SUPERVISED LEARNING

Name : Sabinabanu Kadhar Meeran

Student ID : 23031180

GitHub Link : Sabinabanu11/ML-Supervised-Learning

1. Introduction to Supervised Learning

The main idea of supervised learning is to train models that can generalize to new, unseen cases using labeled data (input-output pairs).

Common Tasks:

- **Classification**: Making predictions about distinct labels (such as spam versus non-spam).
- **Regression:** Predicting continuous quantities, like home prices, is known as regression.

2. Crucial Elements of Supervised Education

- **Information and Labels**:

  With a known target (label) associated with each data point, the model can learn input-output correlations.

- **Training and Testing**:

  Two sets of data are separated for training and testing. To make sure the model generalizes well, it is fitted on the training set and its performance is evaluated on the test set.

- **Metrics**:
  - Regression: Mean Squared Error (MSE), Mean Absolute Error (MAE), R2 score
  - Classification: Accuracy, Precision, Recall, F1-Score

3. Core Algorithms and Their Functions:

Complex neural networks and linear models are examples of supervised learning techniques. We will concentrate on two well-known families: Multilayer Perceptrons (MLPs) and Support Vector Machines (SVMs).

3.a. Support Vector Machines (SVM)

Overview:

- When dividing data into different classes, SVM finds the best hyperplane to maximize the margin between them.

- It is adaptable for a range of datasets since it works well for both linear and non-linear classification tasks.

- For example, SVMs are frequently used in bioinformatics to identify genetic diseases or cancer. They are also used in image recognition, for example, in binary classification to separate face images from non-facial photos.

SVM Kernels:

- Linear: For linearly separable data.
- Polynomial: Maps input features into a higher-degree space.
- RBF (Radial Basis Function): Maps data into an infinite-dimensional space for complex patterns.

Practical Steps:

1. Fit the Model: model = SVC(kernel='linear', C=1.0)
2. Predict Outcomes: predictions = model.predict(X_test)
3. Evaluate: Use accuracy or confusion matrix to measure performance.

3.b. Multilayer Perceptrons (MLPs)

Concept:

- A feedforward neural network made up of several layers of neurons is called an MLP.
- Every neuron uses a non-linear activation function after applying a weighted sum of inputs.

Important Roles and Ideas:

- Depth (Number of levels):
- The model may learn complex, hierarchical characteristics with more levels.
- If not properly handled, having too many layers can result in overfitting and training challenges.
- Width (Number of Neurons per Layer):
- The ability of the model to learn patterns is enhanced with more neurons.
- Overfitting and higher computing costs might result from excessive width.

Training an MLP:

- **Loss Function** This determines how far the actual target values deviate from the model's predictions. Mean Squared Error (MSE) for regression tasks and cross-entropy for classification tasks are typical examples.
- **Optimization (Gradient Descent)**: To reduce the error determined by the loss function, the model's weights are gradually updated. This procedure guarantees that the model's predictive abilities improve with time.
- **Activation Functions**: Activation functions enable the model to capture intricate relationships in the data by introducing non-linear transformations. Tanh, sigmoid, and ReLU (Rectified Linear Unit) are examples of popular activation functions.

4. Supervised Learning real-world applications:

Many sectors use supervised learning techniques to increase efficiency, improve predictions, and automate decision-making. Here are a few significant real-world uses:

1. Medical care:

- **Disease Diagnosis**: To identify diseases like tumors or diabetic retinopathy, models can categorize medical imagery (X-rays, MRIs).

- **Patient risk assessment:** is the process of identifying patients who, based on their medical history and test findings, are most likely to develop a particular disease.

