Capstone 2: Rare Event Detection Model

Sarah Sorlien, MD

2024-06-01

Contents

1 Foreword

I am a retired anesthesiologist and have extensive experience with electronic health records (EHR). I recall when they first came out and we were given the promise of endless opportunities for research and data analysis.

In fact, I was going to do my project using the unique and enormous MOVER database. This anonymized patient database contained EHR records in several files which included, uniquely, waveform data of various monitors of the cardiorespiratory system in the perioperative period. I got permission and downloaded the files (using wget) over the course of 2 days. Unfortunately most of the databases could not even be opened on my M3 Macbook Pro. I could inspect one file - the intraoperative data from an anesthesia program, SIS. For your reference, a handwritten anesthesia record is typically one page and includes graphical data, checkboxes and free text. A comparable anesthesia record documenting an hour or so of the events one patient experiences in a 90 minute operation recorded in SIS contains pages and pages of data.

From my new point of view after taking the courses for this certificate, I could see the inconsistent free text data and missing data so common to the EHR (and apparent in the MOVER dataset) was going to be a major hurdle for anyone trying to use EHR data for those endless opportunities for research we were promised.

In my inspection of the data, however, I realized any model that could predict, for example, patients at risk for surgical complications, would need to be able to predict rare events. For that reason I chose to do my project on a credit card fraud dataset.

2 Executive Summary

In this project, I developed a machine learning model using extreme gradient boosting to detect fraudulent credit card transactions. The dataset, provided by Kaggle, included 28 features generated by principle component analysis (PCA) from the original data. I evaluated and included the additional feature of Benford outliers of the first two digits of the Amount. As fraud was quite rare, comprising 0.17% of the data, I used synthetic minority oversampling (SMOTE) to balance the training set in order to improve model performance. I compared different strategies, using all features and selecting the top features based on their importance. My analysis showed that deploying a machine learning model for fraud detection can significantly reduce the cost associated with fraud. The model using top features performed comparably to the original model by statistical evaluation, however the cost savings were greater with the model trained on all features.

3 Introduction

Credit card fraud is costly to both consumers and businesses. Because of this, there has been a lot of energy directed towards creating machine learning models to detect fraud almost immediately at time the card is presented. The key issue to overcome in the detection of a fraudulent transaction is the rarity of the event. The training data is imbalanced. Because this is important to businesses, there are freely available anonymized datasets for learners to create training models. The dataset I used was from Kaggle and has 28 features generated by principle component analysis (PCA) from the original data. It also contained the features of Amount and Time.

Using this data gave me the opportunity to explore the use of machine learning models to detect rare events. In particular I examined using a method to balance the data in the training set to improve the model performance. I also examined the use of Benford outliers as a feature in the model.

Benford's Law‘ states that the first digit of a number is not uniformly distributed, and many naturally occurring data have well known distributions of the first and subsequent digits. This is a useful tool in fraud detection as it can be used to find transaction amounts that deviate significantly from the expected frequency distributions. I used the first two digits of the Amount to determine if Benford outliers correllated with Class (fraud vs not fraud.) For my purposes, I can see that as useful for fraud detection of data presented in published medical research, which, unfortunately, may not be as rare as surgical complications.

Finally, I concerned myself with determining how the model performed when limiting the features. This has an application in clinical medicine as well. A model with fewer features is easier for a clinician to understand and implement and may save time and money in the long run.

In this paper, I will describe the data, the methods I used to balance the data, the methods I used to generate and evaluate the model, and the results of the model. I will also discuss the implications of the results and future directions I intend to explore.

4 Data

I have provided the creditcard.csv file with my report. The data is derived from 2 days of credit card transactions in September 2013 by European cardholders. The data is reported to contain 28 features generated by PCA, as well as the features of Amount and Time in seconds since the beginning of time period presented. The data is anonymized, so the features cannot be interpreted specifically. It is presumed that the PCA features are derived from the transaction data including such items as who presented the card, how it was presented, time of day etc. There are many ways to generate this sort of raw data into new features to look for anomalous transactions. That will not be a focus of this paper.

