Exploring Class Imbalance with Fraud Detection

James Bush

Springboard

Capstone II

2021-03-16

TABLE OF CONTENTS

Fraud Detection and Imbalance

The Board of Governors of the Federal Reserve System reported that 44.7 billion credit card payments occurred in 2018 for a total of almost $4 trillion USD (FED, 2019). The Nilson Report, a ‘Leading publication covering payment systems worldwide,’ reports that in 2018, 7 card issues made up the top 79% of purchase volume. American Express, the top card issuer on that list, owned 21% of that market share (Nilson, 2021).

The dataset discussed in this paper covers a time span of 2 days, with 0.172% of transactions classified as fraudulent. For American Express in 2018, 2 days of transactions averages to 51.4 million transactions - for a value of $4.6 billion. Using the ratio above, 2 days of fraudulent activity equates to 88,850 fraudulent payments for $7.9 million. In other words, if a Machine Learning (ML) model is trained on these transactions, it would only have around 17 fraudulent transactions for every 10,000 genuine transactions to learn from.

This problem is known as, ‘Class Imbalance,’ and occurs when ML models are used to solve Classification problems. Class Imbalance is when the distribution of samples across classes is greatly skewed. In binary classification, the larger sample class is referred to as the ‘Majority Class,’ and the smaller sample class as the ‘Minority Class.’ Class Imbalance can occur as a property of the domain or caused by data handling. For example, poor sampling practices or misclassified entries during data handling might alter a dataset and create a class imbalance. Some problem domains like spam email detection, rare disease prediction, and fraud detection have imbalanced classes resulting from the environment itself. As a business gets controls in place for spam email detection, the number of genuine emails will begin to dwarf the number of fraudulent ones. With the occurrence of rare diseases, the rarity of the disease results in a limited number of infections for consideration. In fraud detection, like spam detection, as controls reduce the occurrence of fraudulent activity, the number of fraudulent cases become overshadowed by genuine activity – however, as seen above, the *value* for a business related to fraudulent activity remains substantial.

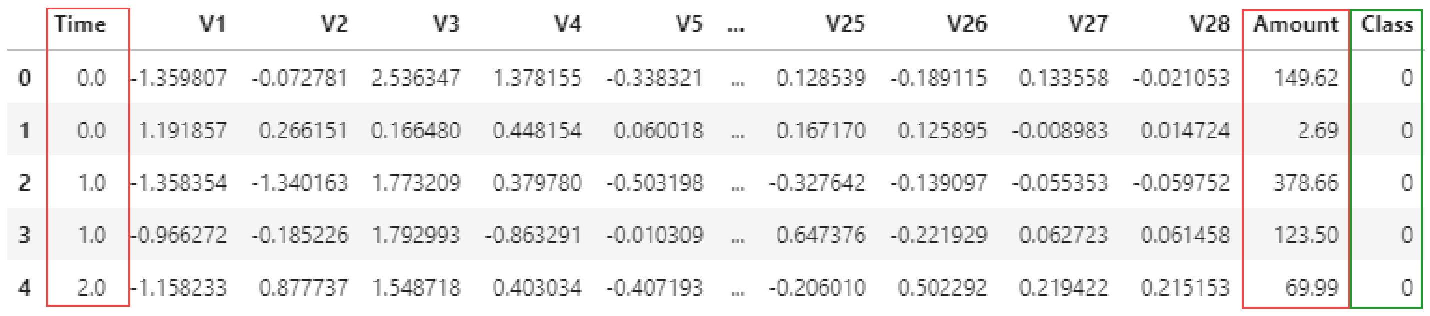
Class Imbalance creates challenges in ML modeling because the imbalance introduces a bias in the feature weight, as well as a bias in the domain representation. Feature imbalance can lead to overfitting, where a model learns the feature set too well and negatively impacts the model’s predictive power. Significant imbalance in the dataset domain has a negative influence on model scoring – for example, when the majority class is imbalanced enough, models scored with

Accuracy scores will predict the majority class almost exclusively. In other words, the fraud detection model would *only* be able to correctly identify genuine transactions.

Exploratory Data Analysis

The data used for this essay is titled, "Credit Card Fraud Detection. Anonymized credit card transactions labeled as fraudulent or genuine," compiled by the Machine Learning Group of ULB and retrieved from Kaggle, Inc. (Kaggle, 2021). The dataset was collected and analyzed as a part of a research collaboration of Worldline and the Machine Learning Group of ULB on big data mining and fraud detection.

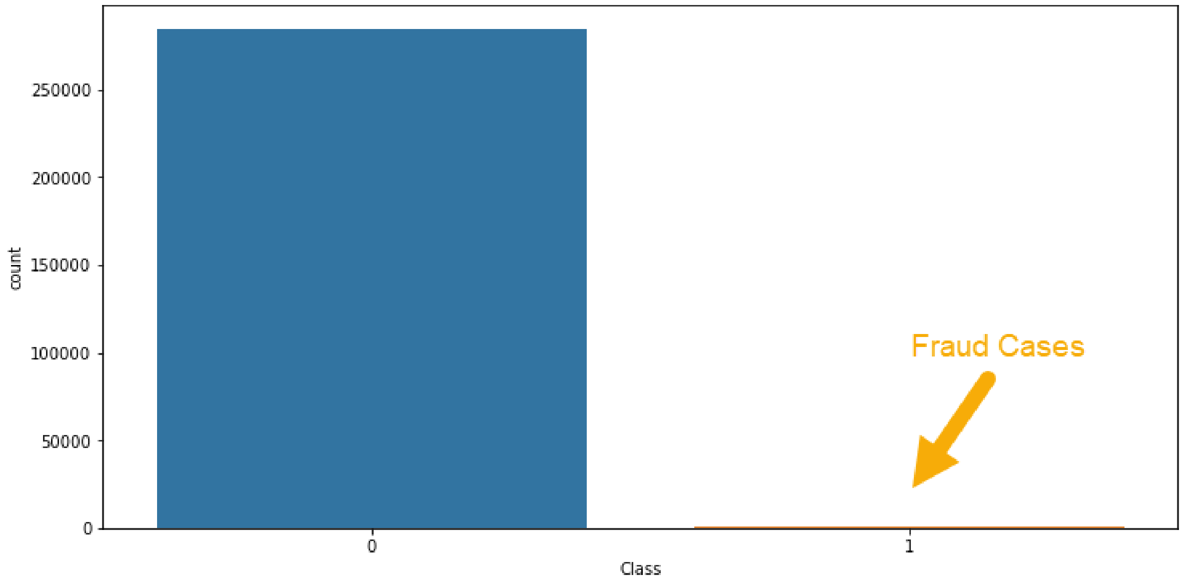
This dataset contains 284,315 majority class *genuine transactions,* and 492 minority class *fraudulent transactions*, for a total of 284,807 transactions. There are 30 independent variables, or features, available for input. The dependent variable (Also called a *response* variable) is the feature 'Class'. Due to confidentiality issues, 28 of the 30 features of the dataset (V1-V28) have been replaced with numerical input variables transformed by Principal Component Analysis (PCA) prior to being released as a public dataset.



