Report for project 1 of MSBD6000B

Training data:

traindata.csv: 3220 data points with 57 continuous features

trainlabel.csv: class label (0 or 1) for the 3220 training data points

Testing data:

testdata.csv: 1380 data points with 57 continuous features

Tool/package used:

scikit-learn module in python 3

General flow of building the classifier:

1. Load the training data and label
2. Perform pre-processing for the training data
3. Evaluate the model with cross validation/metrics
4. Train the final model with all training data
5. Use the final model to predict labels for testing data

Implementation:

For pre-processing:

Perform normalization for the training data. For example, for each attribute, the value x_i should be adjusted to be (x_i - mean_of_the_attribute)/(standard deviation of the attribute). It can be done with following code:

```
mean_train = numpy.mean(features, axis=0)
std_train = numpy.std(features, axis=0)
features -= mean_train
features /= std_train
```

For model selection and evaluation:

The metrics that are used to evaluate a model are confusion matrix and roc auc score, which can show not only the training error but also potential problems in false positive and false negative. Cross validation (k=5) is also used.

I have tried several models, including logistic regression, SVM, decision trees with random forest. The final choice is random forest, as it gives the overall best performance in metrics. It is not surprising as random forest is an ensemble learning method which fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and controls over-fitting.

In this project, the number of estimators is set to be 20 in random forest.