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)**

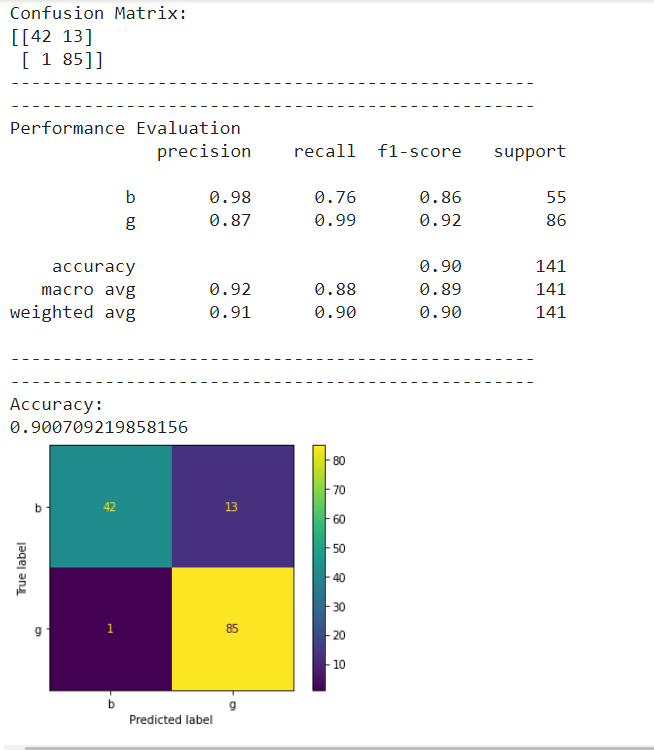
****

**COMPARISON:**

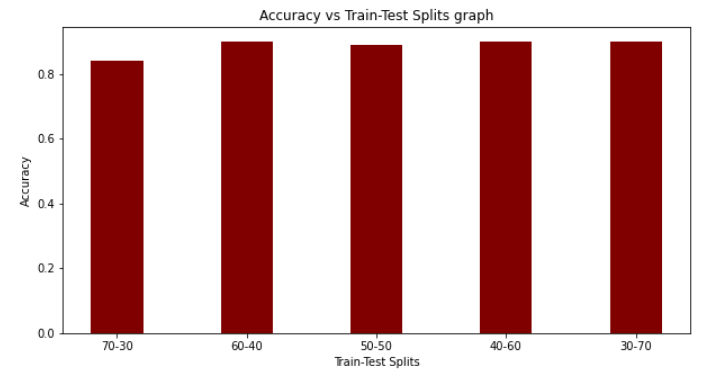
****

**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)**

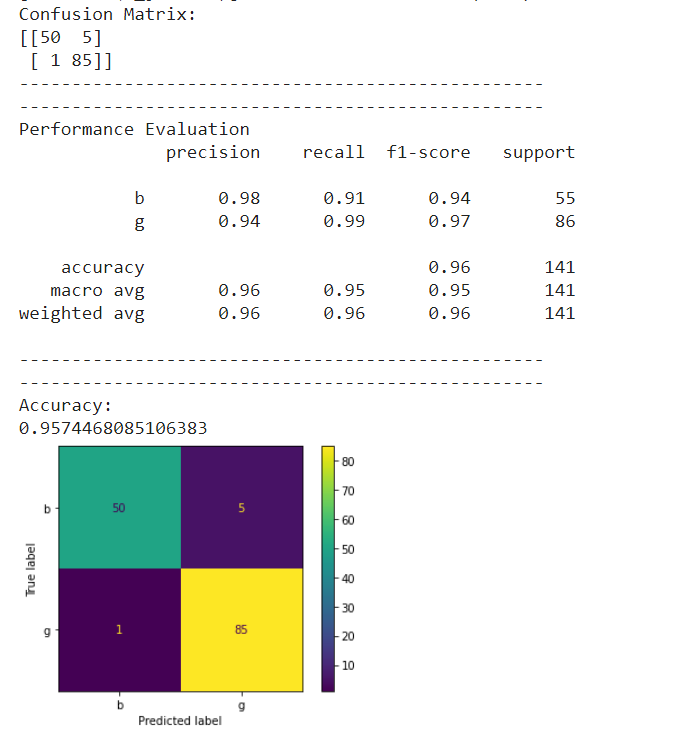
****

**COMPARISON:**

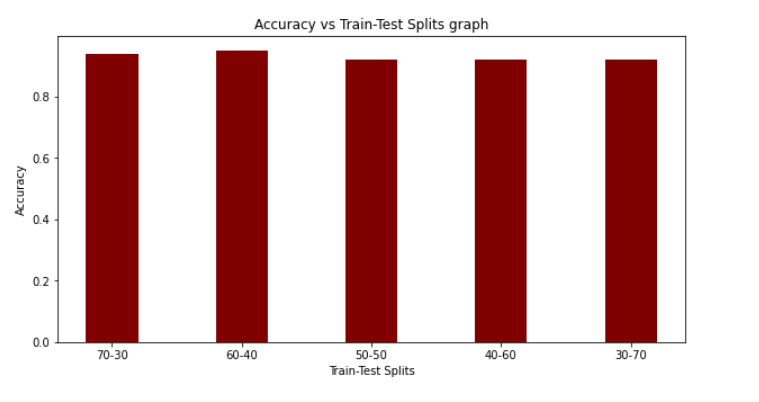
****

**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)**

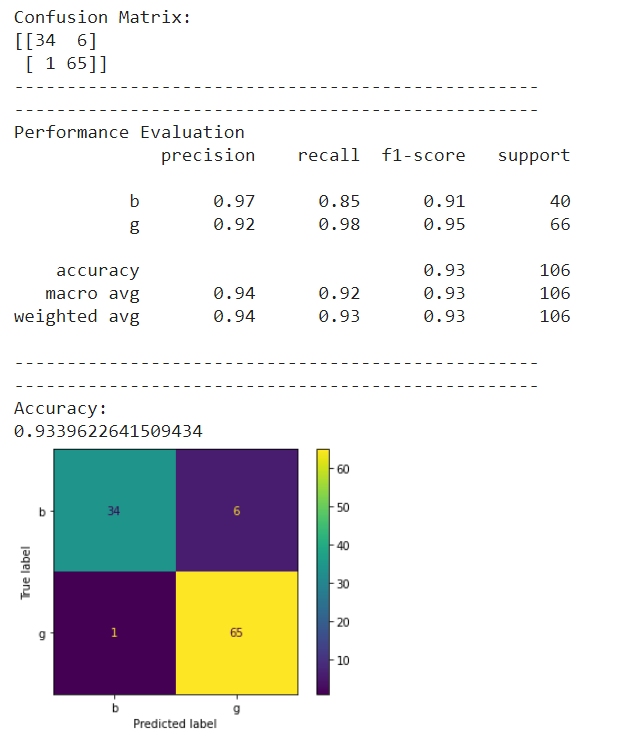
****

**COMPARISON:**

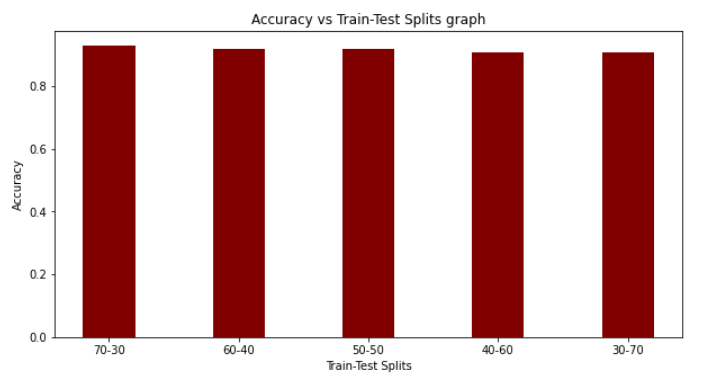
****

**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)**

****

**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. **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))**

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

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

**print("Performance Evaluation")**

**print(classification\_report(y\_test, y\_pred))**

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

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

**print("Accuracy:")**

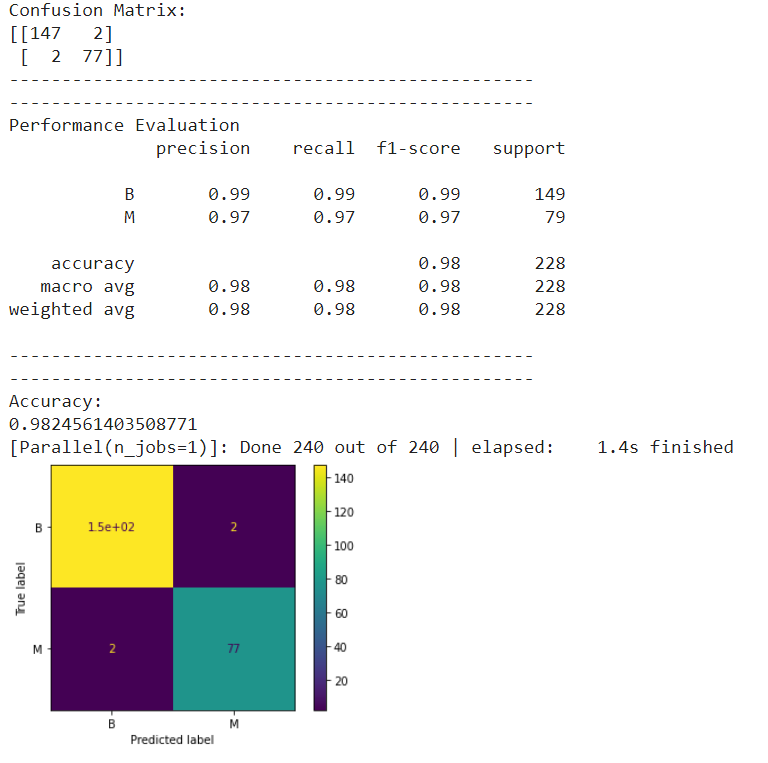
**print(accuracy\_score(y\_test, y\_pred))**

**import matplotlib.pyplot as plt**

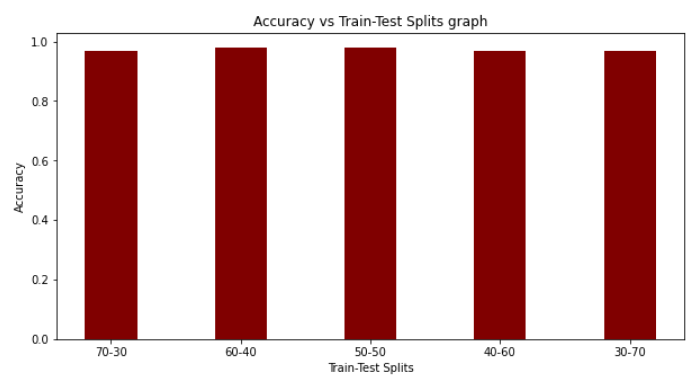
**from sklearn.metrics import plot\_confusion\_matrix**