	Time	V1	V2	V3	V4	V5	...	V25	V26	V27	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	...	0.128539	-0.189115	0.133558	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	...	0.167170	0.125895	-0.008983	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	...	-0.327642	-0.139097	-0.055353	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	...	0.647376	-0.221929	0.062723	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	...	-0.206010	0.502292	0.219422	0.215153	69.99	0

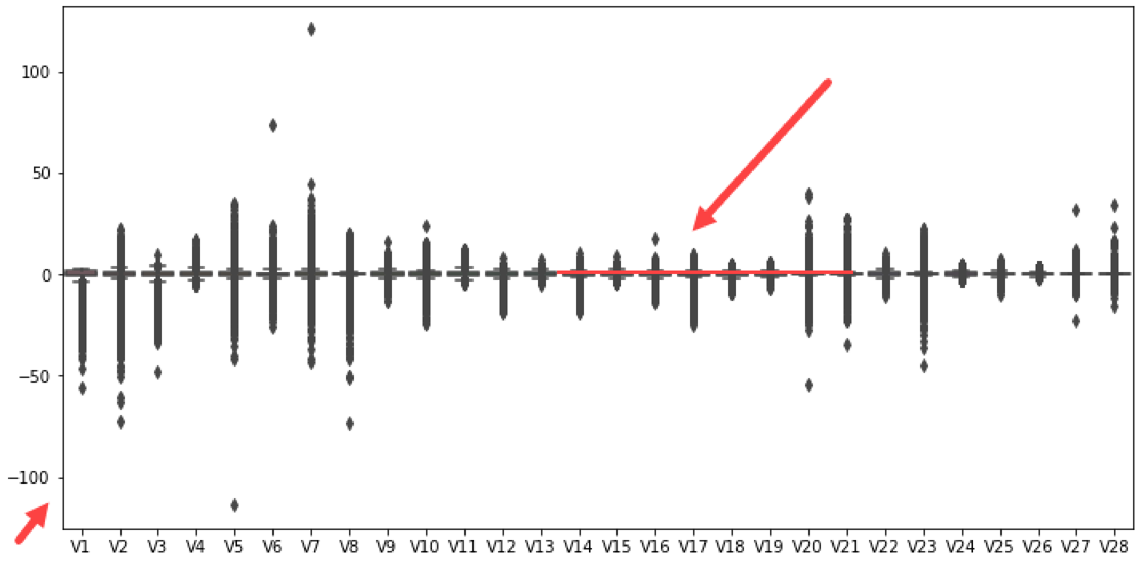
The two features not transformed by PCA are 'Time,' and 'Amount'. The feature 'Time' contains, "The seconds elapsed between each transaction and the first transaction in the dataset," and the feature 'Amount' is the transaction amount. The max 'Time' value is 172,792 *seconds*, showing that the dataframe covers a period of 2 days. The highest transaction amount is $25,691.16, while the average transaction amount is $88.35. The 'Amount' standard deviation is $250 (Kaggle, 2021).

The majority *genuine transactions* are labeled as 0. The minority *fraudulent transactions* are labeled as 1.



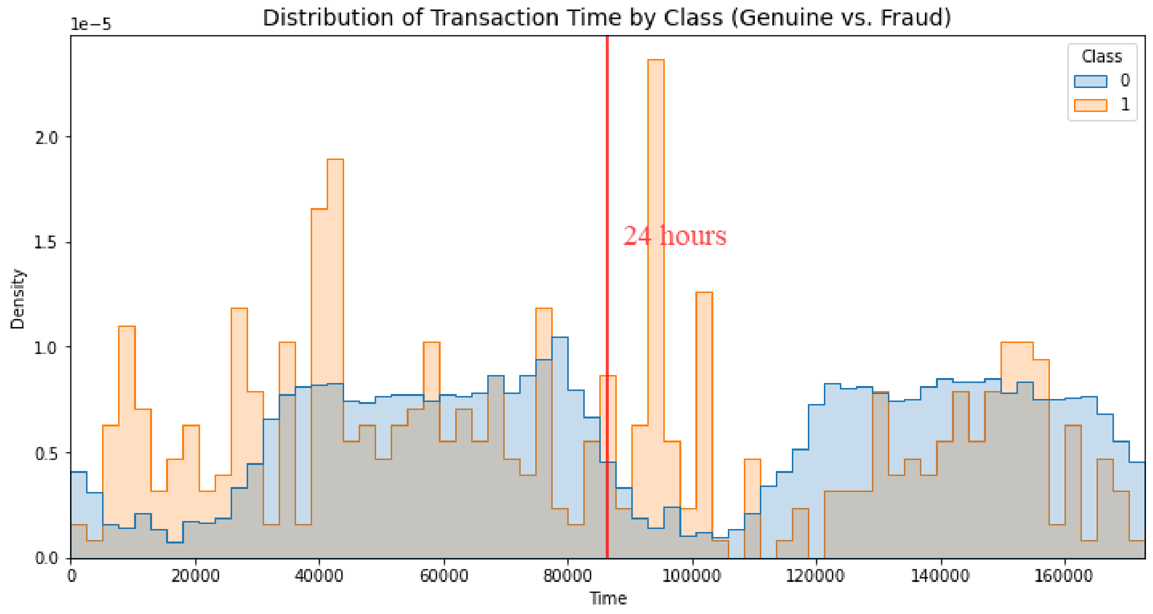
Class Counts by Dependent Variable.

It is a generally accepted practice to apply feature scaling to a dataset prior to performing PCA. Some algorithms are more sensitive to feature scaling than others. Checking boxplots for the PCA-transformed features shows that features are centered on 0, with varying minimum and maximum ranges.

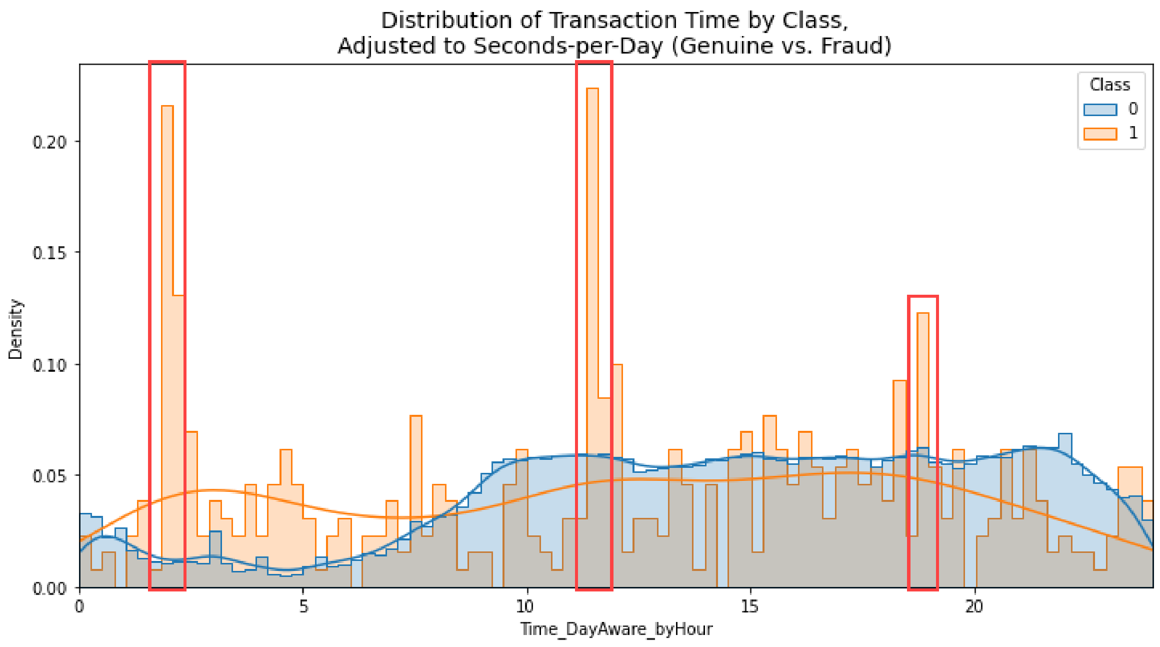


Box Plots of PCA-Transformed Features

Time. The feature 'Time' is recorded as the amount of seconds elapsed between each transaction and the first transaction in the dataset. Breaking this down into seconds-per-day, the feature space can be divided into two 24-hour periods.

