RA2311047010044

38

INDEX

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

STD. _____ DIV. _____ ROLL NO. _____ SCHOOL _____

SR. NO.	PAGE NO.	TITLE	DATE	TEACHER'S SIGN / REMARKS
1		Analyze and exploring the deep learning Platforms	24-07-25	
2		Implement a classifier using open-source data set	7/8/25	
3.		Study of the classifiers with respect to statistical parameters	7/8/25	
4.		Build a simple feed forward neural network to recognize hand written characters	14/8/25	
5.		Study of activation function and its role	28-8-25	
6.		Implement gradient descent & back propagation in deep neural network.	09-9-25	
7.		Build a CNN model to classify Cats & dogs images	16-9-25	

07-08-25

3. Study of the classifiers with respect to statistical parameters

Aim: to study and compare the performance of different classifiers on an open source data set

description

① Accuracy : fraction of predicted observations

$$\text{accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

② Precision : Ratio of correctly predicted positive to the total predicted positives

$$\text{Precision} = \frac{TP}{TP+FP}$$

③ Recall $= \frac{TP}{TP+FN}$

④ F1-score $= \frac{2 \times \text{precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

⑤ Confusion matrix :- A 2D table showing count of TP, FP, F

Algorithm

① Import the libraries
② Load the open source dataset
③ Split the dataset into training (80%) and testing 20%
④ Select and apply model
⑤ train the model with training data
⑥ predict the output on the test data

(7) Compute Evalution metrics and confusion matrix

(8) Repeat 4-5 for all Classifiers

(9) Compare the results.

Code :-

Step: 1 - Import libraries

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
import matplotlib.pyplot as plt
```

Step: 2

```
iris = load_iris()
x = iris.data
y = iris.target

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
                                                    random_state=42)

for name, cls in classifiers.items():
    cls.fit(x_train, y_train)
    y_pred = cls.predict(x_test)

print(f"\n== {name} ===")
print("Accuracy", round(accuracy_score(y_test, y_pred), 4))
print("Precision", round(precision_score(y_test, y_pred,
                                         average='macro'), 4))
print("f1-score:", round(f1_score(y_test, y_pred, average='macro'),
```

Output :-

Accuracy : 1.00

Precision : 1.00

Recall : 1.00

F1-Score : 1.00

Confusion matrix :-

	Setosa	Versicolour	Virginica
Setosa	19	0	0
Versicolour	0	13	0
Virginica	0	0	13

```
cm = confusion_matrix (y_test, y_pred)
disp = confusion_matrix display (confusion_matrix = cm,
    display_labels = iris.target_names)
des.plot = (cmap = plt.cm.Blues)
plt.title (f"confusion matrix - {name}")
plt.show
```

Result :- Implemented different types of ~~data~~ classifiers on the iris dataset successfully. And the accuracy and precision is 1.

[ ]

```python
# Step 1: Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Step 2: Load the Iris Dataset
iris = load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_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 = LogisticRegression(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("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

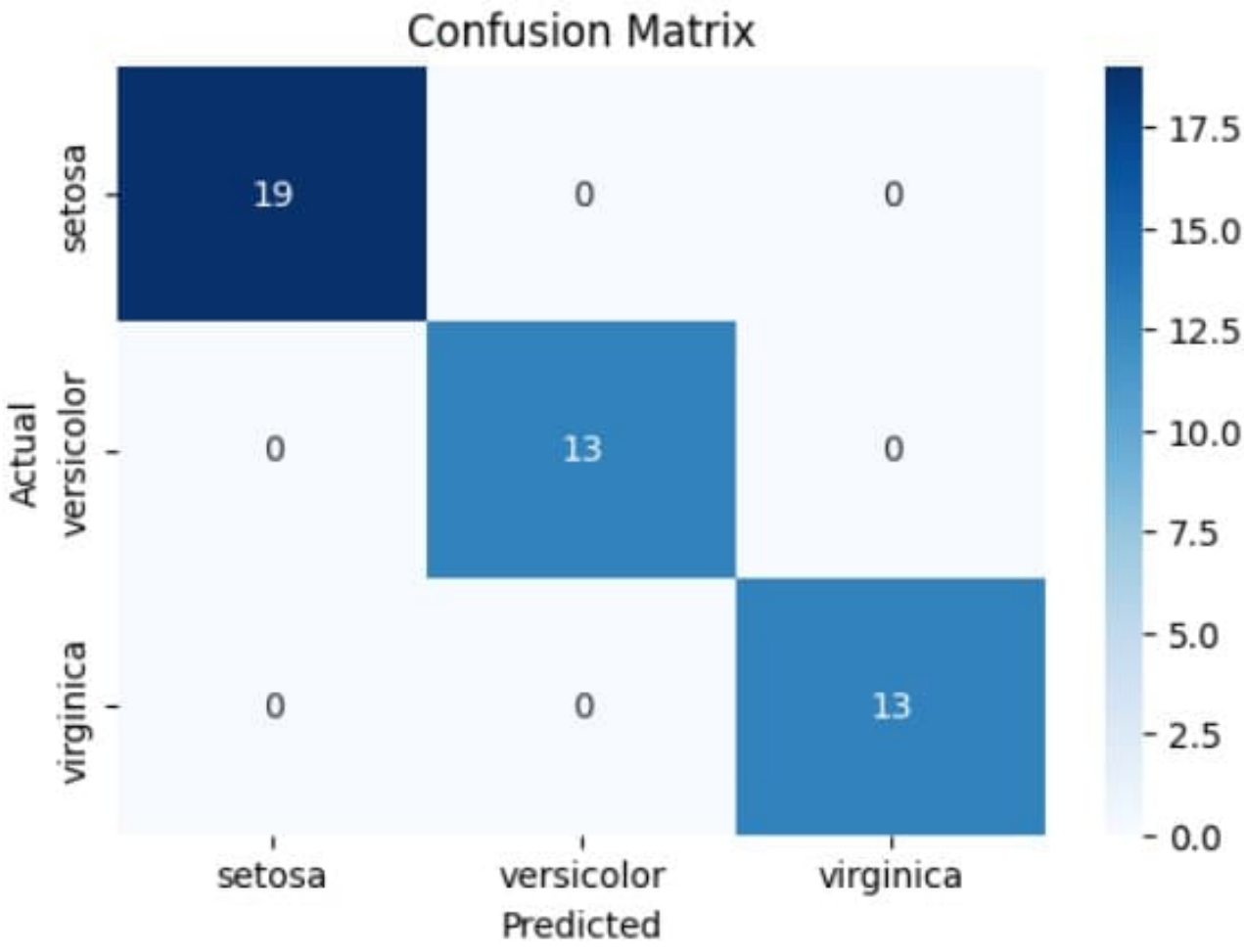
```
# Step 7: Visualize the Confusion Matrix
plt.figure(figsize=(6,4))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues',
xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

```
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
setosa
versicolor
virginica
Actual
19
0
0
0
13
0
0
0
13
setosa
versicolor
virginica
Predicted
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0