University of British Columbia

Advanced Machine Learning Tools for Engineers

EECE 571T

Credit Card Fraud Detection: A performance comparison of machine learning algorithms

---

Author(s):

Ali Magzari

Benjamin Wong

April 2022

Abstract

Fraud has been a part of human civilization from time immemorial. In today's high velocity information age civilization, credit card use—and fraud—continues to grow. Tens of billions of dollars globally are lost every year due to credit card fraud (CCF), making it a thorn in the side of the economic machine. As approximately a billion transactions happen every day, fraud investigators have turned to using machine learning models to help detect and eliminate fraud.

This report seeks to investigate the usefulness of various machine learning algorithms for this task. There are many challenges inherent in using machine learning to detect CCF not present in other data sets. Unbalanced classes means that various preprocessing methods are needed to extract the best performance from algorithms. Additionally, the metrics used to assess performance are different than usual. The confidential nature of the CCF data also means that data is not readily available, and models that are built cannot be deployed nor interpreted unless the actual input variables are known.

Due to time constraints, four machine learning algorithms were used in this study: logistic regression, Light Gradient Boosting Machine, Random Forest, and neural networks. Resampling was done by synthetic minority oversampling technique (SMOTE). Different metrics have to be used to assess this type of unbalanced data, so area under the precision-recall curves and F_1 score were used instead.

Although neural network and random forest algorithms generated relatively satisfactory results, the results of this project were inconclusive. Tools meant to improve performance tended to have mixed results. It is not likely that a model created from this data could be deployed on real data effectively, even if the input and principle component analysis parameters were known. The fraud data points used in the report are not representative of modern global fraud. Additionally, due to the concept of class migration, the data used will be outdated.

Contents

1 Background

Fraud is defined by Lexico as "wrongful or criminal deception intended to result in financial or personal gain" [1]. Not only humans engage in fraudulent behaviour: for certain animals, life and fraud are one and the same. The cuckoo bird's existence relies on its ability to make other birds raise their young for them through deceit [2]. From ancient Egyptian and Greek barter societies to our information age society, fraud can be found in many forms [3]. Fraud and fraud detection have been mainstays of human existence since the very beginning.

It should be no surprise then that the medium in which trillions of dollars of goods and services are exchanged annually—credit cards—is also an arena in which fraud is present. In 2020, nearly 30 billion dollars were lost to online credit card fraud (CCF) across the globe. This number is misleading: the true cost of fraud is usually much higher. 2021 was the first year in which successful CCF attempts outnumbered failed ones [4].

Credit card fraud is not uniform, it comes in many different forms. Dal Pozzolo identifies *stolen card*, *cardholder-not-present*, and *application fraud* as some common forms of CCF [5]. Delamaire adds *bankruptcy fraud*, *theft or counterfeit fraud*, and *behavioural fraud*[6]. Chaudhary distinguishes between online and offline CCF [7]. There is some overlap in definitions between authors, and they state that the lists are non-exhaustive. Fraud is expected to evolve over time adapting to new security measures: new types of CCF may surface in the future.

Different types of fraud are associated with different spending patterns. Some types of fraud are more sophisticated than others, and as a result, are harder to detect. With theft fraud, a card is physically stolen. A criminal will usually attempt to spend as much money as possible before the owner is aware and the card is locked or cancelled [5]. This relatively unsophisticated form of CCF can be identified by looking for deviations in common spending behaviour. Rules-based algorithms, decision trees, and genetic algorithms are common. Other more sophisticated forms of fraud are more difficult to predict [6].

Some types of fraud—namely application or bankruptcy fraud—can allow a perpetrator to use the card normally, as any other person would, while having no intention to pay off the balance. This is either due to having filled in false information, or being protected as they are bankrupt. These are considered particularly difficult to detect. Detecting these types of fraud usually means having a more comprehensive application screening process [6]. These types of fraud will likely not be highly represented in the data used for this report. This is because the data mainly deals with transactions, not applications.

Credit card fraud investigators have the job of looking for the proverbial needle in a haystack, combing through massive quantities of data to separate legitimate transactions from fraudulent ones. It is unrealistic to expect this process to be done manually: approximately a billion transactions are processed around the world, daily [8]. As a result, we can expect computers and algorithms to be an important tool used in CCF detection moving forward.