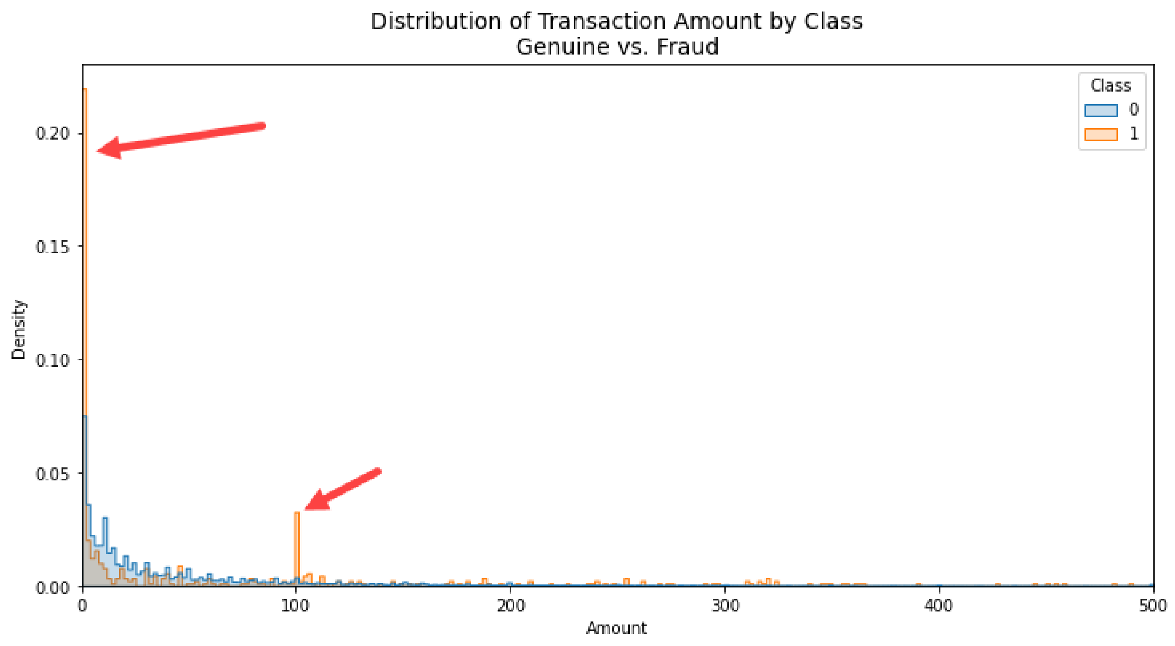


By resetting the 'Time' values over the *24-hour-as-seconds* value (86,400) to x-86400, the second period of values can be transformed into being, 'Day-Aware.' In other words, the feature 'Time' is interpreted with respect to a 24-hour cycle.



Feature 'Time' Adjusted to 24-Hour Periods, with Units as Hours-from-Period-Start.

Amount. The values in the ‘Amount’ feature are heavily skewed to the lower end of the value range. When the ‘Amount’ density distribution is visualized by class and zoomed in to capture an interval between $0 and $500, two significant markers can be observered. A closer look shows the bins aggregating at $1.00 and $100, with a 1:2 and 1:5 genuine-to-fraud ratio, respectively.



Amount Zoomed to Interval of 0, 500 to Show Class Difference Significance at $1, $100 bins.

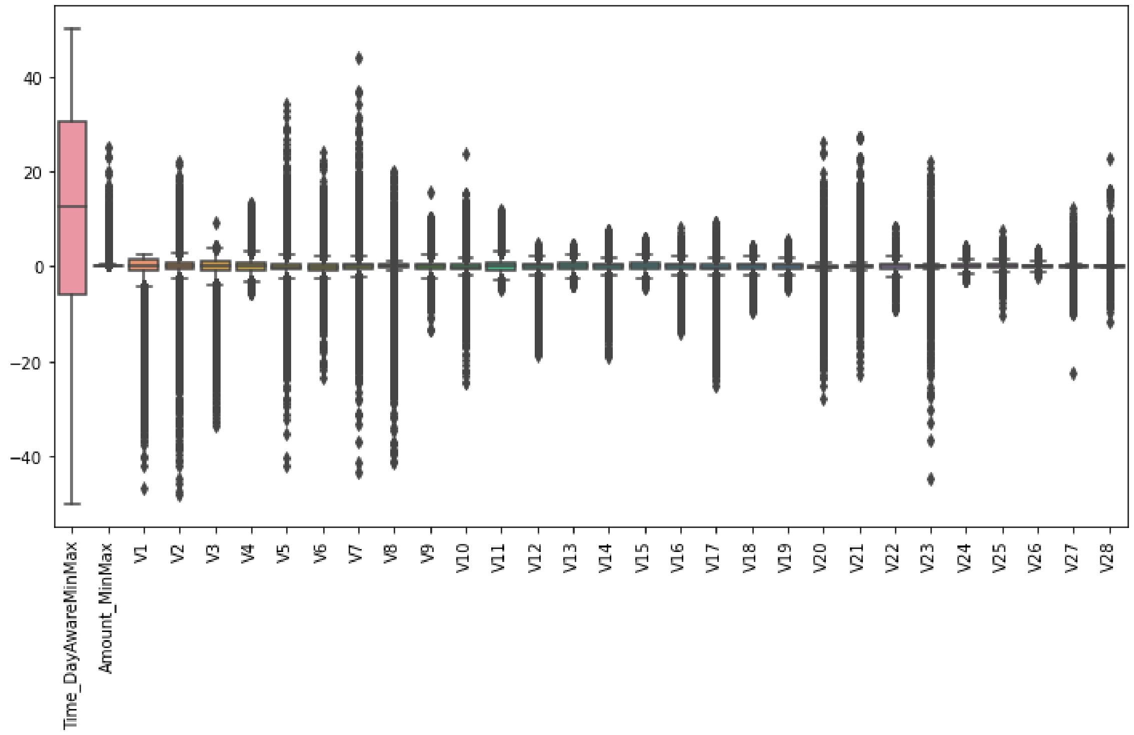
Preprocessing

Preprocessing is used to transform data into a readable format for ML algorithms. Preprocessing can range from activities like separating a feature into smaller parts, removing outliers based on a threshold, or adjusting the scale and unit of a feature. The latter is formally known as Feature Scaling and generally has two forms: Normalization (Where the feature is scaled to a range of 0 to 1), and Standardization (Where the feature is scaled based on the mean and standard deviation). Below, two datasets are created for modeling: 1.) A ‘not-scaled’ sample set where the only preprocessing is the adjustment to the ‘Time’ feature discussed above; and, 2.) A ‘scaled’ sample set, where the ‘Time’ is adjusted to ‘Time_DayAware’, both ‘Time_DayAware’ and ‘Account’ features are normalized, and outliers removed from all features greater than [-50, 50].

Different algorithms are sensitive to different methods of feature scaling (With some even requiring a specific method). For example, Logistic Regression, a Gradient-Decent based algorithm, is sensitive to scale because different scales across features influence the regularization. Support-Vector Machines (SVM) are a distance-based, and are most heavily effected by scale because larger distances impact the time it takes to get returns. Tree-based algorithms like Random Forests are *invariant* to scale – the algorithm's output is unchanged if the feature's scale is changed by a common factor.

Feature Scaling. In the second sample set, the 'Time_DayAware' and 'Account' features are both normalized with Sklearn.MinMaxScaler. 'Time_DayAware' is set to a range of [-50, 50], since the data distribution has two tails. The 'Account' distribution only has one tail, so the MinMaxScaler for 'Account' is set to [0, 50].