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

**plt.show()**

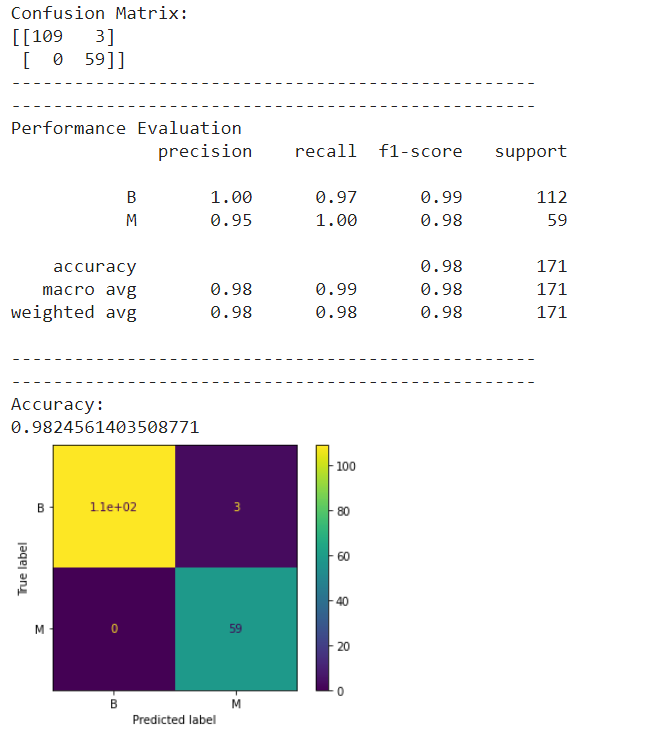
****

**COMPARISON:**

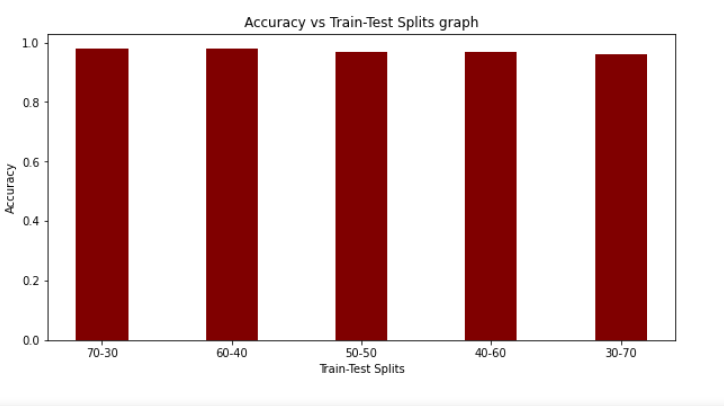
****

**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)**

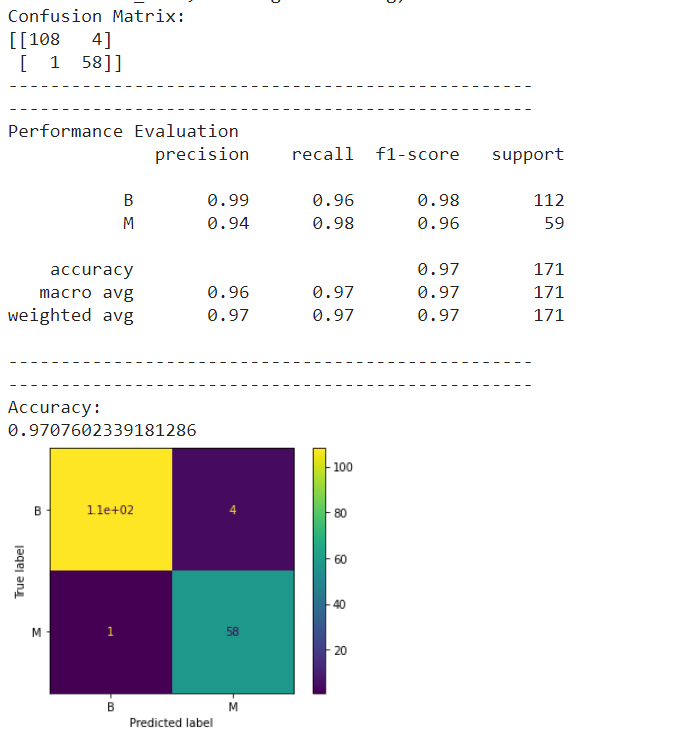
****

**COMPARISON:**

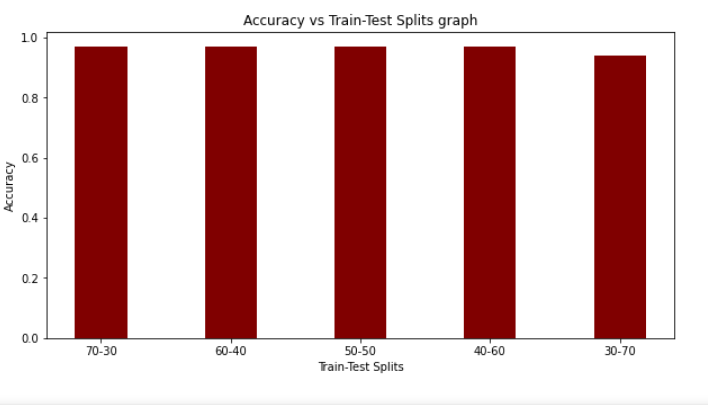
****

**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)**

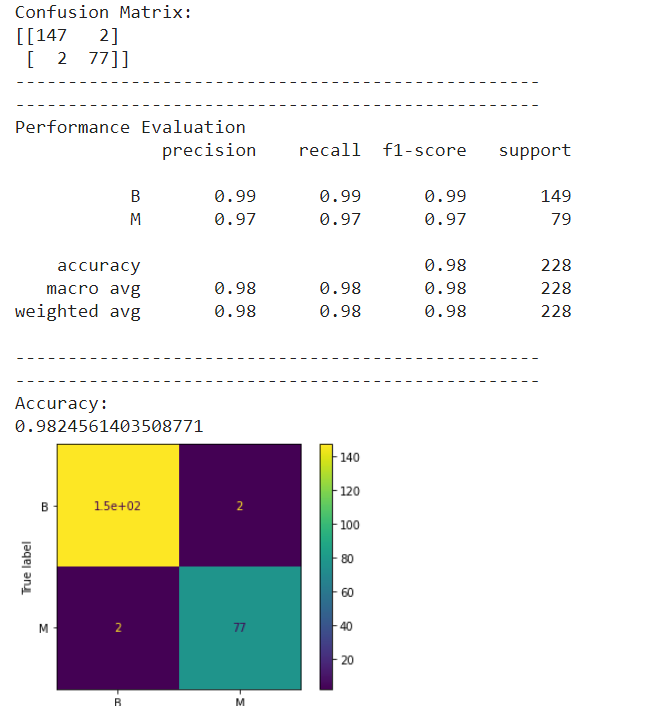
****

**COMPARISON:**

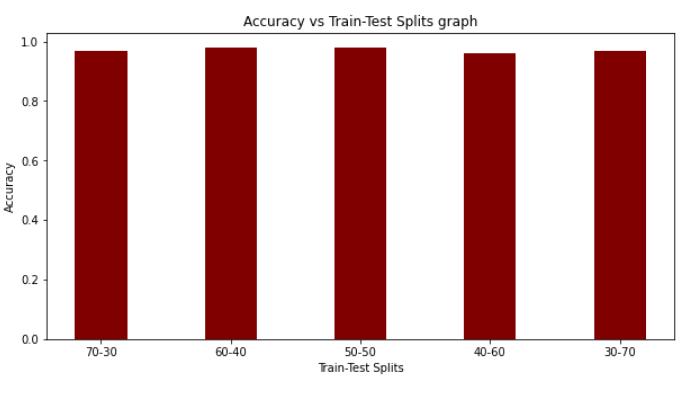
