CO542 - NEURAL NETWORKS AND FUZZY SYSTEMS

MULTI LAYER PERCEPTRONS (MLP)

F.S.MARZOOK

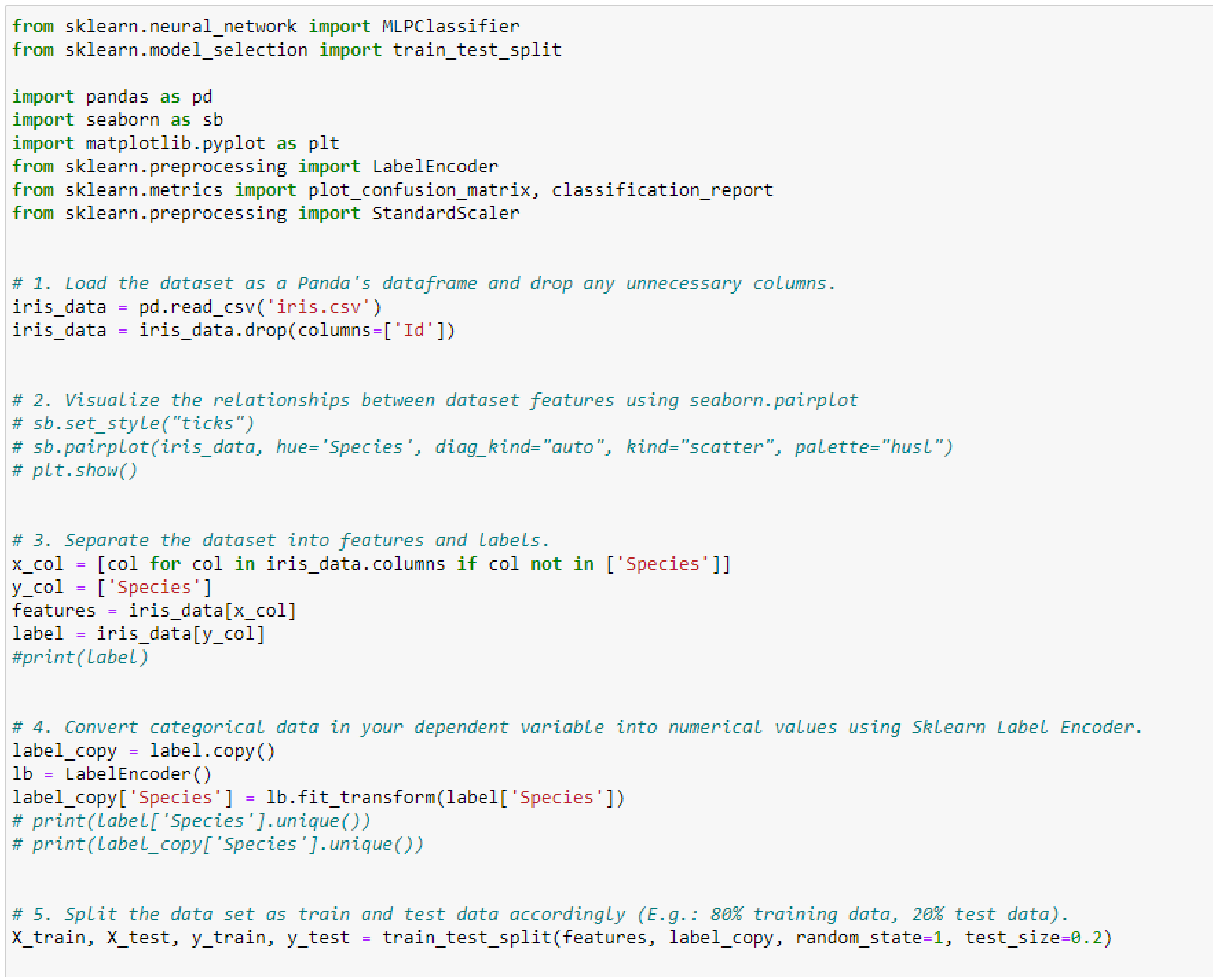
E/16/232

LAB 04

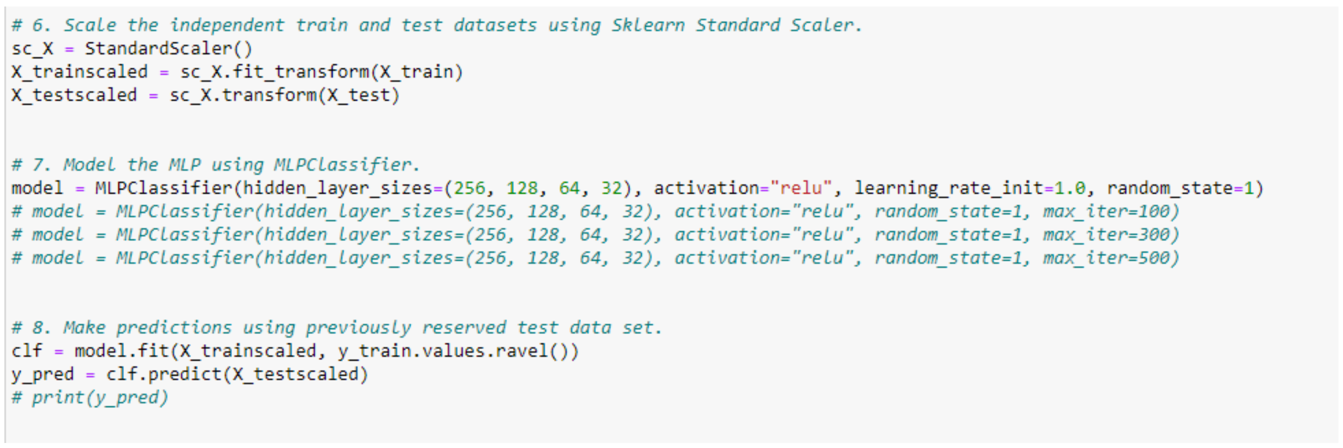
09/10/2021

TASK 3

CODE

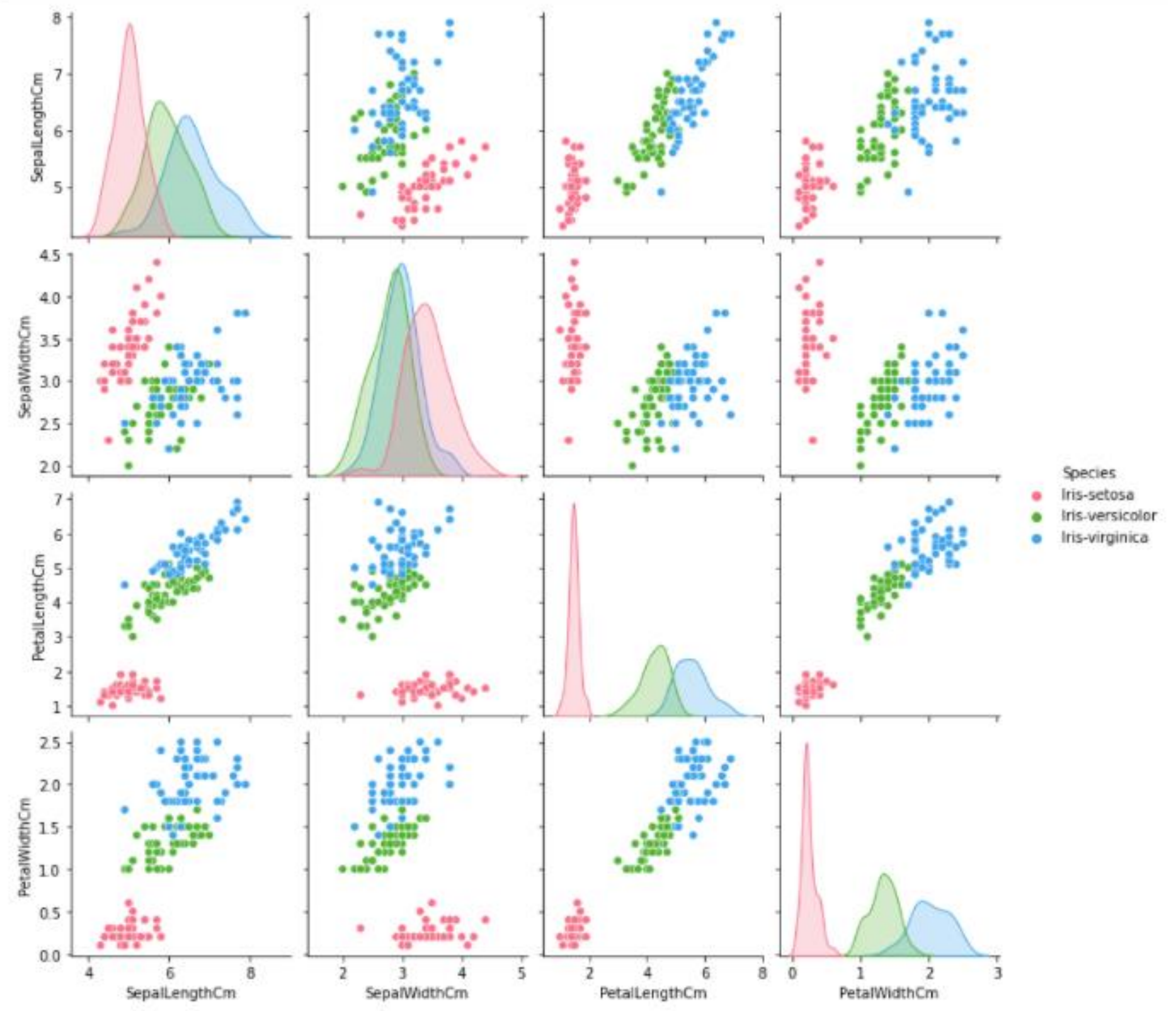


```python
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split

import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import plot_confusion_matrix, classification_report
from sklearn.preprocessing import StandardScaler


# 1. Load the dataset as a Panda's dataframe and drop any unnecessary columns.
iris_data = pd.read_csv('iris.csv')
iris_data = iris_data.drop(columns=['Id'])


# 2. Visualize the relationships between dataset features using seaborn.pairplot
# sb.set_style("ticks")
# sb.pairplot(iris_data, hue='Species', diag_kind="auto", kind="scatter", palette="husl")
# plt.show()


# 3. Separate the dataset into features and labels.
x_col = [col for col in iris_data.columns if col not in ['Species']]
y_col = ['Species']
features = iris_data[x_col]
label = iris_data[y_col]
#print(label)