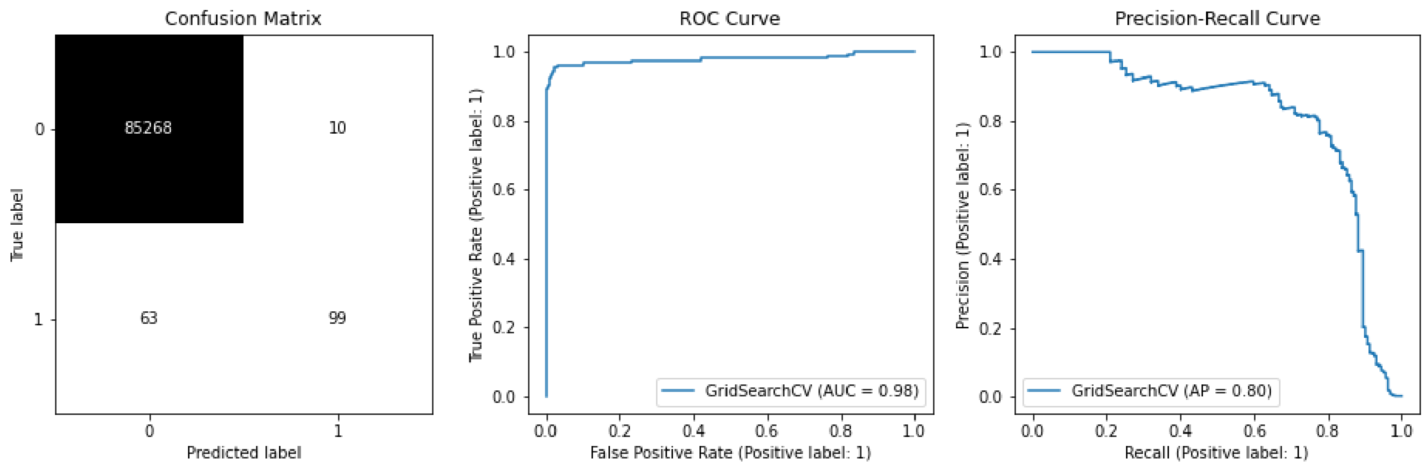
Drop Outliers. Outliers are dropped above and below the [-50,50] threshold for ease. As seen below, *there might be better markers for dropping outliers*, as the quartile lines from the PCA-transformed boxplots are still not visible.



Boxplots show the result after Feature Scaling, Dropping Outliers.

Data Sampling

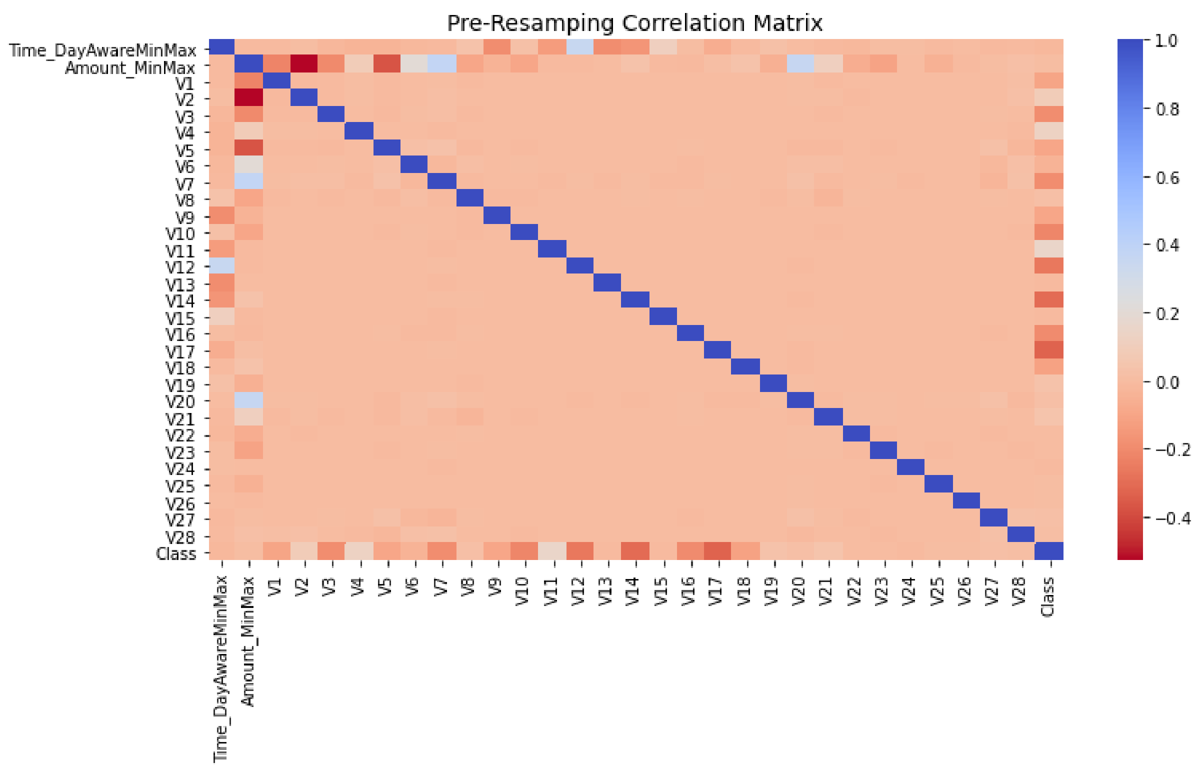
One way to approach a Class Imbalance problem is to resample. Data resampling can be used to make the representation between the two classes significantly more balanced. Three different ways the data can be sampled to stabilize a class imbalance are: 1.) Undersampling, 2.) Oversampling, and 3.) Synthetic sampling. The examples below will use the 'scaled' sample set as input, are evaluated with Logistic Regression, and optimized with grid search across a 5-fold cross-validation. The fit itself is evaluated with *Accuracy*, purposefully to highlight how Accuracy performs with Class Imbalance. A Confusion Matrix (CM), the Receiver Operating Characteristic Curve (ROC) and a Precision-Recall Curve (PRC) will be used for model evaluation - along with a printout of the *Accuracy*, *Precision*, *Recall*, and *F1 Score*. Prior to resampling, the classifier is fit to the dataset below.



```
Accuracy        = 1.00
Precision       = 0.91
Recall          = 0.61
F1 Score        = 0.73
```

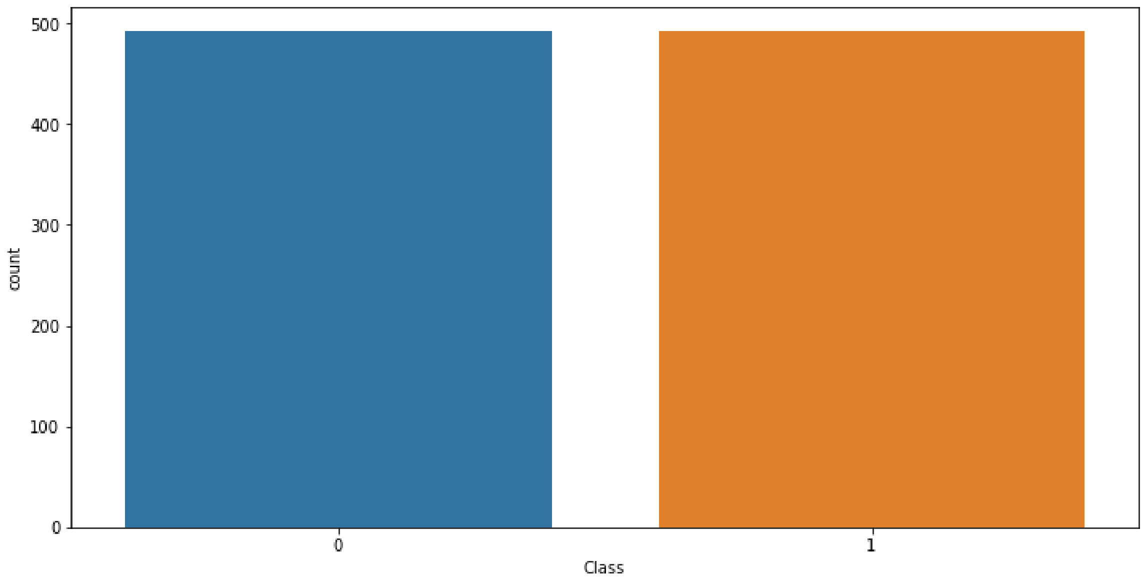
Fitting the imbalanced dataset returns an ROC Area-Under-Curve (AUC) of 0.98, and a PRC Average-Precision (AP) of 0.80. Using the PRC when evaluating models built on imbalanced datasets gives additional insight into the false positive and false negative results; or the 'false alarms' and 'misses' produced by the fit.