So I first inspected the data to see if it lived up to Kaggle's promises.

```
## Number of rows:  284807
```

```
## Number of columns:  31
```

```
##
## PCA Variables Statistics:
```

```
## Means for V1-V28 are approximately equal to 0.
```

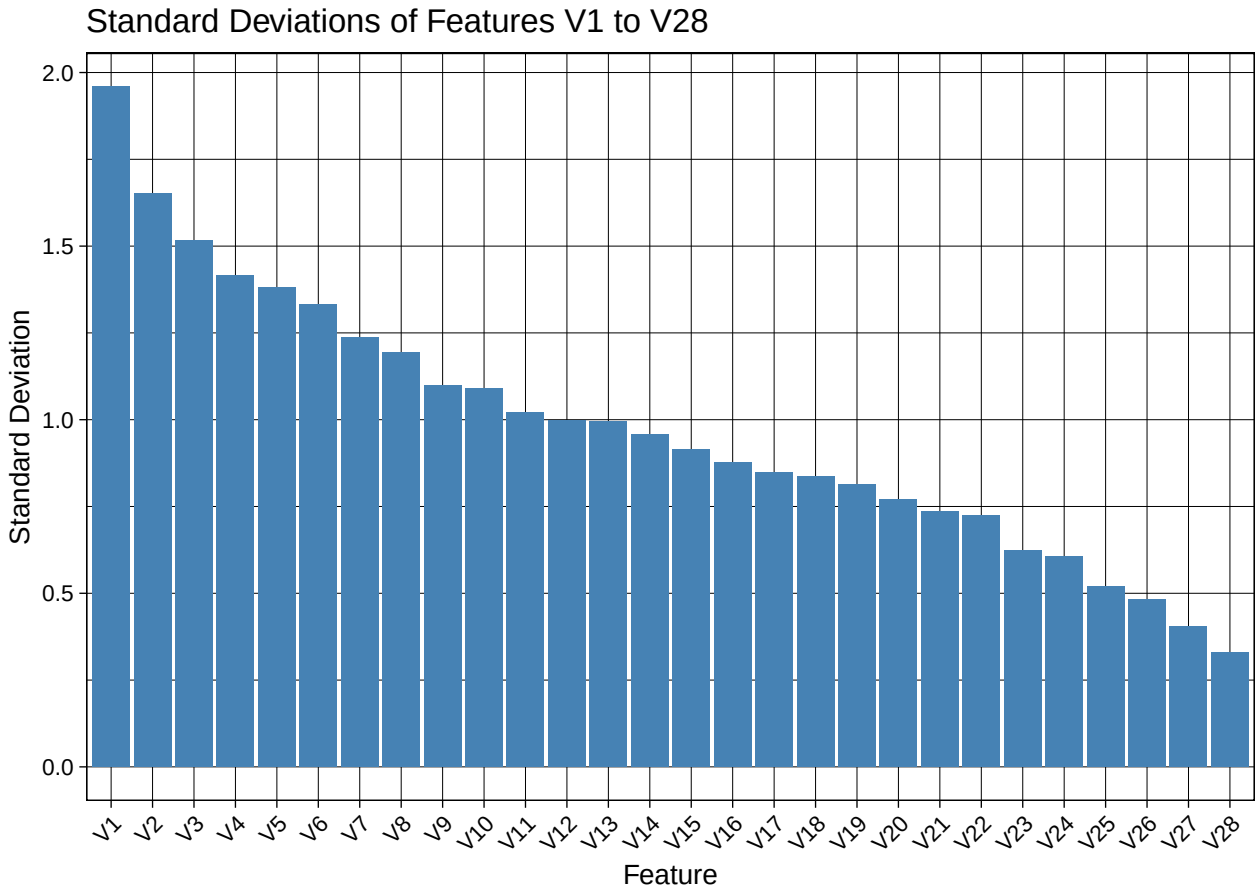
The data contains the advertised 31 columns and 284,807 rows. The columns are labeled V1 to V28, Amount, Time, and Class. The V columns are the PCA features and appear to be regularized as expected with a mean of 0. Amount is the transaction amount, Time is the time in seconds since the beginning of the time period. The target variable, Class, is a binary option, with 0 representing a non-fraudulent transaction and 1 representing a fraudulent transaction.

