Detecting Credit Card Fraud with Machine Learning

Aaron Rosenbaum[1]
Stanford University, Stanford, CA, 94305, USA

I. Introduction

Payments fraud represents a significant and growing issue in the United States and abroad. There was more than $8 billion in fraud over U.S. non-cash payment systems in 2015, a 37 percent increase from 2012 [1]. Financial institutions and payment operators are increasingly relying on machine learning algorithms to develop efficient and effective fraud detection systems [2]. In this applied project, I implement and assess the performance of various machine learning models, including logistic regression, random forests, and neural networks, using a rich dataset from Kaggle. The dataset contains approximately 300,000 credit card transactions occurring over two days in Europe.

A key challenge with payments fraud data is class imbalance. In the Kaggle dataset, roughly 99.8 percent of the transactions are labeled as legitimate and 0.2 percent as fraudulent. Class imbalance can make it difficult for standard models to learn to distinguish between the majority and minority classes [3]. As part of this project, I explore and assess methods to address this issue, including sampling techniques such as undersampling the majority class and oversampling the minority class. Due to privacy concerns, it is not possible to infer meaningful relationships between most of the variables in the dataset. For this reason, the focus of my project is on predictive performance rather than inference.

II. Related Academic Work

There is a long history of using machine learning for fraud detection in the payments industry. Bhattacharyya et. al. note that while fraud algorithms are actively used by banks and payment companies, the breadth of studies on the use of machine learning techniques for payment fraud detection is limited [4], possibly due to the sensitive nature of the data. Their study concluded that random forests, though not widely deployed, may outperform more traditional methods. Roy et. al. found that network size is the strongest driver of a neural network's performance for fraud classification [5]. Chaudhary et. al. note that no single algorithm is pareto optimal across all performance metrics; each has a unique set of strengths and weaknesses [6].

Class imbalance in Machine Learning is a wide area of research. Various techniques are deployed to address this problem, including oversampling the minority class, undersampling the majority class, generating synthetic samples of the minority class, and adjusting the relative cost of misclassifying the minority and majority class. Japkowicz and Stephen conclude that the effects of class imbalance depend on the severity of the imbalance, sample size and the classifier used, noting that sampling techniques may negatively affect the performance of certain classifiers [7]. Cui et. al. highlight research suggesting that oversampling can lead to overfitting, while added noise from synthetic data generation can reduce predictive performance [8].

III. Data and Features

The Kaggle dataset contains 31 variables and nearly 300,000 credit card transactions labeled as either legitimate or fraudulent [9]. Table 1 provides descriptive information about the dataset. An important attribute of the dataset is that it has been processed to protect cardholder privacy. In particular, it contains 28 non-descriptive numerical variables (V1, …, V28) that are the result of a principal components analysis (PCA) transformation of several variables of interest that could not be publicly disclosed.

Table 1: Dataset description

Variable	Description
Time	Time elapsed since first transaction
V1 ⋮ V28	Non-descriptive variables resulting from a PCA dimensionality reduction to protect sensitive data
Amount	Transaction Amount
Class	Class label (1 = Fraud; 0 otherwise)
n = 284,807 observations	d = 31 variables

[1] Department of Statistics, Stanford University

Table 2: Class distribution

	Number	Percent
All transactions	**284,807**	**100**
Fraud	492	0.173
Nonfraud	284,135	99.827

Table 3: Transaction value by class

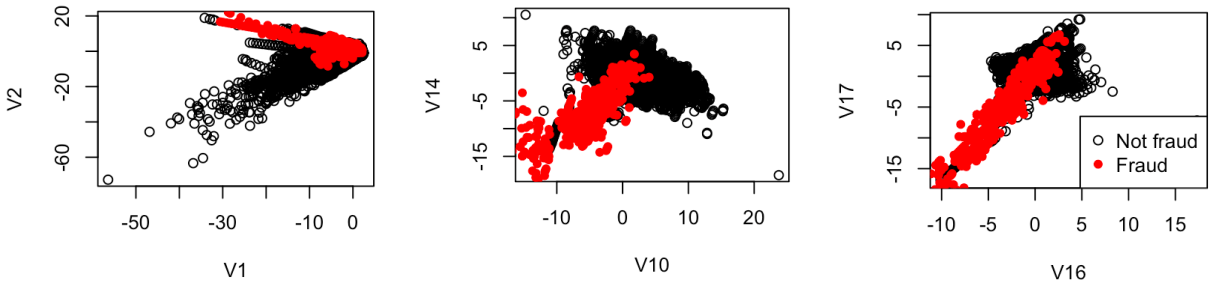
	Total Value	Avg. Value
All transactions	**25,162,590**	**88.35**
Fraud	60,128	88.29
Nonfraud	25,102,462	122.21

