**Lab 4 Report:**

**Task 1**

**Random Forest**

**Final Hyper-Parameters:**

Number of splits: 200

Random state (seed): 5

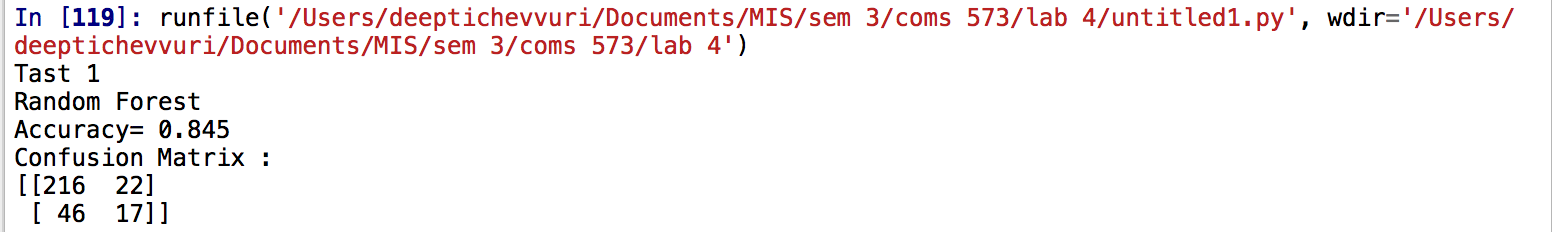
Number of trees: 50

Maximum features: 2

**Table: Tuning Process**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Number of splits** | **seed** | **Number of trees** | **Maximum features** | **Accuracy (%)** | **Conclusion** |
| 200 | 5 | 50 | 2 | 84.5 | Number of |
| 190 | 5 | 50 | 2 | 82.64 | Splits=200 |
| 210 | 5 | 50 | 2 | 83.8 | selected |
| 200 | 4 | 50 | 2 | 84 | Seed= 5 |
| 200 | 6 | 50 | 2 | 82.25 | selected |
| 200 | 5 | 45 | 2 | 84.25 | Number of |
| 200 | 5 | 55 | 2 | 83.75 | Trees= 50 |
| 200 | 5 | 50 | 1 | 84 | Max |
| 200 | 5 | 50 | 4 | 82.75 | Features= 2 |

The performance with above hyper parameters:

**

**Conclusions:**

* It gives a way to control the input variables or features, i.e. the number of variables to be considered is variable.
* The performance is good when the maximum number of features (hyper parameter) is set to square root of the number of number of features (2). Before making this conclusion the model as tested for different possibilities of maximum number of features which is Auto (all), sqrt(all) and 0.2 (0.2% of all).
* Amongst the remaining algorithms, RF gives the best accuracy (84.5%).