This report does not focus on any particular types of fraud, or whether certain models are better at detecting some types than others. The scope of this report will be to look at CCF in general, and the general performance of several different machine learning algorithms for detecting them.

2 Proposal

Designing a machine learning algorithm or model to detect credit card fraud poses some unique challenges. The confidential nature of CCF means that gathering, combining, and classifying data are difficult. The data used in this report [9] has thirty input variables and one output variable. Of the thirty input variables, only two—time and amount—were not encoded via principle component analysis. This encoded information makes models difficult to inter-

pret, if not outright incoherent.

Additionally, CCF data suffers from severe class imbalance. This means that certain classifications have far more entries than others. In the case of CCF data, legitimate transactions generally outnumber fraudulent ones over a hundred times to one [9]. Popular metrics used to assess model performance such as accuracy do not perform particularly well with this level of imbalance. Resampling becomes an important tool for preparing data to address this imbalance [5]. This severe class imbalance coupled with the confidential nature of CCF data means that datapoints for fraud are lacking in quantity, and it is difficult to create new ones from actual data, as opposed to synthesis.

Once a model is trained, it also has to have a viable strategy for dealing with changes in fraudulent behaviour. This class migration, or evolution of fraud over time is referred to as *concept drift*. What this means is that once a model is trained, its performance will deteriorate over time as fraud behaviour changes. Algorithms need to constantly update their models, both by incorporating new data and removing old extraneous datapoints to keep up-to-date. While models trained on current data should maintain relevance, it can be difficult to detect novel and innovative forms of fraud. A model trained on existing data is, by definition, reactionary. There may be methods to preempt new types of fraud, but they are beyond the scope of investigation in this report.

Other work investigating this topic has been done before. A review published by the University of Huddersfield [6] summarizes a number of studies done using various methods. The list is expansive, showing studies done with machine learning such as neural networks, decision trees, clustering. Other types of algorithms like genetic algorithms, bayesian networks, or expert systems were also featured. Some types of fraud are more easily detected by certain techniques [6, 7, 10].

This report seeks to better understand how machine learning can be used to detect CCF. The performance of different machine learning algorithms when trained on CCF data is analyzed, along with the metrics used to analyze them. In addition, the performance of algorithms with unaltered and resampled data is compared. Finally, problems related to class migration and confidentiality are discussed, with a look at how CCF detection algorithms could function in the future.

The investigation in this report will start by analyzing the data available from Kaggle [9]. Due to the unbalanced nature of the data, it will first be subject to resampling. Sythetic minority oversampling technique (SMOTE) was chosen as the resampling method of choice. Once resampled, four different machine learning algorithms will be tested: logistic regression, Light Gradient Boosting Machine (LightGBM), Random Forest, and neural networks. Logistic regression is one of the simpler binary classification algorithms available, so will serve as a useful baseline. LightGBM and Random Forest are both decision tree algorithms, and are usually used due to their interpretability. This will not be particularly useful for this problem as encoded data in the inputs will not help meaningfully explain the trees being built. Neural networks are a machine learning staple, and capable of advanced pattern recognition while being versatile.

3 Methods Used and Explained

3.1 Pseudo-code

The following represents the pseudocode used in this project. Details such as the models used and parameters are detailed in later sections.

1. Load and split data into training and test sets
2. Build preprocessing pipeline
3. Fitting on unaltered data
   a. Fit the models on unaltered training data
   b. Compute training metrics
   c. Compute test metrics
4. Fitting on SMOTE-resampled data

   a. Fit the models on SMOTE training data
   b. Compute training metrics
   c. Compute test metrics
5. Threshold-moving on unaltered data
   a. Search for optimal threshold using PRC of training data
   b. Compute test predictions using best threshold
   c. Compute test metrics
6. Threshold-moving on SMOTE-sampled data
   a. Search for optimal threshold using PRC of SMOTE training data
   b. Compute test predictions using best threshold
   c. Compute test metrics