****

**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)**

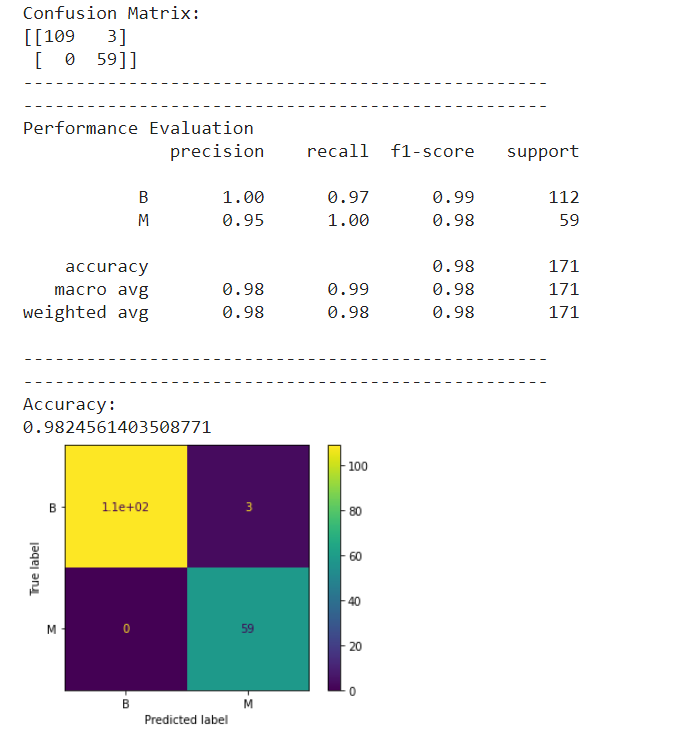
****

**COMPARISON:**

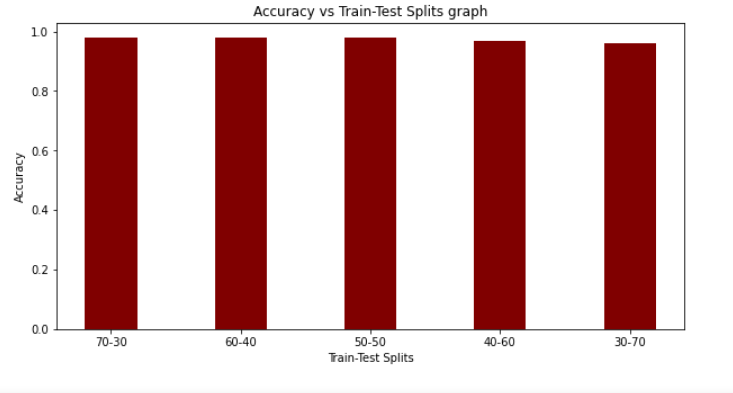
****

**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)**

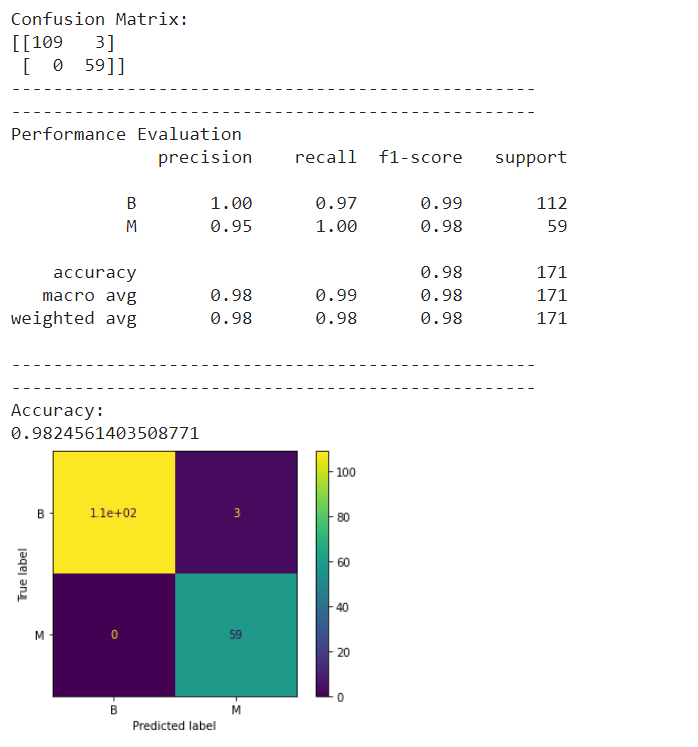
****

**COMPARISON:**

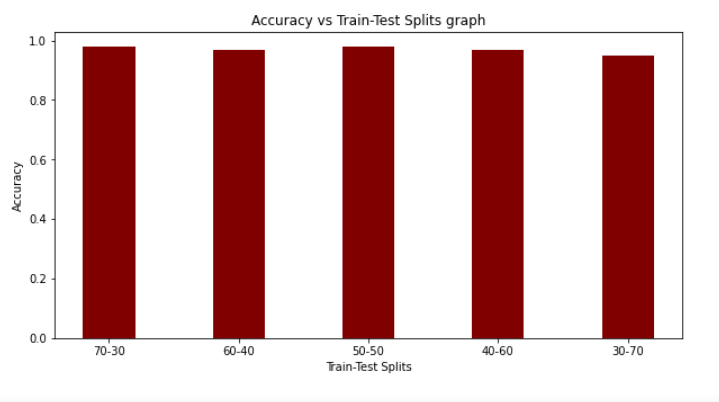
****

**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)**

****

**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.**

1. **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**

