CS 5011: Machine Learning - Contest Report

Due on Tuesday, November 18, 2014

Team 04

Pratik (CS12B023) and Sagar (CS12B039)

Missing Values:

We first tried to remove all the instances and columns of data which had missing values. Both ways didn't work as there were missing values in almost 2411 out of 3500 instances and 1606 out of 1897 columns.
We have tried the following methods to fill in the missing values.
1. Mean.
2. Median.
3. KNN for different values of k(4,8,10).
4. Used weka directly without filling values.

We initially used mean and median. The scores we got were almost same for different classifiers. When we used KNN we got significantly better scores.

Class Imbalance:

There is a class imbalance (700:2800) in the data. Due to this many classifiers were not working well.
Methods used to handle class imbalance:
1. Oversampling using our own algorithm.
2. Giving class weights in some classifiers.
3. Using cost sensitive classifiers available in weka.
4. Smote.

Some of the classifiers had an option of giving weights to the classes but most of them did not have this feature (like random forests etc). So we first tried to oversample the data on our own by repeating the values that were in the training set 3 times to make that class balanced. But this led the classifier to over fit the class A and was not giving good scores even compared to the score without oversampling.
We then tried cost sensitive classifier in weka. Finally we used smote for oversampling of data.

Dimensionality Reduction/Extraction:

We used the following methods for dimensionality reduction/extraction.
1. Principal Component Analysis.
2. Linear Discriminant Analysis.
3. Feature Agglomeration.
4. Clustering.

We used PCA to extract number of features(100, 200, 1000 etc) but nothing worked as better as using all the features directly. So finally we did not do any feature reduction/extraction. We tried clustering to find the similar features because more correlated features fall in to the same cluster.

Normalization:

We initially normalized the data using normal normalize function in sklearn. Then we used Normalizer function which actually regularizes the data in sklearn which gave very good scores compared to normalize and almost the best of all till then (around 82%). But when we submitted we got very less score (26%) and we don't know the reason behind it till now.

Cross Validation:
These are the following methods we used for cross validation.

1. k-fold Cross Validation.
2. Stratified k-fold Cross Validation.

Stratified k-fold tries to preserve the class distribution in the train and test data set. Initially when we were using normal cross validation, our score was nearly random and hence unreliable because of this reason. At some times the class distribution was captured in the train and test datasets and at some times, it wasn't. This caused answers to become meaningless and so we changed our approach. We then used Stratified K-Fold Cross Validation, where we got better and more promising results. With this we were able to distinguish upto a good extent, which were correctly classifying and which weren't.

Classifiers:
These are the different classifiers we used for the classification task.

1. SVM.
2. Random Forest.
3. Adaboost.
4. Bagging.
5. Gradient Boosting.
6. Gaussian Markov Model.
7. Linear Regression.
8. KNN.
9. Logistic Regression.
10. Random Tree.
11. Naive Bayes.

Of all of the above classifiers we tried, some gave very poor results, but gradient boosting and adaboost stood out from the rest with much more promising results. We first started with n = 500 in gradient boosting and increased upto 2500. But we found out that after n = 1500 the scores came down again.

Majority Voting:

After using a lot of classifiers we then started voting using these different classifiers. This will reduce the variance and also error made by one type of classifier won't be made by other type of classifiers. This helped us get more scores at last.

Brief Summary of how we did it:

I think that the data is not linearly separable. This is because none of the linear models we used were working better.We initially used all the above mentioned linear models and finally moved to boosting on decision stump.

We used gradient boosting with n = 1000 on the mean filled missing values data for our first submission (76%). And then we increased n to 1500 and we got (77%). But then when as we increased n further more our score decreased because of over fitting the data.

Then we tried using median data. But this did not work. We tried KNN using different values of k = 4, 8, 10. We found that k = 10 gave better answers on the cross validation.
We then tried random forest which did not show any better results. Then we stared adaboost on ADtrees.

Finally after we had a lot of classification results we used majority voting with the preference to the results which had better scores on cross validation. Our initial submission which gave 77% was giving 188 A and as the A count went to 190 and above the score kept decreasing. So we thought that A's count might be around 180. But to our surprise when we did boosting on smote filtered data we got above 290 A's and we got a score of 73 % when we submitted it. Then we thought that the number of A's must be around 250. Random forest and Random trees gave around 350 - 400 A's. So by using voting we tried to keep the count between 240 - 260 A's and this gave our best submission.

We divided the classifiers into ones which predicted A's correctly and others which predicted B's correctly. Then we took voting such that if classifiers which predicts A's correctly has predicted something as A then we put it as A and if predicts something as B then we put it as C and leave it for later processing. Similarly with the classifiers which predicted B's correctly. If we had a clash between these two class of classifiers then we go with the one that gave the higher score.

The reason we used boosting was that the data was not linearly separable.

You can find our work at
`https://github.com/jpsagarm95/ML_Contest`