SJSU CMPE 239 Data Mining Homework 4 -Report

Submitted By: Niveditha Bhandary [010820550]

Spring 2017

Rank & F1 score at the time of writing the report: 13, 0.6897

Introduction: The main objective of this assignment is to Use/implement a feature selection/reduction technique and experiment with various classification models with imbalanced data. Two modules, Ensemble Dimensionality reducer and Ensemble Classifier are developed to get best performance.

Ensemble Dimensionality reducer is used to check which dimensionality reduction technique or which combination of dimensionality reduction techniques gives the best performance. Ensemble classifier on the other end, combines the predicted values from different classifiers to give one classified output. The idea is, by using an ensemble of classifiers instead of a single classifier, we can improve the performance. The basic flow of the implementation is shown in figure1.

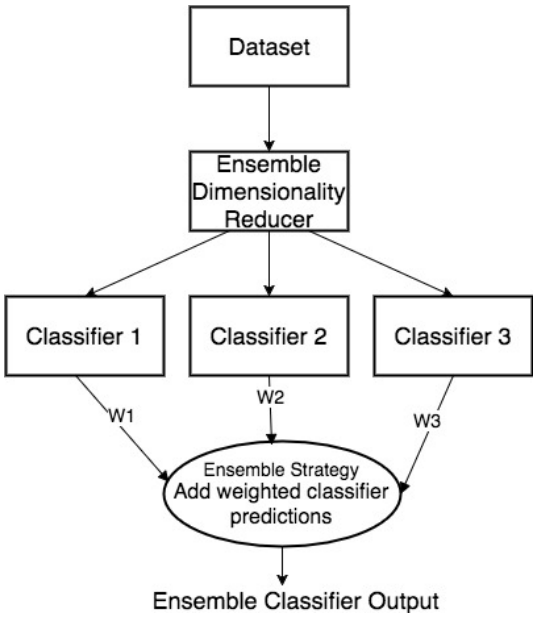

Figure 1: Flowchart for Ensemble classifier

Since the dataset is imbalanced, I tried to remove the imbalance in the data by using a technique called "SMOTE: Synthetic Minority Oversampling Technique". SMOTE is an over-sampling approach in which the minority class is over-sampled by creating "synthetic" examples. Once a good dimensionality reduction technique was found, "Pickle" tool was used to dump the reduced features to a file on the disk. These stored reduced features can be loaded to test different classifiers, instead of re-computing them each time.

Files included in Src folder:

1. classifier.py: Main program which does preprocessing of datasets and also calls different ensemble reduction and ensemble classifier functions
2. EnsembleDimensionalityReduction.py: Helper class to test different DR techniques
3. EnsembleClassifier.py: Helper methods to check different classifiers

Steps followed: Preprocessing on the train and test files are done to get train features, train labels and test features. Dimensionality reduction techniques are used to get reduced train and reduced test features.

For finding the best Dimensionality reduction technique or combination of dimensionality reduction techniques, following steps are followed:

1. A class EnsembleDimensionalityReduction is written with methods such as: __init__, fit, transform, fit_transform. Any number of dimensionality reduction techniques can be added to this class. For this assignment, PCA, LDA and Sparse PCA are used.

2. The objects of this class are initialized with a list of dimensionality reduction techniques and a list of their corresponding parameters.
3. In the main class, first an object of the EnsembleDimensionalityReduction class is created with a list of all the dimensionality reduction techniques that we want to use and their corresponding parameters list.
4. Using the object created in step 3, method fit_transform is called on train features to get the reduced train features.
5. To get test reduced features, transform method of the class is called with the same object and test features.

After completion of these steps, reduced train features and reduced test features are obtained. These are then saved in a file using pickle.

For finding the best classifier or ensemble of classifiers following steps are followed:

1. The cross validation predicted labels from different classifiers, with train documents is stored in different files.
2. To calculate the best combination of weights, random weights in the range [0,1) are generated. The weights are multiplied with predicted labels and added to get ensemble predicted labels for train dataset.
3. The ensemble predicted labels are then compared with actual train labels and the metrics are stored in a csv file.
4. This procedure is repeated 1000 times, and the weights with best metrics are selected.
5. The predicted output from each classifier for test data, is multiplied with the weights and added to get ensemble classifier labels on test data.

Results Obtained:

Technique		**Cross Validation Metric**				**F1 score on Test Data**
Dimensionality Reduction	**Classifier**	**Precision**	**Recall**	**F1 score**	**Score**	
PCA	KNN	0: 0.93 1: 0.81	0: 0.99 1: 0.33	0: 0.96 1: 0.47	0.93	0.4242
PCA + LDA	KNN	0: 0.93 1: 0.81	0: 0.99 1: 0.32	0: 0.96 1: 0.46	0.92	0.2857
Sparse PCA	**KNN**	**0: 0.93 1: 0.79**	**0: 0.99 1: 0.29**	**0: 0.96 1: 0.43**	**0.92**	**0.6897**
Sparse PCA + LDA	KNN	NC				0.4348
Sparse PCA	Decision Tree	0: 0.94 1: 0.48	0: 0.94 1: 0.49	0: 0.94 1: 0.48	0.90	0.4
Sparse PCA	SVM	0: 0.97 1: 0.16	0: 0.51 1: 0.85	0: 0.67 1: 0.26	0.55	NC
Sparse PCA	Naïve Bayes	0: 0.97 1: 0.16	0: 0.51 1: 0.85	0: 0.67 1: 0.26	0.55	NC
Sparse PCA	Logistic Regression	0: 0.9 1: 1.0	0: 1.0 1: 0.01	0: 0.95 1: 0.03	0.90	NC
Sparse PCA	AdaBoost	0: 0.94 1: 0.66	0: 0.97 1: 0.47	0: 0.96 1: 0.55	0.93	0.4706
Sparse PCA	Random Forest	0: 0.94 1: 0.72	0: 0.98 1: 0.40	0: 0.96 1: 0.51	0.92	0.6842
SMOTE + Sparse PCA	KNN	NC				0.6429
Sparse PCA	Ensemble Classifier KNN + Decision Tree + SVM	0: 0.93 1: 0.79	0: 0.99 1: 0.29	0: 0.96 1: 0.43	0.92	0.4478
Sparse PCA	Ensemble Classifier KNN + Decision Tree + Logistic Regression	NC				0.375
NC: Not Collected						

Conclusion: From the experiments it was found that, dimensionality reduction techniques have significant impact on the performance. Sparse PCA technique gave better results when compared to PCA and LDA, probably because of the sparsity of data. Classifiers like KNN, Decision Tree, Logistic Regression performed better compared to Naïve Bayes and SVM. SMOTE and Ensemble of Classifiers did not improve the performance on this sparse dataset.