Preliminary analysis reveals several interesting features about the data. Table 2 shows the split of fraudulent and legitimate transactions. Only 492 transactions, or less than 0.2 percent, are fraudulent, highlighting a stark class imbalance. Table 3 shows that fraudulent transactions have greater transaction value on average than legitimate transactions. A correlation analysis (not shown) reveals that the class labels are correlated with several of the PCA variables. Further, the PCA variables are uncorrelated with each other, consistent with the fact that they are orthogonal and have been standardized to have zero mean. Figure 1 shows a sampling of pairwise plots of the principal components colored by class. These plots reveal striking patterns for the fraudulent and legitimate transactions in the principal component feature space. They suggest that machine learning models have a chance to learn from the data and classify transactions effectively.

Some preprocessing of the data was necessary prior to modeling. To start, I partitioned the data frame into train, validation, and test sets (2/3, 1/6, 1/6 split). For the logistic regression model with a quadratic boundary, I computed all quadratic and interaction terms of the original features. Additionally, prior to fitting the neural network, I applied min-max normalization to the features, as is typically done to promote faster convergence: $z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$.

For most of the models, I used all available variables in the dataset. Feature selection generally did not improve predictive performance given the large sample size. The exception was in fitting the quadratic model, which enlarged the feature space to almost 500 variables, leading to overfitting. In this case, I performed L1 (LASSO) regularization to delete variables from the model and reduce the variance of the fit.

Figure 1: Pairwise plots of selected principal components



IV. Methods

I used R statistical software for all parts of the analysis[2]. Four primary model categories were considered: simple logistic regression, logistic regression with quadratic terms and LASSO regularization, random forests, and neural networks. In fitting the models, I followed a tournament-style procedure where a series of models were trained in each model category. Parameters for each model were tuned using either k-fold cross validation or simple cross validation. The best performing model on the validation set in each category was selected as a finalist. At the end of the process, the four finalists were refit using the combined training and validation sets and assessed against the test set, which had remained untouched throughout the training process.

Accuracy and other standard performance metrics are often misleading in the presence of significant class imbalance. Case in point: for the current dataset, a naive classifier that always predicts "not fraud" will have an accuracy rate of 99.8 percent despite classifying all fraudulent transactions incorrectly. To address this issue, I use area under the precision-recall curve (AUPRC) as my primary performance metric. AUPRC is an attractive metric since it focuses on the accuracy and quality of predictions made on the minority class. Moreover, unlike some other performance metrics, AUPRC is threshold invariant (an example of a threshold is to classify as "fraud" if the model's predicted probability of fraud is greater than 0.5). In an industrial

[2] My R code for the analysis is available at https://github.com/arr1552/CS229-final-project.

setting, a high AUPRC provides some degree of reassurance to the financial institution or payment operator that it can flexibly select a threshold based on its unique business requirements and still obtain reasonable performance.

Given the size of the dataset and computational complexity of some of the procedures, a number of decisions were made to ensure that the training process remained tractable within a reasonable timeframe. For example, as discussed in greater detail in the results and discussion section, in certain cases, I opted for simple cross validation to tune parameters. Of course, in an industrial setting, one would tune parameters to the full model being fit and employ nested cross validation to get the best performance out of the available data.

A. Sampling Methods

To help address the class imbalance issue, I employed four sampling techniques to increase the proportion of minority examples in the training set:

1. Undersampling – balances the data by randomly choosing observations from the majority class to exclude
2. Oversampling – balances the data by randomly oversampling the minority class
3. Both – a hybrid method that employs both undersampling and oversampling
4. ROSE – a synthetic data generation method that balances the data by creating artificial samples of the minority class in the neighborhood of existing examples [10]

Increasing the proportion p of minority class examples using these methods has the effect of increasing the penalty of misclassifying an observation in the minority class with respect to the loss function.

