Cyber Data Analytics

Menno Bezema	4248252
Jonathan Raes	4300343

CDA - CS4035
30 April, 2019

Delft University of Technology

Computer Science

May 10, 2019

1 | Lab 1

This report is structured based on the project description. Starting with the Visualization Task, followed by the Imbalance Task and ending in the Classifciation Task. The report is complemented by the Classification.ipynb file which can be opened using jupyter notebook and which shows all steps of the assignment. All used code can be retrieved from:

https://github.com/mennobo/CS4035_CDA_3/tree/master/Assignment1

If you're having trouble executing the code please consult Appendix 1.4.1

1.1 Visualization Task

Load the fraud data into your favorite analysis platform (R, Matlab, Python, Weka, KNIME, ...) and make a visualization showing an interesting relationship in the data when comparing the fraudulent from the benign credit card transactions. You may use any visualization method such as a Heat map, a Scatter plot, a Bar chart, a set of Box plots, etc. as long as they show all data points in one figure. What feature(s) and relation to show is entirely up to you. Describe the plot briefly
Before starting visualization the data had to be preprocessed. The preprocessing pipeline can be seen in the supplied code starting from the The preprocessing pipeline consists of

- Converting the datetime values so pandas understands them
- Feature hashing the categorical data
- Filtering out refused transactions
- Creating a boolean is_faud colum for clearity
- Replacing cardverificationcodesupplied NaN values with False
- Adding an amount_convert column with the amount converted to GBP
- Replacing NaN values in issuercountrycode and shoppercountrycode
- Mapping cvc response code to 0,1,2,3
- Adding a column spent_in_issuer_country with boolean values indicating wether issuercountrycode equals shoppercountrycode. The rationale behind this is that if a transactions was made in another country then the issuer of the credit-card it may be an indicator of fraud.

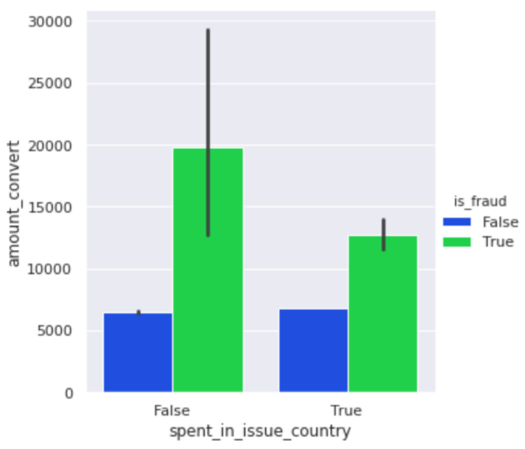


Figure 1.1: Amount of fraud committed in other countries than card issuer

After preprocessing there are 267 features left as well as the is_fraud variable which is our dependent variable or the variable we want to predict. Also, there are no more missing values in the data-set after pre processing.
The visualization task can be seen from the Preprocessing header [1]. The first graph used displays the occurrence of fraud against the amount spent and can be seen in Figure 1.14. It is interesting to see that the average amount spent in fraudulent transactions

[1] All visualizations used to answer this assignments are left in the code, the cells used for this assignment are referenced in the header.

is larger than on normal transactions with 13068.41 and 6.765,74 GBP spent on average respectively. The median values of 8110 and 5795 for fraud vs non-fraud and an std of 11592 and 6008 indicate that the fraudulent group mainly has outliers that are more significant. This data suggests that amount may be a decent predictor for fraud, but mainly for very high amount transactions. Important to note though is that the maximum transacted amount for legitimate transactions is much higher with 2.989,00 GBP for legitimate and 674,54 GBP for fraudulent transactions. So this relation may not hold anymore for very-high-amount transactions.

