IE 582

Final Project

Group: Datatata

Merve Nur Karabulut

Süheyla Yıldız

Halenur Kocaman

15.02.2021

1. **Introduction**

The problem of the project is a two-class imbalanced classification problem. The data is given as train and test sets are separated. There are 60 variables in the data that can be used for prediction of variable y. Variable y consists of two factors which are "a" and "b".

In the train set, 1565 of the observations are labelled as "a" while 509 of the variables are in the class "b". The summary of the data shows that eleven of the variable types are numeric and others are factor. There are also variables that have no variability. Since they do not differentiate the model, they are eliminated during the classification process.
The given data has a class imbalance problem, thus models built on this data are biased towards the class having more observations. They have a better performance on predicting the class "a". However, the aim is to provide a model that can predict both classes well. In order to balance those classes and to increase the predictive accuracy on class "b" oversampling is used.

Random forest model is selected as the best performed model. To measure the performance of the models, ROC curve and balanced error rate are used. These measures are suitable when the data is imbalanced. Since this is a multi-objective problem, the aim is to find the pareto optimal point.

2. **Approach**

2.1. **Preprocessing:**

There are two classes for y value and there is a class imbalance problem in the train data. As seen in the below table, there are 1565 instances with class of "a" and 509 instances with class of "b". So, there are approximately three times more instances for the class of "a" and this causes a class imbalance problem.

Class	a	b
# of instances	*1565*	*509*

Most machine learning algorithms have a poor performance, when they are trained with a data set that has an imbalanced class distribution. Also, metrics that are used to evaluate the performance of the algorithm works better with a balanced data.

If it is possible more samples can be collected to overcome the class imbalance problem; but with the given data, this issue can be solved by resampling the data by over-sampling and under-sampling.

We used the "SMOTE: *Synthetic Minority Over-sampling Technique*" function in the train data to overcome the class-imbalance problem. "This function generates new artificial instances of the minority class using the nearest neighbours and uses under-sampling in majority class to have a balanced data set." [2]

After applying the SMOTE function, we got a balanced data set as seen in the below table.

Class	a	b
# of instances	*1018*	*1018*

2.2. Classification Algorithms:

In this project, we try to classify the class information with two possible values of "a" and "b". Tree-based classification algorithms performs better than other approaches for this data set as we have seen in the experiments throughout the project. Mainly, we used "*Decision Tree*" and "*Random Forest*" approaches.

We used parameter-tuning methods to choose the best combination of parameters considering the accuracy performance.

1) 10-fold validation is used, and model is built using these folds.
2) For the Random Forest, *mtry* (the number of variables randomly sampled as candidates at each split) is selected after tuning. (shown in Figure 1)
3) For the Decision Tree, *the minimal number of observations per tree leaf* and *complexity parameter* are tuned.

Random Forest Approach

The definition of random forest approach is given as follows: "A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting." [3]

We made a parameter tuning to select the "mtry" value of the random forest and according to the results, we used "mtry" as 15 in our model. The tuning results are given below in Figure 1:

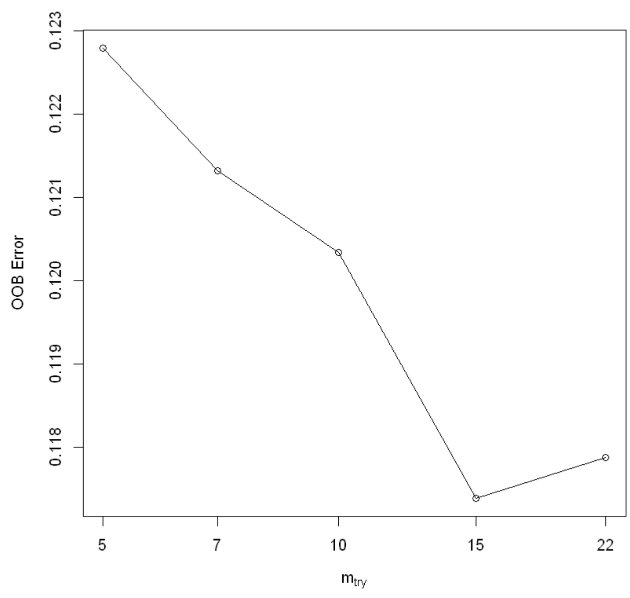


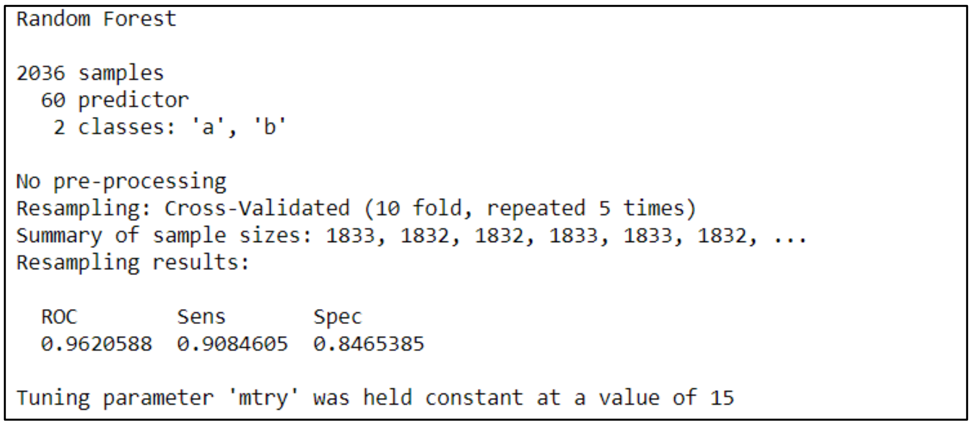
Figure 1*: Tuning result of mtry for random forest*

After pre-processing and tuning processes, the random forest model is built with the following parameters:

- In the “Random Forest” approach, we used the number of trees (J) as 500, mtry as 15.
- 10-fold validation is applied with 5 repeats.
- ROC is used as the performance metric.

3. **Results**:

The output of the random forest approach is given below:



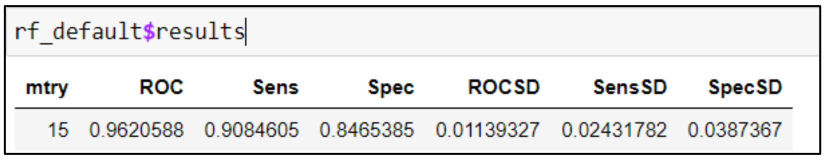
```
Random Forest

2036 samples
  60 predictor
   2 classes: 'a', 'b'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 1833, 1832, 1832, 1833, 1833, 1832, ...
Resampling results:

  ROC        Sens       Spec
  0.9620588  0.9084605  0.8465385

Tuning parameter 'mtry' was held constant at a value of 15
```

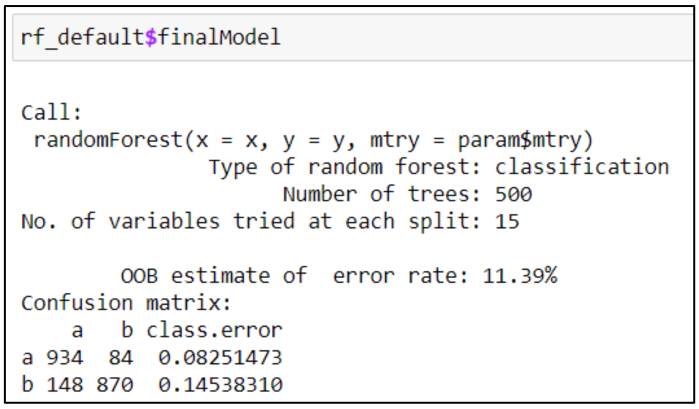


```
rf_default$results
```

mtry	ROC	Sens	Spec	ROCSD	SensSD	SpecSD
15	0.9620588	0.9084605	0.8465385	0.01139327	0.02431782	0.0387367

Using the selected parameters and over-sampled data, we got a random forest model with area under the ROC curve as 0.962 and used this model to predict class values of the test data.

The class error rates of our model are 0.083 for the class of “a” and 0.145 for the class of “b”, results are given below in Figure x. At the beginning, we have also tried to train the model with the given train data but we got higher class error rates due to class imbalance problem.



```
rf_default$finalModel


Call:
 randomForest(x = x, y = y, mtry = param$mtry)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 15

        OOB estimate of  error rate: 11.39%
Confusion matrix:
    a   b class.error
a 934  84  0.08251473
b 148 870  0.14538310
```

4. Conclusions and Future Work:

After several experiments, we concluded that tree-based models give better results compared to other classification methods. Therefore, decision tree and random forest methods were used in the project. One of the key conclusions that we derived during the project is imbalanced class distribution in the training data causes to obtain insufficient models. Over-sampling and under-sampling should be used to get rid of imbalanced class problem.

Decision tree models worked faster than random forest as it can be expected. However, random forest model gave higher area under the ROC curve value. Parameter tuning was applied to select the *mtry* (the number of variables randomly sampled as candidates at each split) value and 10-fold cross validation was performed with 5 repeats.

As a future work, *node size* (minimum size of terminal nodes) and *max nodes* (maximum number of terminal nodes trees in the forest can have) parameters can be tuned for random forest application. Besides that, Support Vector Machine algorithm can be applied for this project as a future work.

5. Code:

IE582-FinalProject

6. References:

[1] https://www.rdocumentation.org/packages/DMwR/versions/0.4.1/topics/SMOTE

[2] https://en.wikipedia.org/wiki/Oversampling_and_undersampling_in_data_analysis#SMOTE

[3] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html