I checked then for any missing data.

```r
# Check for missing values
missing_values <- suppressWarnings(sapply(data, function(x) sum(is.na(x))))
if (all(missing_values == 0)) {
  cat("There are no missing values.")
} else {
  cat("There are missing values.")
}
```

```
## There are no missing values.
```

That's reassuring. After looking at the very messy EHR data a tidy dataset was quite refreshing. I checked the standard deviations of the features determined by PCA to see if the features were numbered and ranked in the order of their contribution to explaining the variance of the data as expected following principle component analysis.



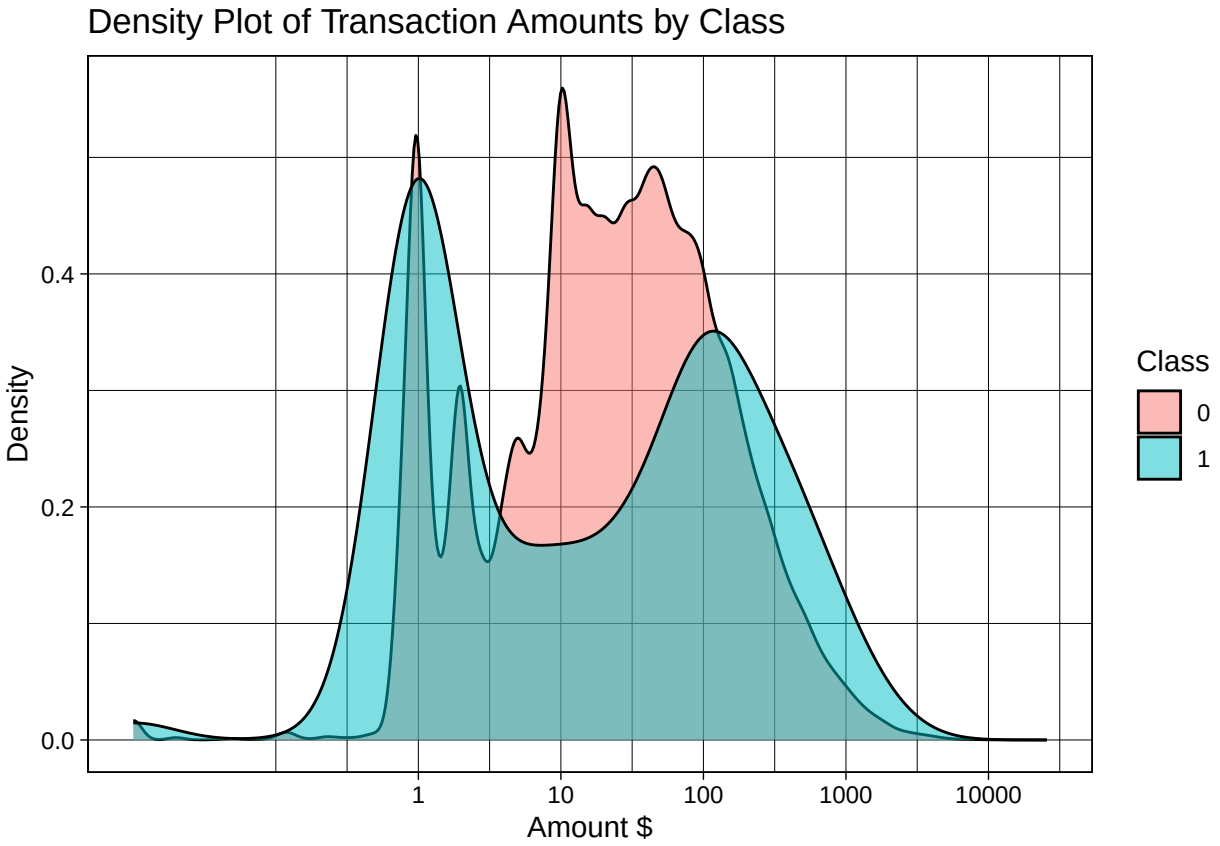
So here I appear to have a tidy dataset with no missing data. The PCA features are regularized as expected. The features are ordered by importance as expected in PCA. Now just to check the distribution of the target variable to see if the dataset is imbalanced as promised.

```
##   Class      n
## 1     0 284315
## 2     1    492
```

```
## Percentage of fraudulent transactions:  0.17 %
```

The dataset is indeed imbalanced, with fraudulent transactions comprising only 0.17% of the data. This is a common issue in anomaly detection problems. I needed to address this issue when training the machine learning model. So the dataset appears to be as advertised. I do want to understand it better with some exploratory data analysis particularly to see if there is any opportunity eke more information from from the Amount feature.

