Ada Boost with Decision Stump

Algorithm Description:

We will get the preprocessed data with the generateData function which will transform the csv file and extract XTrain, YTrain, XTest, YTestActual.

- We will create a hypothesis function using the lambda expression. Also we will initialize the Distribution matrix.
- Using the getweaklearner method we will find out a random attribute having error < 0.5.
  If there are no more weak learner then we will report the hypothesis function which will be a list (X,Y) where X is the hypothesis function and Y is the attribute for the weak learner from the training set of dimension D.
- Update the error e which is calculated as the sum of the Distribution array and the predicted labels with the weak classifier.
- Find out alpha using the AdaBoost algorithm
  - Alpha values:
    [1.6322431680601266, 0.048871142386468454, 0.096349218913841336, 0.050917299646150285, 0.03327198168178299]
- Append the values of alpha and weak learner attribute index is captured and stored.
- This will loop as many as the number of dimension of Training set

Output:

The algorithm will return a model here which consists of three tuples.

(X,Y,Z) where

X➔ List of alpha

Y➔hypothesis function list it will be an nested array like (Y1,Y2) where Y1 is the lambda function and Y2 is the index of the weak identifier

Z➔ version whether it is decision stump or perceptron algorithm

At the end of this of method we will get a model with above parameters

Assumption:

- All values in the training set is mapped to +1, -1
- Republicans are mapped with label +1 and democrats with label -1
- Missing values are mapped as negative points in the training set.
- Overfitting is handled with the getWeakClassifier function which takes care of finding the weak classifier with the less error

Accuracy observed with the weak learners: “**95.412844%”**

Note: Please uncomment the portion of the stump to get the output.

Ada Boost with Threshold Unit(Perceptron)

Algorithm Description:

1. In the first iteration of the ***pla*** method, the while loop will evaluate to true since the initial weight vectors are all zero. This is achieved using the ***misclassified*** method which return a boolean value as true if still there are misclassified points left in the sample otherwise this method returns false.
2. Choose a random misclassified point with the weight vector. This is done using the method ***getMisclassifiedPoint*** which returns a list of misclassified point with their labels.
3. Update the weight vector in **pla** method and then test all the samples with the updated weights. Also increase the count of iterations since the sample was a misclassified point.
4. Check again if all the points are correctly classified with the updated weight vector.
    - If YES exit the while loop and we have found the weights which correctly classify the data.
    - If NO, then repeat the process again from Step (1) until there are no more misclassified points

5. Every time we will keep the lowest error and corresponding weights in temporary variable which will be flipped as soon as the better classifier is observed
6. Since the pla will not converge with the non-linearly separable data so an additional exit condition is placed which will terminate after maximum of 1000 iterations.

At the end of this method we will have a final weight vector, the number of iterations taken, version which will be perceptron will be returned.

Output: A final weight vector would look something like

w=[-13., 5., -5., -7., 25., -3., -1., 9., -3., -5., 7., -9., 3., -1., -3., -5., 1.]

Accuracy = 96.330275%

Note: Currently the accuracy shown by default will be for the perceptron algorithm, to get the accuracy of the other one, uncomment the version for the stump

Comparison of Decision Stump vs Perceptron:

- Ada boosting with multiple weak learners are successfully able to reduce the error after each iteration by increasing the weight of the mis-classified points and reducing the weights of the correctly classified points however as compared to Pocket algorithm decision stump takes more time to train a data as it has to look down for a final hypothesis with the best classification
- As per the observation of the experiment the pocket algorithm accuracy varies frequently with high variance while that of the Decision Stump accuracy remains almost the same.
- Testing data with the Pocket is faster as compared to the Decision Stump algorithm as