# 4. Convert categorical data in your dependent variable into numerical values using Sklearn Label Encoder.
label_copy = label.copy()
lb = LabelEncoder()
label_copy['Species'] = lb.fit_transform(label['Species'])
# print(label['Species'].unique())
# print(label_copy['Species'].unique())


# 5. Split the data set as train and test data accordingly (E.g.: 80% training data, 20% test data).
X_train, X_test, y_train, y_test = train_test_split(features, label_copy, random_state=1, test_size=0.2)
```

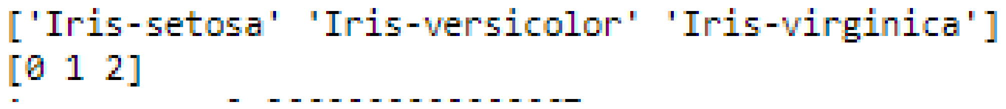


```python
# 6. Scale the independent train and test datasets using Sklearn Standard Scaler.
sc_X = StandardScaler()
X_trainscaled = sc_X.fit_transform(X_train)
X_testscaled = sc_X.transform(X_test)


# 7. Model the MLP using MLPClassifier.
model = MLPClassifier(hidden_layer_sizes=(256, 128, 64, 32), activation="relu", learning_rate_init=1.0, random_state=1)
# model = MLPClassifier(hidden_layer_sizes=(256, 128, 64, 32), activation="relu", random_state=1, max_iter=100)
# model = MLPClassifier(hidden_layer_sizes=(256, 128, 64, 32), activation="relu", random_state=1, max_iter=300)
# model = MLPClassifier(hidden_layer_sizes=(256, 128, 64, 32), activation="relu", random_state=1, max_iter=500)


# 8. Make predictions using previously reserved test data set.
clf = model.fit(X_trainscaled, y_train.values.ravel())
y_pred = clf.predict(X_testscaled)
# print(y_pred)
# 9. Observe the accuracy of the model by plotting the confusion metrix.
print('Accuracy: ', clf.score(X_testscaled, y_test))
fig = plot_confusion_matrix(clf, X_testscaled, y_test, display_labels=["Iris-setosa", "Iris-versicolor", "Iris-virginica"])
fig.figure_.suptitle("Confusion Matrix for Iris Dataset")
plt.show()


# 10. Generate the classification report using Sklearn Classification Report and comment on each of the value you obtained in
report = classification_report(y_test, y_pred)
print('Classification Report:')
print(report)


# 11. Repeat the task with 100,300,500 iterations and comment on your results.
# 12. Repeat the task with varying test and train data set sizes and comment on your observations.
# 13. Repeat the task with different learning rates: 0.002, 0.5, 1.00 and comment on the results obtained.
```