First I examined if the distribution of the Amount feature is different for fraudulent and non-fraudulent transactions. I created a subset of the data with only positive amounts and plotted the density of the transaction amounts by class. I used a logarithmic scale to allow for a more detailed view of the distribution of transaction amounts, especially for lower value transactions.



The density plot shows that fraudulent transactions tend to cluster within the $10 to $100 range. Specifically, the density of fraudulent transactions (Class 1) is significantly higher within this range compared to non-fraudulent transactions. From this I expected the Amount feature itself will be important to the model.

I wanted to see if I could wring any more information out of the Ammount feature by examining the distribution of the first two digits of the Amount for adherence to Benford's Law. Benford's Law describes an expected distribution of first digits of naturally occurring, random (not assigned) numeric data, and it applies to many types of data including monetary transaction data. I used Benford's Law to determine if the combination of the first two digits of the Amount feature are outliers based on the expected distribution according to Benford's Law. This is a common method for detecting human made fraudulent data. In fact, there is a very elegant package for R that calculates the feature I am using: the Benford Outlier. I calculated whether the first 2 digits of each amount fell within 3 standard deviations from the expected mean of the Benford distribution.

I used the Benford.analysis package to perform the analysis and plot the results. I then extracted the Benford Outliers and added a column to the data indicating whether an amount was categorized as a Benford_Outlier or not. Finally, I looked at the relationship between Benford Outliers and fraudulent transactions by creating a contingency table of Benford Outliers vs. Class.