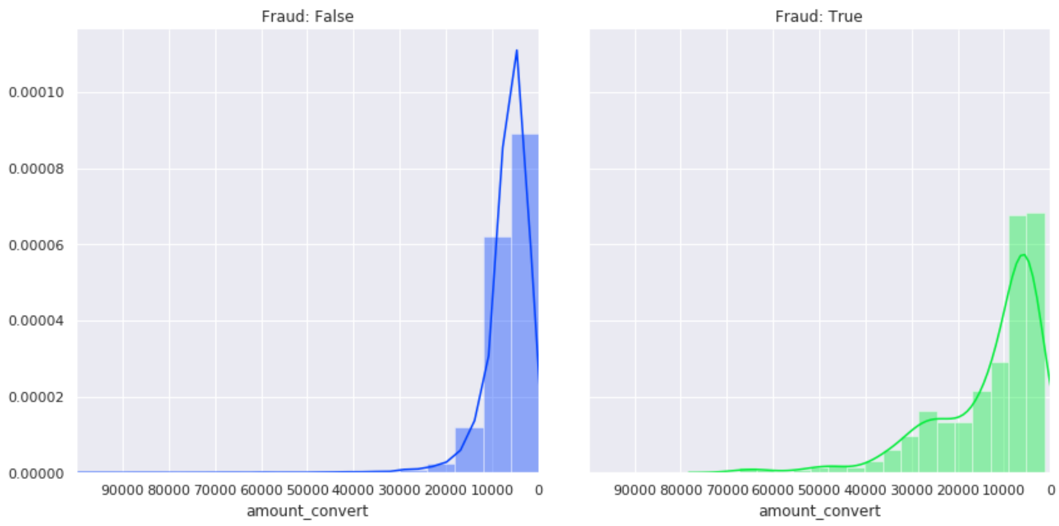
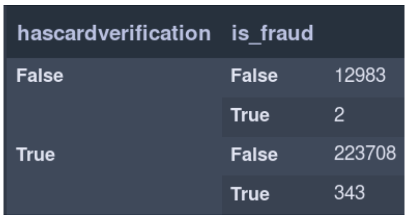


Figure 1.2: Fraud vs Amount spent

Interesting to point out is that the cardverificationcodesupplied column is a boolean column but has missing values. If we add a column called hascardverification and give it a value of false if the corresponding cardverificationcodesupplied has a null value, and true otherwise, and then group and count the values we see the following result:



hascardverification	is_fraud	
False	False	12983
	True	2
True	False	223708
	True	343

Figure 1.3: hascardverification counts

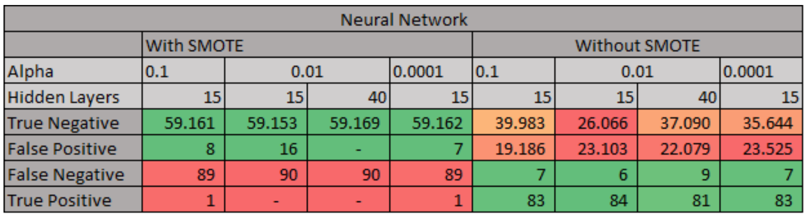
If no card verification code was supplied we see only 2 cases of fraud. However since we do not know what it means if this column has no value, it is unlikely this could be used as a predictor in real life.

1.2 Imbalance Task

Process the data such that you can apply SMOTE to it. SMOTE is included in most analysis platforms, if not you can write the scripts for it yourself. Analyze the performance of at least three classifiers on the SMOTEd and UNSMOTEd data using ROC analysis. Provide the obtained ROC curves and explain which method performs best. Is using SMOTE a good idea? Why (not)
To perform this task we set up the classifiers and then ran them using SMOTEd data and unSMOTEd data, recording the performance per classifier for each version. Below is a short description of the results we found. We have added the confusion matrix values to the description. In the appendices we have added the ROC curves per classifier.

- Random Forest
  For random forests SMOTE seems to offer an improvement to the classifier. Running the classifier on un-smoted data gets very bad results.
  - With SMOTE: True Neg: 54462, False pos: 4707, False neg: 42, True pos: 48
  - Without SMOTE: True Neg: 59158, False pos: 11, False neg: 89, True pos: 1
- Logistic Regression
  - With SMOTE, best case: 10074 False positives and 81 True positives
  - Without SMOTE, best case: 0 False positives and 0 True positives

  Logistic Regression works better with SMOTEd data. UnSMOTEd logistic regression actually classifies nothing as positive.
- Neural Networks Results:



Neural Network								
	With SMOTE				Without SMOTE			
Alpha	0.1	0.01		0.0001	0.1	0.01		0.0001
Hidden Layers	15	15	40	15	15	15	40	15
True Negative	59.161	59.153	59.169	59.162	39.983	26.066	37.090	35.644
False Positive	8	16	-	7	19.186	23.103	22.079	23.525
False Negative	89	90	90	89	7	6	9	7
True Positive	1	-	-	1	83	84	81	83

Figure 1.4: Neural Network Performance

  Using a multi-layer perceptron the results are not very good. Using SMOTEd data we get more true positives but also a lot more false positives. The best results are achieved at an alpha of 0.01 and 15 hidden layers. Even with the optimal parameters the results are still not good.