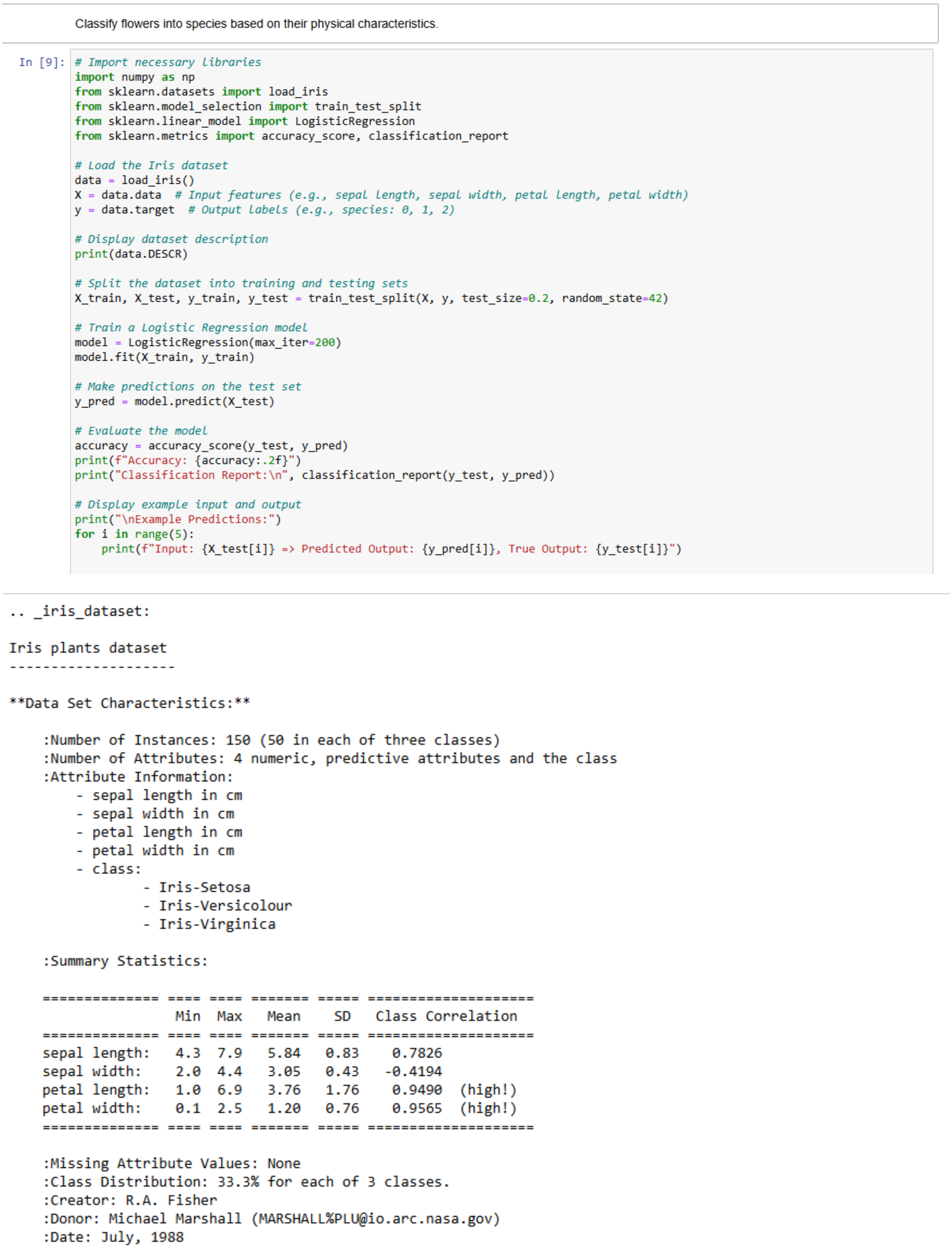
2. Finance:

- **Credit scoring**: Assisting financial organizations in making well-informed lending decisions by forecasting a loan applicant's likelihood of default.
- **Fraud Detection**: Finding questionable transactions or odd trends in credit card usage that point to fraud is known as fraud detection.

