Splitting The Data Into Train And Test

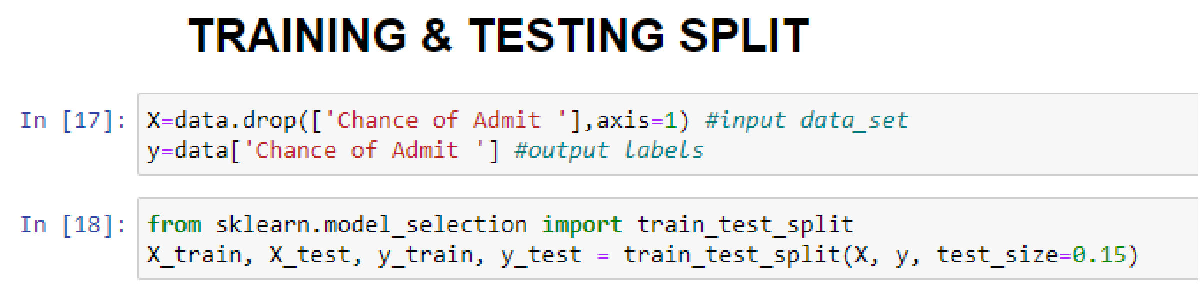
UNIVERSITY ADMIT ELIGIBILITY PREDICTOR

TEAM ID : PNT2022TMID36734

When you are working on a model and you want to train it, you obviously have a dataset. But after training, we have to test the model on some test dataset. For this, you will need a dataset that is different from the training set you used earlier. But it might not always be possible to have so much data during the development phase. In such cases, the solution is to split the dataset into two sets, one for training and the other for testing.

To help us with this task, the Scikit library provides a tool, called the Model Selection library. There is a class in the library which is, 'train_test_split.' Using this we can easily split the dataset into the training and the testing datasets in various proportions.
The train-test split is a technique for evaluating the performance of a machine learning algorithm.

- **Train Dataset:** Used to fit the machine learning model.
- **Test Dataset:** Used to evaluate the fit machine learning model.



TRAINING & TESTING SPLIT

```
In [17]: X=data.drop(['Chance of Admit '],axis=1) #input data_set
         y=data['Chance of Admit '] #output labels
```

```
In [18]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15)
```

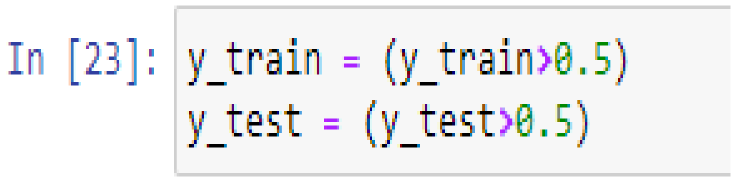
In general, you can allocate 80% of the dataset to the training set and the remaining 20% to the test set. We will create 4 sets

- **x_train**
- **x_test**
- **y_train**
- **y_test .**

There are a few other parameters that we need to understand before we use the class:

Splitting The Data Into Train And Test

- test_size: this parameter decides the size of the data that has to be split as the test dataset. This is given as a fraction. For example, if you pass 0.5 as the value, the dataset will be split 50% as the test dataset and remaining a train dataset
- random_state: here you pass an integer, which will act as the seed for the random number generator during the split. Or, you can also pass an instance of the Random_state class, which will become the number generator. If you don't pass anything, the Random_state instance used by np.random will be used instead.



```
In [23]: y_train = (y_train>0.5)
         y_test = (y_test>0.5)
```