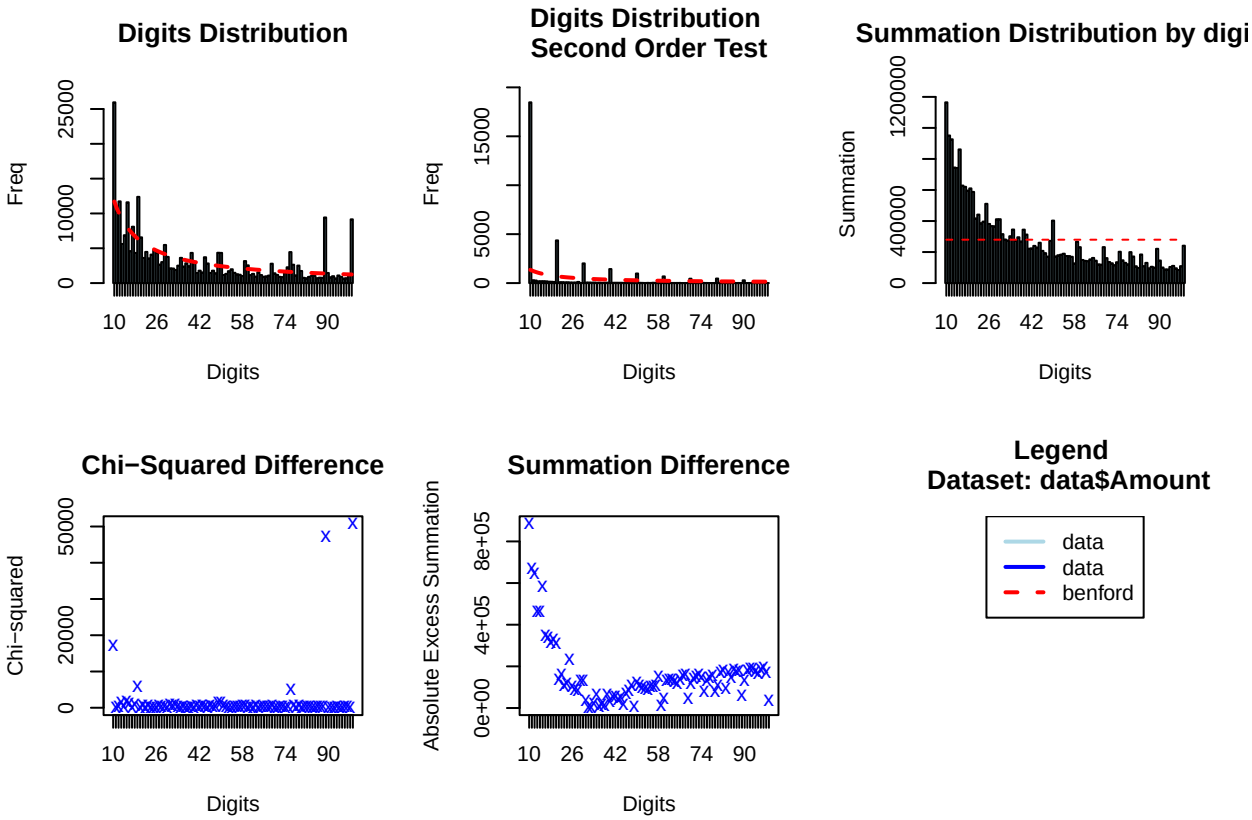


Table 1: Contingency Table of Benford Outliers vs. Fraudulent Transactions

	No Fraud	Fraud	No Fraud (%)	Fraud (%)
Not Outlier	249065	359	87.6	72.97
Benford Outlier	35250	133	12.4	27.03

Simple inspection suggests that the Benford Outliers are significantly more likely to be fraudulent transactions which was confirmed by Chi Squared Analysis. I decided this could be a useful feature to include in the model. In this case it is appropriate to preprocess the balanced training sets and test sets separately to add the Benford Outlier feature.

5 Methods

5.1 Preprocessing and Featurization

All preprocessing of the data needs to take place after splitting the data into training and test sets. I used a 70/30 split for training and testing. The next step was to balance the training set. There are several ways to do this, but I chose to use the Synthetic Minority Oversampling Technique (SMOTE) to balance the training set. SMOTE is a popular method for oversampling the minority class by generating synthetic samples. I used the `SMOTE` function from the `smotefamily` package to create an oversample the minority class in the training set. The synthetic values are generated by looking at sets of 5 nearest neighbors of each of the minority class and creating a new value interpolated between the minority class data point and its nearest neighbor without duplicates. This is a common form of increasing the number of data points in the minority class and was handled using the SMOTE function.

I then checked the structure of the balanced training set and the class distribution of the balanced training data to confirm that the the SMOTE function has worked as expected to produce a balanced training set (balanced_train) with identical columns to the original.

```
## Number of rows in balanced training set: 397852
```

```
## Number of columns in balanced training set: 31
```

Table 2: Class Distribution in Balanced Training Data

Class	Count
0	199020
1	198832

The balanced training set has 398,040 rows and 31 columns, which is the same as the original dataset. The class distribution of the balanced training data shows that the classes are now balanced, with 199,020 transactions in each class. The next step was to calculate the Benford Outliers for the balanced_training and test sets and add the Benford Outlier feature.

```
# Calculate Benford Outliers for Training Set
benford_results_train <- benford(balanced_train$Amount, number.of.digits = 2)
benford_outliers_train <- getSuspects(benford_results_train,
                                      data.frame(Amount = balanced_train$Amount))
balanced_train$Benford_Outlier <- balanced_train$Amount %in% benford_outliers_train$Amount

# Calculate Benford Outliers for Testing Set
benford_results_test <- benford(test_data$Amount, number.of.digits = 2)
benford_outliers_test <- getSuspects(benford_results_test, data.frame(Amount = test_data$Amount))
test_data$Benford_Outlier <- test_data$Amount %in% benford_outliers_test$Amount
# Ensure columns are in the same order
common_cols <- intersect(names(balanced_train), names(test_data))
balanced_train <- balanced_train[, common_cols]
test_data <- test_data[, common_cols]
```

5.2 Model Development

I had chosen to use an extreme gradient boost model using the XGBoost package. This is an ensemble model that learns to make classifications based on iterations of decision trees. In this way many weak models combine to provide a more accurate model. It is fast, lightweight, reliable, and, honestly, really fun to watch while it is going through iterations. The XGBoost model has become very popular for these reasons.