**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))**

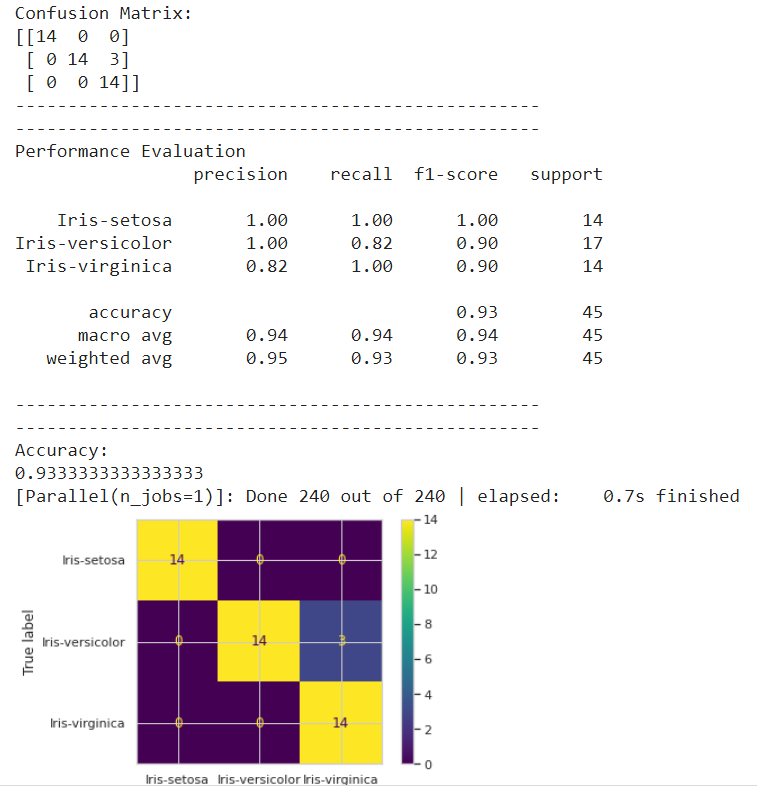
**import matplotlib.pyplot as plt**

**from sklearn.metrics import plot\_confusion\_matrix**

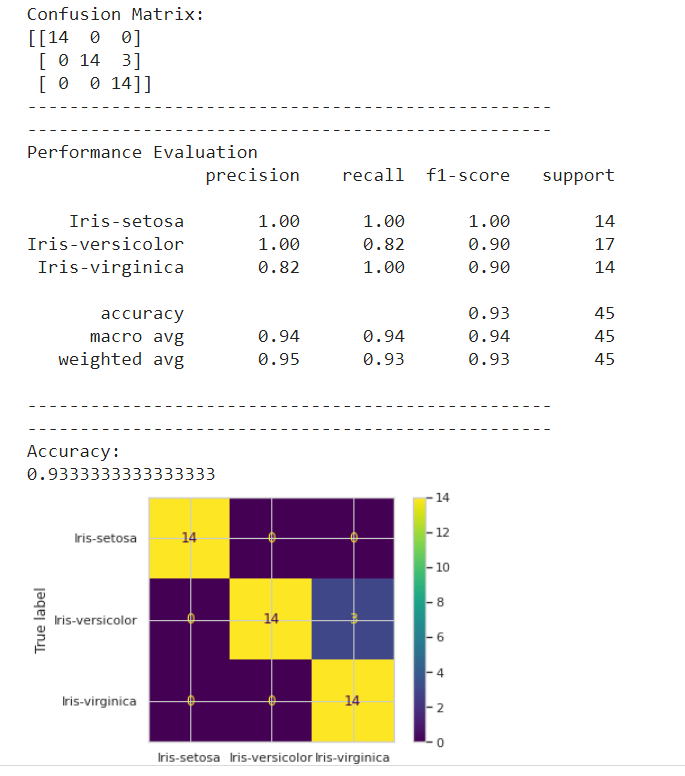
**plot\_confusion\_matrix(rf\_random, X\_test, y\_test)**

**plt.show()**

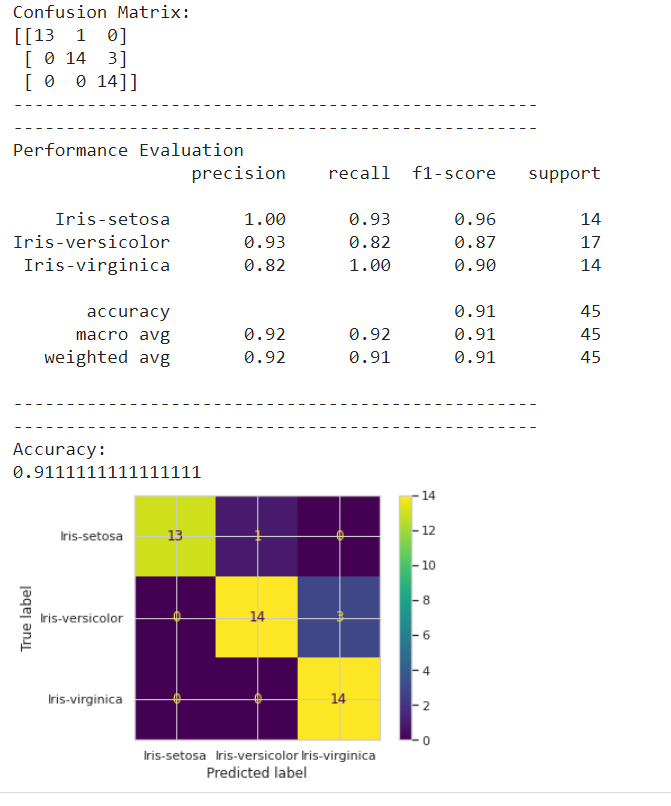
**5.1.1 SVM Classifier(With Tuning)**

****

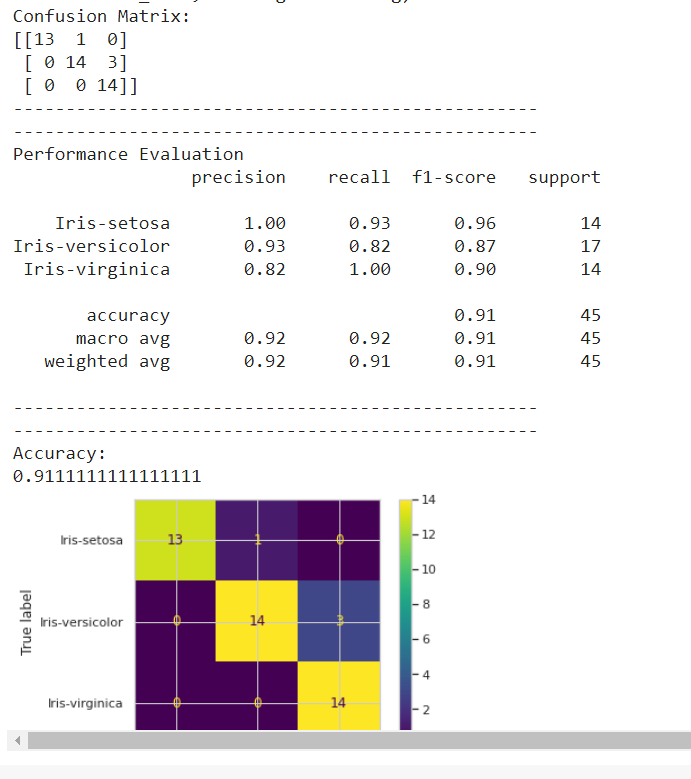
**5.1.2 SVM Classifier(Without Tuning)**