B. Simple Logistic Regression

Logistic regression is a widely used discriminative learning algorithm that uses a hypothesis of the form $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$. In its simplistic form, logistic regression fits a linear decision boundary in the feature space to model a binary dependent variable. Parameters are fit by maximum likelihood, where the log likelihood is given by

$$\ell(\theta) = \sum_{i=1}^{n} y^{(i)} \log h\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h\left(x^{(i)}\right)\right)$$

C. Logistic Regression with Quadratic Terms and LASSO Regularization

In addition to fitting simple logistic regression models, I also fit models with a quadratic decision boundary in the original feature space by expanding the feature set to include all quadratic and interaction terms. As discussed above, enlarging the feature space in this way can lead to significant overfitting. To address this issue, I used LASSO regularization to reduce the variance of the fit using the glmnet package in R.

The LASSO adds an L_1 penalty term to the loss function, $\lambda\|\theta\|_1$, where $\lambda > 0$ is a parameter that determines the amount of regularization. The L_1 penalty has the tendency to zero out coefficients in the model as λ grows, and so the LASSO is often described as a method that implicitly performs feature selection. I computed λ using 3- or 10-fold cross validation, depending on the computational complexity of the specific model.

D. Random Forests

Tree-based models stratify the predictor space into simple regions that can be used to classify the dependent variable. Although simple tree methods are generally not competitive with other machine learning techniques, combining a large number of trees using methods such as bagging, boosting, and random forests often results in significant improvements in prediction accuracy, at the expense of some loss interpretation. That's not a bad tradeoff for the project at hand since the goal is to maximize predictive performance over interpretability.

I fit random forest (RF) models to the dataset. The RF method is similar to bagging in that it builds a large number of decision trees on bootstrapped training samples [11]. In particular, a simple decision tree is fit on B bootstrap samples and predictions for each tree are averaged together to obtain a consensus prediction: $\hat{f}_{RF}(x) = \frac{1}{B}\sum_{b=1}^{B} f^{*b}(x)$. The difference between RFs and bagging lies in how the simple trees are fit. When building a RF, a random set of predictors is chosen for each split in the tree. This method has the effect of decorrelating the trees, reducing bias when their predictions are ultimately averaged together.

E. Neural Networks

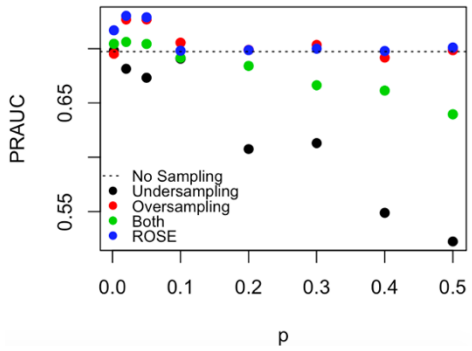
Neural Networks are a popular set of machine learning algorithms that are widely used for credit card fraud detection. Conceptually, a neural network is composed of simple elements called neurons that receive inputs, change their internal state based on that input, and produce an output based on an activation function. These systems can learn to detect complex patterns in data without using task-specific rules.

I employed a simple fully connected neural network using the nnet package in R. The models fit consist of a single hidden layer and use the logistic activation function. To reduce computational complexity, I selected the number of hidden units to use for all models fit by simple cross validation for one of the models. I also tested the weights option in the nnet function as an alternative method for dealing with class imbalance. The weight method directly changes how observations in each class are penalized in the loss function. The weight w value was also selected by simple cross validation.

V. Results and Discussion

Four subsamples of the training dataset were created using the sampling methods discussed in section IV (A). Simple cross validation was used to select the value of the proportion of minority class (p) using the PRAUC criterion. Given the substantial run times (hours) for the quadratic and Random Forest models, p was selected only on the basis of performance in the simple linear logistic regression models on the validation set. In an industrial setting, this parameter should be tuned separately for each model.