Overall we can clearly see that using SMOTE is a good idea and produces far better classification results. This agrees with the observations done in the paper "SMOTE for Learning from Imbalanced Data: Progress and Challenges". All ROC curves are provided in the Appendix.

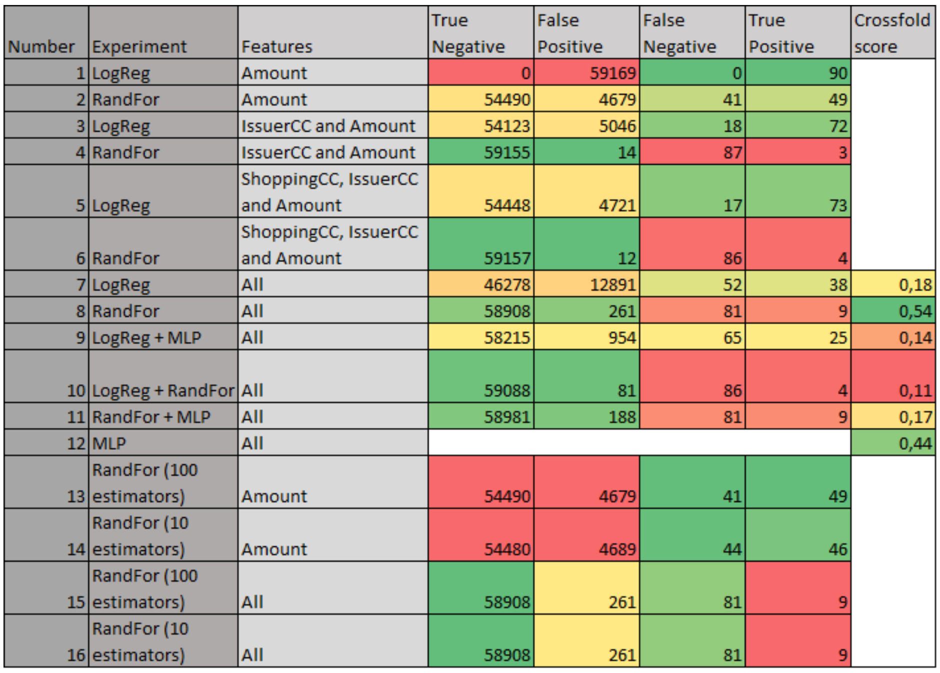
1.3 Classification Task

Build two classifiers for the fraud detection data as well as you can manage: 1.A black-box algorithm, ignoring its inner workings: it is the performance that counts. 2.A white-box algorithm, making sure that we can explain why a transaction is labeled as being fraudulent. Explain the applied data pre-processing steps, learning algorithms, and post-processing steps or ensemble methods. Compare the performance of the two algorithms, focusing on performance criteria that are relevant in practice, use 10-fold cross-validation. Write clear code/scripts for this task, for peer-review your fellow students will be asked to run and understand your code

The preprocessing of the previous task was used. In this task we experimented with different features and feature combinations to see if that resulted in better results. For this task, the following algorithms were chosen:

1. Black box algorithm: Random Forest
2. White box algorithm: Logistic Regression

All executed experiments are shown in Figure 1.5. Experiment number 1 till 8 attempt to answer the Classification task while Experiment number 9 till 12 are used to get a better insight into the different classifiers and attempt to get even better results. Finally experiment 13 till 16 are used to explore parameters for the Random Forest.



