MACHINE LEARNING
CSC 8850

FINAL PROJECT REPORT

By

Sai Harika Punyamurthula

Mothi Balaji Srinivasan

Problem 1(Classification):

Techniques: RandomForest, Kernel SVM – RBF

Procedure Outline:

1. Data pre-processing – Eliminated Zero Variance and Near Zero Variance features
2. Feature Selection was done with Boruta – (JMLR Paper).
3. 100 repeats of K-Fold Cross Validation
4. Computed model using Best Parameters.
5. Made Predictions for Test Data.

Parameter tuning for Random Forest:
A grid of Mtry and ntree parameters was constructed using different values and repeat cross-validation was done. Parameters for which the cross validated area under ROC was maximum were selected as the best parameters. The table 1.0 illustrates the best parameters and the cross validated area under ROC. The figure 1.0 illustrates parameter tuning and shows best parameters for dataset 1. Similarly, grid of gamma and Cost parameters were tuned for SVM.

Results:

Algorithm	Data Set	Parameter		Cross Validation AUC
		mtry	ntree	
Random Forest	Data Set 1	15	1000	0.881
	Data Set 2	9	1050	0.787
	Data Set 3	16	1025	0.89

Table 1.0 Random Forest best parameters

Algorithm	Data Set	Parameter		Cross Validation AUC
		Gamma	cost	
Kernel SVM	Data Set 1	0.31	2	0.73
	Data Set 2	1	2	0.667
	Data Set 3	1.5	3	0.60

Table 1.1 SVM best parameters

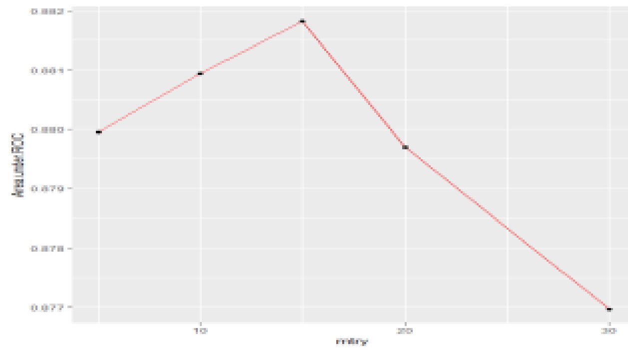

Figure 1.0 Random Forest: Mtry vs Cross-validated Area under ROC

Better performance was achieved for Random Forest. Therefore, Random Forest training model was selected for all three datasets to predict the test labels.

Problem 2(Missing Value Estimation):

Techniques assessed: softImpute – R Package, impute.knn method from Impute package

Soft-Impute is an algorithm developed by Trevor Hastie and Robert Tibshirani that uses convex relaxation techniques to provide a sequence of regularized low-rank solutions for large-scale matrix completion problems. SOFT-IMPUTE iteratively replaces the missing elements with those obtained from a soft-thresholded SVD (Soft-Impute JMLR paper).

We benchmarked our technique on external micro array data as well: lung_data_label.mat, prostate_data_label.mat, srbct_data_label. Table 2.0 shows the results of testing done on different datasets.

Algorithm	**Data Set**	**% of missing and RMSE Error**		
		4%	**10%**	**83%**
Soft Impute	srbct_data_label.mat	0.1451714	0.2231146	0.8455188
Knn Impute	srbct_data_label.mat	0.1459565	0.217029	0.8958327
Soft Impute	prostate_data_label.mat	0.0386309	0.06055933	0.1956854
Knn Impute	prostate_data_label.mat	0.05552823	0.05543156	0.3325293
Soft Impute	lung_data_label.mat	0.05245209	0.08378761	0.2541248
Knn Impute	lung_data_label.mat	0.04666785	0.07329735	0.3559154

Table 2.0 Imputation testing on other micro-array datasets

RMSE calculation:

On the given datasets, some percentage of data was made missing, imputation was done with both knn and softImpute. The RMSE error was calculated for these values. For knn, we calculated error for different k values from 2 to 10. For different datasets, best k was different. But, the rmse error was lesser for softImpute.

Results: SoftImpute was chosen for imputation for all 3 datasets.

Problem 3 (Multi label Classification):

Algorithms Used: RPart, Random Forest.

