MACHINE LEARNING

PRACTICALS

train_test_split

- The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.
- **Train Dataset**: Used to fit the machine learning model.
- **Test Dataset**: Used to evaluate the fit machine learning model.

train_test_split

- The objective is to estimate the performance of the machine learning model on new data: data not used to train the model.
- common split percentages include:
- Train: 80%, Test: 20%
- Train: 67%, Test: 33%
- Train: 50%, Test: 50%

train_test_split

- Split your original dataset into input (*X*) and output (*y*) columns.
- ***x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)***
- ***“test_size”***: wil take percentage (float) of the size of the dataset between 0 and 1
- ***“random_state”*** *:* This parameter is used to control the shuffling applied to the data before applying the split. Random number parameter that allows you to reproduce the same exact train test split each time you run the code.

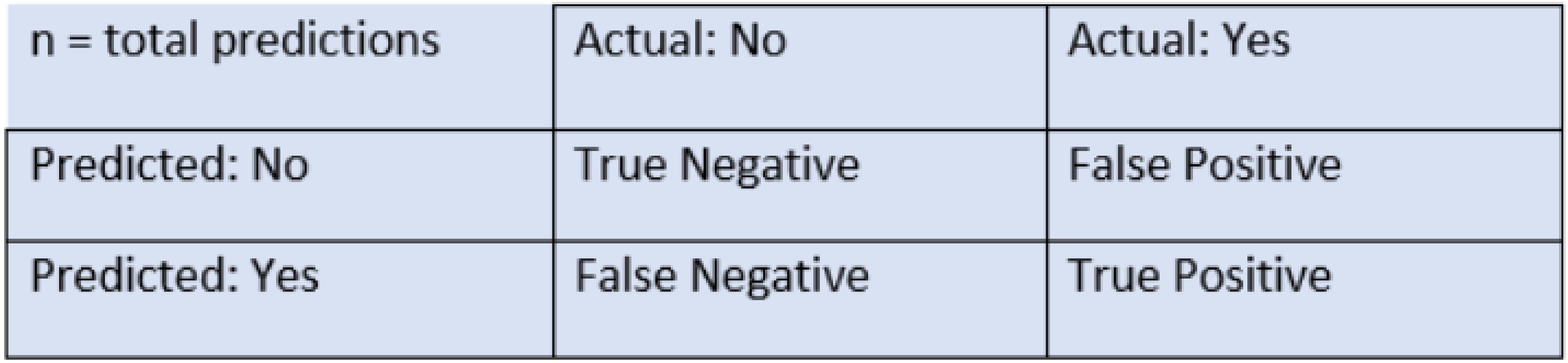
- ***“shuffle”***: This parameter is used to shuffle the data before splitting. Its default value is true.

StandardScaler()

- StandardScaler comes into play when the characteristics of the input dataset differ greatly between their ranges, or simply when they are measured in different units of measure.
- StandardScaler comes into play when the characteristics of the input dataset differ greatly between their ranges, or simply when they are measured in different units of measure.

confusion_matrix

- Determine the performance of the method.
- A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes.
- The matrix compares the actual target values with those predicted by the machine learning model.



n = total predictions	Actual: No	Actual: Yes
Predicted: No	True Negative	False Positive
Predicted: Yes	False Negative	True Positive

- **True Negative:** Model has given prediction No, and the real or actual value was also No.
- **True Positive:** The model has predicted yes, and the actual value was also true.
- **False Negative:** The model has predicted no, but the actual value was Yes, it is also called as **Type-II error**.
- **False Positive:** The model has predicted Yes, but the actual value was No. It is also called a **Type-I error.**

metrics.accuracy_score()

- Computes the accuracy of the model.