Number	Experiment	Features	True Negative	False Positive	False Negative	True Positive	Crossfold score
1	LogReg	Amount	0	59169	0	90	
2	RandFor	Amount	54490	4679	41	49	
3	LogReg	IssuerCC and Amount	54123	5046	18	72	
4	RandFor	IssuerCC and Amount	59155	14	87	3	
5	LogReg	ShoppingCC, IssuerCC and Amount	54448	4721	17	73	
6	RandFor	ShoppingCC, IssuerCC and Amount	59157	12	86	4	
7	LogReg	All	46278	12891	52	38	0,18
8	RandFor	All	58908	261	81	9	0,54
9	LogReg + MLP	All	58215	954	65	25	0,14
10	LogReg + RandFor	All	59088	81	86	4	0,11
11	RandFor + MLP	All	58981	188	81	9	0,17
12	MLP	All					0,44
13	RandFor (100 estimators)	Amount	54490	4679	41	49	
14	RandFor (10 estimators)	Amount	54480	4689	44	46	
15	RandFor (100 estimators)	All	58908	261	81	9	
16	RandFor (10 estimators)	All	58908	261	81	9	

Figure 1.5: Experimental results

Random Forest

A Random Forest is an algorithm that applies multiple decision trees to classify a result. In order to select the best feature set to use for the Random Forest many different combinations of features were tried. Increasing the amount of columns does a lot to reduce the amount of false positives, although it also has detrimental effects to the amount of true positives. By choosing all the columns in the preprocessed data set we can effectively reduce the amount of false positives to near-0. However this also does the same for the true positives. As can be seen above the Random forest scores decently wit True positives when using only the Amount but still has a significant amount of False positives as well. Using more features only creates worse scores.
For parameters using 100 estimators or 10 estimators does not really make a difference as can be seen in experiment 13-16 in Figure 1.5.

Logistic Regression

Logistic regression scores better with more columns, using issuer country code and shopping country code it has a high amount of true positives but also a decent amount of false positives as well.

10-fold cross-validation

Using cross fold validation predictions did not get higher than 54% correctly classified over 10 folds. The Random Forest scores the best in cross validation and the combination of logistic regression with a Random Forest scores the worst.

Ensemble + Bonus

Finally we looked at the scores of multiple classifiers. In order to do this the classifiers were initialized and added into an ensemble using multiple combinations and using multiple parameters.

The ensemble results are displayed in Figure 1.5 as you can see the 'best' combination reaches 25 True Positives with 954 False Positives. Based on what you prefer another algorithm is a better classifier. If you prefer low false positive rates then a combination of Logistic Regression with a Neural Network is the best. In credit card fraud detection you want high True positive detection and want to minimize False Positive rates so based on the information gathered in this report we would recommend using Logistic Regression in Combination with a Neural Network.

1.4 Appendix

1.4.1 Running the code

To run the solution, open the file *Classification.ipynb* in jupyter notebook. Make sure you have installed the dependencies: imblearn, seaborn, pandas, sklearn, numpy, matplotlib. Install them using pip or any other method of choosing.