Figure 2: PRAUC vs. minority class proportion (simple logistic regression)



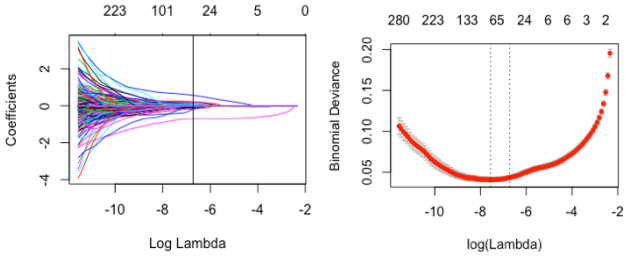
As shown in Figure 2, increasing p at first increases the performance of the sampling methods relative to fitting on the original training dataset (no sampling). As p continues to increase, performance declines in all of the sampling methods and eventually stabilizes in Oversampling and ROSE and continues to decrease in Undersampling and Both. As p increases, the sample size of the Undersampling dataset drops significantly due to the class imbalance, yielding significant loss of information and performance degradation. I find $p = 0.02$ leads to the best performing models. Using this value of p increases the proportion of minority samples by roughly a factor of 10 relative to the original dataset.

Table 4 shows the performance of all models assessed against the validation set after the first stage of training. In addition to AUPRC (our primary performance metric), area under the receiver operating characteristic (AUROC) curve is included for reference. In the linear logistic regression category, all of the sampling methods except Undersampling outperform the original training dataset. ROSE and Oversampling performed the best, with ROSE winning the category by a small margin.

Figure 3: LASSO plots for undersampling strategy

Coefficient shrinkage path (left);

CV results for regularization parameter (right)



In the quadratic logistic regression category, the optimal value of the regularization parameter λ was selected by 3- or 10-fold cross validation depending on the computational complexity of the model. Figure 3 shows coefficient shrinkage paths and error rates as a function of λ for the Undersampling dataset. From the right plot, we see evidence of significant overfitting when λ is close to zero. For the optimal values of λ, we see from table 4 that there is a significant improvement in PRAUC for the Oversampling method relative to the corresponding model without quadratic terms, indicating that expanding the feature space may improve model performance. The performance of Undersampling marginally increased, Both largely remained stable, while ROSE's performance decreased. The algorithm did not converge for the no-sampling group.

The RF models had great predictive performance. Every sampling method in this category outperformed corresponding methods in the other categories. Moreover, they were simple to train, though they had long run times. Within the RF category, oversampling modestly outperformed other sampling methods. The no-sampling group also performed well, suggesting that RFs may be robust to the class imbalance issue, at least with this particular dataset. Due to the high computational demands of training these models, I opted to use the default value for the parameter that controls the number of variables the model randomly considers at each tree split. In practice, one would tune this parameter with cross validation to optimize performance. Nonetheless, performance with the default value is impressive.

The neural network models had disappointing performance. As discussed in the methods section, I selected the number of hidden units by simple cross validation and found the best performance using three hidden units. These algorithms exhibited unstable performance across multiple runs, though the ROSE sample generally performed well and was the clear winner in this category. Due to the long run times, it was difficult to diagnose whether the poor performance was the result of insufficient tuning, the dataset, or another factor. Given these issues, I also tried the weighted option in the nnet package on the regular training set. I selected the value of the weights by cross validation and found that giving the minority class a weight of 100 times the majority class led to the best performance, but the final model yielded unsatisfactory results. In addition, I tried training the neural networks using other R packages without significant improvement. Given additional time and computing resources, I would investigate these performance issues further.

Table 5 and Figure 4 shows the performance of the finalist models on the test set after they were retrained using the combined data in the training and validation sets. All finalist models performed quite well, indicating that they generalize to new examples. The random forest combined with oversampling was the best performing model across almost every objective performance metric.