Before using the model, the dataframe must be transformed to a DMatrix which differs from a typical dataframe in several ways that help to speed up the model. In the DMatrix, the data is stored column-wise. The DMatrix is excellent at handling data that contains a lot of zeroes through compression, storing the data in a dense format. The DMatrix also stores the labels in a separate vector. I used the `xgb.DMatrix` function from the `xgboost` package to convert the balanced training and test sets to DMatrix format.

Using the xgb.train() function from the xgboost package, I trained the XGBoost model on the balanced training set. I used the `xgb.train` function using AUC (area under the curve) as the evaluation metric. Other parameters I used included identifying the model as "binary:logistic" and asking the model to run through 100 rounds of training. These parameters and others can be tuned to improve the model. You can also introduce an instruction to stop early if the model is not improving after a number of iterations. I used the `watchlist` parameter to monitor the performance of the model on the training and test sets during training. I set the `verbose` parameter to 1 to display the training progress (which, as I mentioned, is fun to watch, but I will not include that output in this report).

6 Results

The XGBoost model was trained on the balanced training set and evaluated on the test set. Calculating the AUC on the test set shows the model at this point is pretty accurate.

```
## AUC:  0.9801828
```

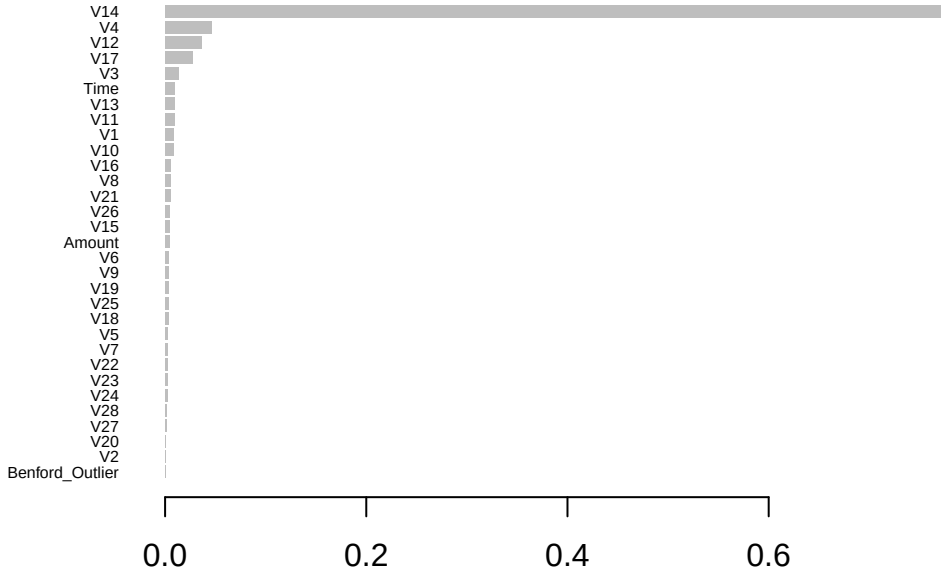
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 85268    23
##          1    27   125
##
##                Accuracy : 0.9994
##                  95% CI : (0.9992, 0.9996)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.833
##
##  Mcnemar's Test P-Value : 0.6714
##
##             Sensitivity : 0.9997
##             Specificity : 0.8446
##          Pos Pred Value : 0.9997
##          Neg Pred Value : 0.8224
##              Prevalence : 0.9983
##          Detection Rate : 0.9980
##    Detection Prevalence : 0.9982
##       Balanced Accuracy : 0.9221
##
##        'Positive' Class : 0
##
```

```
## Precision:  0.9997303
```

```
## Recall:  0.9996835
```

```
## F1 Score:  0.9997069
```

The model achieved an AUC of 0.98 on the test set, which is a good result. The confusion matrix shows that the model correctly classified the majority of transactions. The precision, recall, and F1 score are also high, indicating that the model is performing well. I also examined the feature importance of the model to see if I could reduce the number of features and still maintain a high level of performance. This was trivial to do with the xgboost package. I used the `xgb.importance` function to calculate the feature importance and the `xgb.plot.importance` function to plot the top 28 features.



The plot shows the top 32 features based on their importance in the model. Sadly my Benford_Outlier contributed very little to the model which I suspect meant it had already been integrated in the orginal features in some way. I then selected the top 15 features to train a simplified model. I used the xgb.importance() function to generate and importance matrix from which to select the top 15 features. I then made a subset of both the balanced training and test sets to include those 15 features. I then prepared the data for XGBoost with the top features and trained the model on the top features. I then evaluated the model on the test set using the same metrics as before.