2.



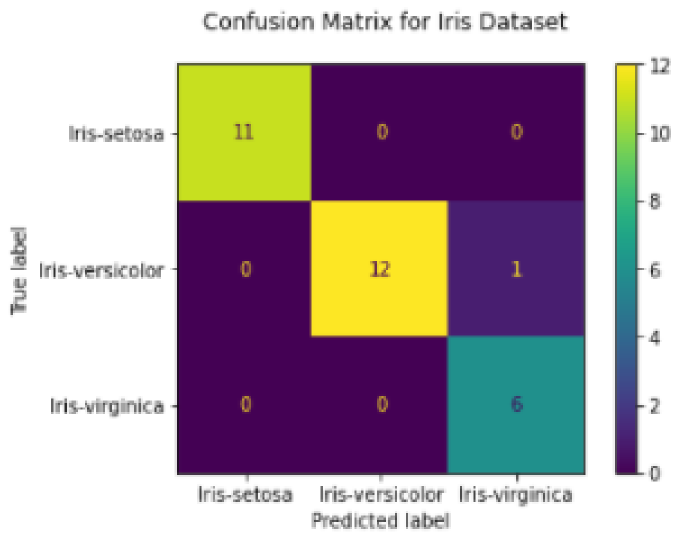
3.



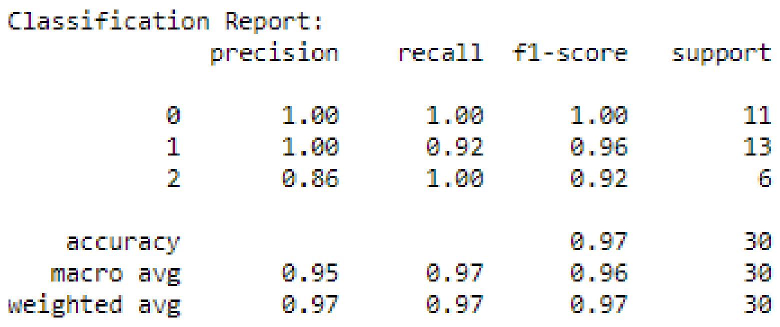
```
['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
[0 1 2]
```

9.

```
Accuracy:  0.9666666666666667
```



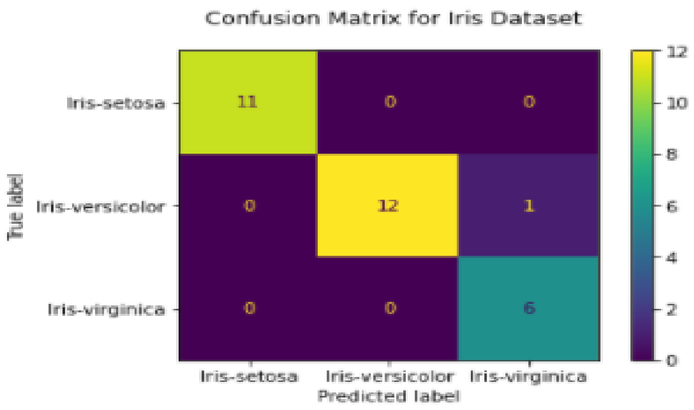
10.



```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      0.92      0.96        13
           2       0.86      1.00      0.92         6

    accuracy                           0.97        30
   macro avg       0.95      0.97      0.96        30
weighted avg       0.97      0.97      0.97        30
```