Table 4: Model performance on validation set

	AUROC	AUPRC
Simple logistic: linear terms only		
No sampling	0.9822538	0.6973303
Undersampling	0.9821079	0.6813981
Oversampling	0.9800541	0.7267684
Both	0.9844079	0.7061640
ROSE	**0.9792802**	**0.7301219**
Logistic: quadratic terms with LASSO*		
Undersampling	0.9827175	0.6915803
Oversampling	**0.9824838**	**0.7935913**
Both	0.9510685	0.7036836
ROSE	0.9628794	0.5475979
*No sampling did not converge		
Random forest		
No sampling	0.9540286	0.8448476
Undersampling	0.9595123	0.8177367
Oversampling	**0.9457366**	**0.8504548**
Both	0.9844079	0.8381937
ROSE	0.9160344	0.7703227
Neural network*		
Undersampling	0.9769187	0.6919809
Oversampling	0.6711547	0.2757658
Both	0.6848203	0.2979033
ROSE	**0.9611862**	**0.7118781**
Weighted option	0.6506091	0.2266319
*No sampling did not converge		

Bolded models denote the category winner

Table 5a: Finalist performance on test set

*Assumes decision boundary at prob > ½

	AUROC	AUPRC	Accuracy*	Sensitivity*	Specificity*	F1*
Linear logistic (ROSE)	0.9836798	0.8347616	0.9995	0.797619	0.999831	0.842673
Quad logistic (Over)	0.9839819	0.8840936	0.9993	0.734043	0.999873	0.816568
Random Forest (Over)	0.9839583	0.9095681	0.9997	0.929577	0.999810	0.904110
Neural Net (ROSE)	0.9828006	0.7316877	0.9995	0.831169	0.999768	0.842105

Table 5b: Confusion matrices*

Linear logistic (ROSE)

Pred \ Truth	0	1
0	47426	17
1	8	67

Quadratic logistic (Over)

Pred \ Truth	0	1
0	47418	25
1	6	69

RF (Over)

Pred \ Truth	0	1
0	47438	5
1	9	66

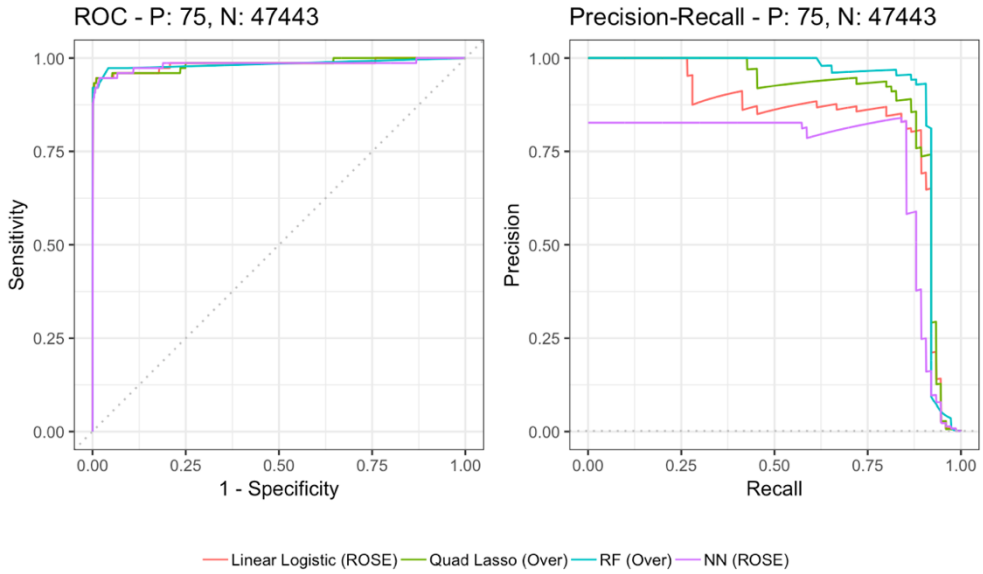
Neural net (ROSE)

Pred \ Truth	0	1
0	47430	13
1	11	64

VI. Conclusion

Payments fraud is a significant and growing issue in the United States but detecting it can be like finding a needle in a haystack. I find that oversampling and synthetic data generation methods, when properly tuned, can lead to superior predictive performance in the face of class imbalance. RFs are highly effective, easy to implement, and appear to be robust to class imbalance, at least in the Kaggle dataset. Given that payments fraud is constantly evolving, future areas of work might include the application of reinforcement learning to a real-time data stream. Although fraudsters will never retire, machine learning algorithms can give them a run for their money.

