Ex. No. : **2**	**IMPLEMENTATION OF SIMPLE LAYER PERCEPTRON**

Aim :

To build, train, and evaluate a single-layer perceptron model to predict the species of iris flowers based on their features and measure the model's performance.

Algorithm :

// Import Libraries //

1. ***numpy*** and ***matplotlib*** for numerical operations and plotting.
2. ***sklearn*** libraries for loading and splitting data, standardizing features, creating the perceptron model, and measuring performance.

// Load the Dataset //

3. Load the Iris dataset using ***load_iris()*** from sklearn.datasets.

// Separate Features and Labels //

4. Extract features (***X***) and labels (***y***) from the dataset.

// Standardize the Features //

5. Use ***StandardScaler()*** to normalize the feature values.

// Split the Data //

6. Split the standardized data by ***train_test_split()*** as 70% training and 30% for testing.

// Initialize the Perceptron Model //

7. Create a single-layer perceptron model using ***Perceptron()*** with parameters.

// Train the Model //

8. Train the model by calling ***fit()*** on the model with ***X_train*** and ***y_train***.

// Make Predictions //

9. Predict the labels for the test data using ***predict()*** on the model with ***X_test***.

// Evaluate the Model //

10. Calculate the accuracy of the model using ***accuracy_score()*** by comparing ***y_test*** and ***y_pred***.
11. Generate a confusion matrix with ***confusion_matrix()*** and a detailed classification report using ***classification_report()*** for metrics like precision, recall, and F1-score.

// Display Results //

12. Print the accuracy, confusion matrix, and classification report for a detailed evaluation of the model.
13. Use ***ConfusionMatrixDisplay*** to plot the confusion matrix, labeling each class appropriately.

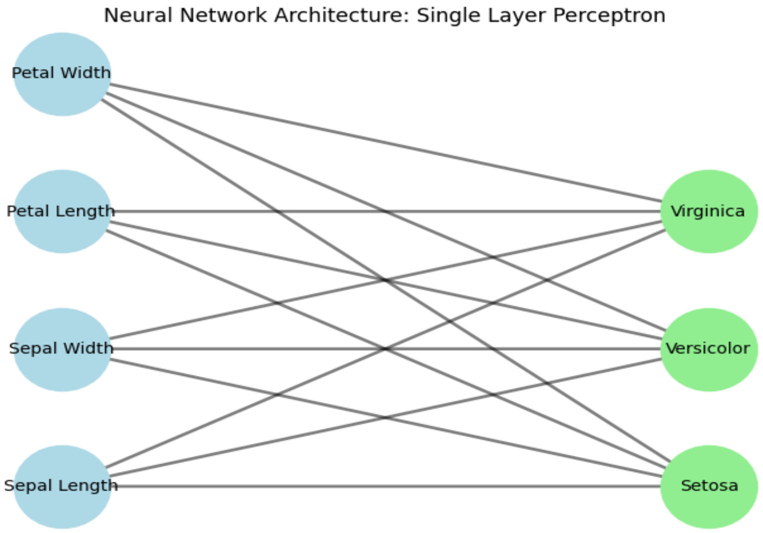
Program :

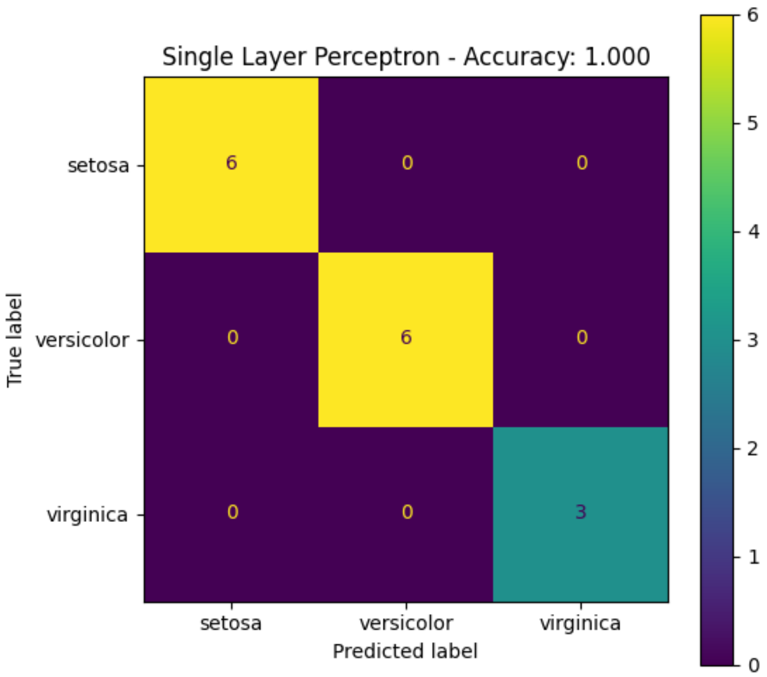
```
        # IMPLEMENTATION OF SIMPLE LAYER PERCEPTRON #

# Importing Libraries
import numpy as np
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Loading Data Set
iris = load_iris()
X = iris.data[:, [2, 3]]
y = iris.target

# Standardize the data
scaler = StandardScaler()
X1 = scaler.fit_transform(X)

# Data Split
x_train, x_test, y_train, y_test = train_test_split(X1, y,
test_size = 0.1, random_state = 42, shuffle = True)

x_train.shape, x_test.shape, y_train.shape, y_test.shape

# Create the Perceptron Model
myperceptron = Perceptron(max_iter = 2000, tol = 1e-3,
random_state = 42)

# Train the Model to Fit
myperceptron.fit(x_train, y_train)

# Prediction
y_pred = myperceptron.predict(x_test)

# Calculate performance metrics
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred,
target_names = iris.target_names)
```

```
# Print the Performance Metrics
print(f'Accuracy: {accuracy:.3f}')
print("Confusion Matrix:\n", conf_matrix)
print("\nClassification Report:\n", class_report)

# Plot the Confusion Matrix
fig, ax = plt.subplots(figsize=(6, 6))
ConfusionMatrixDisplay.from_estimator(myperceptron, x_test,
y_test, display_labels=iris.target_names, ax=ax)
plt.title(f'Single Layer Perceptron - Accuracy:
{accuracy:.3f}')
plt.show()
```

Outputs :

```
Accuracy: 0.98
Confusion Matrix:
     [[6 0 0]

      [0 6 0]

      [0 0 3]]

Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00         6
  versicolor       1.00      1.00      1.00         6
   virginica       1.00      1.00      1.00         3

    accuracy                           1.00        15
   macro avg       1.00      1.00      1.00        15
weighted avg       1.00      1.00      1.00        15
```





Confusion Matrix

Result

Building, training, and evaluating a single-layer perceptron for predicting iris species has been successfully completed.