'Mark Watney'
April 29th, 2016
CS 514 Project 5: Santander Bank Customer Satisfaction Machine Learning Project

I decided to approach this project using the popular Python based libraries Pandas, for data processing, and Scikit-Learn, for applying machine learning algorithms. The first step I took was preprocessing the training dataset available on Kaggle. I loaded the training csv file into a Pandas data frame. As suggested in the Kaggle competition forums, I looked for features that were duplicates of other features in the dataset - this was achieved by comparing columns in the csv file to other columns. When duplicate features were found, I removed the features from the Pandas data frame. I also checked for features that had a constant value throughout the dataset. When features with a constant value were found, they were also removed from the Pandas data frame. These steps can be seen in my TrainingDataExtractor class in data_extractor.py.

At this point, I decided to train some machine learning classifiers from SciKit-Learn with default parameters to determine which classification models work best out-of-the-box with this dataset. I chose to withhold 25% of the training data as a validation set rather than using k-folds cross-validation at this point to save time. The metric I chose to gauge the success of a given machine learning classifier was Area under a ROC curve as this was the metric discussed in class and what the associated Kaggle competition is judged by. The preliminary training code associated with these steps can by found in preliminary_classification_test.py. The results are as follows:

Table 1: Preliminary Results with Santander Bank Data

Machine Learning Algorithm / Model	Area Under ROC Result
Gradient Boosted Classifier	0.827
Random Forest Classifier	0.657
Ada Boosted Classifier	0.824
Logistics Regression Classifier	0.609

The Gradient Boosted Classifier performed the best on the dataset so I decided I would go forward with the Gradient Boosted Classifier model trying to maximize Area under the ROC curve. Also, because I read on the competition forums that the Random Forest Classifier can perform well if its default parameters are adjusted, I also decided to try to improve the results of the Random Forest Classifier.

My first attempt at improving my preliminary results was by using feature selection algorithms to remove features that had little correlation to the dataset labels. The feature selection algorithms I chose from SciKit-Learn were Recursive Feature Elimination (RFE) and SelectFPR. However, removing features based off the results of RFE and SelectFPR did not improve the results of the Gradient Boosted Classifier

and Random Forest Classifier. My attempts at feature selection can be seen in both feature_selection.py and feature_selection_2.py. I did not include the results of feature selection in this report as it did improve my results.

Given that feature selection did not improve my results, I decided to focus on changing the parameters of the classification models to maximize their performance. From reading the Kaggle competition forums, I decided to focus on the parameters n_estimators, max_depth, and max_features for the Random Forest Classifier and n_estimators for the Gradient Boosted Classifier. The code for these parameter tunings can be found in randomforest_tuning.py and gradientboosting_tuning.py. The following tables show the progression of parameter values I decided to try for each of the classification models:

Table 2: Changing Random Forest Classifier parameters and resulting area under ROC curve

n_estimators	max_depth	max_features	Area under ROC
300	10	0.4	0.852
300	15	0.4	0.848
300	15	0.6	0.842
400	15	0.5	0.845
500	15	0.5	0.841
500	10	0.4	0.852

Table 3: Changing Gradient Boosted Classifier parameters and resulting area under ROC curve

n_estimators	Area under ROC
100	0.830
300	0.831
500	0.829
750	0.833

Based off these results, I decided my final classifier would be a Random Forest Classifier with n_estimators at 500, max_depth at 10, and max_features at 0.4. Up to this point, Area under the ROC curve was measure with a 25% withdrawn validation set. To get a better estimation on how my final classifier would perform on the testing dataset, I decided that k-folds cross-validation was needed. I decided on 6 folds. The code can be found in finalmodel_results.py. The results are as follows:

Table 4: Final Results

	Area under ROC
Final Classifier	.83785

Because the Kaggle competition just closed to new competitors when I performed this work, I was not able to test my final model on the testing dataset for a score. However, assuming that the testing dataset was drawn from the same distribution of data that the training dataset was drawn from, the 6-folds cross-validation results gives a good estimate on how I would have preformed in the competition. Based off the public leader board when I wrote this report, I estimate I would have finished at position 2,907 out of 5,236 teams (roughly at the 45th percentile).

Table 5: Snapshot of where I estimate I would have finished with my final classifier

2905	↑549	Marcin W.	0.836856	16	Thu, 28 Apr 2016 22:30:28 (-24h)
2906	new	Dmytro Karabash	0.836851	8	Sun, 24 Apr 2016 07:49:32
2907	↓444	Vijayan Nagarajan	0.836849	14	Sun, 27 Mar 2016 00:06:59
2908	↓444	mtakushi	0.836839	19	Tue, 29 Mar 2016 04:43:56 (-14.8h)
2909	↑694	FutureDataScientist	0.836836	17	Mon, 25 Apr 2016 16:41:00 (-0.3h)