Figure 4: Finalist performance on test set: ROC and PRC



References

[1] Board of Governors of the Federal Reserve System. *Federal Reserve Payments Study finds U.S. payments fraud a small but growing fraction of overall payments*. 2018. Web. 11 June 2019.

[2] van Liebergen, Bart, 2017. "Machine learning: A revolution in risk management and compliance?" Journal of Financial Transformation, Capco Institute, vol. 45, pages 60-67.

[3] Laurikkala, Jorma. "Improving Identification of Difficult Small Classes by Balancing Class Distribution." AIME (2001).

[4] Siddhartha Bhattacharyya , Sanjeev Jha , Kurian Tharakunnel , J. Christopher Westland, Data mining for credit card fraud: A comparative study, Decision Support Systems, v.50 n.3, p.602-613, February, 2011.

[5] Roy, A., Sun, J., Mahoney, R., Alonzi, L., Adams, S., and Beling, P. Deep learning detecting fraud in credit card transactions. Systems and Information Engineering Design Symposium (SIEDS) (2018), pp. 129–134.

[6] Chaudhary, Khyati & Yadav, Jyoti & Mallick, Bhawna. (2012). A review of Fraud Detection Techniques: Credit Card. International Journal of Computer Applications. 45.

[7] Japkowicz, N., and S. Stephen. "The Class Imbalance Problem: A Systematic Study." Intelligent Data Analysis, vol. 6, no. 5, pp. 429–449.

[8] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie. Class-balanced loss based on effective number of samples. CoRR, abs/1901.05555, 2019.

[9] Worldline and the Machine Learning Group. Credit Card Fraud Detection. Retrieved April 25, 2019 from https://www.kaggle.com/mlg-ulb/creditcardfraud.

[10] Lunardon, Nicola & Menardi, Giovanna & Torelli, Nicola. (2014). ROSE: a Package for Binary Imbalanced Learning. R Journal. 6. 79-89. 10.32614/RJ-2014-008.

[11] Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani. An Introduction to Statistical Learning: with Applications in R. New York :Springer, 2013.

[12] Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi. Calibrating Probability with Undersampling for Unbalanced Classification. In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015.

[13] Dal Pozzolo, Andrea; Caelen, Olivier; Le Borgne, Yann-Ael; Waterschoot, Serge; Bontempi, Gianluca. Learned lessons in credit card fraud detection from a practitioner perspective, Expert systems with applications,41,10,4915-4928,2014, Pergamon.

[14] Dal Pozzolo, Andrea; Boracchi, Giacomo; Caelen, Olivier; Alippi, Cesare; Bontempi, Gianluca. Credit card fraud detection: a realistic modeling and a novel learning strategy, IEEE transactions on neural networks and learning systems,29,8,3784-3797,2018,IEEE.

[15] Dal Pozzolo, Andrea Adaptive Machine learning for credit card fraud detection ULB MLG PhD thesis (supervised by G. Bontempi).

[16] Carcillo, Fabrizio; Dal Pozzolo, Andrea; Le Borgne, Yann-Aël; Caelen, Olivier; Mazzer, Yannis; Bontempi, Gianluca. Scarff: a scalable framework for streaming credit card fraud detection with Spark, Information fusion,41, 182-194,2018,Elsevier.

[17] Carcillo, Fabrizio; Le Borgne, Yann-Aël; Caelen, Olivier; Bontempi, Gianluca. Streaming active learning strategies for real-life credit card fraud detection: assessment and visualization, International Journal of Data Science and Analytics, 5,4,285-300,2018,Springer International Publishing.

[18] Bertrand Lebichot, Yann-Aël Le Borgne, Liyun He, Frederic Oblé, Gianluca Bontempi Deep-Learning Domain Adaptation Techniques for Credit Cards Fraud Detection, INNSBDDL 2019: Recent Advances in Big Data and Deep Learning, pp 78-88, 2019.

[19] Fabrizio Carcillo, Yann-Aël Le Borgne, Olivier Caelen, Frederic Oblé, Gianluca Bontempi Combining Unsupervised and Supervised Learning in Credit Card Fraud Detection Information Sciences, 2019.