**Adaboost:**

Number of splits: 300

Random state (seed): 1

Maximum depth:1

Minimum sample leaf:1

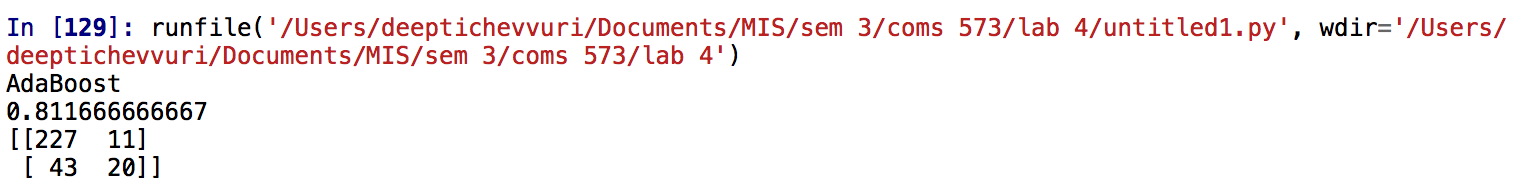
Learning rate:1

Number of trees: 500

**Table: Tuning Process**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Number of splits** | **seed** | **Max Depth** | **Min sample Leaf** | **Learning rate** | **Number of trees** | **Accuracy (%)** | **Conclusion** |
| 300 | 1 | 1 | 1 | 1 | 500 | 81.17 | Number of |
| 290 | 1 | 1 | 1 | 1 | 500 | 80.96 | Splits=300 |
| 310 | 1 | 1 | 1 | 1 | 500 | 80.68 | selected |
| 300 | 2 | 1 | 1 | 1 | 500 | 81.17 | Seed= 1 |
| 300 | 1 | 0.9 | 1 | 1 | 500 | 78.16 | Max |
| 300 | 1 | 01.1 | 1 | 1 | 500 | 81.17 | Depth=1 |
| 300 | 1 | 1 | 2 | 1 | 500 | 81.17 | Min leaf=1 |
| 300 | 1 | 1 | 1 | 0.9 | 500 | 80.6 | learning |
| 300 | 1 | 1 | 1 | 2 | 500 | 78.17 | rate= 1 |
| 300 | 1 | 1 | 1 | 1 | 510 | 78.16 | Num of |
| 300 | 1 | 1 | 1 | 1 | 490 | 78.17 | Trees= 500 |

The performance with above hyper parameters:



**Conclusions:**

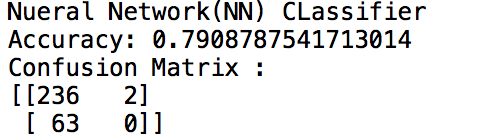
* It takes maximum time to train, when compared to all the other algorithms.
* AdaBoost deals with the class imbalance problem by maintaining a set of weights on the training data set in the learning process
* The number of trees in adaboost okay a significant role in imporving the performance of the classifier, this is because larger trees have low or zero redistribution error rate.

