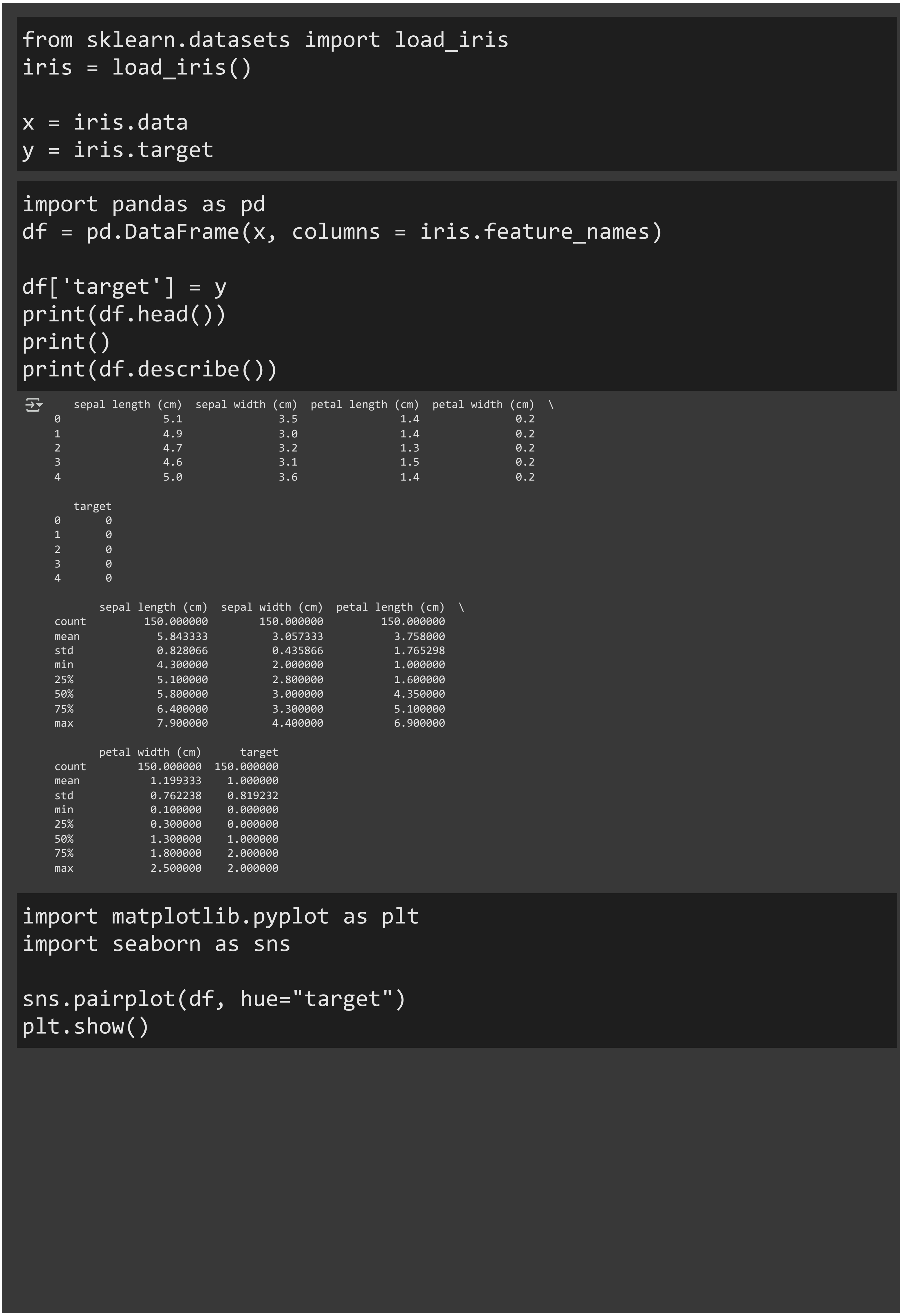
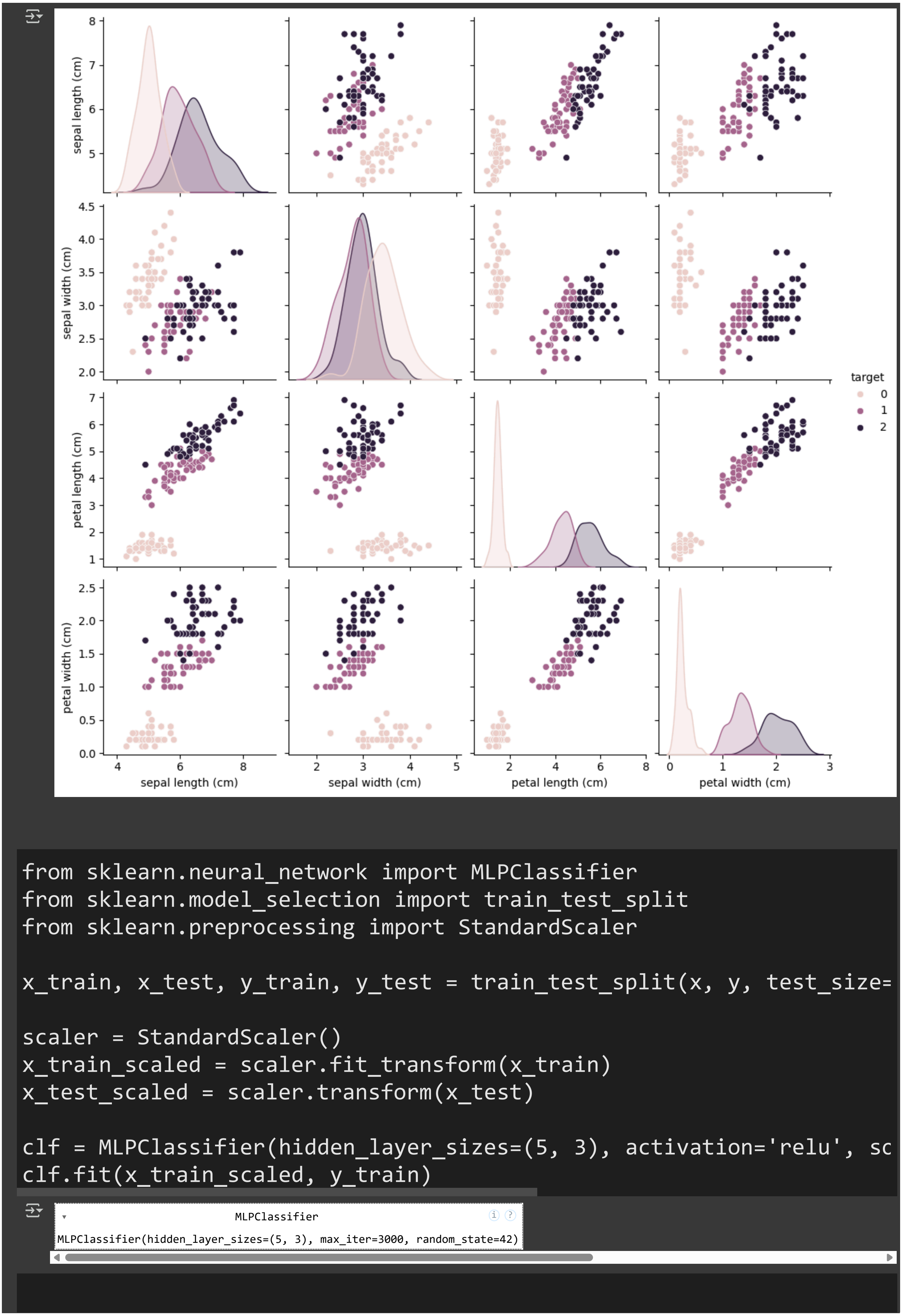


```
from sklearn.datasets import load_iris
iris = load_iris()

x = iris.data
y = iris.target
```

```
import pandas as pd
df = pd.DataFrame(x, columns = iris.feature_names)

df['target'] = y
print(df.head())
print()
print(df.describe())
```