5. Pros and Cons of Supervised Learning

Pros	Cons
High Precision: For jobs where labeled data is plentiful and of high quality, supervised learning models typically attain high accuracy. Simple Training Procedure: Compared to unsupervised approaches, training is made simpler by the obvious link between input attributes (X) and labeled outputs (y). Broad Applicability: Performs well on a range of tasks, including regression (e.g., sales forecasting, house price prediction) and classification (e.g., spam identification, disease diagnosis). Capabilities for prediction: Supervised models can generalize to predict unknown data once they have been trained. Interpretability: Some supervised models offer insights into the relationships between inputs and outputs and are very interpretable, such as decision trees and linear regression. Scalability: When parallel computation is feasible, many supervised learning techniques can scale to effectively handle big datatsets.	Reliance on Labeled Information: Requires a lot of labeled data, which in some sectors can be difficult, costly, or impracticable to gather. Risk of Overfitting: It might memorize the training data rather than generalizing to new data if the model is overly complicated or the training data is too little. Restricted to Known Results: Only outputs found in the labeled dataset can be predicted using supervised learning. It has trouble in new situations or classrooms. Time-consuming Instruction: Training can be time-consuming and computationally costly, particularly for complicated models like deep neural networks and massive datasets. Problems with bias and data quality: The quality of the labeled data has a significant impact on the quality of the predictions. Bad models are the result of biased or poorly labeled datasets. Engineering Features: A lot of supervised models need meticulous feature selection and preprocessing, which might make implementation more difficult.

Python Code Insights



Classify flowers into species based on their physical characteristics.

In [9]:

```python
# Import necessary libraries
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Load the Iris dataset
data = load_iris()
X = data.data  # Input features (e.g., sepal length, sepal width, petal length, petal width)
y = data.target  # Output labels (e.g., species: 0, 1, 2)

# Display dataset description
print(data.DESCR)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a Logistic Regression model
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:\n", classification_report(y_test, y_pred))

# Display example input and output
print("\nExample Predictions:")
for i in range(5):
    print(f"Input: {X_test[i]} => Predicted Output: {y_pred[i]}, True Output: {y_test[i]}")
```

```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988
```

```
Accuracy: 1.00
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

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


Example Predictions:
Input: [6.1 2.8 4.7 1.2] => Predicted Output: 1, True Output: 1
Input: [5.7 3.8 1.7 0.3] => Predicted Output: 0, True Output: 0
Input: [7.7 2.6 6.9 2.3] => Predicted Output: 2, True Output: 2
Input: [6.  2.9 4.5 1.5] => Predicted Output: 1, True Output: 1
Input: [6.8 2.8 4.8 1.4] => Predicted Output: 1, True Output: 1
```

