EMORY UNIVERSITY

Machine Learning II

Homework #3 **Due: turned in by Monday 03/23/2020 before class**

Adil Ashish Kumar

(put your name above)

Total grade: ________ out of ___100___ points

Please answer the following questions and submit your assignment as a single PDF file by uploading it to the HW3 drop-box on the course website.

Hands-on predictive modeling (100 points)

Download the dataset on spam vs. non-spam emails from http://archive.ics.uci.edu/ml/datasets/Spambase. Specifically, (i) file "*spambase.data*" contains the actual data, and (ii) files "*spambase.names*" and "*spambase.DOCUMENTATION*" contain the description of the data. This dataset has 4601 records, each record representing a different email message. Each record is described with 58 attributes (indicated in the aforementioned .*names* file): attributes 1-57 represent various content-based characteristics extracted from each email message (related to the frequency of certain words or certain punctuation symbols in a message as well as to the usage of capital letters in a message), and the last attribute represents the class label for each message (spam or non-spam).

Task: The general task for this assignment is to build *two separate models* for detecting spam messages (based on the email characteristics that are given) using RapidMiner or any other tool you prefer (e.g., Python, Spark, etc.):

1. [Start with this task] The best possible model that you can build in terms of the *overall predictive accuracy*;
2. The best cost-sensitive classification model that you can build in terms of the *average misclassification cost*.

Some specific instructions for your assignment/write-up:

- Reading the data into RapidMiner (or your software of choice): Data is in a comma-separated-values (CSV) format. You may also want to add the attribute names (which are in *spambase.names* file) to the data file as the first line. It might be easiest to read data into Excel, save it as Excel file, and then import it to RapidMiner.
- Exploration: Make sure to explore multiple classification techniques. You should definitely consider decision trees, *k*-nearest-neighbors, and Naïve Bayes, but you are free to experiment with any other classification techniques you know (for example, you can try applying some meta-modeling techniques, etc.).
    - Also, explore different configurations of each technique (for example, you should try varying some key parameters, such as values of *k* for *k*-NN, etc.) to find which configurations work best for this application. You can use parameter optimization approaches to help you with that.
- Make sure to explore the impact of various data pre-processing techniques, especially normalization and attribute selection.
- When building cost-sensitive prediction models, use 10:1 cost ratio for different misclassification errors. (I think it should be pretty clear which of the two errors –

classifying a non-spam message as spam vs. classifying a spam message as non-spam – is the costlier one in this scenario.)

- In general, use best practices when evaluating the models: evaluate on validation/test data, discuss the confusion matrix and some relevant performance metrics (not just the required accuracy and average misclassification cost, but also precision, recall, f-measure – especially for your best/final models…), show some visual indications of model performance (such as ROC curves).
- Finally, as a deliverable, produce a write-up (i.e., a single PDF file) describing your aforementioned explorations. Report the performances of different models that you tried (i.e., using different data mining techniques, different attribute selection techniques, etc.) What was the performance of the best model in the cost-unaware task (i.e., in terms of accuracy)? What was the performance of the best model in the cost-aware task (i.e., in terms of expected cost)? Discuss the best models in two different tasks (as well as their performance) in detail, provide some comparisons. Draw some conclusions from the assignment.

Evaluation: 100 points total.

- Performance: 35 points (based on the performance achieved by your best reported models).
- Exploration/write-up: 65 points (based on the comprehensiveness of your exploration, i.e., when searching for the best performing model, did you evaluate and report just one or two techniques, or did you try a number of different variations, based on what you know from the class?).

After reading the data into excel and adding in the headers, I saved the file as a CSV and read it into python. Data was split in ratio 70:30 for train and test respectively.