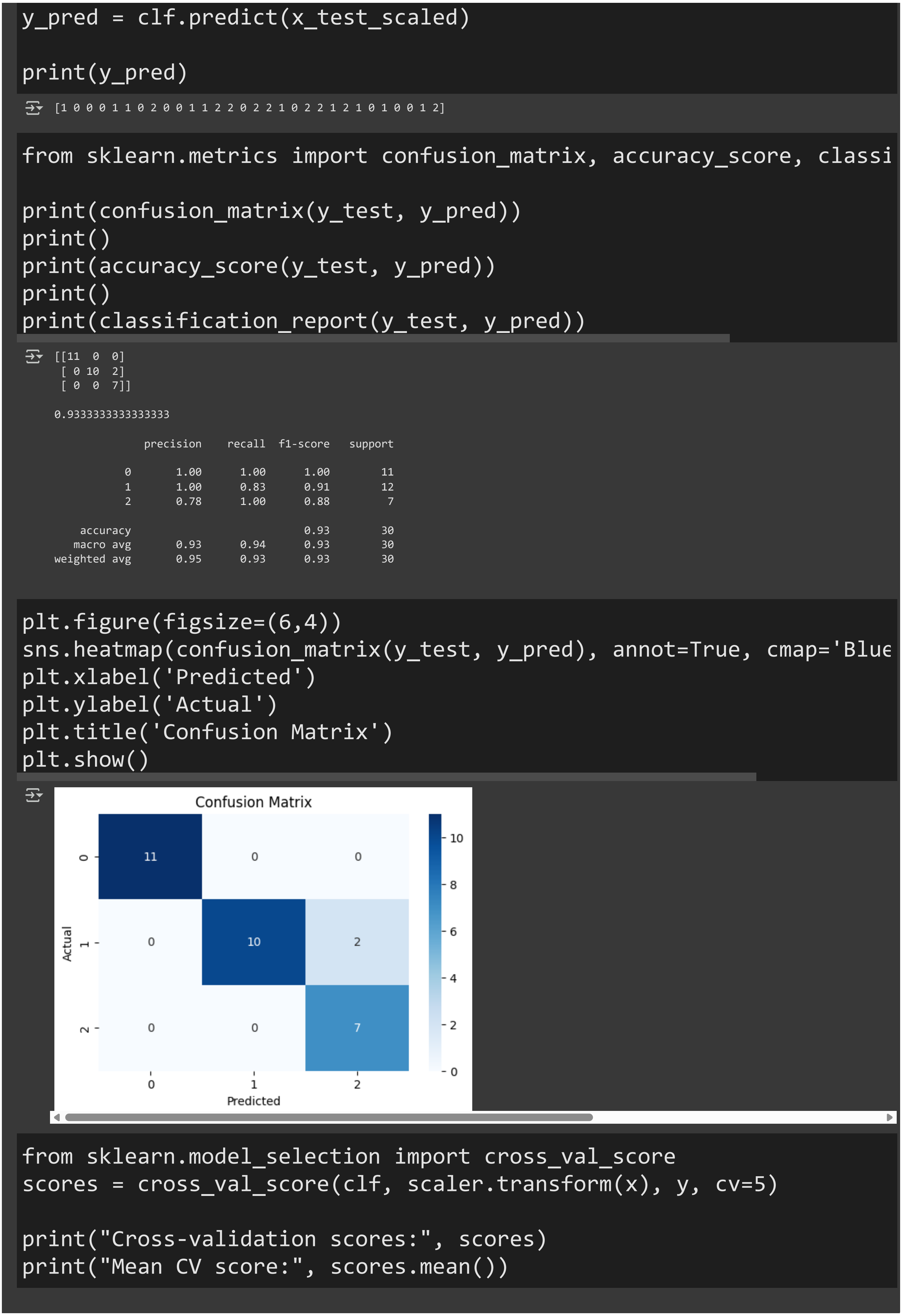
```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2   
1                4.9               3.0                1.4               0.2   
2                4.7               3.2                1.3               0.2   
3                4.6               3.1                1.5               0.2   
4                5.0               3.6                1.4               0.2   

   target  
0       0  
1       0  
2       0  
3       0  
4       0  

       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000   
mean            5.843333          3.057333           3.758000   
std             0.828066          0.435866           1.765298   
min             4.300000          2.000000           1.000000   
25%             5.100000          2.800000           1.600000   
50%             5.800000          3.000000           4.350000   
75%             6.400000          3.300000           5.100000   
max             7.900000          4.400000           6.900000   

       petal width (cm)      target  
count        150.000000  150.000000  
mean           1.199333    1.000000  
std            0.762238    0.819232  
min            0.100000    0.000000  
25%            0.300000    0.000000  
50%            1.300000    1.000000  
75%            1.800000    2.000000  
max            2.500000    2.000000  
```