Another concern for Regression modeling is multicollinearity, or "The statistical phenomenon in which predictor variables in a logistic regression model are highly correlated" (Midi, Sarkar & Rana, 2010). To visualize correlation across features, a Correlation Matrix can be used.



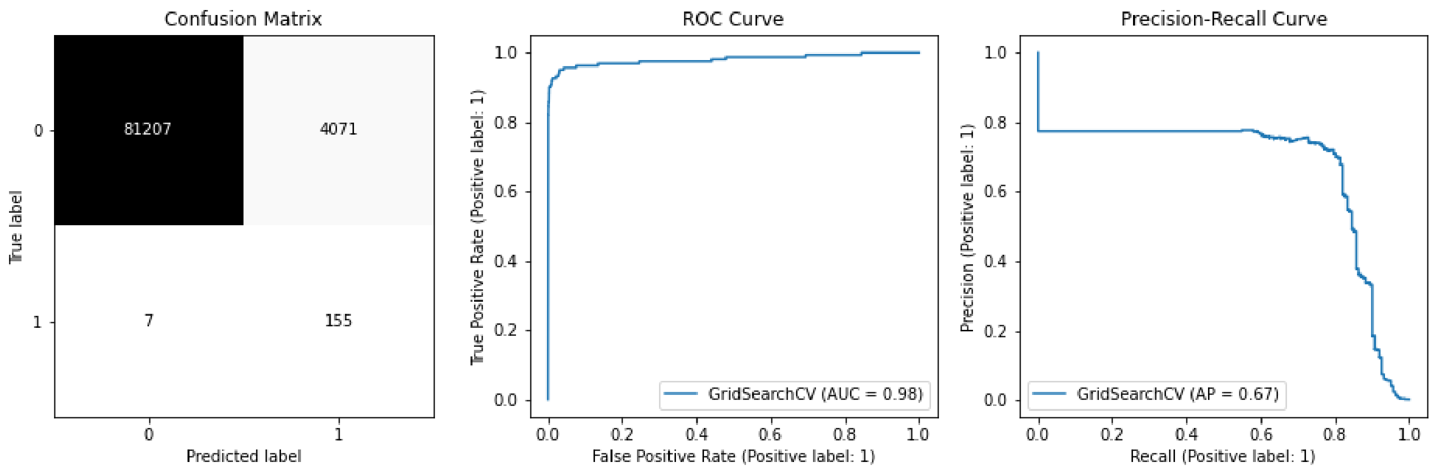
Prior to Resampling, Few Correlations are even Visible Across Features in an Imbalanced Dataset.

Undersampling. Undersampling is an adjustment to the distribution of a dataset where the majority class is randomly sampled at the same rate as the minority class. This resampling method **reduces** the training size, making it useful for quick analyses.



Undersampling the Majority Class to Create a 1:1 Ratio.

Fitting the undersampled set returns an ROC Area-Under-Curve (AUC) of 0.98, and a PRC Average-Precision (AP) of 0.67.

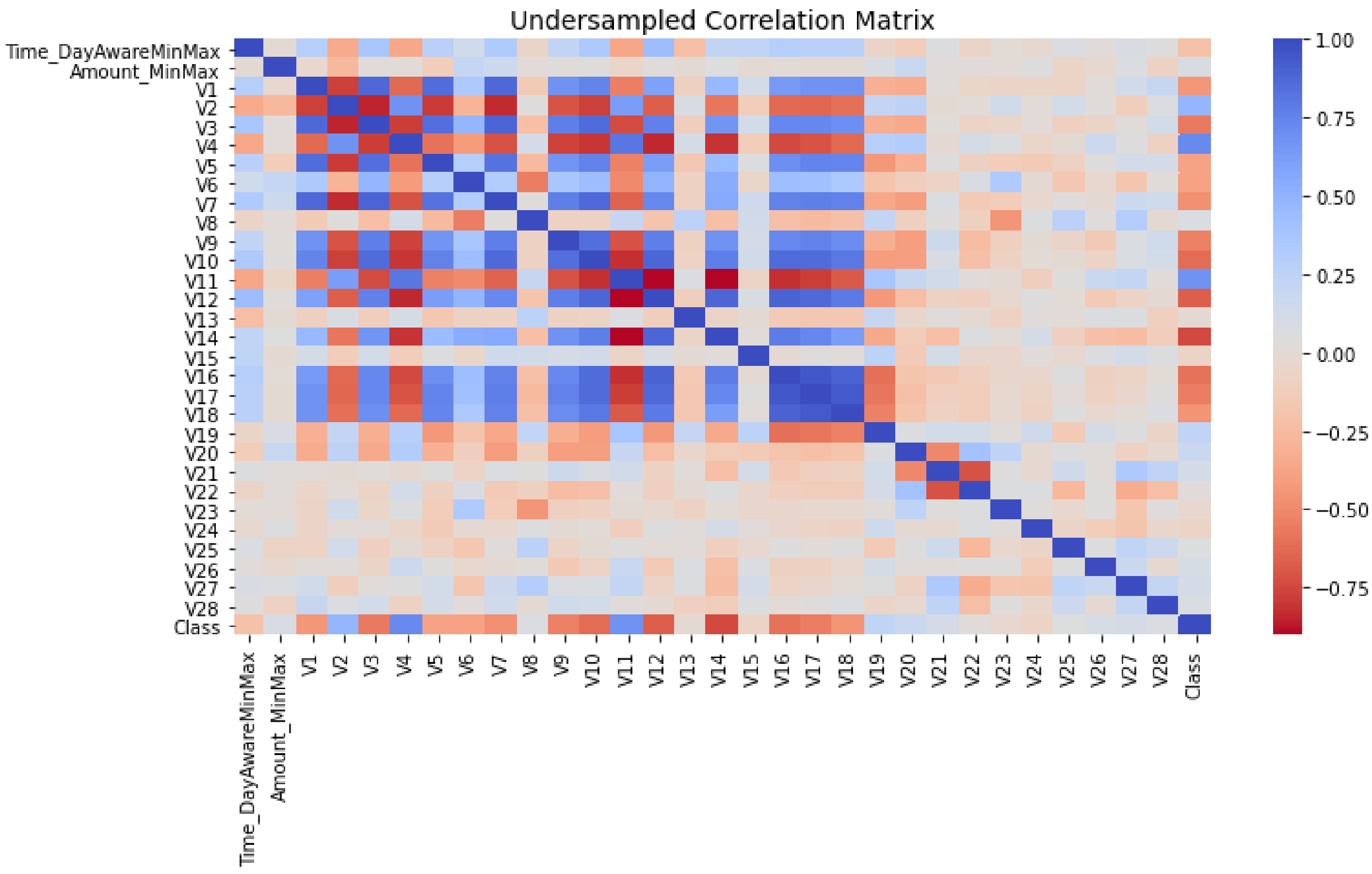


```
Accuracy        = 0.95
Precision       = 0.04
Recall          = 0.96
F1 Score        = 0.07
```

The undersampled set represents less than 1% of the entire data sample – resulting in over 99% of the dataset going unused. The undersampled dataset has a 1:1 class ratio, but less than 1% of the majority's features are being considered as predictors for the majority class – creating an under-representation of the majority class's features. As a result, undersampling reduces the model's predictive power for the majority class. This loss of majority class predictive power is best seen in the decline in *precision* from 0.86 to 0.05. Precision is calculated as:

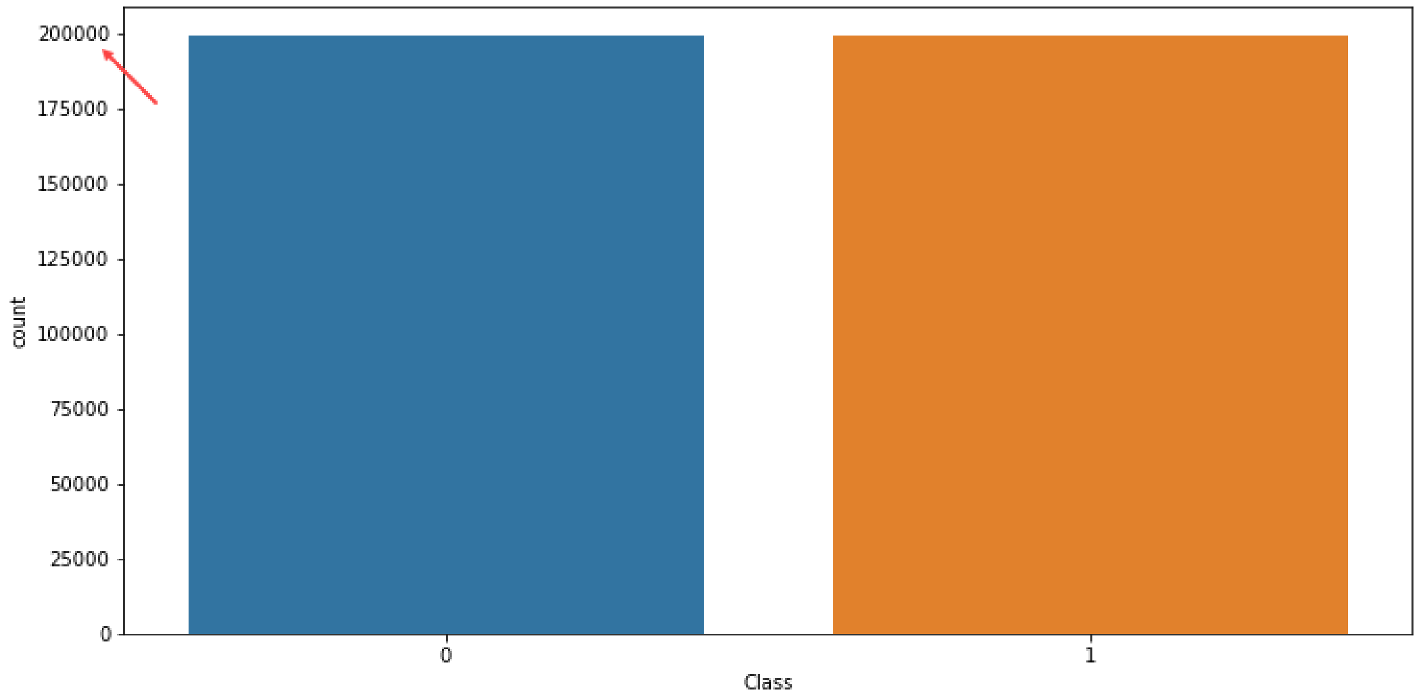
$$\frac{TP}{TP + FP}$$

Feature correlations have increased across the feature set too. The increase of feature correlations might prompt additional data preprocessing, to ease the possibility of multicollinearity on the Logistic Regression Classifier.



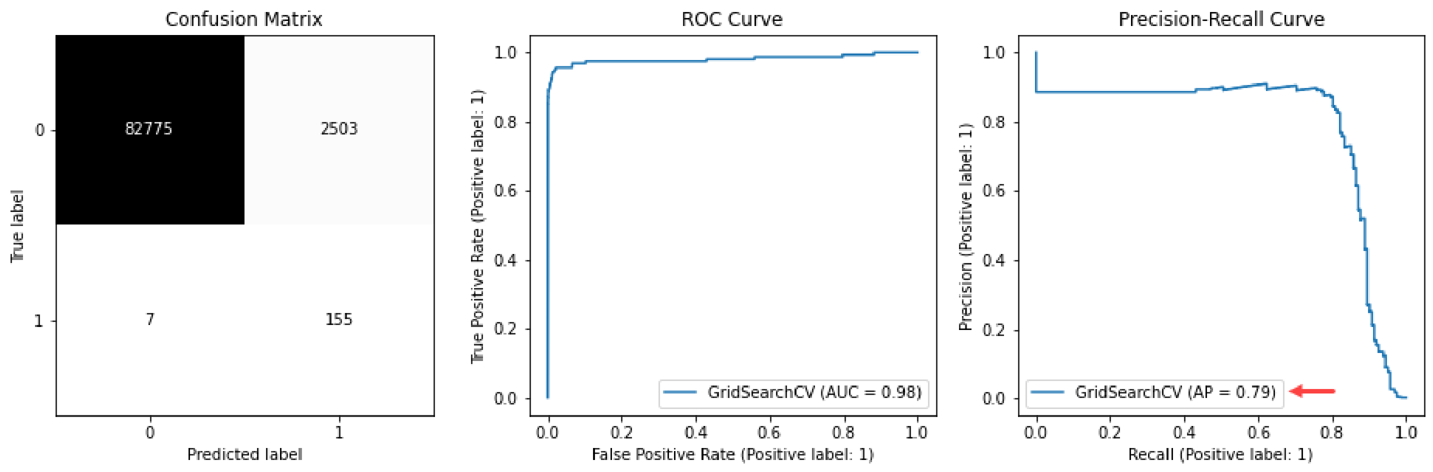
Though Undersampling Removed the Class Imbalance, it Introduced Collinearity Across the Feature Set.

Oversampling. Oversampling is another method of adjusting the dataset distribution. Different types of oversampling exist, but in this example, *imblearn.over_sampling.RandomOverSampler* is used. RandomOverSampler picks samples from the minority class at random with replacement.



Oversampling the Minority Class to Create a 1:1 Class Ratio.

Oversampling *resamples* the minority class, so the imbalance problem is solved by increasing the minority class sample size. In doing so, oversampling avoids losing 99% of the majority class information (like undersampling does).

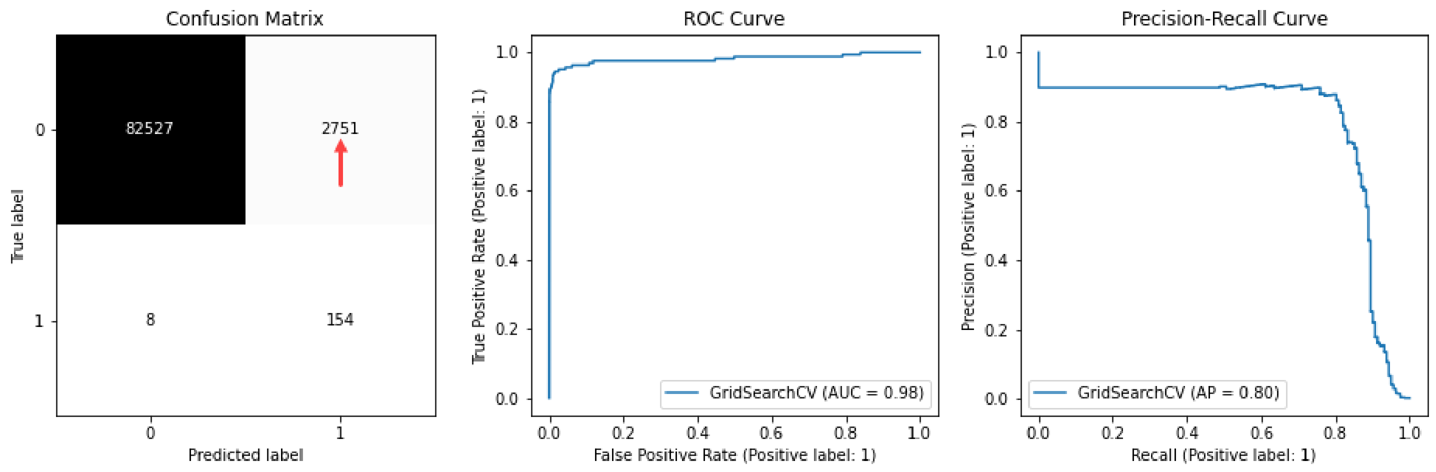


```
Accuracy        = 0.97
Precision       = 0.06
Recall          = 0.96
F1 Score        = 0.11
```

Oversampling the dataset returns an ROC AUC of 0.98, and an PRC AP of 0.79. While the PRC AP value increased from 0.67 to 0.79 between the two sampling methods, the *Precision* and *Recall* scores remained within 0.02. Precision and

Recall are both scores that represent a relationship between actual and predicted labels, so a lack of significant change in both scores while the sample size is increased implies that no change occurred with the relationship between actual and predicted labels. Since oversampling repeated the features of the minority class over 500 times to increase it from 352 samples to 199,006 samples, it is highly unlikely that new information was acquired.