****

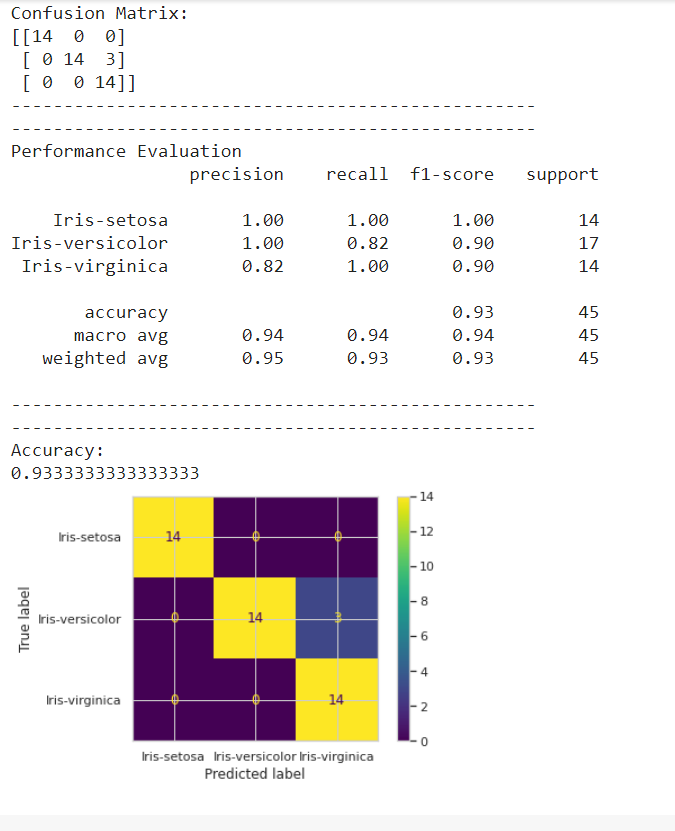
**5.1.3 MLP Classifier(With Tuning)**

****

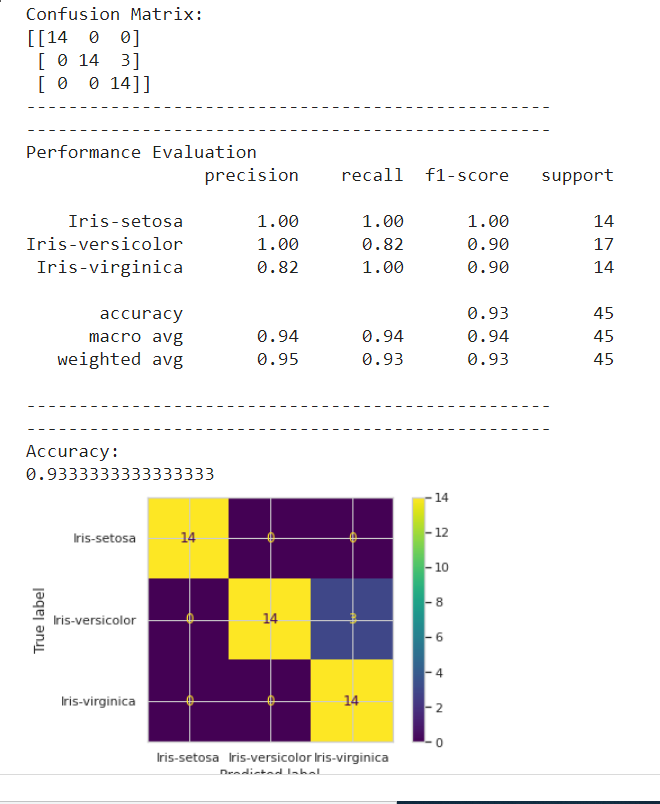
**5.1.4 MLP Classifier(Without Tuning)**

****

**5.1.5 Random Forest Classifier(With Tuning)**

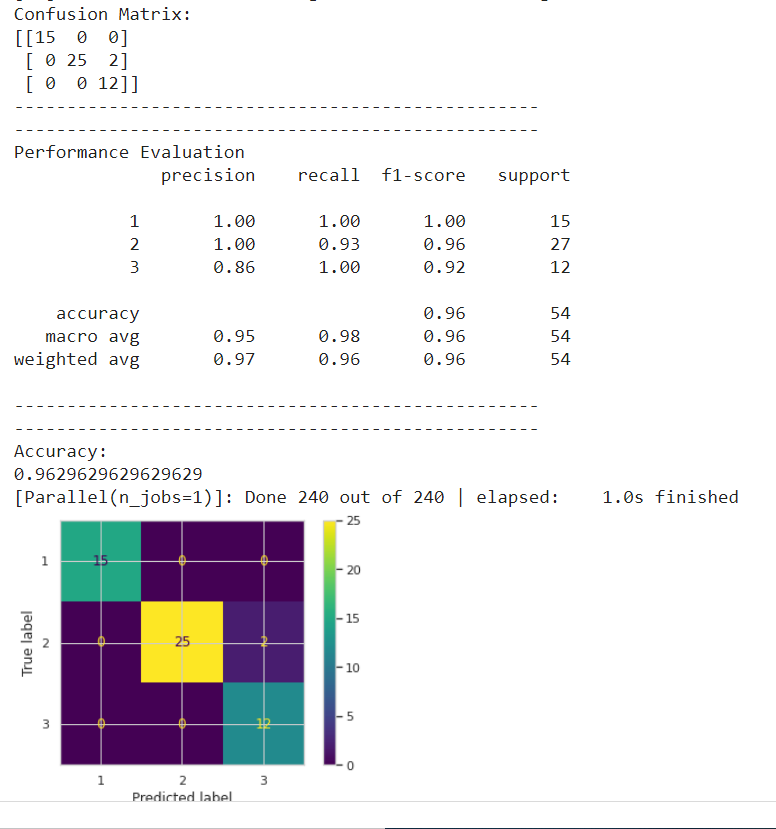
****

**5.1.6 Random Forest Classifier(Without Tuning)**

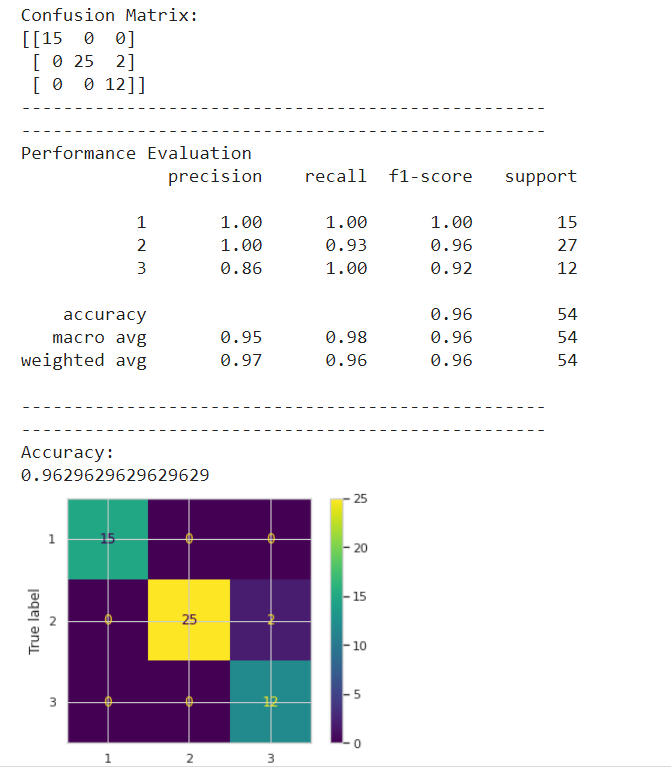
****

**5.2 Wine Dataset**

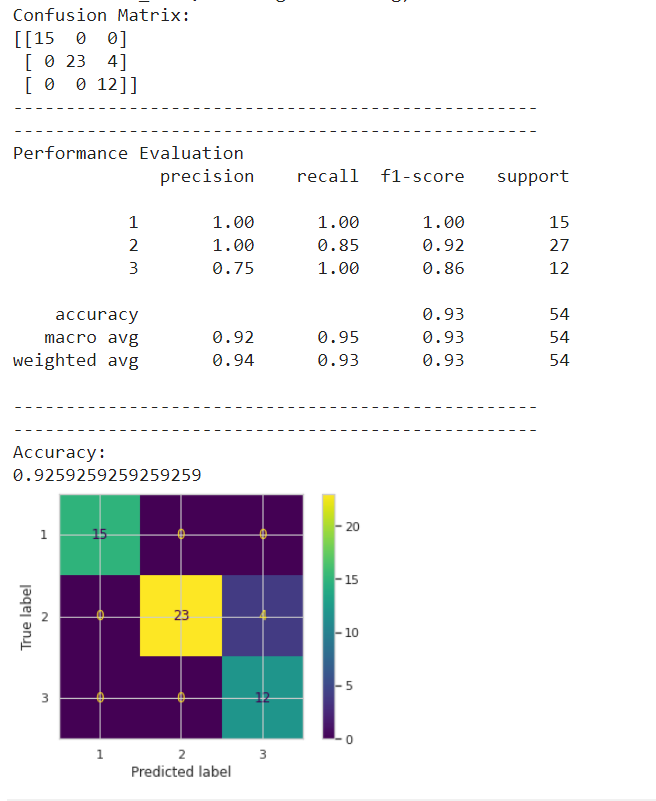
**5.2.1 SVM Classifier(With Tuning)**

****

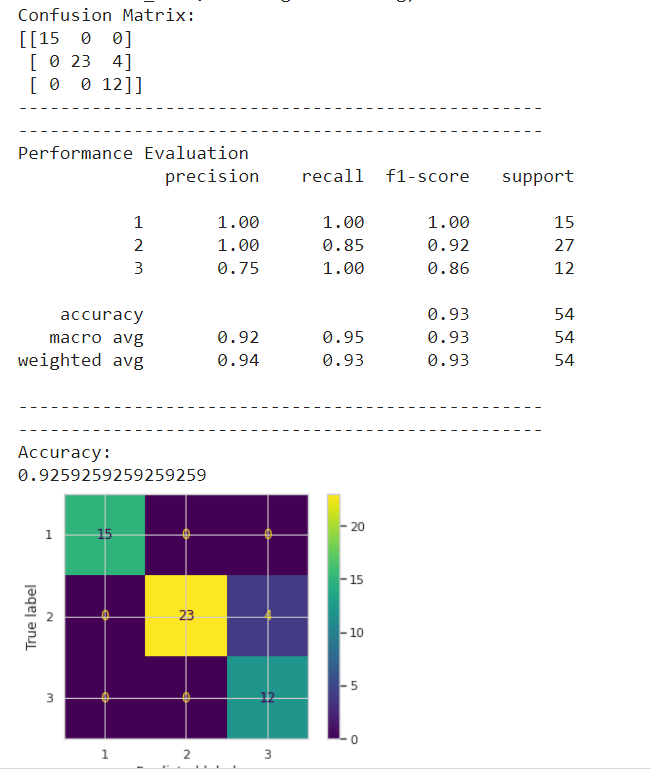
**5.2.2 SVM Classifier(Without Tuning)**