3.2 Data Exploration and Preprocessing

The dataset used is created by the Machine Learning Group at l'Université Libre de Bruxelles and was made available to the data science community on Kaggle [9]. It contains a total of 284,807 credit card transactions, 792 of which are fraudulent, which accounts for less that 0.17% of the dataset, which is a clear sign of data imbalance. The data consists of 30 explanatory variables and one response variable, as summarized below:

- Time: number of seconds elapsed from the first transaction in the dataset;
- Amount: amount of the transaction;
- V1-V28: principal components generated due to confidentiality reasons;
- Class: response variable describing the transaction as legitimate or fraudulent, 0 or 1 respectively

Respecting the golden rule of machine learning, the dataset is split into two distinct segments: training (70%) and test (30%). The training set has 199,364 observations, 350 of which are fraudulent cases (0.18%). All the columns being numeric, and no missing data detected, the preprocessing pipeline was simple, and only involved standard scaling:

$$z = \frac{x - \mu}{\sigma}$$

3.3 Combating Class Imbalance

3.3.1 Resampling with SMOTE

Class imbalance is known to affect the performance of classification algorithms [11], which calls for resampling. Out of the several resampling methods, synthetic minority oversampling technique (SMOTE) was used, as it allowed to generate better results than the ones produced via basic version of both under and oversampling techniques. SMOTE was first described in 2002 [12], and its main idea could be summarized as follows [13].

- An instance from the minority class is randomly selected;
- a set of k-nearest neighbors for the previously selected instance is obtained;
- one of the neighbors is randomly selected;
- the original observation and its neighbor are connected by forming a line in the feature space;
- the synthetic point is randomly positioned along the feature line;

3.3.2 Threshold-moving

Problems suffering from extreme class imbalance underperform at the default threshold. One of the ways to avoid such a flaw is to tune the threshold until the acquisition of desired precision and recall scores. For this reason, the threshold generating the maximum f1-score was calculated based on the training precision-recall curve for both models fitted on unaltered and resampled data.

3.4 Metrics

Since the majority class represents 99.82% of the training set, a simple dummy classifier trained to choose the most frequent class would generate an accuracy of 99.82%. As this may seem beyond satisfactory on paper, it misses the principal objective

of the classifier, which is to detect fraudulent cases. While accuracy is a useless score in this scenario, other metrics should be considered, which requests the definition of the confusion matrix in Figure 2.

A confusion table illustrates the performance of an algorithm by reporting the following:

- TP: The cases that are classified by the model as positives, and are actually positive.
- FP: The cases that are classified by the model as positives while they are actually negative.
- FN: The cases that are classified as negatives, while they truly are positives.
- TN: The cases that are classified as negatives, and are actually negative.

From these four elements stem the definitions for two important metrics, precision and recall:

- Precision: The fraction of true positives over the number of observations classified as positives, or—*out of the cases classified as positives, how many are actually positive?*

$$Precision = \frac{TP}{TP + FP}$$

- Recall: The fraction of true positives over the total number of actual positive cases, or—*out of the number of truly positive cases, how many were we able to classify as so?*

$$Recall = \frac{TP}{TP + FN}$$

In an effort to detect the highest number of fraudulent transactions, a high recall score is desired. However, since precision and recall are inversely proportional, a high recall would also result in a low precision score, which is not convenient because a low precision score would mean that the algorithm classifies wrongly a large number of legitimate transactions as being fraudulent. The concerned financial institution will then have to manually verify a large quantity of transactions; that requires a tremendous amount of financial and human resources, which defeats the purpose of using machine learning to solve the issue at hand.

F1-score, or the harmonic mean of precision and recall is also favored:

$$F_1 = \frac{Precision \cdot Recall}{Precision + Recall}$$

Another important metric to probe is the area under the precision-recall curve (AUPRC). The precision-recall curve itself is very insightful in the sense that it depicts the evolution of both precision and recall at various prediction probability thresholds. Figure 3 is an example of such a curve. Note that AUPRC is comprised between 0 and 1.