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.pairplot(df, hue="target")
plt.show()
```



```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=

scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)

clf = MLPClassifier(hidden_layer_sizes=(5, 3), activation='relu', sc
clf.fit(x_train_scaled, y_train)
```

```
MLPClassifier(hidden_layer_sizes=(5, 3), max_iter=3000, random_state=42)
```



```
y_pred = clf.predict(x_test_scaled)

print(y_pred)
```

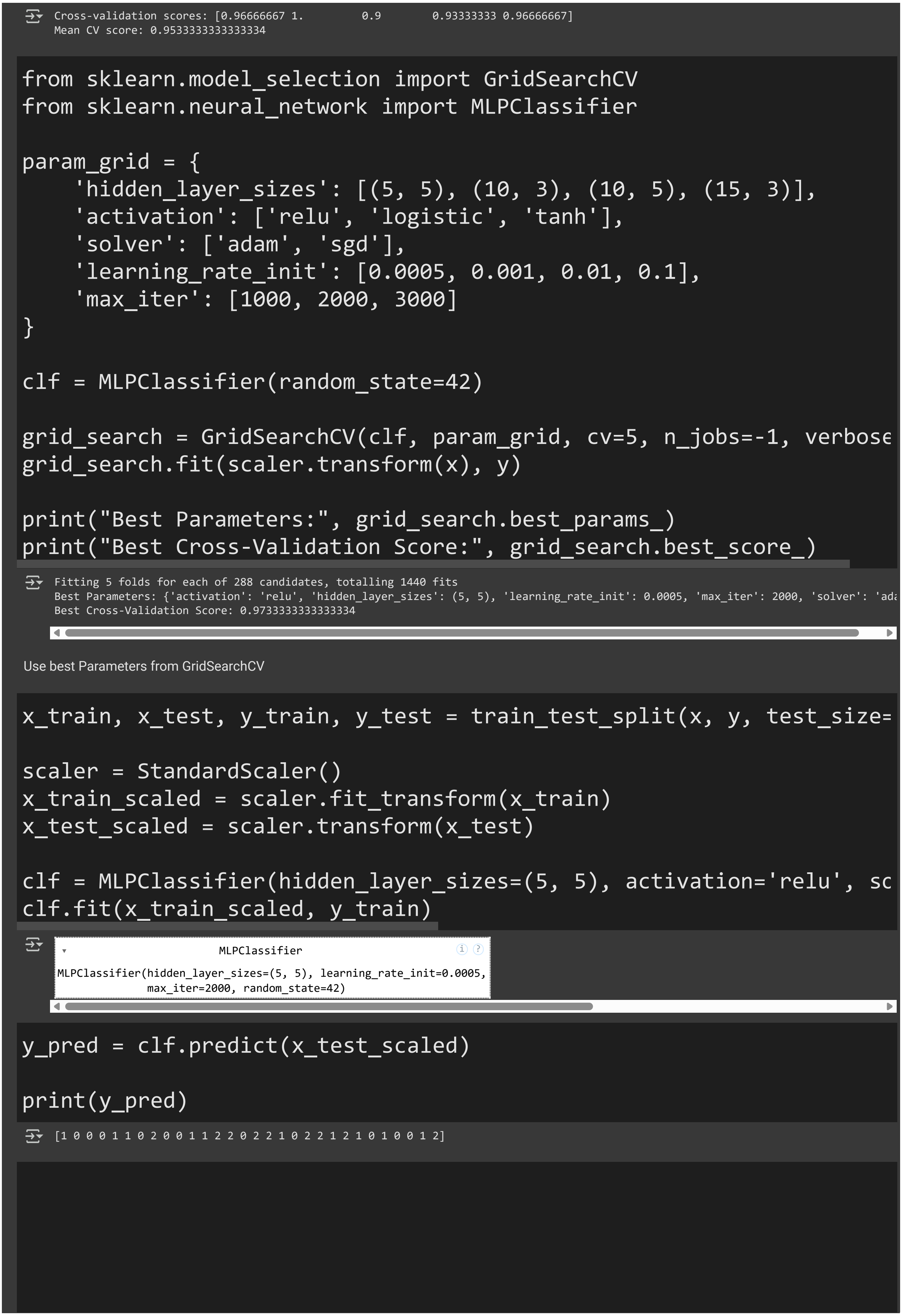
```
[1 0 0 0 1 1 0 2 0 0 1 1 2 2 0 2 2 1 0 2 2 1 2 1 0 1 0 0 1 2]
```