11. 100 iterations

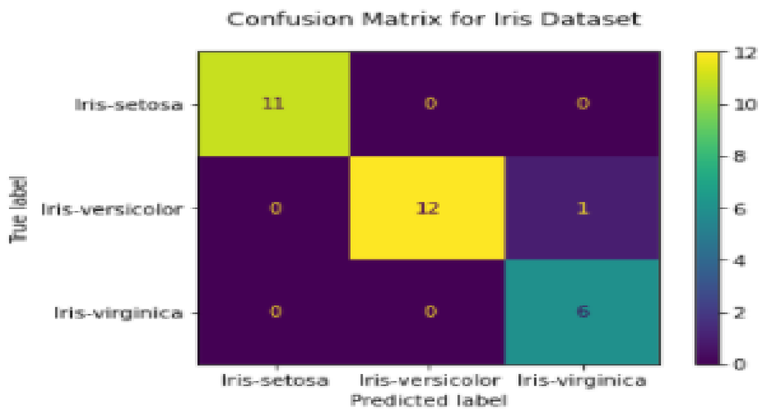
```
Accuracy:  0.9666666666666667
```



```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      0.92      0.96        13
           2       0.86      1.00      0.92         6

    accuracy                           0.97        30
   macro avg       0.95      0.97      0.96        30
weighted avg       0.97      0.97      0.97        30
```

300 iterations

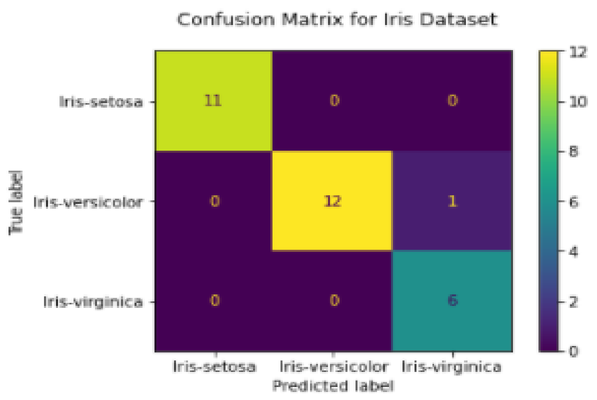
```
Accuracy:  0.9666666666666667
```



```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      0.92      0.96        13
           2       0.86      1.00      0.92         6

    accuracy                           0.97        30
   macro avg       0.95      0.97      0.96        30
weighted avg       0.97      0.97      0.97        30
```

500 iterations

```
Accuracy:  0.9666666666666667
```

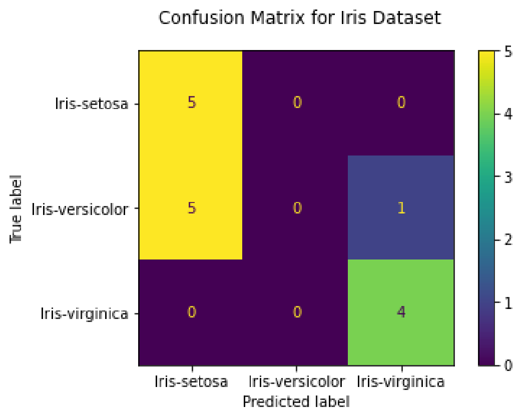


```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      0.92      0.96        13
           2       0.86      1.00      0.92         6

    accuracy                           0.97        30
   macro avg       0.95      0.97      0.96        30
weighted avg       0.97      0.97      0.97        30
```

When the number of iterations increases, the accuracy and other matrices doesn’t change much. It is similar.