1.4.2 ROC Curves

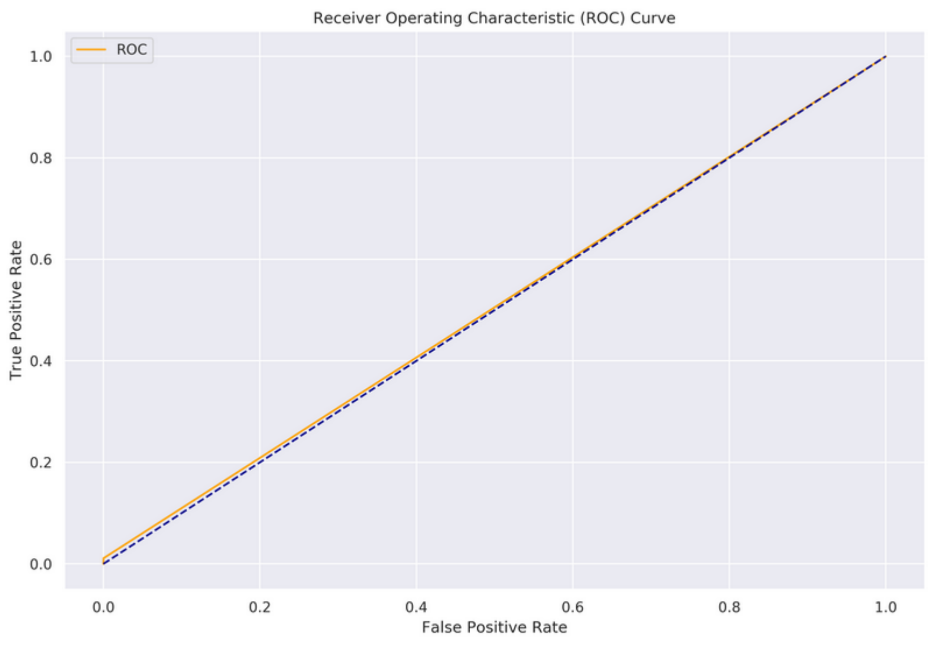


Figure 1.6: Random Forest without SMOTE

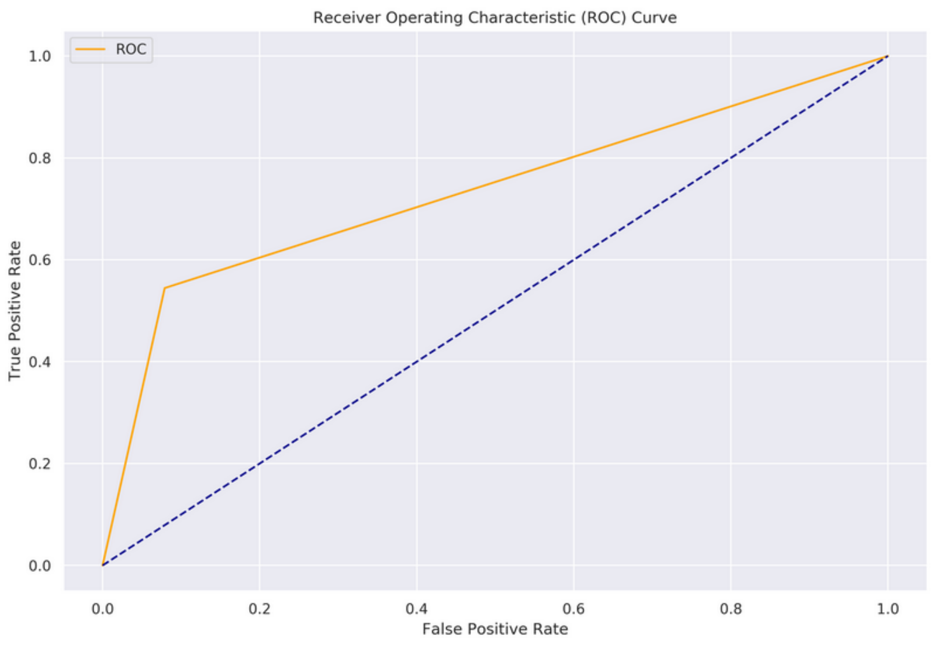


Figure 1.7: Random Forest with SMOTE

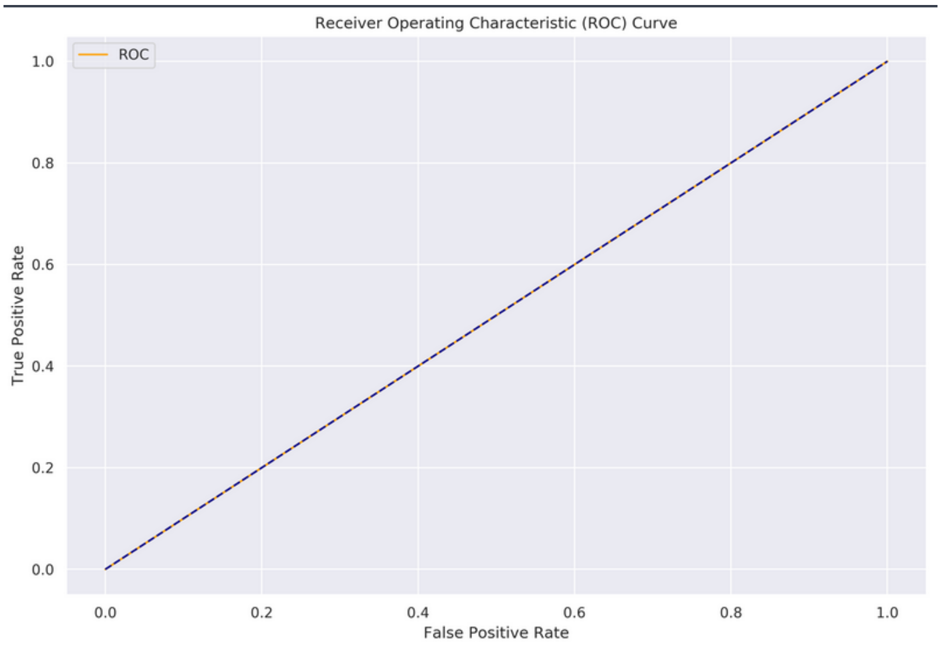