**Task 2:**

**Neural Network(NN) Classifier:**

Random state:1

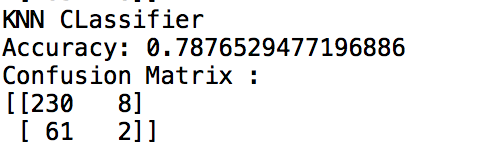
Cv=10



**KNN Classifier:**

Random state:20

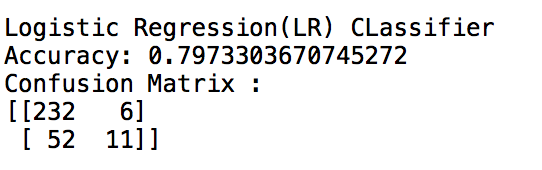
Cv=10



**Logistic Regression(LR) Classifier:**

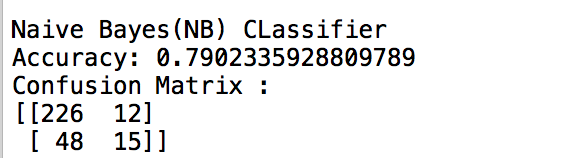
Random state:1

Cv=10



**Naive Bayes(NB) Classifier:**

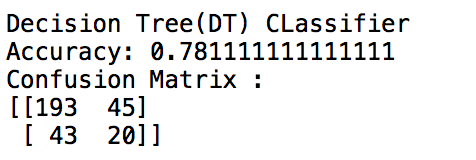
Cv=10



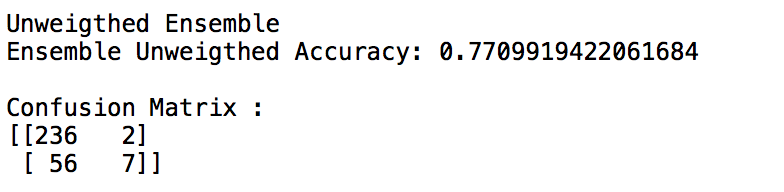
**Decision Tree(DT) Classifier:**

Random state:10

Cv=60



**Unweighted Majority Vote Ensemble Classifier Performance:**

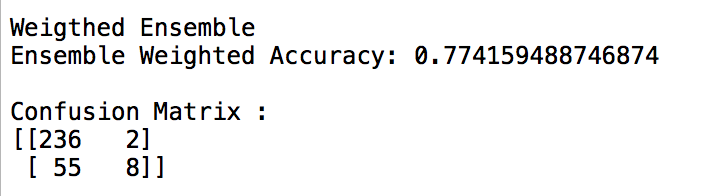
****

**Weighted Majority Vote Ensemble Classifier Performance:**

The weights have been calculated as proportional to the classification accuracy

Accuracy: {79.088,78.765,79.733,79.023,78.111}

Weights: {1.01,1.01,1.02,1.01,1}



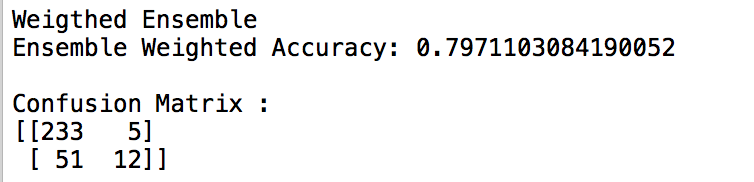
Other Strategies:

1. When weight of one classifier is increased keeping the other common, for example {2,1,1,1,1}; {1,2,1,1,1} etc the accuracy of the classifier is:

{nn, knn ,lr, gnb, dt}=> {77.11, 77.74, 77.75, 78.07, 76.10}

From this one conclusion can be made, if more weightage is give to Naïve Bayes the performance of the ensemble classifier is high.

1. Even by increasing the value of weights i.e.e taking them as {100,100,100,100,100} doesn’t increase the performance, if the weights are proportionally changed amongst them then the accuracy can be changed.
2. Weights = [20,60,50,100,2] give an accuracy of 79.71



**Table: Weights Selection**

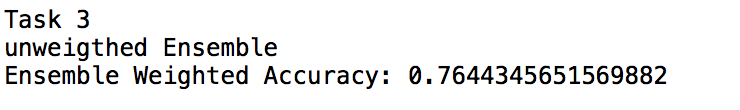
|  |  |
| --- | --- |
| **Weights** | **Accuracy** |
| {1.01,1.01,1.02,1.01,1} | 77.415 |
| {2,1,1,1,1} | 77.11 |
| {1,2,1,1,1} | 77.74 |
| {1,1,2,1,1} | 77.75 |
| {1,1,1,2,1} | 78.07 |
| {1,1,1,1,1} | 76.10 |
| {20,60,50,100,} | **79.71** |

**Conclusions:**

* Ensemble methodology imitates our nature to seek several opinions before making a crucial decision.
* The main principle is to weigh several individual pattern classifiers, and combine them to reach a classification that is better than the one obtained by each of them separately.
* The majority rule voting approach might not always work so well, especially if the ensemble consists of more “weak” than “strong” classification models.

**Task 3:**

**Unweighted Majority Vote Ensemble Classifier Performance (7 models):**

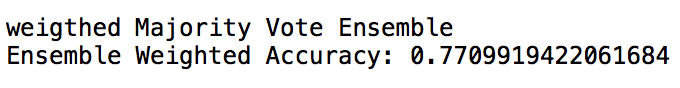
****

**Weighted Majority Vote Ensemble Classifier Performance (7 models):**

Weights given are proportional to their classification accuracy,

Accuracy: {84.5.81.17,79.088,78.765,79.733,79.023,78.111}

Weights: {1.08,1.04,1.01,1.01,1.02,1.01,1}



**Conclusions:**

* The additional two classifiers added Random forest and Adaboost have higher accuracies so, if the weight given to these classifiers is more the performance of the ensemble is higher.
* The results shown are for a proportional ensemble, but the model has been tested for weights of RF and Ada greater than the weight of Gnb has given higher performance.
* It has also been observed that the performance of the models is a little less when run for the first time, if the models are run continuously for 5 to 6 times, the performance settles in a maximums stable value. This is because all the models are probabilistic in nature.