12. Training dataset - 0.9% Testing dataset – 0.1%

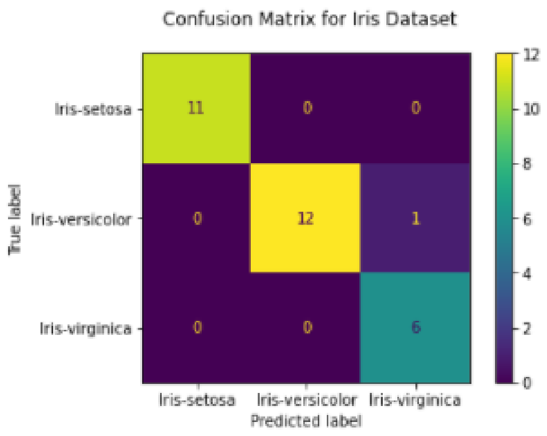
```
Accuracy:  0.6
```



```
Classification Report:
              precision    recall  f1-score   support

           0       0.50      1.00      0.67         5
           1       0.00      0.00      0.00         6
           2       0.80      1.00      0.89         4

    accuracy                           0.60        15
   macro avg       0.43      0.67      0.52        15
weighted avg       0.38      0.60      0.46        15
```

Training dataset - 0.8% Testing dataset – 0.2%

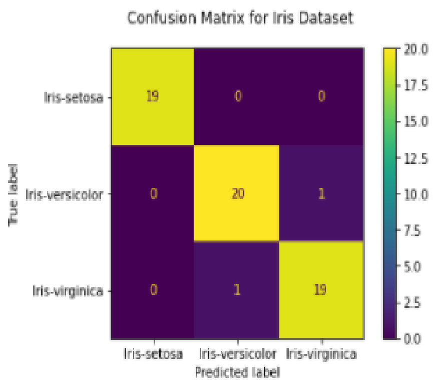
```
Accuracy:  0.9666666666666667
```



```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      0.92      0.96        13
           2       0.86      1.00      0.92         6

    accuracy                           0.97        30
   macro avg       0.95      0.97      0.96        30
weighted avg       0.97      0.97      0.97        30
```

Training dataset - 0.6% Testing dataset – 0.4%

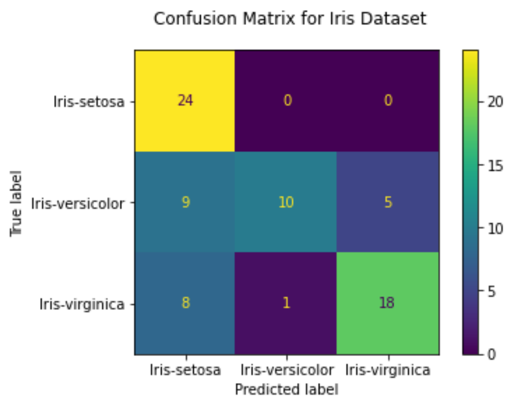
```
Accuracy:  0.9666666666666667
```



```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       0.95      0.95      0.95        21
           2       0.95      0.95      0.95        20

    accuracy                           0.97        60
   macro avg       0.97      0.97      0.97        60
weighted avg       0.97      0.97      0.97        60
```

Training dataset - 0.5%   Testing dataset – 0.5%

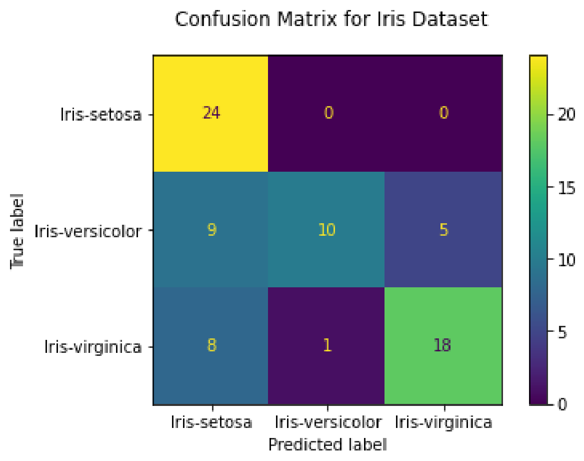
```
Accuracy:  0.6933333333333334
```



```
Classification Report:
              precision    recall  f1-score   support

           0       0.59      1.00      0.74        24
           1       0.91      0.42      0.57        24
           2       0.78      0.67      0.72        27

    accuracy                           0.69        75
   macro avg       0.76      0.69      0.68        75
weighted avg       0.76      0.69      0.68        75
```