```
## AUC (Top Features):  0.9734684
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 85251    23
##          1    44   125
##
##                Accuracy : 0.9992
##                  95% CI : (0.999, 0.9994)
##     No Information Rate : 0.9983
##     P-Value [Acc > NIR] : 6.467e-14
##
##                   Kappa : 0.7883
##
##  Mcnemar's Test P-Value : 0.01455
##
##             Sensitivity : 0.9995
##             Specificity : 0.8446
##          Pos Pred Value : 0.9997
##          Neg Pred Value : 0.7396
##              Prevalence : 0.9983
##          Detection Rate : 0.9978
##    Detection Prevalence : 0.9980
##       Balanced Accuracy : 0.9220
##
##        'Positive' Class : 0
##
```

```
## Precision (Top Features):  0.9997303
## Recall (Top Features):  0.9994841
## F1 Score (Top Features):  0.9996072
```

The two models appear statistically to be very similar and they both identify the same amount of true positives and false positives. If there are two models that provide similar results and one used about half the features you might conclude we would save time and computing resources without sacrificing much in the way of performance. But desired outcome isn't a statistic in the real world - it is measured in dollars.

Looking at a hypothetical cost to deploy the model and a hypothetical cost of the misery of a false positive, I calculated the cost of each model naturally using the actual amount of the missed fraud for the cost there. Of course I had the hypothesis that the top feature model would be less expensive to deploy. From this analysis we can see how much less expensive it would have to be to be the better choice.

My assumptions were the deployment cost of the model was $5000, the cost of a false positive was $100, and the cost of any missed fraud was the actual amount of the missed fraud.