Cost Unaware task

1) Decision Tree
   As a first step I ran a Decision tree model which was trained and validated on an inner and outer cross validation (5 fold validation for inner and outer cv) on training data. I ran a grid search to find the optimum values for parameters max depth, information gain criteria, min leaf samples and min split samples. This method had a nested CV accuracy score of 90.77%

   I then ran a decision tree model with optimum parameters from above model on test data and the accuracy score was 92.1%

2) KNN

Since KNN requires data to be normalized, I first normalized the X variables using the standard scaler function. I then ran a KNN model with default as p=2 to use the Euclidean distance metric to find nearest neighbors. Model was trained and validated on an inner and outer cross validation (5 fold validation for inner and outer cv) on training data. I ran a grid search to find the optimum values for parameters no of neighbors and weights(distance,uniform). This method had a nested CV accuracy score of 90.86%

I then ran a KNN model with optimum parameters from above model on test data and the accuracy score was 92.46%

3) Naïve Bayes

For Naïve Bayes, I used the multinomial Naïve Bayes model since it was more suited to word count type features, which is similar to the application we are dealing with. Model was trained and validated on an inner and outer cross validation (5 fold validation for inner and outer cv) on training data. I ran a grid search to find the optimum values for parameters alpha(smoothing factor) and fit_prior(Whether to learn class prior probabilities or not). This method had a nested CV accuracy score of 78.9%

I then ran a Multinomial Naïve Bayes model with optimum parameters from above model on test data and the accuracy score was 81.1%

4) Random Forest

I ran a Random forest model which was trained and validated on an inner and outer cross validation (5 fold validation for inner and outer cv) on training data. I ran a grid search to find the optimum values for parameters no of estimators, max depth, min impurity decrease and max features. This method had a nested CV accuracy score of 93.35%

I then ran a decision tree model with optimum parameters from above model on test data and the accuracy score was 94.1%

Based on the above, Random forest had the best accuracy. I decided to proceed with Random forest model and see if I could improve the accuracy by applying data processing techniques like Feature selection and normalization

5) Random Forest + Normalization

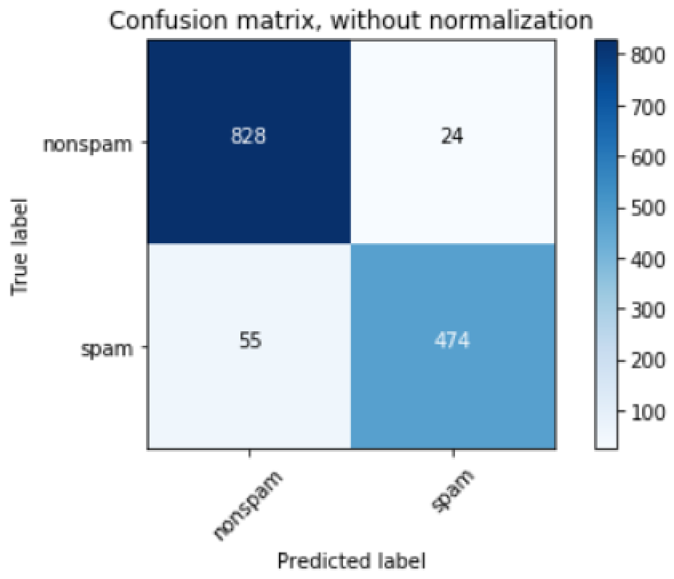
I ran a random forest model with optimum parameters from previous grid search on data that was normalized using standard scaler. The random forest model has a accuracy of 94.1% on test data, which was same as random forest model on data without normalization. It seems like normalization did not have any effect on accuracy, so I decide to exclude it for the next step

6) Random Forest + Feature Selection

For feature selection, I decided to try the Recursive feature elimination (RFE) method. This method recursively removes attributes and builds a model on the features that remain. Using RFE, the no of features was reduced to 54. I then ran the random forest model with optimum parameters on the reduced feature set. The model had an accuracy of 94.2% on test data. This is a slight increase in accuracy to our previous random forest accuracy.

Overall, considering multiple modeling methods like KNN, Decision Tree, Naïve Bayes and Random Forest, the model with best accuracy is Random forest on non normalized data with RFE applied on the data. The final model had an accuracy of 94.2% on test data.

The confusion matrix for the model is below. We see that the model makes more errors while classifying spam as non spam compared to no spam as spam.