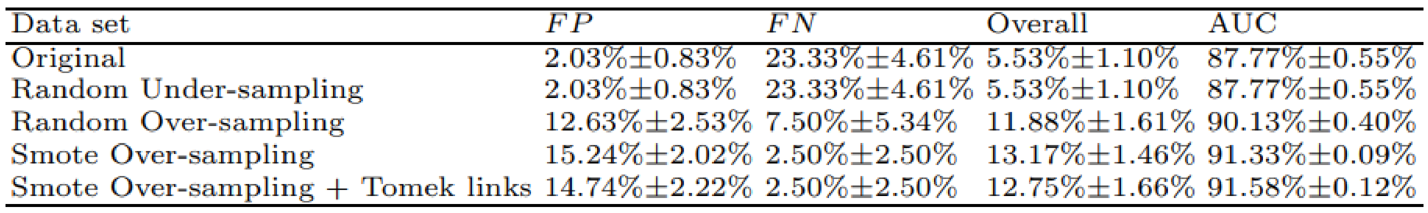
Synthetic Minority Oversampling Technique (SMOTE). Many improvements have been made to SMOTE since its inception. With SMOTE, new *synthetic* samples are generated for the minority class by applying a random multiplier to vectors between minority class samples and k-nearest neighbors.



```
Accuracy      = 0.97
Precision     = 0.05
Recall        = 0.95
F1 Score      = 0.10
```

Again, because of the magnitude of samples returned from SMOTE to bring the severely imbalanced dataset to a 1:1 ratio, the resulting scores for *Precision* and *Recall* are relatively the same as they were with the undersampled dataset – suggesting that the model hasn't gained any predictive power.

In Batista, Bazzan, & Monard (2003), various sampling methods were applied to an imbalanced dataset and evaluated with AUC; false-positive rate (FP) and false-negative rate (FN) were also recorded.



Data set	FP	FN	Overall	AUC
Original	2.03%±0.83%	23.33%±4.61%	5.53%±1.10%	87.77%±0.55%
Random Under-sampling	2.03%±0.83%	23.33%±4.61%	5.53%±1.10%	87.77%±0.55%
Random Over-sampling	12.63%±2.53%	7.50%±5.34%	11.88%±1.61%	90.13%±0.40%
Smote Over-sampling	15.24%±2.02%	2.50%±2.50%	13.17%±1.46%	91.33%±0.09%
Smote Over-sampling + Tomek links	14.74%±2.22%	2.50%±2.50%	12.75%±1.66%	91.58%±0.12%

FP rate, FN rate, Overall error rate and AUC after cross validation. Batista et al. (2003).

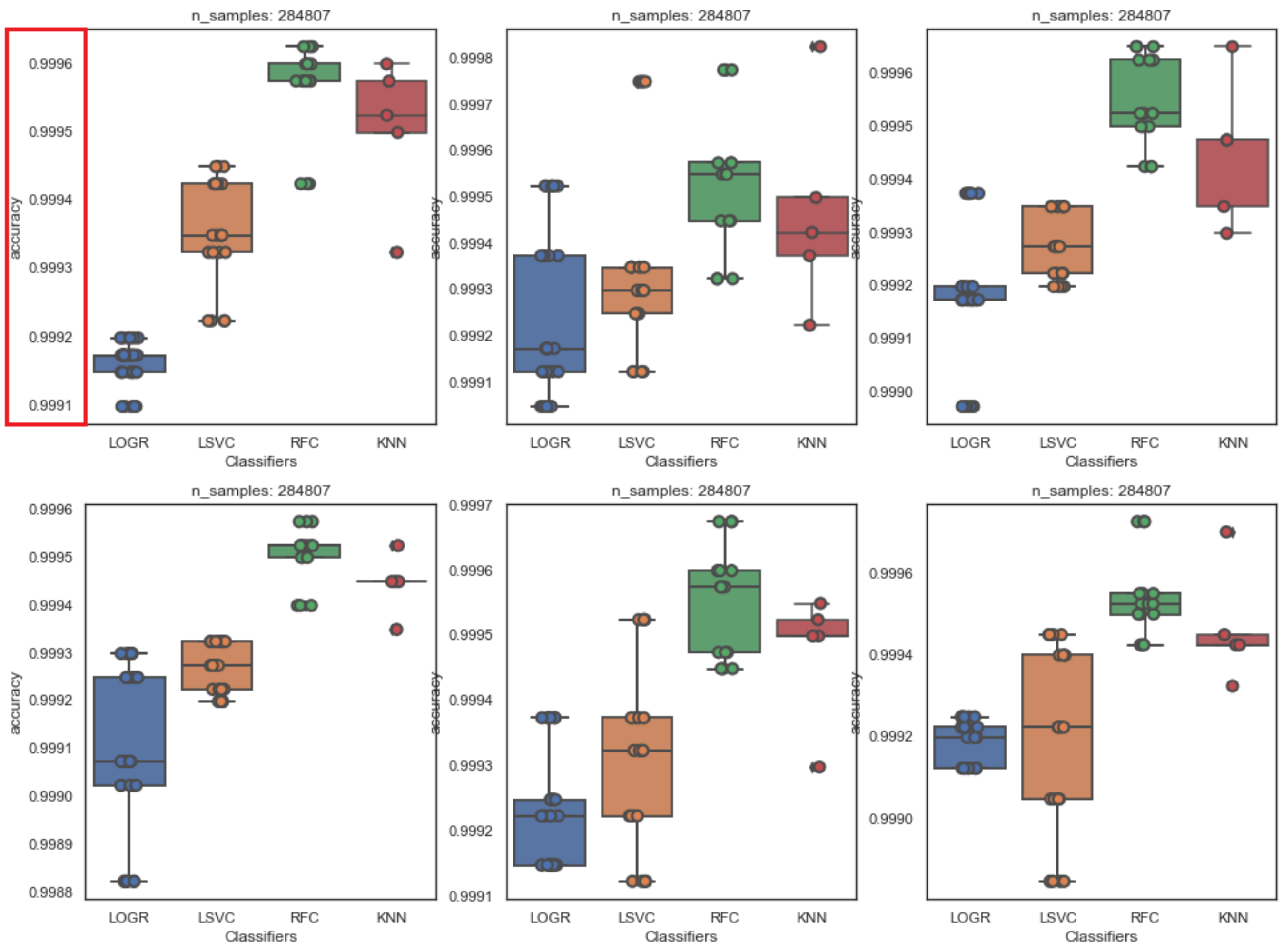
Barista et al. (2003) observed a decrease in FN when applying random oversampling, and again when applying SMOTE. Accompanying the decrease in FN was an increase in FP. Finally, improvements in AUC across the sampling methods were observed when applying oversampling, SMOTE, and again with SMOTETomek.