3.5 Algorithms

3.5.1 Logistic Regression

Logistic Regression (LR) is a predictive analytics technique commonly used when the dependent variable is binary in nature [14]. This works well for creating a CCF model, as the outputs are binary—"fraud", and "not fraud". Linear regression is similar to LR, but any probabilities that are established still need to be fed into a non-linear function for classification. This non-linear function usually takes the form of a softmax or sigmoid function [15]. Now that the values are confined between 0 and 1, an output value can be predicted. In this case, the output will be a binary value, either true or false.

3.5.2 Light Gradient Boosting Machine (LightGBM)

LightGBM is a decision tree-based learning algorithm. The algorithm starts off with a guess. Residuals are then calculated from an initial guess then a tree is created. The effect of the leaves in a tree are determined by its performance for predicting y-values compared to the previous guess. The process repeats by calculating a new residual from this tree before a new tree is created. Trees will be created

until a threshold is reached or performance no longer increases.

3.5.3 Random Forest

Random Forest is a decision tree-based algorithm. While decision trees are generally easy to understand, they do not perform well in practice. Random Forest seeks to remedy this by creating a 'forest' of trees. A random forest first creates a bootstrapped dataset from the original. Datapoints not bootstrapped are in the 'out-of-bag' dataset. Trees with branches are then created by randomly selecting variables. This is usually done via bagging; a subsample of the bootstrap data is selected to train a random tree. This continues until a threshold is reached or the new trees do not improve model performance. The data that does not make it into the bootstrapped dataset is used to test the accuracy of the model. Samples are put through the model, where the trees vote on what the final prediction should be.

3.5.4 Neural Network

Artificial neurons and neural networks (NN) have been around for a long time as an idea: The Threshold Logic Unit was proposed in 1943. Artificial NNs are made up of interconnected 'neurons', usually organized in layers. A NN is made up of an input layer, and output layer, and one or more hidden layers. The output layer can consist of one or more neurons and can be used for different types of problems: classification, regression/approximation, and data processing.

4 Results and Discussion

4.1 Model Parameters

The models constructed here are the ones introduced earlier:

- Logistic regression (LR): basic binary regression model with default L2 norm regularization to avoid overfitting, and maximum number of iterations set to 1000 to ease convergence. The rest of the parameters are set to default values of *sklearn* library logistic regression model [16].
- Light Gradient Boosting Machine (LightGBM): The default parameters of the LGBMClassifier of *lightgbm* framework [17] were chosen as no insight was present to choose specific parameters.
- Random Forest (RF): The default parameters of *sklearn* RandomForestClassifier [18] were chosen, maximum tree depth set to "None" for full node expansion and obtention of pure leaves.
- Neural Network (NN): A simple neural network is created by implementing the default *sklearn* MLPClassifier [19] with one hidden layer of 600 neurons, ReLU activation function, and nesterov optimization. The number of hidden neurons was chosen to be 600 according to the following equation:

$$N_h = \frac{N_s}{\alpha(N_i + N_o)}$$

Where:
N_h: number of hidden neurons
N_s: number of training observations
N_i: number of input neurons
N_o: number of output neurons
α: arbitrary scaling factor from 2 to 20
Setting the scaling factor to an arbitrary value between 10 and 11 results in $N_h = 600$.

4.2 Discussion of Results

Table 1 summarizes both training and test results after the models were fitted on unaltered training data. The random forest and neural network models reported a significant gap between test and training scores: signifying that they failed to generalize well. Having said that, they still generated satisfactory test metrics superior to the ones obtained by logistic regression. The test AUPRC of 0.81 and 0.77 for the random forest and neural network models,

respectively, are worth mentioning. On the other hand, LightGBM produced the poorest results, at 0.20.

Figure 4 illustrates the precision-recall curves of the test data and corroborates the observations mentioned above. RF and NN curves had larger areas under the curve when compared to LR. They are able to achieve higher levels of precision while retaining an equivalent recall value.

Table 2 lists the test scores obtained after the models were fitted on SMOTE-oversampled training data. Random forest performance seem to improve slightly in terms of recall and f1-score, but remains unchanged for both precision and AUPRC. On the other hand, neural network underperforms in all aspects after being trained on SMOTE-resampled data. The only models that seem to benefit from SMOTE are logistic regression and LGBM (whose value increased from 0.20 to 0.65).

