Homework #3 **Due: turned in by Monday 02/19/2018 before class**

__Chengyao (Leo) Ma__

(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?).

Report

Chengyao (Leo) Ma

February 19, 2018

1 Overview

The dataset is consisting of 4601 rows and 58 columns. While the target value, column no.58 is a categorical value (1 stands for spam email and 0 stands for non-spam email), the other 57 columns are all continuous values measuring value of different metrics of email.

2 Data Preprocessing

2.1 Standardization

Given all the attributes here are continuous values, we consider using normalization to make our data consistent and clear. To verify that standardization would improve the accuracy of model, we make a simple comparison to train the same logistic regression model with raw data and standardized data.

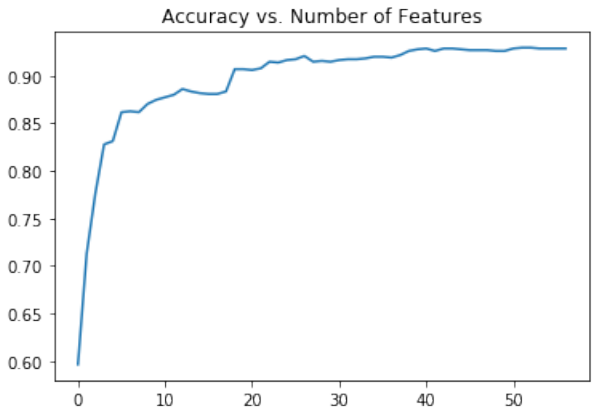
$$Accuracy_{raw\_data} = 0.92696$$

$$Accuracy_{standardized\_data} = 0.92870$$

So we are going to use standardized data for the following analysis.

2.2 Feature Selection

We use recursive feature elimination(RFE) to test the effect of feature selection. RFE recursively removing attributes and building a model on the attributes remain. We apply RFE to logistic regression. The following plot shows that feature selection does not improve the accuracy of logistic regression.
So we are not going to use feature selection for the following analysis.

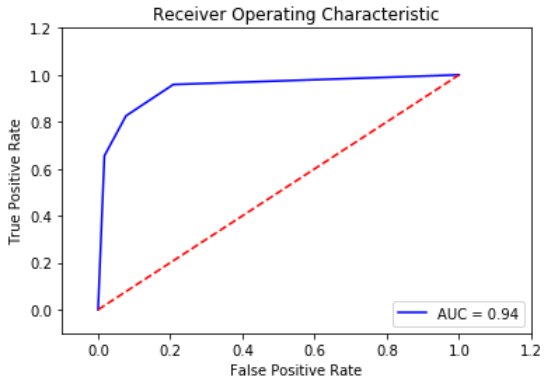


3 Models Building for Overall Predictive Accuracy

3.1 KNN

The first model we train is KNN. We use grid search to find the best parameter *k* out of [1,30] which is 3. The performance is :

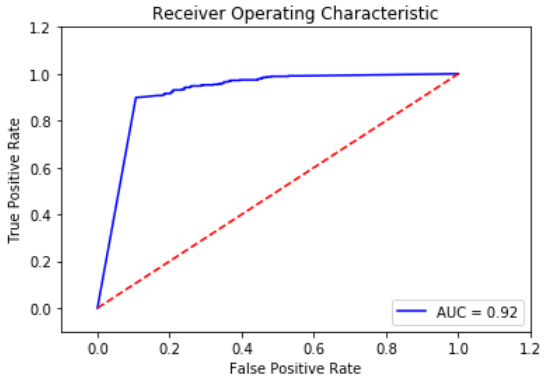
```
accuracy_score = 0.88347826087
precision_score = 0.878440366972
recall_score = 0.825431034483
f_measure = 0.851111111111
```



3.2 Naive Bayesian

The nave Bayesian model we are going to train is the simplest Gaussian Nave Bayes algorithm. For this model, there is no parameter to tune and the performance is:

```
accuracy_score = 0.814782608696
precision_score = 0.699523052464
recall_score = 0.948275862069
f_measure = 0.805123513266
```



3.3 Decision Tree

The parameters we tune for decision tree model includes:

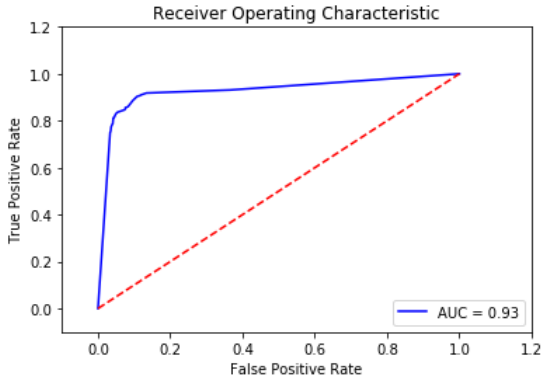
Parameters	Interval
Max Depth	[5,10,20,30,40,50,60,70,80,90,100,None]
Max Features	['auto','sqrt','log2']
Min Samples Leaf	[1,2,4]
Min Samples Split	[2,5,10]