Figure 1.8: Neural Network without SMOTE

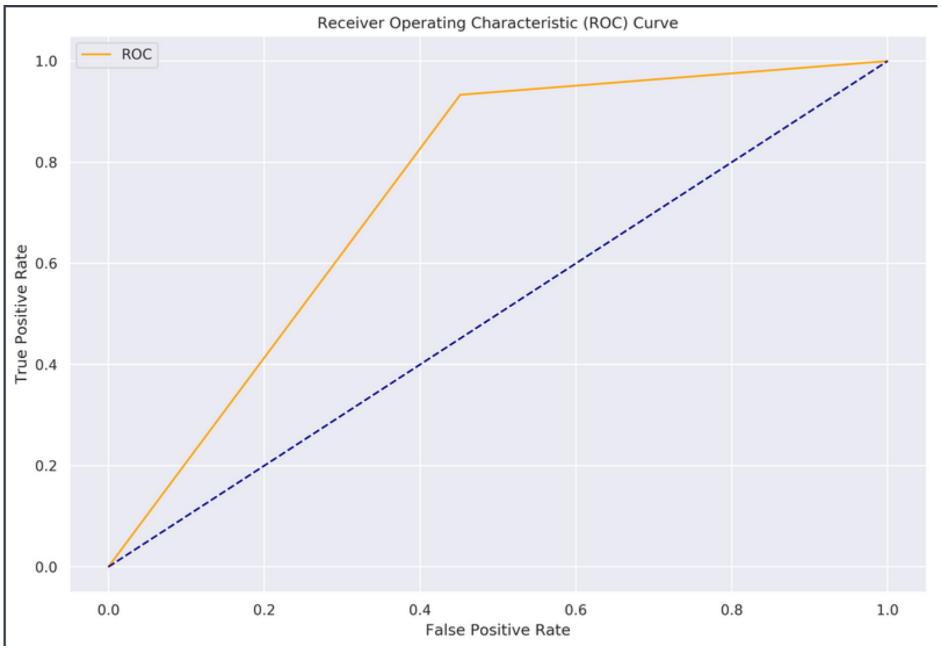


Figure 1.9: Neural Network with SMOTE

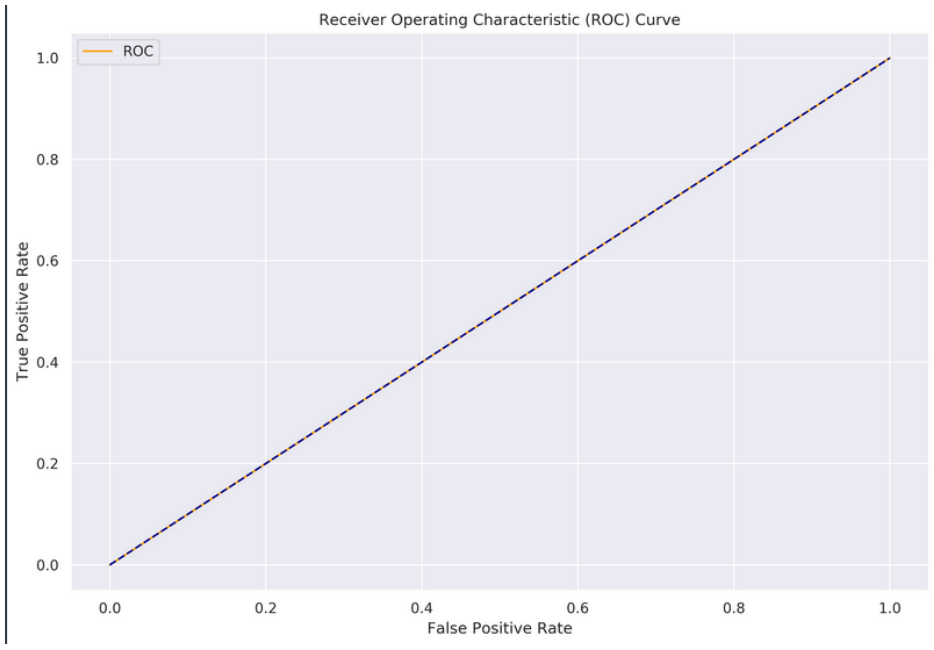


