In [1]:
```
import os
import pandas as pd
import seaborn as sns
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
```

In [2]:
```
from sklearn.datasets import load_iris
iris=load_iris()
targets= iris.target
```

In [3]:
```
from sklearn.model_selection import train_test_split
train_data, test_data, train_label,test_label=train_test_split(iris.data,target
```

In [4]:
```
#scaling the data
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(train_data)
```

Out[4]:
```
StandardScaler

StandardScaler()
```

In [5]:
```
train_data=scaler.transform(train_data)
test_data=scaler.transform(test_data)
print(train_data[:3])
```

```
[[ 0.61303014  0.10850105  0.94751783  0.736072  ]
 [-0.56776627 -0.12400121  0.38491447  0.34752959]
 [-0.80392556  1.03851009 -1.30289562 -1.33615415]]
```

In [6]:
```
#training the data
from sklearn.neural_network import MLPClassifier
mlp=MLPClassifier(hidden_layer_sizes=(10,5), activation="relu", solver='adam',
mlp.fit(train_data, train_label)
```

Out[6]:
```
MLPClassifier

MLPClassifier(hidden_layer_sizes=(10, 5), max_iter=1000)
```

In [7]:
```
#testing the data
#Prediction
from sklearn.metrics import accuracy_score
prediction_train=mlp.predict(train_data)
print(accuracy_score(prediction_train, train_label))
prediction_test=mlp.predict(test_data)
print(accuracy_score(prediction_test, test_label))
```

```
0.975
1.0
```

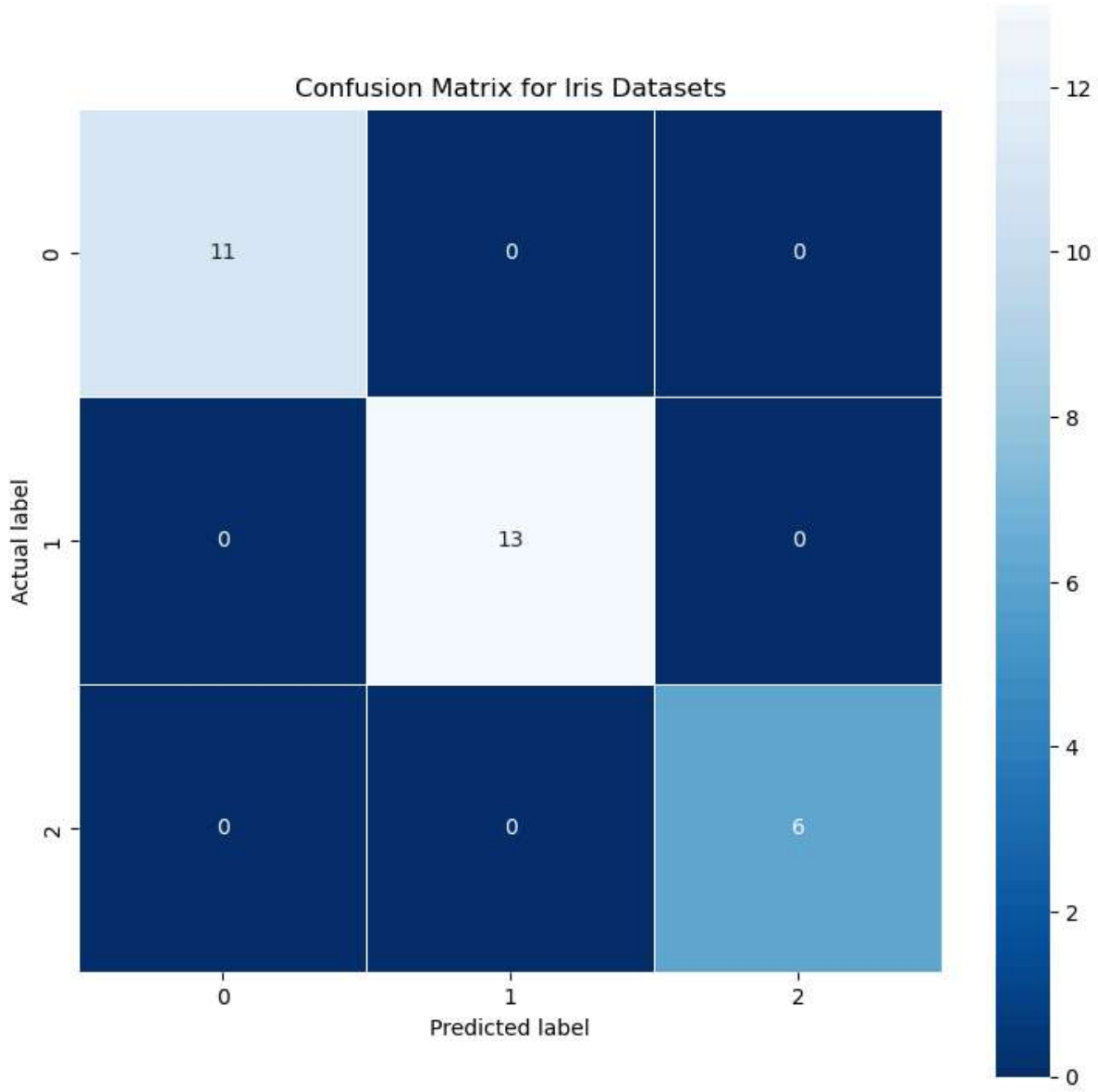
In [8]:
```
from sklearn.metrics import classification_report
print(classification_report(prediction_test,test_label, labels=mlp.classes_.tol
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00         6

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

In [9]:
```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
prediction=mlp.predict(test_data)
cm=confusion_matrix(prediction, test_label)
print (cm)
```

```
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
```

In [10]:
```
plt.figure(figsize=(9,9))
plt.title('Confusion Matrix for Iris Datasets')
sns.heatmap(cm,annot=True,linewidth=0.5,square=True,cmap='Blues_r',fmt='g')
plt.xlabel('Predicted label')
plt.ylabel('Actual label')
plt.show()
```



In [ ]:
```

```