The best parameters by grid search is :

$$MaxDepth = 20$$
$$MaxFeatures =' auto'$$
$$MinSamplesLeaf = 1$$
$$MinSamplesSplit = 10$$

The performance of this best decision tree model is:

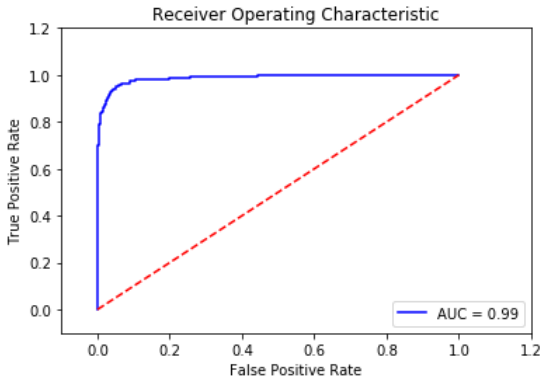
```
accuracy_score = 0.901739130435
precision_score = 0.912941176471
recall_score = 0.836206896552
f_measure = 0.872890888639
```



3.4 AdaBoost

We use boost as an example of ensemble method. The specific technique is AdaBoost. It uses the best decision tree we trained previously as the base classifier. The iterative training process constantly fits into the training data by increasing relative weight of misclassified training instances. The learning rate of this model is 0.5. The performance is:

```
accuracy_score = 0.941739130435
precision_score = 0.950113378685
recall_score = 0.903017241379
f_measure = 0.925966850829
```



3.5 Random Forest

Random Forest is another ensemble model we used. The parameters to be tuned includes:

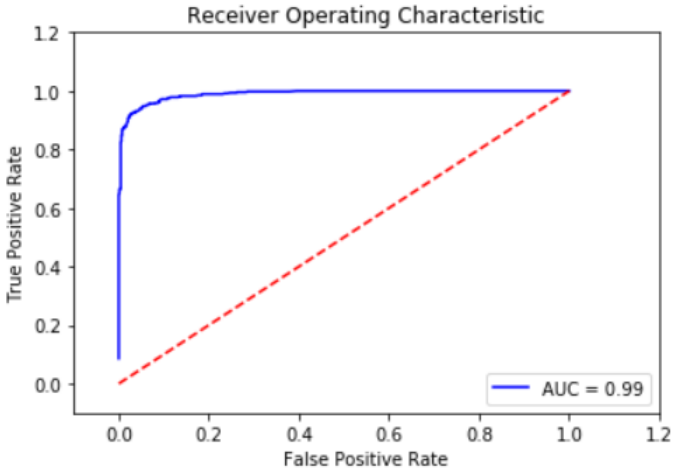
Parameters	Interval
Max Depth	[5,10,20,30,40,50,60,70,80,90,100,None]
Max Features	['auto','sqrt']
Min Samples Leaf	[1,2,4]
Min Samples Split	[2,5,10]
Number of Estimators	[200,400,600,800,1000,1200,1400,1600,1700,2000]

The best parameters by grid search is :

$$MaxDepth = 100$$
$$MaxFeatures =' sqrt'$$
$$MinSamplesLeaf = 1$$
$$MinSamplesSplit = 2$$
$$NumberofEstimators = 400$$

The performance of the best random forest model is :

```
accuracy_score = 0.951304347826
precision_score = 0.957399103139
recall_score = 0.92025862069
f_measure = 0.938461538462
```



3.6 Summary

Based on the measures of accuracy score, precision, recall, f measure and AUC score, the model overall predictive accuracy is random forest.

4 Models Building for Cost-sensitive Classification

4.1 Cost Function

When doing spam email classification, classifying non-spam email as spam email is far more costlier, cause it will create great negative impact on user experience. Hence, we set the cost of classifying non-spam email as spam email as 10/11, and the cost of classifying spam email as non-spam email as 1/11.

We build the self-defined scorer function in Python by:

The basic model is Logistic Regression, and the corresponding cost = 0.25565

```
def cost_calc(Y_true, Y_pred):
    cost = 0
    for i in range(len(Y_pred)):
        if list(Y_true)[i] == 1 and list(Y_pred)[i] == 0:
            cost += 1
        if list(Y_true)[i] == 0 and list(Y_pred)[i] == 1:
            cost += 10
    return cost/len(Y_pred)
cost = make_scorer(cost_calc, greater_is_better = False)
```

4.2 Decision Tree

The KNN and GausianNB methods in scikit-learn do not support class_weight as paratmeres, so we mainly use Grid Search to train the best cost-sensitive Regression Tree model.

The best parameters by grid search is :

$$MaxDepth = 5$$
$$MaxFeatures =' auto'$$
$$MinSamplesLeaf = 1$$
$$MinSamplesSplit = 10$$

The cost of this best decision tree model = 0.18956. The AdaBoost does not contribute to improve model in this situation.

4.3 Summary

The best cost-sensitive classification model is Decision Tree with an average misclassification cost of 0.18956