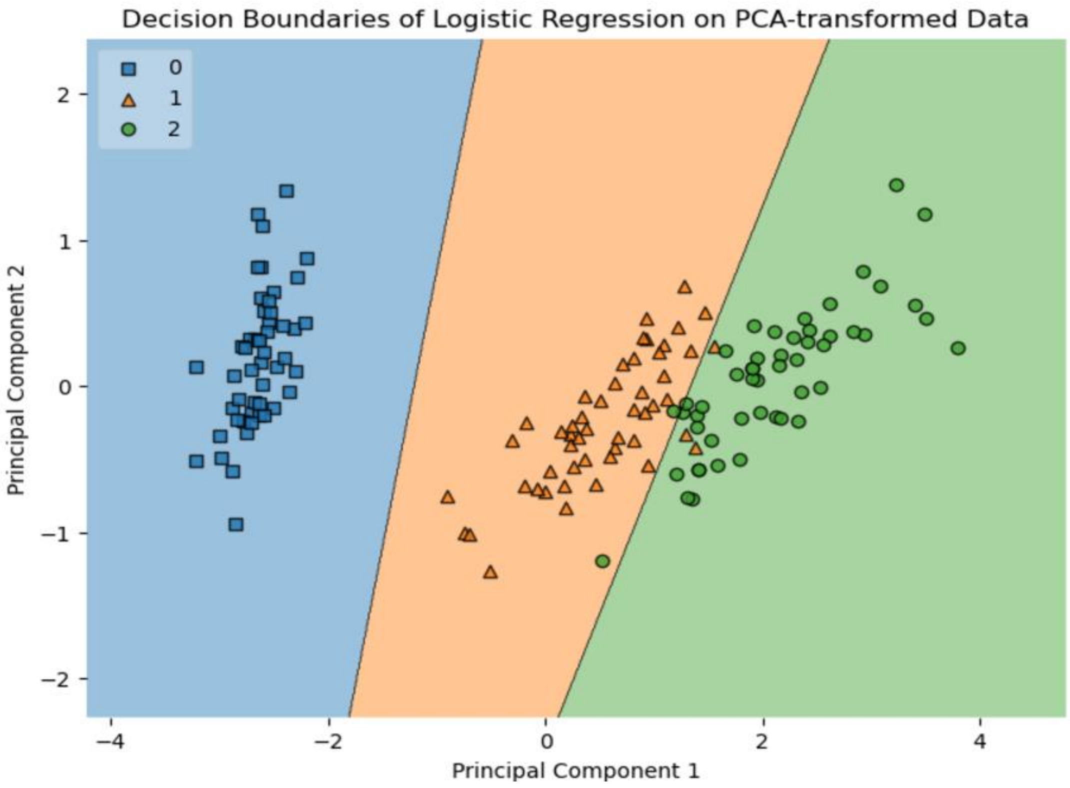
```
pip install mlxtend
```

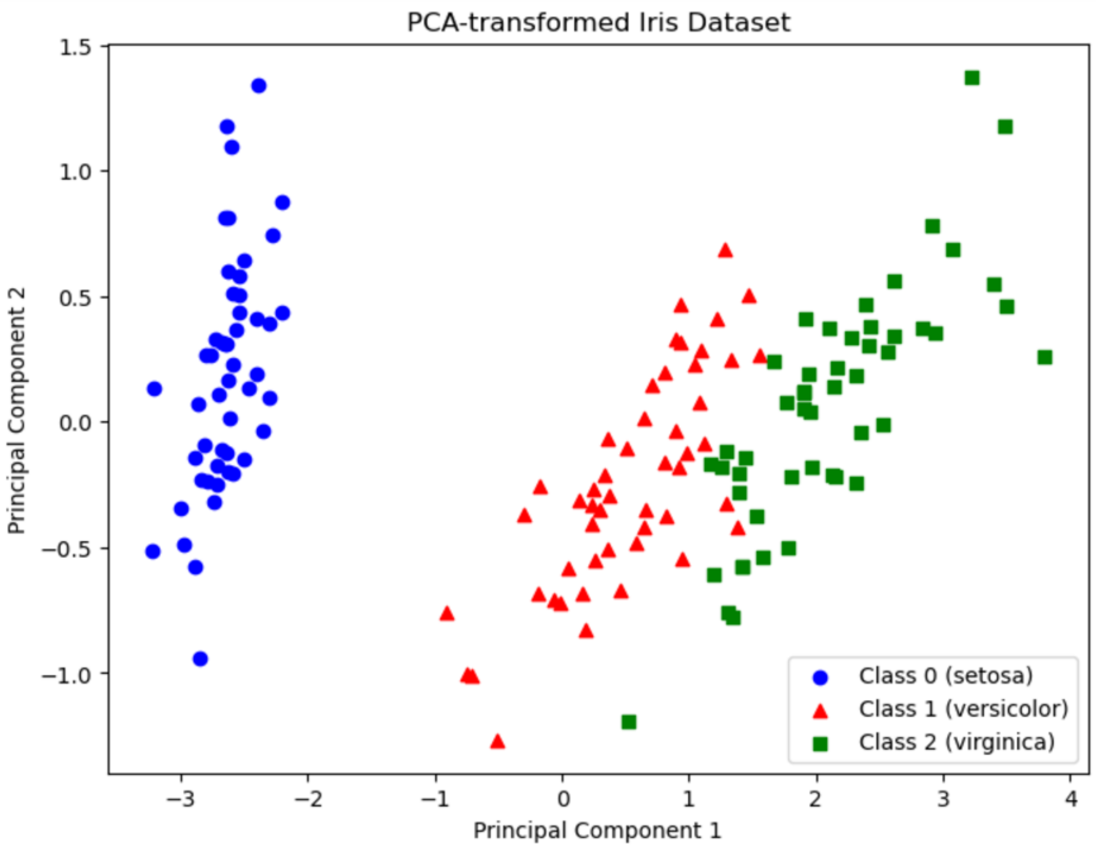
```
!pip install mlxtend
```

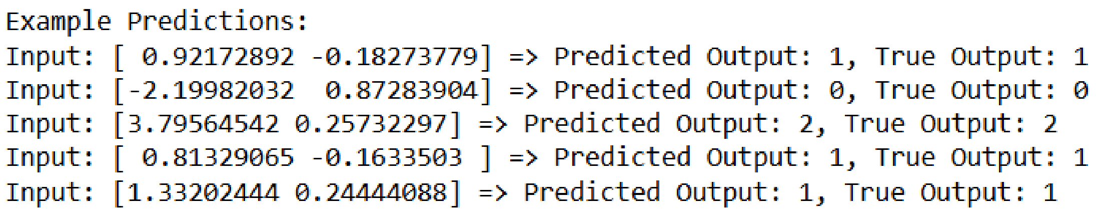
```python
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
from sklearn.decomposition import PCA
from mlxtend.plotting import plot_decision_regions

# Load the Iris dataset
data = load_iris()
X = data.data  # Input features (sepal length, sepal width, petal length, petal width)
y = data.target  # Output labels (species: 0, 1, 2)

# Reduce the data to 2 dimensions using PCA for visualization
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.2, random_state=42)

# Train a Logistic Regression model
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
print("Classification Report:\n", classification_report(y_test, y_pred))

# Visualize decision boundaries
plt.figure(figsize=(8, 6))
plot_decision_regions(X_pca, y, clf=model, legend=2)
plt.title("Decision Boundaries of Logistic Regression on PCA-transformed Data")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
```

```
# Visualize the original data distribution
plt.figure(figsize=(8, 6))
for label, marker, color in zip(range(3), ('o', '^', 's'), ('blue', 'red', 'green')):
    plt.scatter(X_pca[y == label, 0], X_pca[y == label, 1],
                label=f"Class {label} ({data.target_names[label]})",
                marker=marker, color=color)
plt.title("PCA-transformed Iris Dataset")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend()
plt.show()

# Display example input and output
print("\nExample Predictions:")
for i in range(5):
    print(f"Input: {X_test[i]} => Predicted Output: {y_pred[i]}, True Output: {y_test[i]}")
```

```
Accuracy: 1.00
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

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







```
Example Predictions:
Input: [ 0.92172892 -0.18273779] => Predicted Output: 1, True Output: 1
Input: [-2.19982032  0.87283904] => Predicted Output: 0, True Output: 0
Input: [3.79564542 0.25732297] => Predicted Output: 2, True Output: 2
Input: [ 0.81329065 -0.1633503 ] => Predicted Output: 1, True Output: 1