```
from sklearn.metrics import confusion_matrix, accuracy_score, classi

print(confusion_matrix(y_test, y_pred))
print()
print(accuracy_score(y_test, y_pred))
print()
print(classification_report(y_test, y_pred))
```

```
[[11  0  0]
 [ 0 10  2]
 [ 0  0  7]]

0.9333333333333333

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      0.83      0.91        12
           2       0.78      1.00      0.88         7

    accuracy                           0.93        30
   macro avg       0.93      0.94      0.93        30
weighted avg       0.95      0.93      0.93        30
```

```
plt.figure(figsize=(6,4))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cmap='Blue
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(clf, scaler.transform(x), y, cv=5)

print("Cross-validation scores:", scores)
print("Mean CV score:", scores.mean())
```



```
Cross-validation scores: [0.96666667 1.         0.9        0.93333333 0.96666667]
Mean CV score: 0.9533333333333334
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.neural_network import MLPClassifier

param_grid = {
    'hidden_layer_sizes': [(5, 5), (10, 3), (10, 5), (15, 3)],
    'activation': ['relu', 'logistic', 'tanh'],
    'solver': ['adam', 'sgd'],
    'learning_rate_init': [0.0005, 0.001, 0.01, 0.1],
    'max_iter': [1000, 2000, 3000]
}

clf = MLPClassifier(random_state=42)

grid_search = GridSearchCV(clf, param_grid, cv=5, n_jobs=-1, verbose
grid_search.fit(scaler.transform(x), y)

print("Best Parameters:", grid_search.best_params_)
print("Best Cross-Validation Score:", grid_search.best_score_)
```

