NumPy :

- It is a numeric Python module that provides fast math functions for calculations.
- It is used to read data in NumPy arrays and for manipulation purposes.

Pandas :

- It is used to read and write different files.
- Data manipulation can be done easily with data frames.

scikit-learn / sklearn :

- In Python, sklearn is a machine learning package that includes a lot of ML algorithms.

Matplotlib:

Matplotlib is a visualization library in Python. It is built on NumPy arrays and consists of several plots like line, bar, scatter, histogram, etc.

Pyplot:

Pyplot makes Matplotlib work like Matlab.

Standard Scaler: It performs the normalization operation.

scaler.fit(): Similar to fitting a model, we can fit our scaling to the data using the fit method on the training set. This simply estimates the mean and standard deviation.

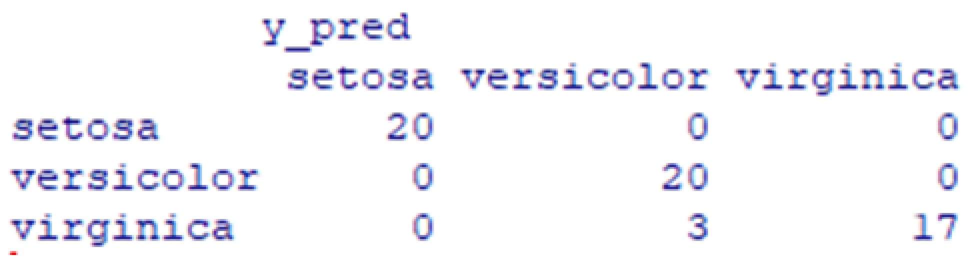
scaler.transform(): To apply the scaling, we can use the transform method, which will subtract the mean and divide by the standard deviation.

classifier.fit(): The learning algorithm / estimator instance is first fitted to the model, i.e., it must *learn* from the model. This is done by passing our training set to the fit method.

classifier.predict(): The predict() will perform a prediction for each test instance, and it usually accepts only a single input.

confusion_matrix: Confusion matrix represents the prediction summary in matrix form. It shows how many predictions are correct and incorrect per class.

A demo of Confusion Matrix interpretation:



```
            y_pred
             setosa versicolor virginica
setosa           20          0         0
versicolor        0         20         0
virginica         0          3        17
```

20 Setosa and 20 Versicolor are correctly classified as Setosa and Versicolor respectively. Out of 20 Virginica, 17 are correctly classified as Virginica, and 3 are wrongly classified as Versicolor.

classification_report: Build a text report showing the main classification metrics.

(i) Precision: TP / (TP + FP)

(ii) Recall / Sensitivity: TP / (TP + FN)

(iii) F1 Score = (2 * Precision * Recall) / (Precision + Recall)

True Positive (TP): A true positive is an outcome where the model correctly predicts the positive class.

True Negative (TN): A true negative is an outcome where the model correctly predicts the negative class.

False Positive (FP): A false positive is an outcome where the model incorrectly predicts the positive class.

False Negative (FN): A false negative is an outcome where the model *incorrectly* predicts the *negative* class.

Support: Support is a measure of the number of times an item set appears in a dataset.

Macro Average: Average the precision, recall, and F1 scores across all classes to get the final macro-averaged precision, recall, and F1 scores. You may do the hand calculations and cross-check with the output you got.

Weighted Average:

Weighted Average of Precision = [Sum(Individual Class Precision * Individual Class Support)] / Total Support

In a similar way, you can do it for the rest of the recall and F1 score. You may do the hand calculations and cross-check with the output you got.

accuracy_score: In Python, the accuracy_score function of the sklearn.metrics package calculates the accuracy score for a set of predicted labels against the true labels.

Accuracy: (TP + TN) / (TP + TN + FP + FN)

The IRIS data used in the example has a total of 150 data samples that comprise 50 from each class of IRIS Setosa, IRIS Virginica, and IRIS Versicolor.

During the training and test split, 70% was used for training and 30% for testing purposes.

30% of 150 is 45. That's the reason the total support count is 45.

For instance, depending on the output, among the 45, there can be 11 Iris Setosa, 17 Iris Virginica, and 17 Iris Versicolor. The same number has been reflected during the prediction evaluations in the confusion matrix.