Figure 1.10: Logistic Regression without SMOTE

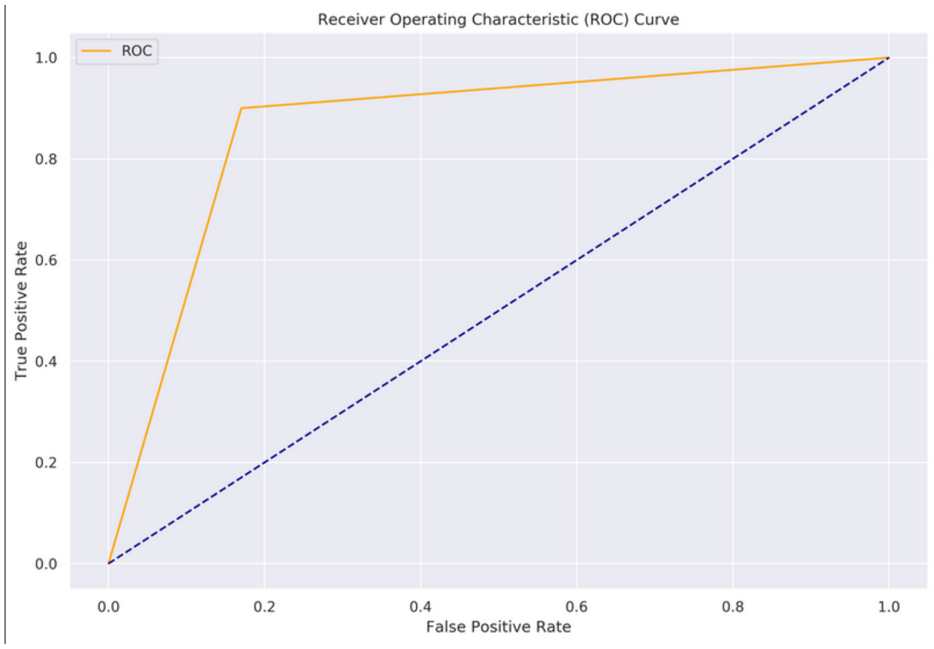


Figure 1.11: Logistic Regression with SMOTE

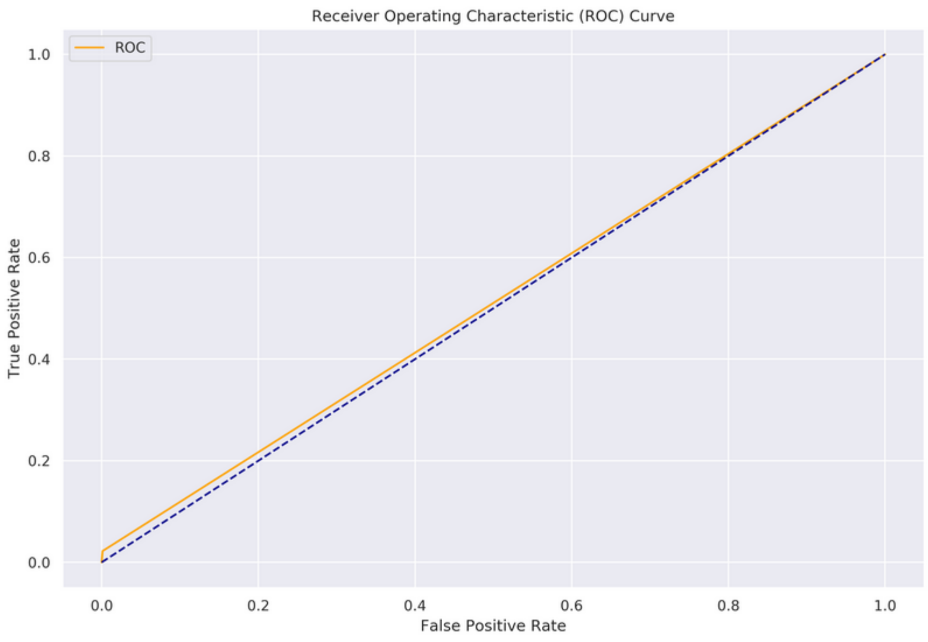


Figure 1.12: Ensemble Logistic Regression + Random Forrest

