Daniele | Hossein

We want to use the variables to predictive the Pulsars.

We need the data to Train and Test the methods. Regarding to the two datasets offered, we will use all of the Train Dataset for Training and all of the Test Dataset for Testing.

|  |  |
| --- | --- |
| Module: data\_loading.py load\_train\_data() | Module: data\_loading.py load\_test\_data() |
| """  Load training data and divide it in labels and features.  Returns  -------  features : ndarray  labels: ndarray  """ | """  Load test data and divide it in labels and features.  Returns  -------  features : ndarray  labels: ndarray  """ |

Then, when a new patient (Test Dataset) shows up, we can measure these variables and predict if the Pulsar has existed or not exist.

However, first we have to decide which machine learning method would be best. also, we will use Cross-validation, Confusion Matrix, Logistic Regression, K-nearest neighbours and also support vector machines methods.

Cross-validation allows us to compare different machine learning methods and get a sense of how well they will work in practice.

In this method, we divided the Train Dataset into 5 blocks (4 blocks for train, 1 for test). This is called Five-Fold cross-validation.

|  |
| --- |
| Module: cross\_validation.py |
| """  Shuffle feautures and labels consistently and create samples.  Parameters  ----------  features : np.ndarray  labels : np.ndarray  samples : int  Number of partitions.  Returns  -------  sampled\_features : List[np.ndarray]  sampled\_labels : List[np.ndarray]  """ |

We need to do two things with the datasets:

1. Estimate the parameters for the machine learning methods.

* In other words, to use Logistic Regression, we have to use some of the data to estimate the shape of the curve. Here, estimating parameters is called Training the algorithm.

1. Evaluate how well the machine learning methods work.

* In other words, we need to find out if the curve will do a good job categorizing new data. Here, evaluating methods is called Testing the algorithm.

Then, we can compare methods by seeing how well each one categorized the test data. We assume that this data for training and testing are the best way to divide up the data.

By the way, Cross-validation uses them all one at a time to which block would be best for testing, and summarizes the results at the end. It keeps track of how well the method did with the test data. Then it uses combination of blocks to train the method.

Beside, we need to summarize how each method performed on the Testing data. One way to do this is by creating a **Confusion Matrix** for each method.

The rows in a Confusion Matrix (It is especially called here predicted\_labels) corresponds to what the machine learning algorithm predicted and the columns (It is especially called here true\_labels) corresponds to the know truth.

|  |
| --- |
| Module: confusion\_matrix.py |
| """  Compute confusion matrix from predicted labels and true labels.  Parameters  ----------  true\_labels : ndarray  Real labels to make test on.  predicted\_labels : ndarray  Labels predicted by model.  Returns  -------  confusion\_matrix : ndarray  elemnt[i, j] is predicted as part of i class, but its class is j  """ |

Since there are only two categories to choose from: “sotun samte chap” or “sotun samte rast”, then the top left corner contains True Positives. These are the pulsars that had “sotun samte chap” that were correctly identified by the algorithm.

The true negatives are in the bottom right-hand corner. These are the pulsars that did not have “sotun samte rast” that were correctly identified by the algorithm.

The bottom left-land corner contains the False Negatives. These are when a patient has “sotun samte chap”, but the algorithm said they didn’t.

Lastly, the top right-hand corner contains the False Positives. These are pulsars that “sotun samte rast”, but the algorithm says they are.

The numbers along the diagonal (The True Positives and True Negatives) tell us how many times the samples were correctly classified.

The numbers not on the diagonal (the False Positives and False Negatives) are samples the algorithm messed up.

We can apply Logistic Regression to the Testing Dataset and create a Confusion Matrix.

In the end, every block of data is used for testing and we can compare methods by seeing how well they performed.

In this case, since the support vector machine did the best classifying the test datasets, we will use it.