It is crucial to underline that although linear regression outputted an AUPRC of 0.72, its recall and F1-score were surprisingly low. That is due to the fact that these metrics are calculated at the default threshold of 0.5, which is not necessarily the point at which the model classifies optimally. As mentioned in Methods, such an issue could be remediated by tuning the threshold that maximises the F1-score.

Table 3 depicts the test results obtained after threshold optimization. This is achieved by fitting the model on the training data; calculating its precision and recall scores at various probability thresholds; choosing the threshold at which the precision and recall generate the highest F1-score; this newly obtained threshold is then used to predict classes on untouched test data. This operation was executed on both unaltered and SMOTE-resampled training data.

The first note to be made regarding table 3 is that F1-optimization via threshold-moving worsens the results when coupled with SMOTE-resampled data. This may largely be due to the fact that after oversampling the training data—while adds noise—the newly generated data has a different class distribution to the test data. Consequently, the optimal threshold for the PRC obtained from SMOTE-resampled training data is different to the threshold that would optimize scores for test data.

On the other hand, threshold-moving based on unaltered data improved model performance. Besides neural network, where the performance actually worsened, random forest and linear regression saw their f1-score improve: 0.81-to-0.82, and 0.70-to-0.72, respectively. This might not be a significant improvement, however.

Key points to summarize the above results would be:

- Neural network performed best on unaltered data.
- The three best models in terms of f1-score are: NN fitted on unaltered data; RF fitted on SMOTE-resampled data; RF fitted on unaltered training data with threshold-moving.
- LR generated satisfactory results but inferior to NN and RF.
- Although LGBM metrics improved after SMOTE, the model showcased the lowest results.

4.3 Suggested Improvements

A number of changes could have benefited this report greatly:

- Automatic hyper-parameter optimization: A grid search that reports the scores of the models while using different hyper-parameters would have given us better insight (i.e. reporting the neural network scores while using different number of hidden neurons and activation functions).
- Reporting of model fitting time: Some models may generate superior results, but would require a significantly longer fitting time, which is problematic, as in the case of random forest.

- Implementation of other algorithms or techniques to combat class imbalance: A large variety of classification algorithms and imbalance remediation techniques are available, and should to be tested.
- Comparison of results with the data science community and trusted literature: A number of ML-enthusiasts do not solve machine learning problems following recommended guidelines. Additionally, published journal articles do not necessarily treat the exact same dataset explored in this project. These factors made it hard to compare our results. Nevertheless, more search and time could have been devoted to that effect.

5 Conclusions and Future Work

5.1 Conclusions

Handling CCF data was more challenging than anticipated. The unbalanced data meant that the metrics we were familiar with—such as accuracy—lost meaning. Other metrics such as precision, recall, f1-score and AUPRC had to be used. Techniques utilized to combat class imbalance had mixed results. The confidential nature of the data was also challenging to work with. While it was easy to understand numerically, there was little intuition that could assist in understanding how the principle components affected the end result or why they were useful. Decision tree algorithms, favored for their interpretability, were incoherent.

While the models built in this report have not been deployed to test their performance, it would not be surprising if the performance of the model was subpar. It is questionable if the few fraud samples available—800 or so—are representative. They were taken over a two day span in Europe in 2012. Concept drift would mean that the data may be outdated, and the model might not work well in another region. Additionally, as we do not know what the initial parameters are, and how the principal component analysis was done, it is virtually impossible to deploy and test these models on any data other than the data from Kaggle.

5.2 Future of CCF Detection

The future of CCF detection will likely have certain properties. Due to the rapid concept drift of fraud distributions, a successful algorithm will be able to take in and train on data in real time, and perhaps create data points of its own. Generative Adversarial Networks (GANs) have been used to synthesize images particularly well, as the website "thispersondoesnotexist.com" showcases [20]. GANs could possibly be used to replace SMOTE as a form of oversampling.