Training dataset - 0.3%   Testing dataset – 0.7%

```
Accuracy:  0.6933333333333334
```



```
Classification Report:
              precision    recall  f1-score   support

           0       0.59      1.00      0.74        24
           1       0.91      0.42      0.57        24
           2       0.78      0.67      0.72        27

    accuracy                           0.69        75
   macro avg       0.76      0.69      0.68        75
weighted avg       0.76      0.69      0.68        75
```

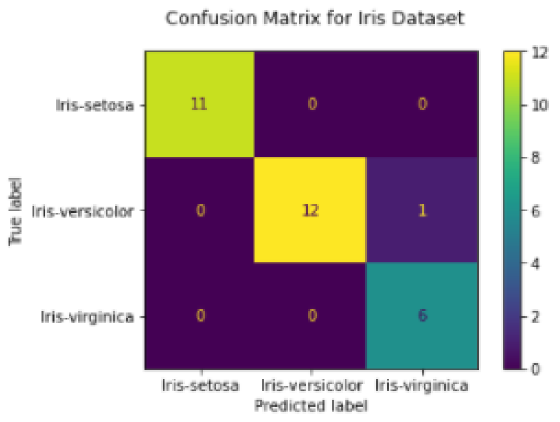
The f1 score is good when the train:test = 80:20 or 60:40. However, when the training dataset is too low or the testing dataset is too low, the F1 score is not much good. This is not recommended because at times the testing set maybe smaller than the training set and vice versa. We can use acceptable ratio as 60:40 or 80:20

Direct comment on how the ratio affect the performance cannot be made, but there should be enough data to train the network and test it.

13.

Learning rate – 0.002

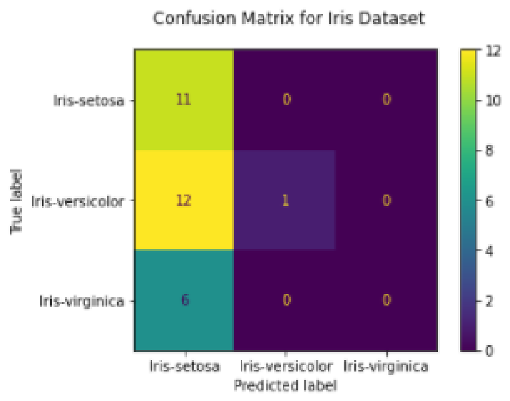
```
Accuracy:  0.9666666666666667
```



```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      0.92      0.96        13
           2       0.86      1.00      0.92         6

    accuracy                           0.97        30
   macro avg       0.95      0.97      0.96        30
weighted avg       0.97      0.97      0.97        30
```

Learning rate – 0.5

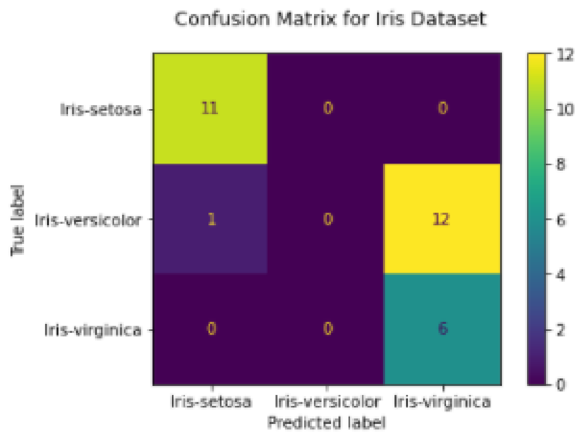
```
Accuracy:  0.4
```



```
Classification Report:
              precision    recall  f1-score   support

           0       0.38      1.00      0.55        11
           1       1.00      0.08      0.14        13
           2       0.00      0.00      0.00         6

    accuracy                           0.40        30
   macro avg       0.46      0.36      0.23        30
weighted avg       0.57      0.40      0.26        30
```

Learning rate – 1.00

```
Accuracy:  0.5666666666666667
```



```
Classification Report:
              precision    recall  f1-score   support

           0       0.92      1.00      0.96        11
           1       0.00      0.00      0.00        13
           2       0.33      1.00      0.50         6

    accuracy                           0.57        30
   macro avg       0.42      0.67      0.49        30
weighted avg       0.40      0.57      0.45        30
```

It can clearly be seen that the performance of the model decreases when the learning rate increases. However, low learning rates can increase the training time of the model