KH2311047010044

INDEX

NAME K.D.D.SaiAbhiram SUBJECT Deep learning lab obdb

STD. ____ DIV. ____ ROLL NO. ____ SCHOOL ____

07-08-25

2. Implement a classifier using open-Source dataset

Aim :- to implement a classifier using an open source dataset, to classify the data set using support vector machine.

Algorithm :

① Import necessary libraries like sklearn, matplotlib etc...

② Load the data set

③ Preprocess the data

* Select first two features for 2D-visualization
* split the dataset into training and testing sets.
* Standardize the features using standard scalar.

④ train the svm model

* Initialize svm with a linear model
* fit the model using training data.

⑤ visualize the decision boundaries

* Create a meshgrid of input size
* Predict the class for each point in the grid
* Plot the regions along with actual training points.

Code :-

```
# Step:1 : Import libraries
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import
from sk learn.datasets import load-iris
from sk learn.linear-model import logistic regression
from sk learn.model-selection import train-test-split
from sk learn.metrics import classification-report, confusion-matrix,
    accuracy-score

# step2 :- load the iris dataset
  iris = load-iris()

  X = pd.Data frame(iris.data, columns = iris.features_names)
  y = pd.series( iris.target, name = 'species')

# step-3 :- split the dataset
x-train, x-test, y-train, y-test = train-test-split (X, y, test-size=0.3
                                               random-state= 42)

# Step:4 - train the logistic Regression model
    model = logistic regression (max-iter= 200)
    model.fit (x-train, y-train)

# Step:5 - Make Predictions
   y-pred = model.predict (x-test)
```

```
# Step: 6 - Evaluate the model

Print ("Accuracy:", accuracy_score (y-test, y-pred))

Print ("\n classification Report : \n", classification_report
                                          (y-test, y-pred))

Print ("\n confusion matrix : \n", confusion_matrix (y-test, y-pred))

# Step: 7 - visualize the confusion matrix

    Plt.figure (figsize = (6,4))

    x = iris.target_names
    y = iris.target_name

  Plt.title ("confusion matrix")
  Plt.xlabel (" Predicted")
  Plt.ylabel ("Actual")
  Plt.show()
```

Result :- the svm classifier was successfully implemented on the iris dataset using two features.

14/8/25

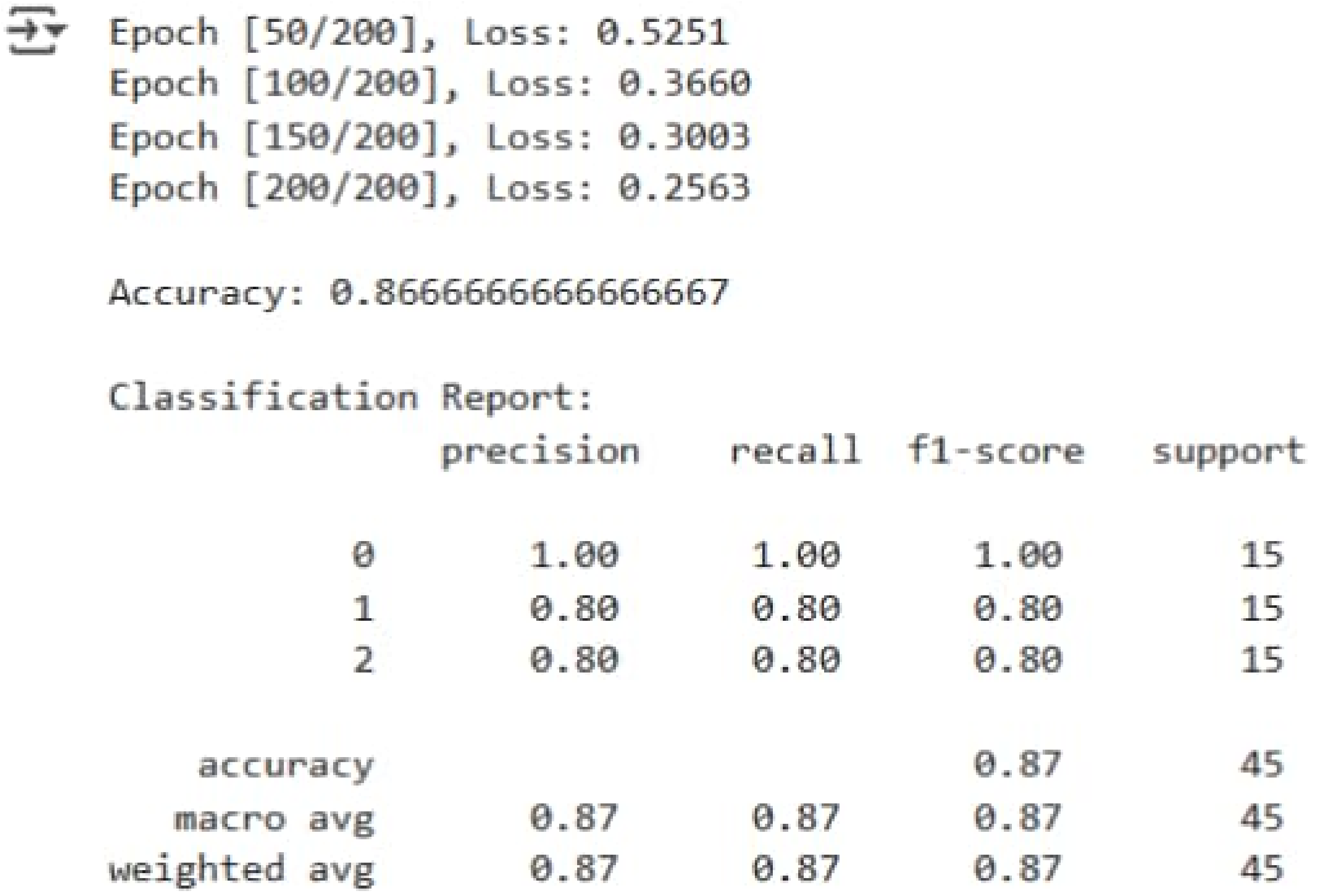
output :-

```
Accuracy : 1.0

Classification report :
               Precision   recall   f1-score   support
   0             1.00       1.00      1.00       19
   1             1.00       1.00      1.00       13
   2             1.00       1.00      1.00       13
accuracy          _          _        1.00       45
macro avg        1.00       1.00      1.00       45
weighted
    avg          1.00       1.00      1.00       45
```