****

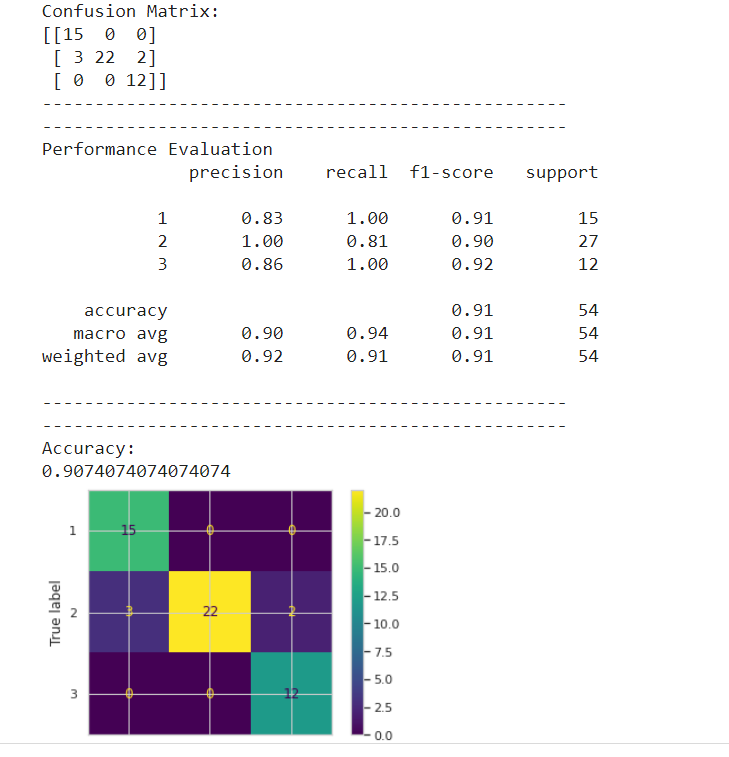
**5.2.3 MLP Classifier(With Tuning)**

****

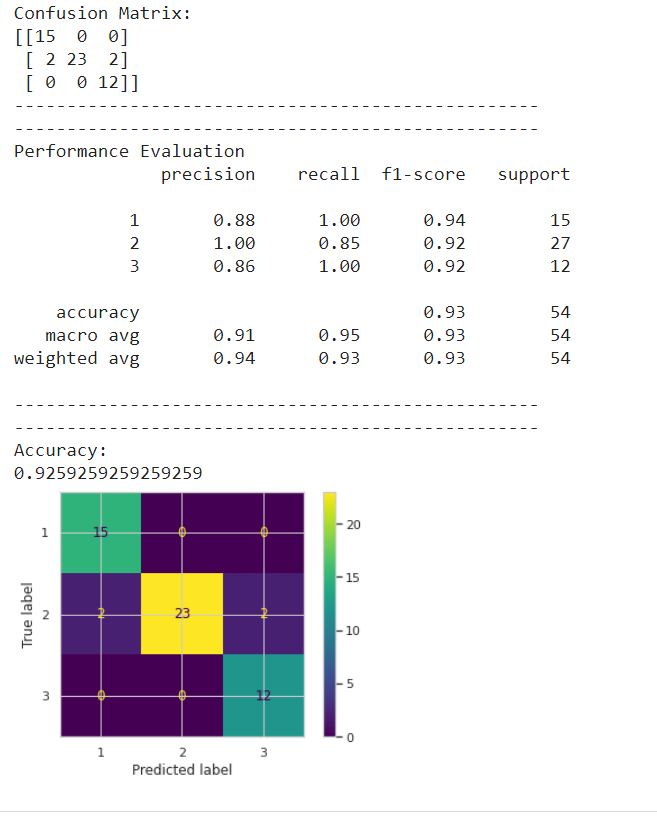
**5.2.4 MLP Classifier(Without Tuning)**