Yet, even a GAN model—trained on existing data—would not be able to predict how fraud could evolve and deploy countermeasures ahead of time. That might require a self-play or reinforcement learning component. Having the ability to predict and defend against new types of fraud before they come into existence would allow the CCF detection algorithm to play the role of an extremely proficient white-hat hacker. This likely will not eliminate CCF for good though. A counter CCF algorithm might have to defend against fraud committed by other algorithms. If history has taught us anything, it is that fraudsters will always find a way, as they always have throughout human history.

6 References

[1] Lexico. *Fraud*. 2022. URL: https://www.lexico.com/definition/fraud.

[2] Nicky Featherstone. *Where Do Cuckoos Lay Their Eggs?* Aug. 2021. URL: https://www.forestwildlife.org/where-do-cuckoos-lay-their-eggs/.

[3] Trulioo. *A history of fraud: from ancient Egypt to the modern pandemic*. May 2020. URL: https://www.trulioo.com/blog/history-fraud.

[4] Chargebacks911. *Key Credit Card Fraud Statistics to Know for 2022*. Accessed April 2022. 2022. URL: https://chargebacks911.com/credit-card-fraud-statistics/.

[5] Andrea Dal Pozzolo. "Adaptive Machine Learning for Credit Card Fraud Detection". PhD thesis. Dec. 2015. URL: http://di.ulb.ac.be/map/adalpozz/pdf/Dalpozzolo2015PhD.pdf.

[6] Linda Delamaire, Hussein Abdou, and John Pointon. "Credit card fraud and detection techniques: a review". In: *Banks and Bank Systems* 4.2 (Apr. 2009), pp. 57–68. URL: http://eprints.hud.ac.uk/id/eprint/19069/.

[7] Khyati Chaudhary, Jyoti Yadav, and Bhawna Mallick. "A review of Fraud Detection Techniques: Credit Card". In: *International Journal of Computer Applications* 45 (Jan. 2012). URL: https://research.ijcaonline.org/volume45/number1/pxc3878991.pdf.

[8] Erica Sandberg. *The Average Number of Credit Card Transactions Per Day & Year*. Nov. 2020. URL: https://www.cardrates.com/advice/number-of-credit-card-transactions-per-day-year/.

[9] *Credit Card Fraud Detection, anonymized credit card transactions labeled as fraudulent or genuine*. Accessed April 2022. 2022. URL: https://www.kaggle.com/mlg-ulb/creditcardfraud.

[10] Andrea Dal Pozzolo et al. "Learned lessons in credit card fraud detection from a practitioner perspective". In: *Expert Systems with Applications* 41 (Aug. 2014), 4915–4928. DOI: 10.1016/j.eswa.2014.02.026.

[11] Michal Koziarski and Michal Wozniak. "CCR: A combined cleaning and resampling algorithm for imbalanced data classification". In: *International Journal of Applied Mathematics and Computer Science* 27 (2017), pp. 727 –736.

[12] Nitesh V. Chawla et al. "SMOTE: Synthetic Minority over-Sampling Technique". In: *J. Artif. Int. Res.* 16.1 (June 2002), 321–357. ISSN: 1076-9757.

[13] Swastik Satpathy. *Overcoming Class Imbalance using SMOTE Techniques*. Oct. 2020. URL: https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/.

[14] IBM. *What is Logistic regression?* URL: https://www.ibm.com/topics/logistic-regression.

[15] DeepAI. *Logistic Regression*. May 2019. URL: https://deepai.org/machine-learning-glossary-and-terms/logistic-regression.

[16] 2019. URL: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html.

[17] 2022. URL: https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html.

[18] Scikit-learn. 2018. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[19] 2010. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html.

[20] 2018. URL: https://thispersondoesnotexist.com/.

[21] scikit learn. *Precision-Recall*. URL: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html#sphx-glr-auto-examples-model-selection-plot-precision-recall-py.