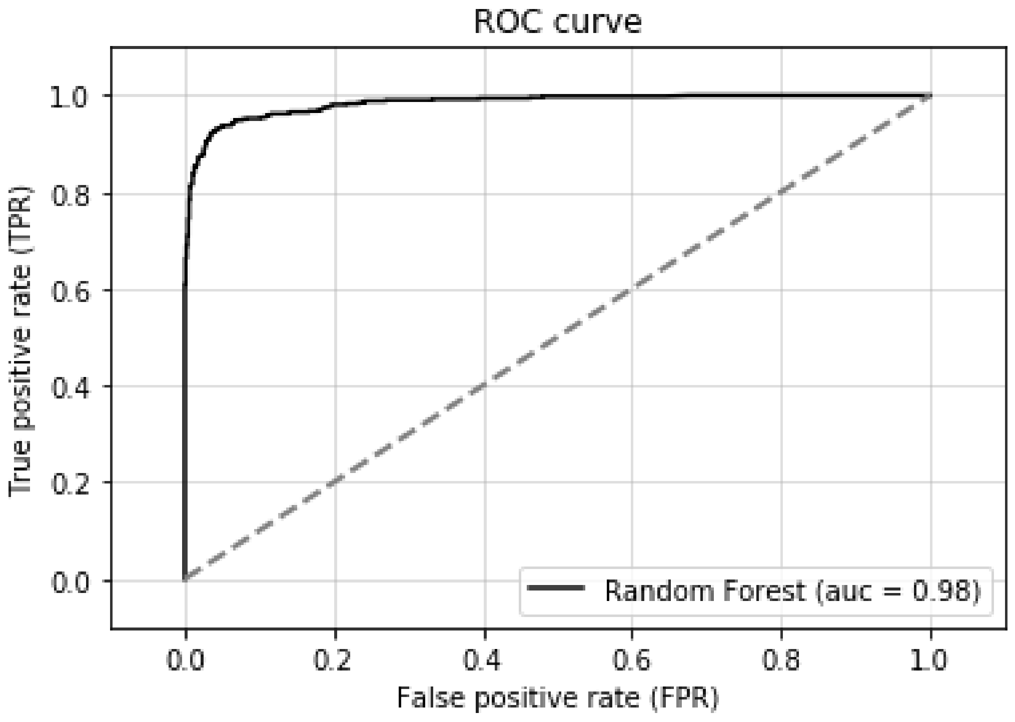
Below we have the precision, recall and F1 scores for the final model. The F1 scores indicate that the model has better performance in predicting non spam vs predicting spam.

```
              precision    recall  f1-score   support

           0       0.94      0.97      0.95       852
           1       0.95      0.90      0.92       529

   micro avg       0.94      0.94      0.94      1381
   macro avg       0.94      0.93      0.94      1381
weighted avg       0.94      0.94      0.94      1381
```

Below is the ROC curve for our Random forest model. It has a high AUC of 0.98, indicating that the model has good performance in precision and recall metrics



Cost Aware Task

Since we need to take into consideration the cost ratio of 10:1, I first setup a cost matrix based on length of y variable in train and test. The assumption is that it is more costly to classify a non spam mail as spam, because an important email could be categorized as spam, hence the cost matrix is setup accordingly.

1) Decision Tree
   I used the Cost Sensitive Decision Tree Classifier implementation in python to fit a decision tree that incorporates our cost matrix. After fitting the model, I calculated the savings score using the predictions on test data and cost matrix. I then ran a loop to pick the model with lowest cost by iterating over different parameter values for max depth and min gain. The lowest cost decision tree model had a cost of 0.3

2) Random Forest
   I used the Cost Sensitive Random Forest Classifier implementation in python to fit a random forest model that incorporates our cost matrix. After fitting the model, I calculated the savings score using the predictions on test data and cost matrix. I then ran a loop to pick the model with lowest cost by iterating over different parameter values for no of estimators and max features(none, sqrt,log2). The lowest cost Random forest model had a cost of 0.035

3) Bagging
   I used the Cost Sensitive Bagging Classifier implementation in python to fit a model that incorporates our cost matrix. After fitting the model, I calculated the savings score using the predictions on test data and cost matrix. I then ran a loop to pick the model with lowest cost by iterating over different parameter values for no of estimators and max samples. The lowest cost model had a cost of 0.31

   Based on the above, Random forest had the best cost aware performance. I decided to proceed with Random forest model and see if I could improve the cost aware performance by applying data processing techniques like Feature selection and normalization.