****

**5.2.5 Random Forest Classifier(With Tuning)**

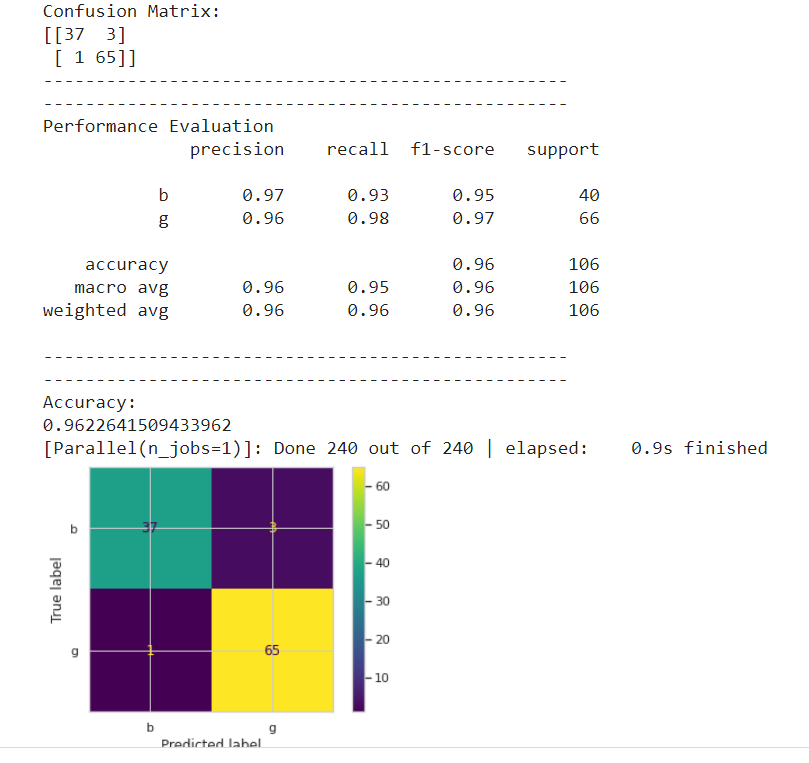
****

**5.2.6 Random Forest Classifier(Without Tuning)**

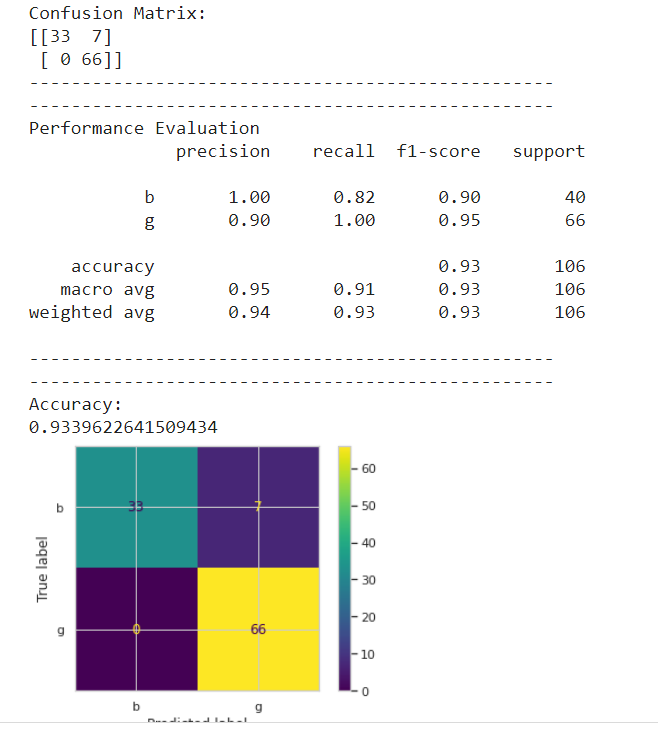
****

**5.3 Ionosphere Dataset**

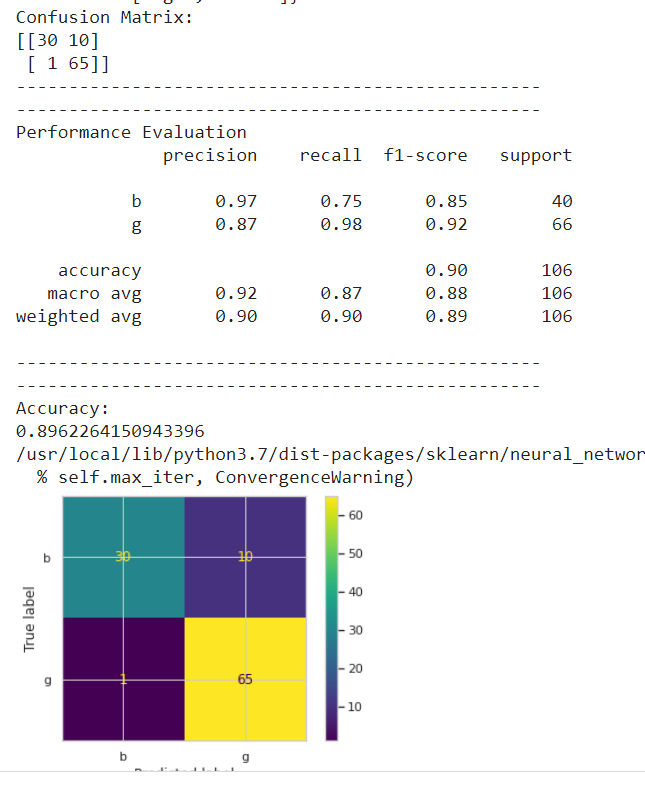
**5.3.1 SVM Classifier(With Tuning)**

****

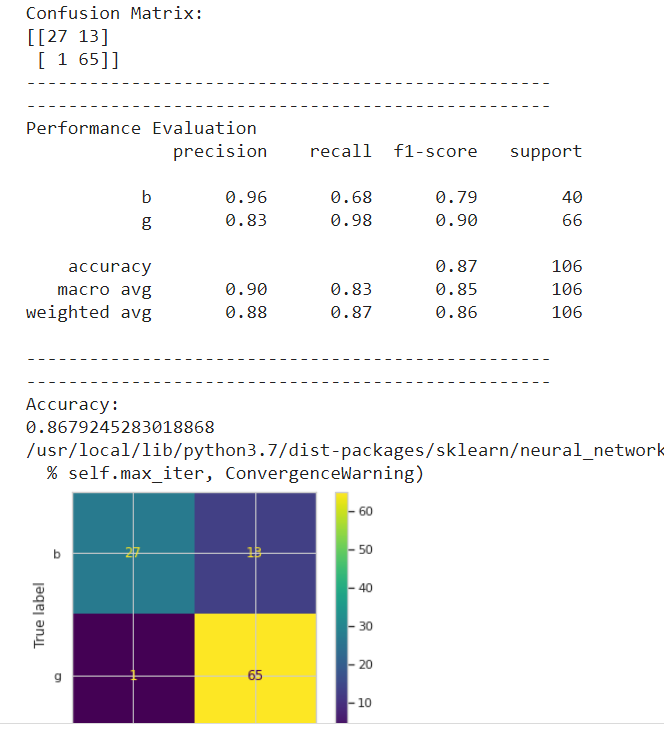
**5.3.2 SVM Classifier(Without Tuning)**

****

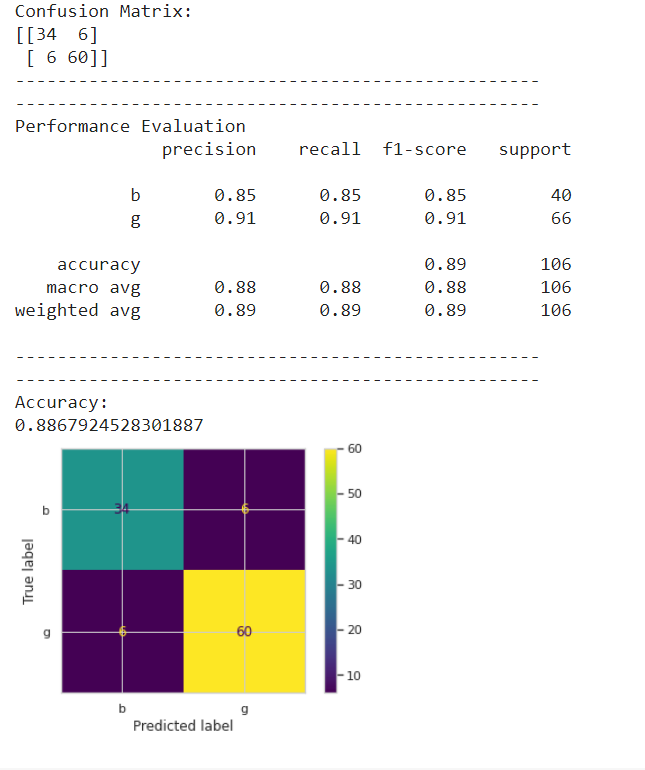
**5.3.3 MLP Classifier(With Tuning)**

****

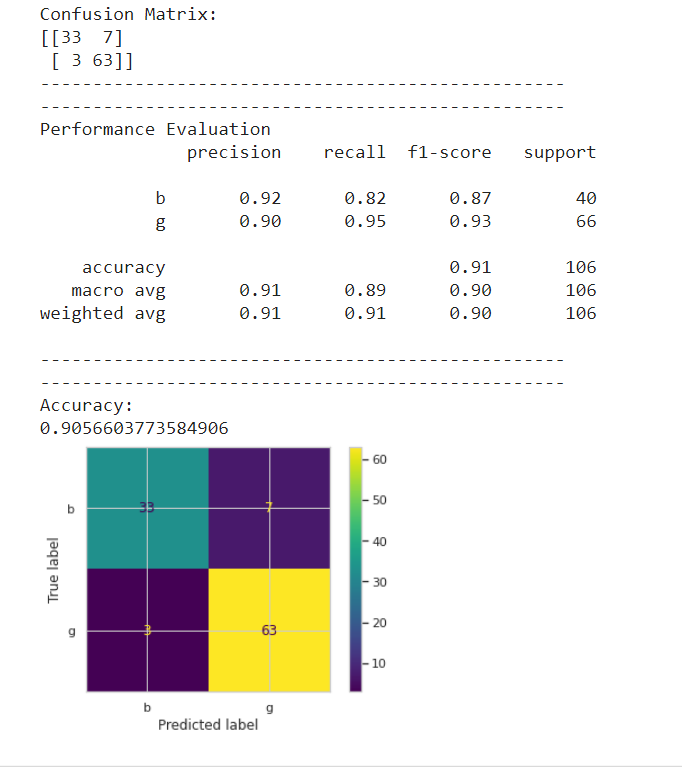
**5.3.4 MLP Classifier(Without Tuning)**

