CS5691 - PRML DATA CONTEST REPORT

A. Pragnya - cs17b004
V.V.S.S. Mounik - cs17b031

Aim: Predict whether an employee remains in the company

Input: List of ratings and remarks given by an employee for a company. Also a list of whether an employee supports a remark or not. Along with training data.

Output: Prediction on given test data

Public Leaderboard accuracy produced (de-randomized code) : 0.88349

Corresponding private leaderboard score: 0.85481

Libraries used: sklearn, numpy

Algorithms used : SVM(From Sklearn), Random Forests (Implemented)

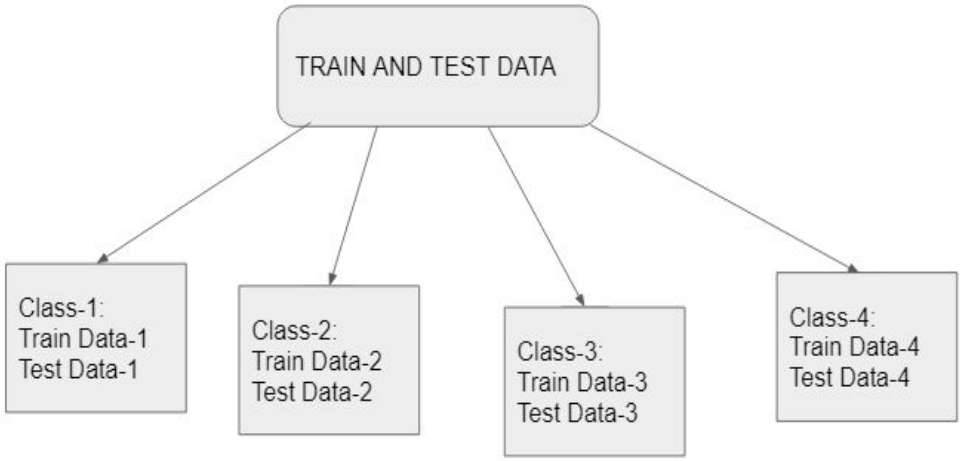
Instructions to run the code:

- Make sure you put the .py file in the same directory containing the ratings.csv, remarks.csv and other .csv input files.
- Run the .py file using python or python3 and wait for around 1.5-2 hrs to get the output written.

REPORT :

- As employee id is relative inside a company, a better unique identifier can be considered as (company+employee id).

- Our approach is given a unique identifier, construct the attribute vector by adding all features and get train data and test data as 2D arrays and run the algorithms.
- We've classified the employees into 4 categories and done learning on each category.
- A better view is attached below.



Where
Class-1: Employee hasn't given any remarks, also hasn't supported or opposed any remark.
Class-2: Employee has given remarks, but hasn't supported or opposed any remark.
Class-3: Employee hasn't given any remarks, but supported or opposed some remarks.
Class-4: Employee has given remarks, and also supported or opposed some remarks.

- Both train and test data are splitted into 4 classes.
- Based on the information available in each class, prediction was made for the test data.

- Given a test point, find which class it belongs to and predict using the classifier of that class.
- We've used dictionaries to store the information from input files.

Preprocessing:

- For employees in any class, the following features are first computed.
  No of ratings, First and last rating dates, no of days,first, last and average ratings, no of days for each rating using the ratings dictionary.
- For class-2 and class-4, the following attributes are considered from remarks file using remark dictionary:
  No of remarks, First and last remark dates, no of days,first, last and average remark length, no of days for each remark.
- For class-3 and class-4, no of remarks supported and opposed are considered from remarks_supp_opp file using it's dictionary.
- Dates are measured relatively from 1 Jan,2015.
- Therefore, we get 8D data from class-1 employees,16D data from class-1 employees,10D data from class-1 employees and 18D data from class-1 employees.

Classification:

- The following is done in each class.
- Split the train data into 80-20 percent to get training and validation sets. Validation sets are used for hyper parameter tuning.
- The algorithms and hyperparameter choices used are
  - SVM with gaussian kernel with C in [1e-4,0.001,0.01,0.1,1,10,100,1000] and gamma in [1e-5,1e-4,0.001,0.01,0.1,1,10,100,1000]
  - Random forests with num_trees in [40,60,80], criterion in ['accuracy','entropy'], a in [0.5,0.75], b in [0.5] and num_nodes_stop in [10,30,50]
- Classifier which maximises the weighted (5-1) accuracy for validation set is considered the classifier for the class and prediction on the test data in the class is made using the same.

- The predictions are made 3 times by re-splitting the training set again and based on the majority of the values, the final prediction is made.
- It is seen that SVM is a good classifier for class-1 and rfs are good classifiers for the other classes.

Attempts which gave lower accuracies in public leaderboard:

- We've considered many additional features such as last k ratings, last k remark lengths, avg remark length for supported and opposed remarks etc. which haven't produced any improvements. Also it led to the same prediction for every datapoint in the class leading to less accuracy.
- Also, we've tried logistic regression, 2-NN, Bayes classifier with appropriate assumptions but they also failed to give good numbers.
- We've tried to make categories based on companies also. We get around 36 classes, but the classification inside a class sometimes has more test data than train data. So it also didn't produce any quality output.

Why did we go for class division:

- This is because, not all employees gave remarks or supported them.
- So maintaining 18D data for an employee which is in class-1 leads to some default values in 10 attributes which can dilute the algorithm.
- If the algorithm predicts based on these default values it isn't producing any quality output.
- So, we've thought it would be better if we gave the class it's own attribute vector dimension.

Any better ideas for increasing accuracies further:

- We can increase the list of hyper parameter choices in random forests.
- We can also use XGBoost, ANNs etc.
- But these might increase the time complexity and we've to keep running our code for hours.