Confusion matrix

```
[ [19  0  0 ]
  [0  13  0 ]
  [ 0  0 13] ]
```



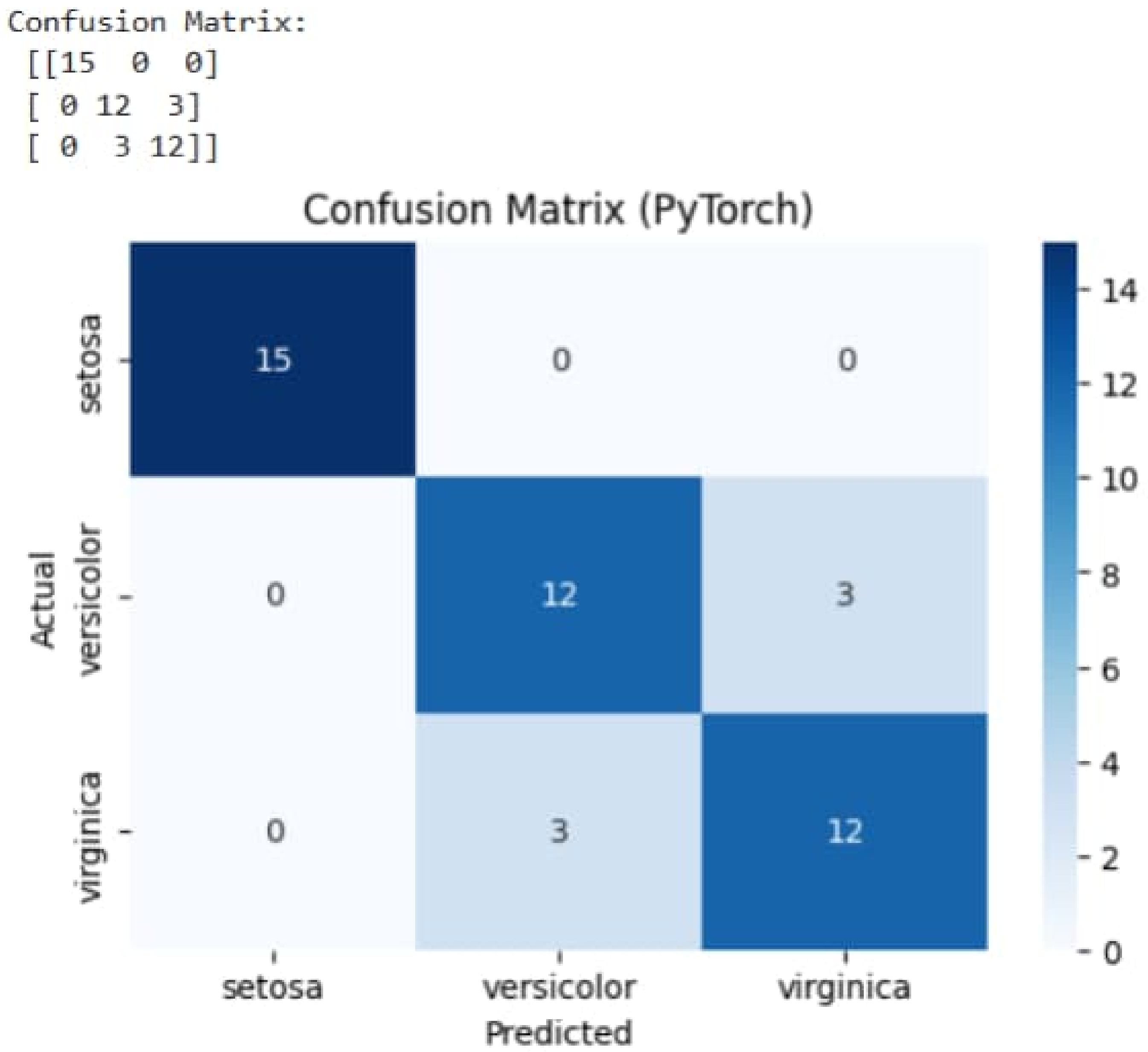
```
Epoch [50/200], Loss: 0.5251
Epoch [100/200], Loss: 0.3660
Epoch [150/200], Loss: 0.3003
Epoch [200/200], Loss: 0.2563

Accuracy: 0.8666666666666667

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        15
           1       0.80      0.80      0.80        15
           2       0.80      0.80      0.80        15

    accuracy                           0.87        45
   macro avg       0.87      0.87      0.87        45
weighted avg       0.87      0.87      0.87        45
```



```
Confusion Matrix:
 [[15  0  0]
 [ 0 12  3]
 [ 0  3 12]]
```

Confusion Matrix (PyTorch)