Input: [1.33202444 0.24444088] => Predicted Output: 1, True Output: 1
```

Outputs:

- A confusion matrix to visualize correct and incorrect predictions.
- Classification report with metrics like Precision, Recall, and F1-score.

How the Code Works

1. Dataset:

- Four input features make up the Iris dataset: petal length, petal width, sepal length, and sepal width (represented by X).
- Three criteria are used to classify the flower species represented by the target variable (y):
- 0: Setosa 1: Virginica 2: Versicolor

2. Model:

- Classification tasks are handled using the simple supervised learning method known as logistic regression.

3. Process:

- o 20% of the dataset is used to test the model's performance, and the remaining 80% is used for training.
- o The training data is used to train the model.
- o The test data that is not visible is used to make predictions.
- o A thorough categorization report and measures like accuracy are used to evaluate the model's performance.

Output Example

1. **Accuracy**: Measures how often the model correctly predicts the species.

2. **Classification Report**:

```
precision    recall  f1-score   support

0      1.00      1.00      1.00       10

1      0.92      0.92      0.92       12

2      0.92      0.92      0.92        8

accuracy                        0.95       30
```

3. **Example Input/Output**

```
Example Predictions:

Input: [6.1 2.8 4.7 1.2] => Predicted Output: 1, True Output: 1

Input: [5.8 2.7 3.9 1.2] => Predicted Output: 1, True Output: 1

Input: [5.1 3.3 1.7 0.5] => Predicted Output: 0, True Output: 0
```

Conclusion:

- Labelled data is used in supervised learning to direct the learning process.
- SVMs: The accuracy and flexibility of the boundary are influenced by the choice of kernel and parameter tweaking.
- MLPs: Modifying the network's width and depth alters its ability to represent complexity, but it necessitates careful tuning and regularization.
- Equipped with these understandings, practitioners can modify their models to better suit their data, enhancing the accuracy and comprehensibility of their findings.