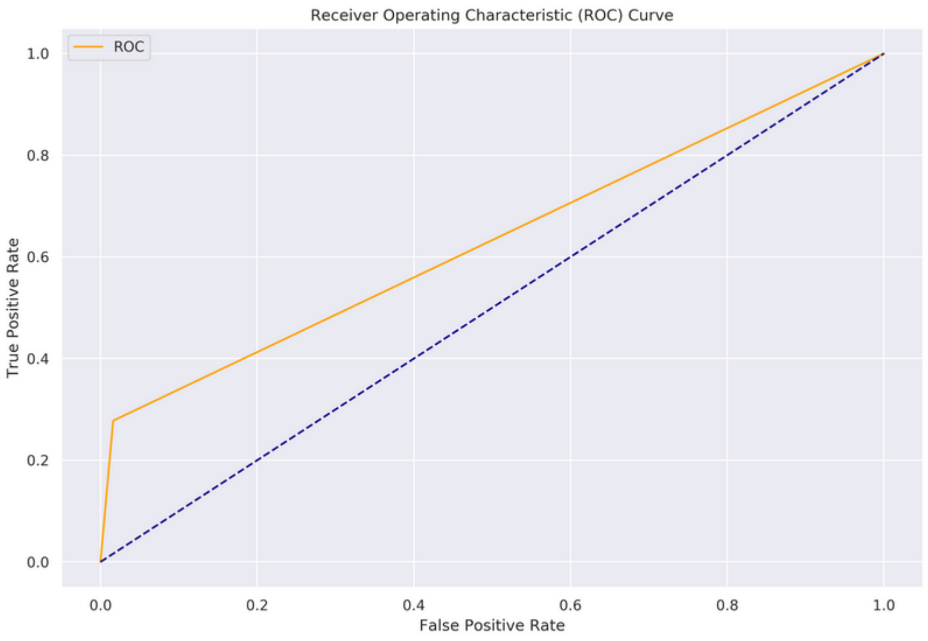


Figure 1.13: Ensemble Logistic Regression + Neural Net

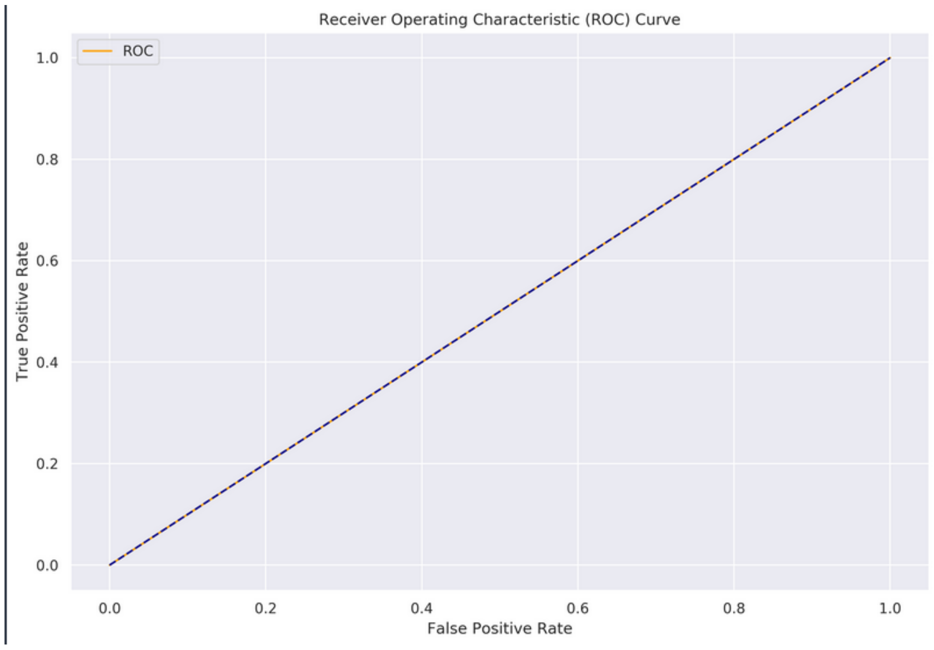


Figure 1.14: Logistic Regression without SMOTE