```
Fitting 5 folds for each of 288 candidates, totalling 1440 fits
Best Parameters: {'activation': 'relu', 'hidden_layer_sizes': (5, 5), 'learning_rate_init': 0.0005, 'max_iter': 2000, 'solver': 'ada
Best Cross-Validation Score: 0.9733333333333334
```

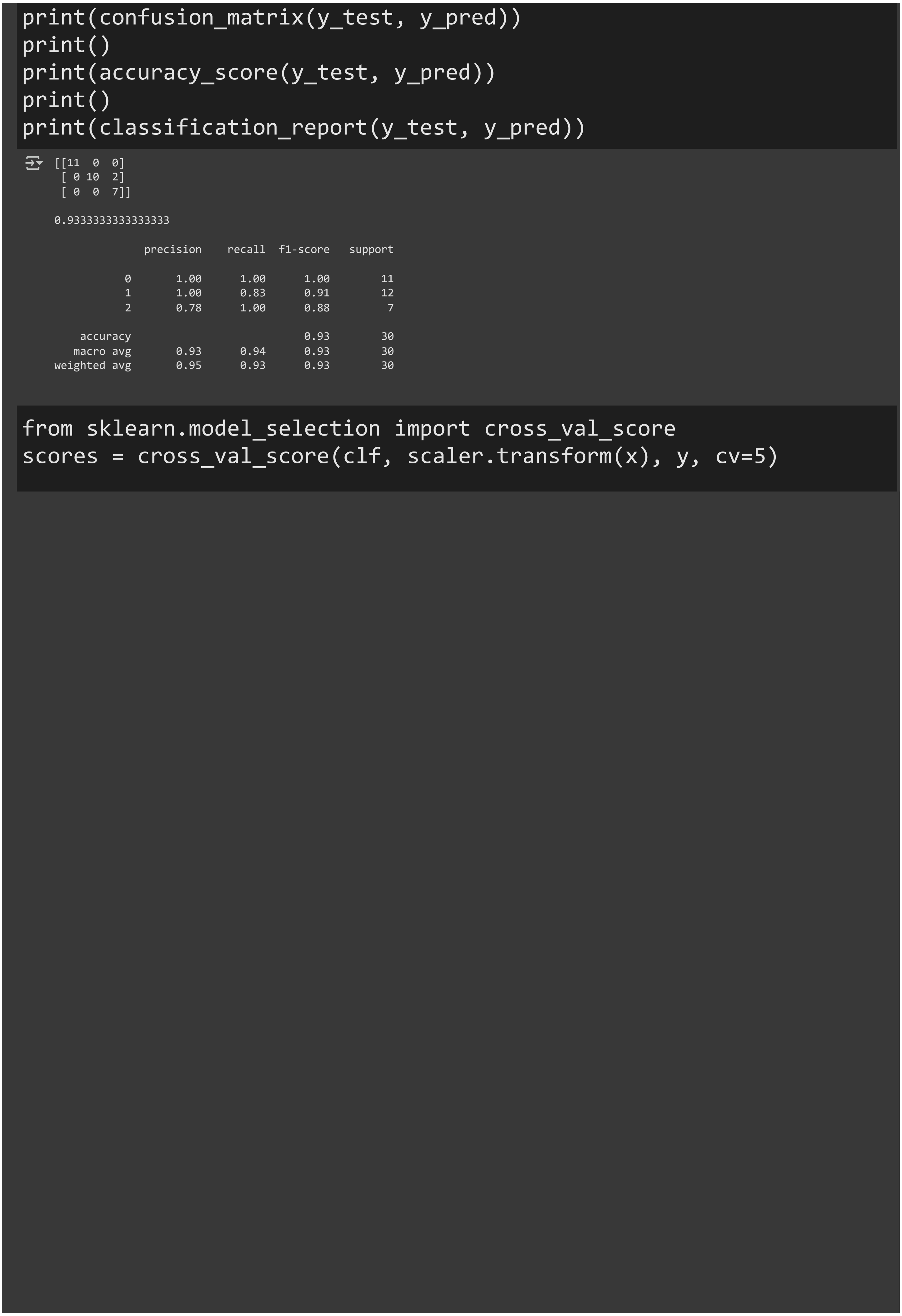
Use best Parameters from GridSearchCV

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=

scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)

clf = MLPClassifier(hidden_layer_sizes=(5, 5), activation='relu', sc
clf.fit(x_train_scaled, y_train)
```

```
MLPClassifier
MLPClassifier(hidden_layer_sizes=(5, 5), learning_rate_init=0.0005,
              max_iter=2000, random_state=42)
```

```
y_pred = clf.predict(x_test_scaled)

print(y_pred)
```

```
[1 0 0 0 1 1 0 2 0 0 1 1 2 2 0 2 2 1 0 2 2 1 2 1 0 1 0 0 1 2]
```



```
print(confusion_matrix(y_test, y_pred))
print()
print(accuracy_score(y_test, y_pred))
print()
print(classification_report(y_test, y_pred))
```

```
[[11  0  0]
 [ 0 10  2]
 [ 0  0  7]]

0.9333333333333333

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      0.83      0.91        12
           2       0.78      1.00      0.88         7

    accuracy                           0.93        30
   macro avg       0.93      0.94      0.93        30
weighted avg       0.95      0.93      0.93        30
```

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(clf, scaler.transform(x), y, cv=5)
```