Biostat 626 Midterm 1

Jamie Ki

1. Set up a Github repository and upload all your code used for training, evaluation, and generating results of test data. Provide the url to your Github repository as the answer to this question.

https://github.com/jamkiiiii/626midterm1.git

2. Write a text file (name the file "README.md") to provide necessary instructions, so that other people can reproduce all your results.

Created

3. Describe your baseline algorithm and provide necessary tables and/or figures to summarize its performance based on the training data.

The baseline algorithm used is random forest. The package "randomForest" was used to build the algorithm. The following table summarizes the performance accuracies of random forest in the binary and multi-class classification.

	Binary Classification	Multi-class Classification
Random Forest	0.9996	0.9793

The following is the screenshot of the outcome of the random forest algorithm for binary classification:

```
Call:
 randomForest(formula = as.factor(results) ~ ., data = train_sub[,      3:564])
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 23

        OOB estimate of  error rate: 0.09%
Confusion matrix:
     0    1  class.error
0 3109    3 0.0009640103
1    2 2288 0.0008733624
Confusion Matrix and Statistics


pred    0    1
   0 1368    0
   1    1  996

               Accuracy : 0.9996
                 95% CI : (0.9976, 1)
    No Information Rate : 0.5789
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.9991
```

```
Mcnemar's Test P-Value : 1

            Sensitivity : 0.9993
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9990
             Prevalence : 0.5789
         Detection Rate : 0.5784
   Detection Prevalence : 0.5784
      Balanced Accuracy : 0.9996

       'Positive' Class : 0
```

Now, the following is the screenshot of the result for the multi-class classification for random forest.

```
Call:
 randomForest(formula = as.factor(train_sub$activity) ~ ., data = train_sub[,      3:563], importance = TRUE,
proximity = TRUE)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 23

        OOB estimate of  error rate: 2.19%
Confusion matrix:
    1   2   3   4   5   6   7 class.error
1 849   5   7   0   0   0   0  0.01393728
2   3 740   6   0   0   0   0  0.01201602
3   5   3 682   0   0   0   0  0.01159420
4   0   0   0 847  46   0   5  0.05679287
5   0   0   0  25 981   0   0  0.02485089
6   0   0   0   0   0 985   6  0.00605449
7   2   0   1   1   1   3 242  0.03200000
Confusion Matrix and Statistics


pred   1   2   3   4   5   6   7
   1 361   0   1   0   0   0   0
   2   3 323   5   0   0   0   2
   3   1   1 291   0   0   0   0
   4   0   0   0 375  12   0   0
   5   0   0   0  19 405   0   0
   6   0   0   0   0   0 419   0
   7   0   0   0   1   0   3 100
```

```
Overall Statistics

               Accuracy : 0.9793
                 95% CI : (0.9727, 0.9847)
    No Information Rate : 0.1817
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9755

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6 Class: 7
Sensitivity            0.9890   0.9969   0.9798   0.9494   0.9712   0.9929  0.98039
Specificity            0.9995   0.9950   0.9990   0.9938   0.9900   1.0000  0.99820
Pos Pred Value         0.9972   0.9700   0.9932   0.9690   0.9552   1.0000  0.96154
Neg Pred Value         0.9980   0.9995   0.9970   0.9897   0.9937   0.9984  0.99910
Prevalence             0.1572   0.1395   0.1279   0.1701   0.1796   0.1817  0.04393
Detection Rate         0.1555   0.1391   0.1253   0.1615   0.1744   0.1804  0.04307
Detection Prevalence   0.1559   0.1434   0.1262   0.1667   0.1826   0.1804  0.04479
Balanced Accuracy      0.9943   0.9960   0.9894   0.9716   0.9806   0.9964  0.98930
```

4. Describe your final algorithm and provide necessary tables and/or figures to summarize its performance based on the training data.

The final algorithm used is SVM. The following table is the accuracy for the binary and multi-class classification for SVM algorithm.

	Binary Classification	Multi-class Classification
SVM	0.9991349	0.9741379

5. Use a figure or a table to show your leaderboard performance. Describe your efforts to improve the performance.

	Binary classification	Multi-Class classification
Random Forest	0.998	0.936
SVM	0.999	0.952

The first algorithm used on this midterm was random forest. After viewing my first leaderboard results, I tried tuning the random forest in order to increase the performance accuracy. However, the accuracy I had as my result on my laptop did not improve.

Then, I tried the KNN algorithm for both binary classification and multi-class classification. However, the accuracy was lower than 0.95. Therefore, instead of using this algorithm to submit on the leaderboard, I decided to try other algorithms that would have a higher accuracy.

Lastly, the SVM algorithm was used to improve the performance. The accuracy on my laptop was higher than the random forest for binary classification and was very similar for multi-class classification. Therefore, I submitted my result to the leaderboard this time, and the result was higher than the random forest algorithm, which is 0.999 for binary classification and 0.952 for

multi-class classification. Thus, since the SVM algorithm outperformed the random forest algorithm, the SVM algorithm is my final algorithm chosen for this midterm.

6. Comment on your final results and potential ways to further improve the classification accuracy.

The final results for my final algorithm are pretty high, but there were many people with a higher accuracy.

For the binary classification, on the leaderboard, there were students with 1.000 accuracy. Therefore, potentially, there could be other algorithms that could outperform the SVM algorithm. Or, tuning the SVM algorithm could also increase the accuracy and performance.

Now, for multi-class classification, there were pretty high accuracies on the leaderboard, such as 0.982, but no one with a 1.000 accuracy. Therefore, more processes might be needed to increase the accuracy for the multi-class classification. Similarly, using different algorithms could increase the accuracy, or tuning the parameters could also enhance the performance.