```
## Total Fraud Amount (No Model): $ 15713.83
## Original Model Cost: $ 11206.64
## Top Features Model Cost: $ 12906.64
```

7 Discussion

As a physician, I am well aware of the apparent paradox of positive predictive value when a very sensitive test is applied to a population where very few people have the disease. It is possible for a positive test to mean the person probably does not have the condition. This is the bane of population screening tests.

In translating this to the model of fraud detection, it is important to look at the real world endpoint rather than solely a statistical measure of performance. There are costs to missing the fraud, to incorrectly classifying the fraud, to deploying the model and to aggravation and poor performance if the detection is too slow. Such an analysis can help you decide what model to deploy as well as set a benchmark for how much cheaper a more lightweight model will have to be to make a difference.

So if we just ignored fraud because it was such a rare event, in two days in Europe the company loses almost $16,000. Assuming the original model cost only $5000 to deploy, and each false positive loses you $100 of good will and administrative costs, Those 2 days will only cost you $11,207. Sounds good!

The top features model, which looks so similar statistically to the original model, costs $12,907, or $1700 more than the original model. So the original model is the better choice if the top features model costs more than $1700 less to deploy. This is of course an example with a lot of oversimplification. In a real world situation, I would make a set of assumptions of costs based on existing data then refine them by analyzing the ongoing data to correct the assumptions as needed.

8 Conclusion

In this project, I developed a machine learning model using extreme gradient boosting to detect fraudulent credit card transactions. I used the XGBoost package to train the model on a training set balanced using SMOTE and evaluated the model on the test set. I also examined the use of Benford outliers as a feature in the model and compared the performance of the model using all features to a model using only the top 15 features. I found that the model using all features performed better than the model using only the top features in terms of cost. This is a good example of how a model that performs well statistically may not be the best choice in the real world.

9 Future Directions

I hope to apply this method to EHR data in the future in looking for predictors of rare events. In medicine I believe that the use of machine learning models to detect anomalies can be a valuable tool to predict rare surgical complications or aid with the diagnosis of rare disease. I also believe that the use of Benford outliers can be a useful feature in models for detection of fraudulent medical research data. I also would like to use methods I learned in the DataCamp course for featurizatioin of time of day. Unfortunately that could not be used with this dataset.

Avenues I did not explore in this project include tuning the hyperparameters of the XGBoost model to improve performance, using other methods to balance the training set, and exploring other features that may improve the model. For the latter, I would need to use data that was not already so highly preprocessed.

Of course I also realize if I plan to go forward working on EHR data I will need to leverage my domain knowledge of how electronic health records data is collected. (Usually collected by overtired and overworked professionals who are not particularly trained in data collection - they are more on the saving lives pathway.) That kind of knowledge will be critical in cleaning the messy data that is abundant in electronic health records. I expect I will need several more online classes to get all the tools needed there.

Finally I want to say how grateful I am to Dr. Irizarry and the staff at HarvardX/edX for providing this pathway to certification. I have learned so much and have a new appreciation for the power of machine learning in medicine. Taking these courses has been an amazing way to start my retirement and, I hope, a second act in data science. Thank you.

10 References

I do not know the best way to indicate the help I received by taking classes via DataCamp. I was introduced to DataCamp through HarvardX and went on to get my own account. I used these classes to supplement my learning throughout all the courses for this certificate. Specifically for this project I took courses:

Extreme Gradient Boosting with XGBoost (Python) with Sergey Fogelson Fraud Detection in R with Bart Baesens, Sebastiaan Höppner, and Tim Verdonck This course went over the Benford Analysis package, ways of balancing the data, and concepts of featurization that I could not do with this dataset but will be useful in the future. It introduced the concept of cost analysis as well. I practiced what they taught in the course, but I do not believe my code and process is a direct copy of the course material.

Other references: Benford.analysis package documentation: https://cran.r-project.org/web/packages/benford.analysis/benford.analysis.pdf SMOTE: Synthetic Minority Over-sampling Technique: https://arxiv.org/abs/1106.1813 XGBoost: A Scalable Tree Boosting System: https://arxiv.org/abs/1603.02754 https://medium.com/data-reply-it-datatech/fraud-detection-modeling-user-behavior-6d4f7bba1422

Schaefer, J., Lehne, M., Schepers, J. et al. The use of machine learning in rare diseases: a scoping review. Orphanet J Rare Dis 15, 145 (2020). https://doi.org/10.1186/s13023-020-01424-6

Samad M, Angel M, Rinehart J, Kanomata Y, Baldi P, Cannesson M. Medical Informatics Operating Room Vitals and Events Repository (MOVER): a public-access operating room database. JAMIA Open. 2023 Oct 17;6(4):ooad084. doi: 10.1093/jamiaopen/ooad084. PMID: 37860605; PMCID: PMC10582520. https://mover.ics.uci.edu/index.html

Data Source: https://www.kaggle.com/mlg-ulb/creditcardfraud

And a note about AI: I do have both codeGPT and used chatGPT 4 and 4o. codeGPT: Like the kid in class who always has his hand raised. Kind of annoying for this project especially when it tries to write your Rmarkdown file. I constantly had to delete even whole paragraphs of text it wrote. And the text it produces is very generic and not at all what I wanted to say. It was helpful in quickly filling in bits of code with correct "punctuation". I have used it more successfully to speed up writing Python code for my other projects that are not coursework but even there it is wrong often enough that if you don't know what you are doing it is a mess. I would not recommend it for a beginner.

chatGPT 4 and 4o: I used these like a souped up Google search primarily. It helped me get a deeper understanding of things like SMOTE, PCA etc. I could not only ask for a definition, but I could ask it questions about how I understood the implications of the definition. Also, I put in results and my interpretations to see if I was on the right track. It was enormously helpful when I got errors (pasting in the code and the error message) and it would give find the unmatched parenthesis or explain how my syntax was wrong. All that without the snark of StackOverflow. As I am taking the class at home without any student colleagues it gave me the feeling of sitting around the dorm room with three clever, kind and patient friends all working together.

It also helped me tremendously with my RMarkdown. This is only my second RMarkdown document. I got an idea of things it could do from grading the student Movielens projects. ChatGPT 4o was super with teaching me how to use more of the features of RMarkdown. I would recommend it for a beginner. Also I could paste in my graph and say - how can I make this prettier? It would give me a list of things to try.

So all in all, the Code is mine but I owe a lot of my knowledge base and any beauty you have found here to the AI. I suppose we will soon develop a standardized method of citing AI, but for the time being I thought my narrative would serve.