Problem transformation method we used in our method transforms the multilabel classification into binary classification problem. Algorithm adaptation methods adapt multiclass algorithms so they can be applied directly to the problem.

The following is the step by step process followed for our problem.

1) Creating a task

The first thing we did is to get the data in the right format, then we create a Multilabel Task. Instead of one target name we specified a vector of targets which correspond to the names of logical variables in the data frame.

2) Constructing a learner

We are doing Multi label classification in mlr using Problem transformation method by applying simple binary classification algorithm, this is done by creating a binary (or multiclass) classification learner with makeLearner.

After that we used a function like makeMultilabelBinaryRelevanceWrapper to convert a learner that uses the respective problem transformation method.

3) Train

We train a model as usual with a multilabel learner and a multilabel task as input.

4) Cross Validation

K-fold Cross Validation is done, k=5

5) Predictions for Test data were done with the model with best parameters that were selected through cross validation.

6) Performance

The performance of your prediction can be asessed via function performance. You can specify via the measures argument which measure(s) to calculate. The default measure for multilabel classification is the Hamming loss. Fig. 3.1 shows a graph for cpGrid vs cross-validated Hamming loss. For rPart, Hamming loss of 0.22 was observed and classification accuracy was 94.4%. For Random Forest, Hamming loss of 0.207 was observed but classification accuracy was 86.6%.

Results: Since classification accuracy was better for rPart and Hamming loss was only about 0.02 greater, we chose rPart for predictions.

Algorithm	**Best Parameters**			**Classification Accuracy**
	maxDepth	**cpGrid**	**minSplit**	
RPart	1	0.09	10	94.4%

Table 3.1 rPart best parameters and classification Accuracy

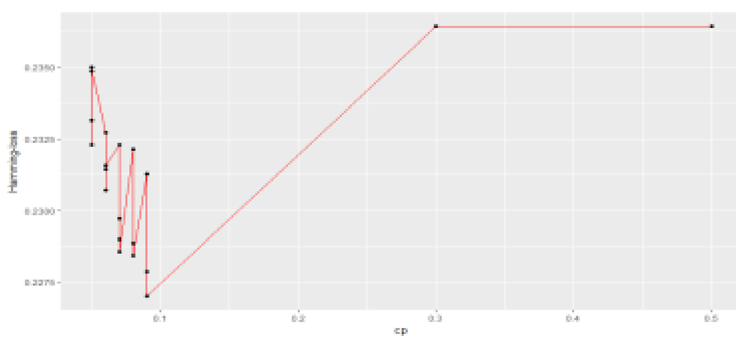

Fig 3.1 cpGrid vs cross-validated Hamming Loss for rPart

Algorithm	Best Parameters		Classification Accuracy
	mtry	ntree	
Random Forest	8	2000	86.6%

Table 3.2 Best parameters and accuracy Random forest

Algorithm	**Best Parameters**		**Classification Accuracy**
	mtry	**ntree**	
Random Forest	8	2000	86.6%

Table 3.3 Best parameters and accuracy for Random Forest

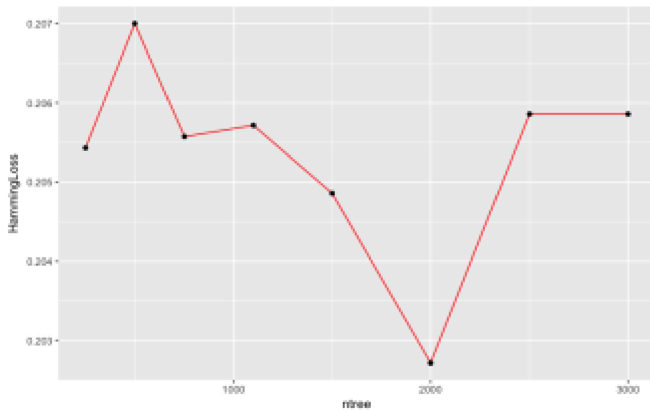

Fig 3.2 ntree vs cross-validated Hamming loss for Random Forest

References:

https://web.stanford.edu/~hastie/Papers/mazumder10a.pdf

https://www.r-bloggers.com/feature-selection-all-relevant-selection-with-the-boruta-package/

https://mlr-org.github.io/mlr-tutorial/release/html/