****

**5.3.5 Random Forest Classifier(With Tuning)**

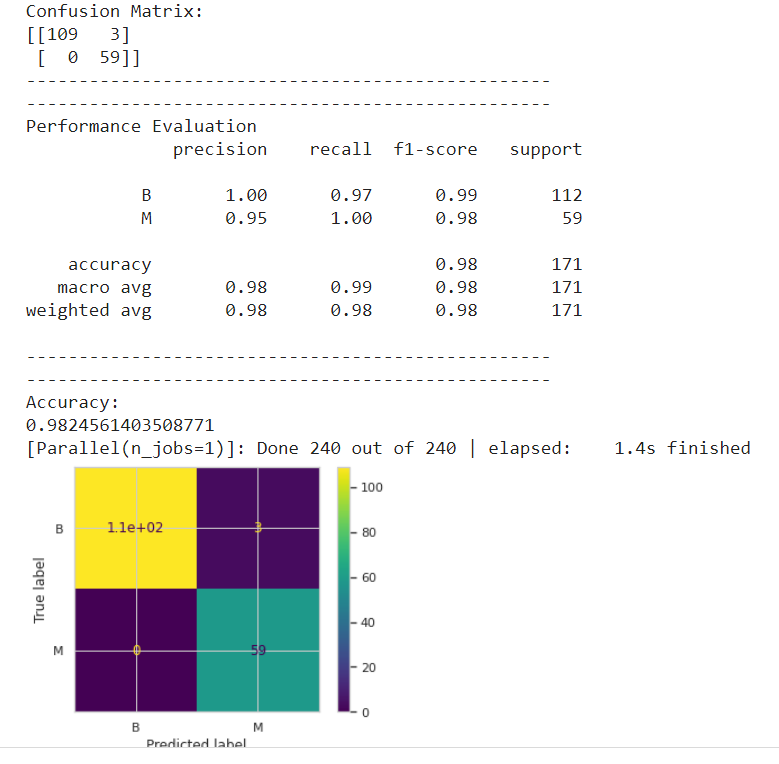
****

**5.3.6 Random Forest Classifier(Without Tuning)**

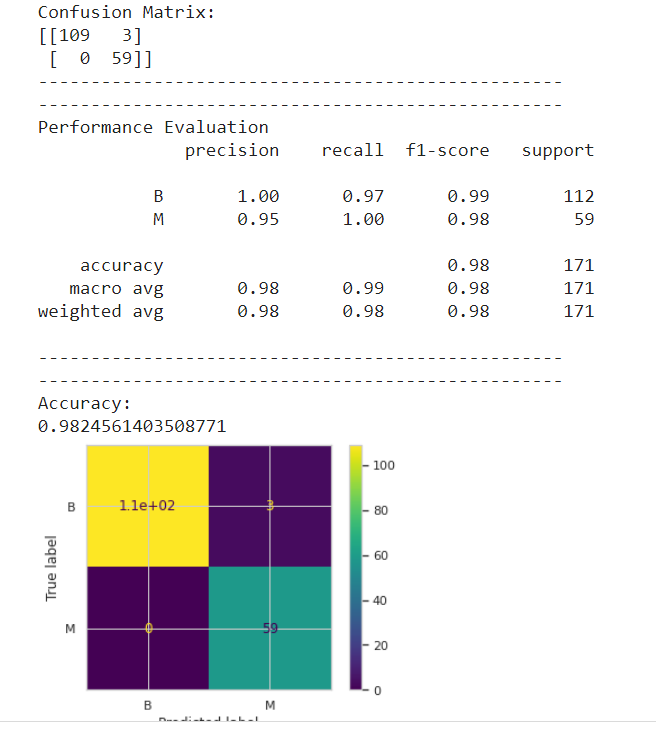
****

**5.4 Iris Plant Dataset**

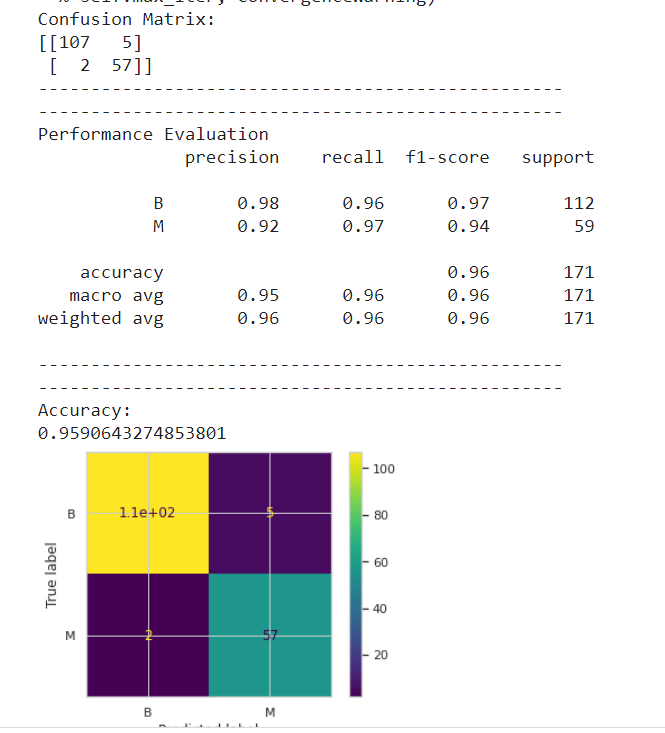
**5.4.1 SVM Classifier(With Tuning)**

****

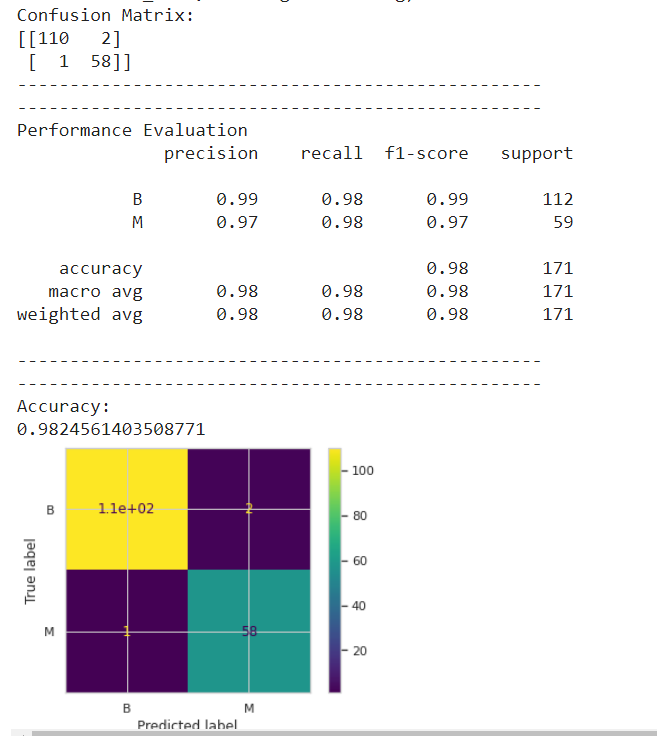
**5.4.2 SVM Classifier(Without Tuning)**

****

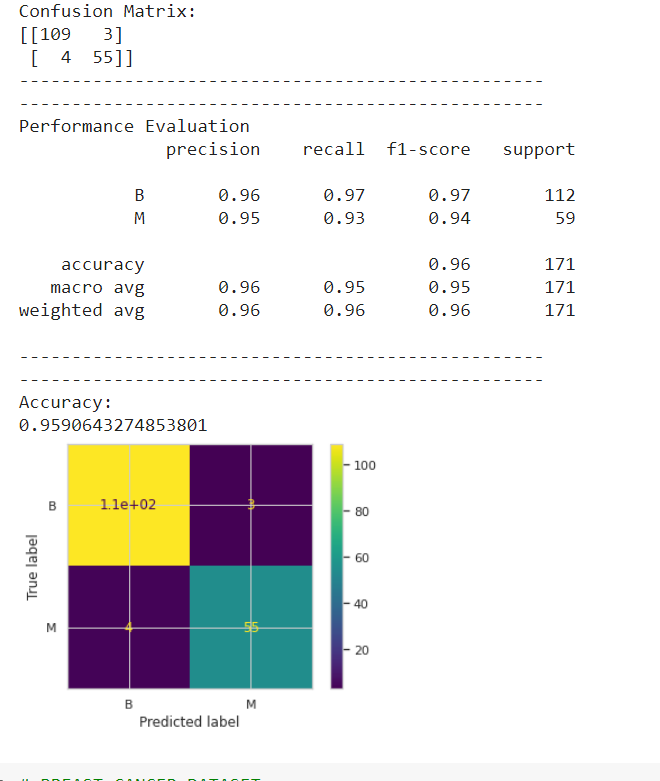
**5.4.3 MLP Classifier(With Tuning)**

****

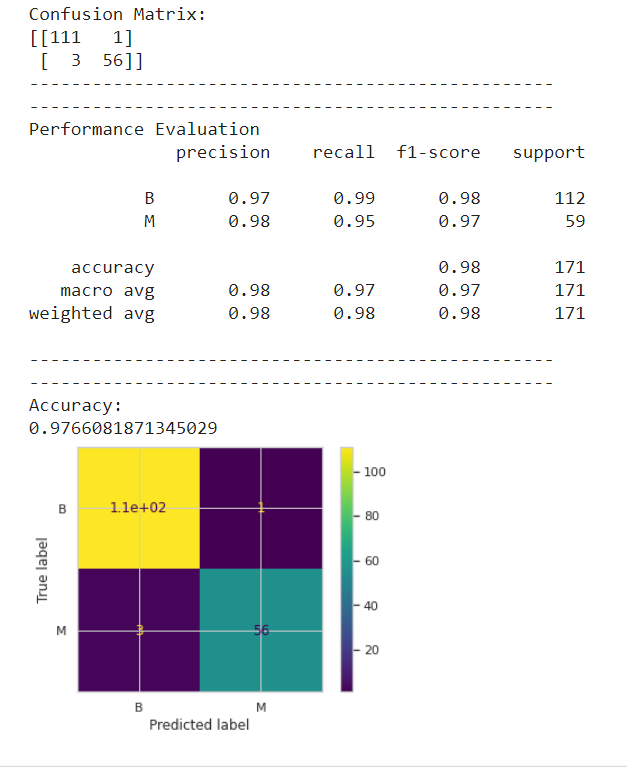
**5.4.4 MLP Classifier(Without Tuning)**

****

**5.4.5 Random Forest Classifier(With Tuning)**

****

**5.4.6 Random Forest Classifier(Without Tuning)**

****

**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.**