4) Random forest + Normalization

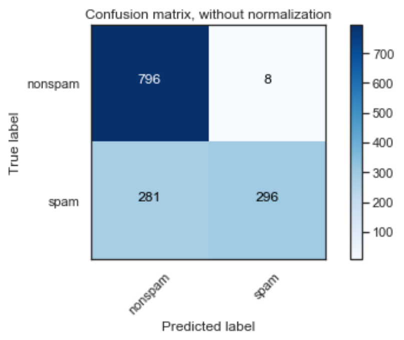
   I ran a cost sensitive random forest model on data that was normalized using standard scaler. The random forest model had a cost of 0.035, which was same as cost sensitive random forest model on data without normalization. It seems like normalization did not have any effect on accuracy, so I decide to exclude it for the next step

5) Random forest model + RFE

   I ran a cost sensitive random forest model on data which had RFE applied on it. The RFE applied dataset had no of variables reduced to 54. The cost sensitive random forest model had a cost of 0.031 which was slightly better than cost sensitive random forest model on non normalized data. I decide to use this as my final cost sensitive model since it has the lowest cost.

Overall, considering multiple modeling methods like Bagging, Decision Tree and Random Forest, the model with lowest cost is Random forest on non normalized data with RFE applied on the data. The final model had a cost of 0.031.

The confusion matrix for the final model is below. We see that the model tries to minimize predicting non spam as spam mails. Obviously this will increase the other type of error, ie classifying spam as non spam, but because of the way the cost matrix works, this is acceptable for our application



Below is the precision, recall and F1 measure for our final model. We see very high recall for non spam and lower recall for spam, which is in accordance with our cost matrix to reduce cost.

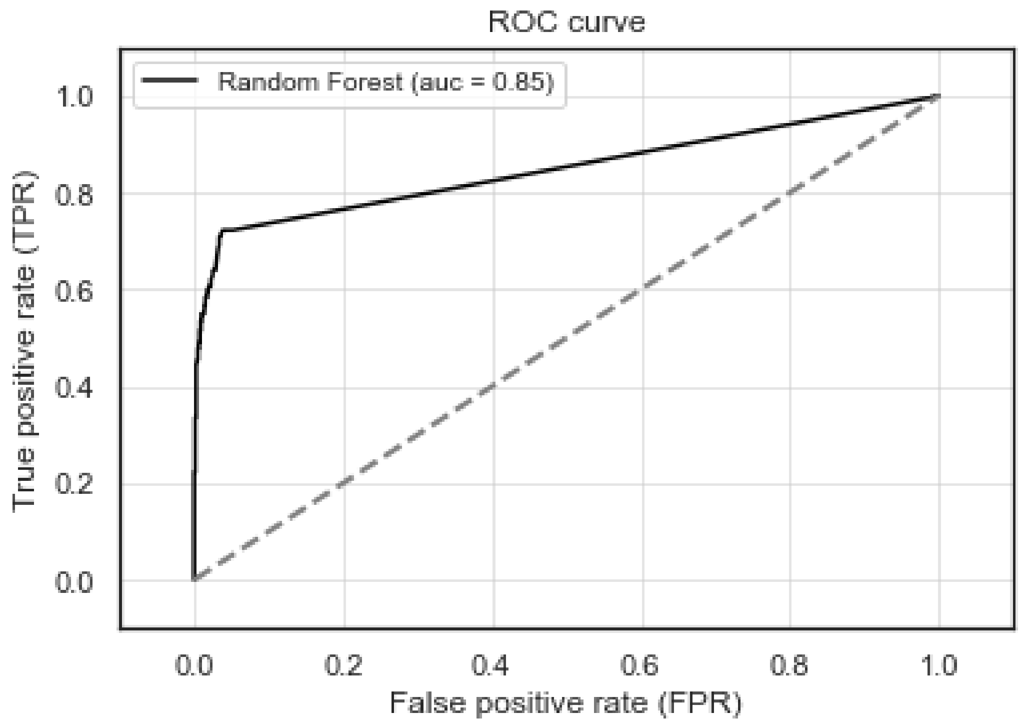
```
              precision    recall  f1-score   support

     nonspam       0.74      0.99      0.85       804
        spam       0.97      0.51      0.67       577

    accuracy                           0.79      1381
   macro avg       0.86      0.75      0.76      1381
weighted avg       0.84      0.79      0.77      1381

Accuracy: 0.79073
F1 score: 0.75916
```

Below is the ROC curve for our final model. The model has an auc of 0.85, which is pretty good considering that there will be more errors in predicting spam as non spam.


ROC curve
Random Forest (auc = 0.85)
True positive rate (TPR)
False positive rate (FPR)