7 Tables and Figures

7.1 Tables

Model-set	Recall	Precision	F_1	AUPRC
RF-train	1.00	1.00	1.00	1.00
RF-test	0.75	0.88	0.81	0.81
NN-train	0.95	0.98	0.96	0.97
NN-test	0.75	0.88	0.81	0.77
LR-train	0.68	0.91	0.78	0.80
LR-test	0.65	0.79	0.71	0.70
LGBM-train	0.58	0.42	0.49	0.26
LGBM-test	0.54	0.35	0.43	0.20

Table 1: Unaltered Data Results

Model	Recall	Precision	F_1	AUPRC
RF	0.77	0.88	0.82	0.81
NN	0.75	0.81	0.78	0.76
LR	0.86	0.15	0.26	0.72
LGBM	0.76	0.78	0.77	0.65

Table 2: SMOTE Resampled Test Results

Model	Recall	Precision	F_1
RF-opt	0.75	0.91	0.82
RF-SMOTE-opt	0.84	0.29	0.43
NN-opt	0.73	0.78	0.75
NN-SMOTE-opt	0.74	0.78	0.76
LR-opt	0.74	0.70	0.72
LR-SMOTE-opt	0.93	0.03	0.06
LGBM-opt	0.54	0.37	0.44
LGBM-SMOTE-opt	0.80	0.19	0.31

Table 3: Optimal Threshold Scores, *opt means optimal

7.2 Figures

Figure 1: SMOTE diagram

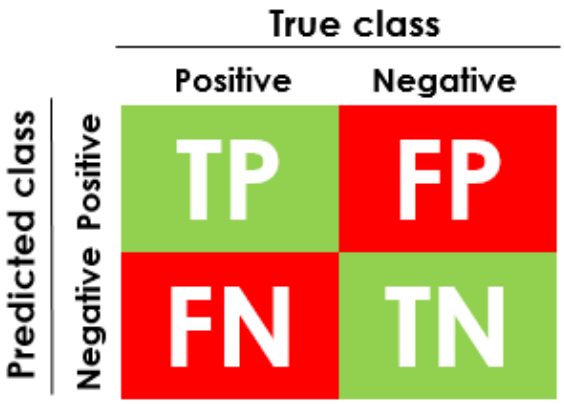

Figure 2: Confusion Matrix

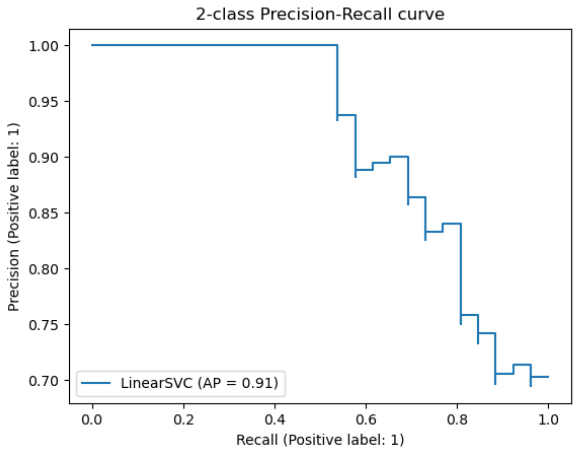

Figure 3: Precision-Recall Curve Example [21]

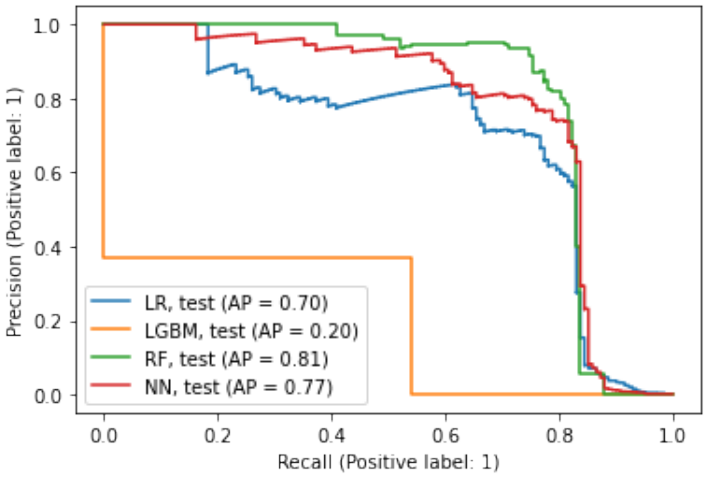

Figure 4: Test PRC of models fitted on unaltered training data