Model Evaluation

Accuracy. Accuracy is a go-to metric for model scoring because it is easy to calculate and comprehend. Accuracy is calculated as the *number of true predictions by the model* divided by the *total number of predictions.* It can also be evaluated as the sum of *true-positives* (TP) and *true-negatives* (TN) divided by the total number of predictions.

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

To demonstrate the results of scoring an imbalanced dataset with accuracy, the ULB Fraud dataset is evaluated below with four algorithms: Logistic Regression, Linear Support Vector Classifier, Random Forest Classifier, and a K-Nearest Neighbors Classifier. The visual below is generated by inputting the full dataset (284,807 records) into a procedure that models a 70/30 train test split with 5-fold cross validation, across 6 iterations of resampling.



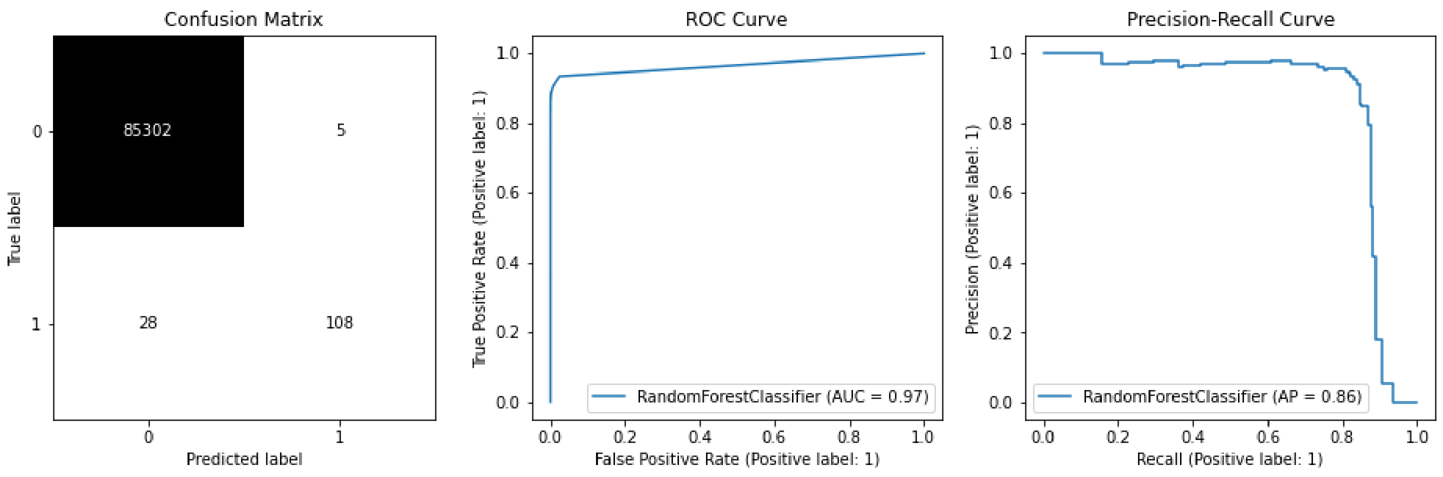
As seen above, the accuracy score is close to the number of not-fraudulent transactions divided by the total number of transactions: 284,315 / 284,807 = 99.83%. However, a model designed to detect fraud is concerned with increasing the detection of fraud (TP) and reducing fraud misses (FN). Because Accuracy only considers TP + TN (I.e., correct fraud predictions and correct not-fraud predictions), Accuracy is heavily influenced by the majority class (TN). The TP, FN, TN, and FP are listed in a *confusion matrix*. Note: the orientation of the confusion matrix might be different depending on the source library. For these examples, Sklearn was used to generate the metrics. Because the majority class is passed as 0 and our minority class as 1 for the dependent variables, Sklearn orients the confusion matrix as:

TN	FP
FN	TP

The Receiver Operating Characteristic Curve. The ROC is a plot of the *true-positive rate* (TPR) on the y-axis, and the *false-positive rate* (FPR) on the x-axis. For any classifier, there can be multiple decision thresholds used to classify samples. Lowering a classification threshold allows the classifier to classify more samples as positive – increasing the number of both FP and TP. The ROC curve plots a line across the various decision thresholds, connecting all of the model's classification thresholds in terms of their TPR and FPR relationship.

Area Under Curve. AUC quantifies the area under the ROC. AUC is desirable as a performance metric because is it scale-invariant, and because it is classification-threshold-invariant (Google Developers, 2021). AUC can also be interpreted as the probability of detection versus a false alarm. An AUC at 0.5 is said to have, 'No skill,' because it cannot detect positives or false alarms. An AUC above 0.5 has a positive outcome, while an AUC below 0.5 has negative outcome.

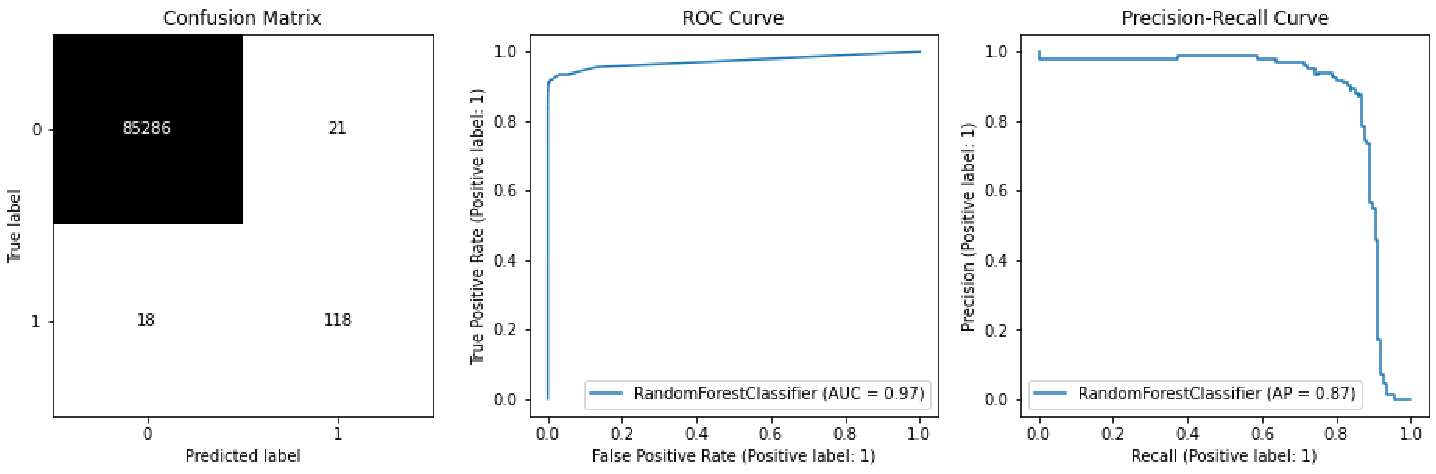
Random Forest Classifier (RFC). Another option for handling Class Imbalance is to use algorithms that are not as sensitive to scale or imbalance. In this section, the 'unscaled' dataset is input into a Random Forest Classifier (RFC). Remember, this dataset has not been scaled, and only has 'Time' adjusted to be *Day-Aware*.



```
Accuracy        = 1.00
Precision       = 0.96
Recall          = 0.79
F1 Score        = 0.87
```

Fitting a Random Forest Classifier to the Training Set As-Is

The RFC returns an ROC AUC of 0.97, and a PRC AP of 0.86 – returning a slightly worse AUC but much better AP than the previous Regression models. When inputting a SMOTE resampled, 'unscaled' dataset into the RFC, the AUC and AP remain the same while the *Precision* and *Recall* scores change.



```
Accuracy      = 1.00
Precision     = 0.85
Recall        = 0.87
F1 Score      = 0.86
```

Fitting a Random Forest Classifier to the SMOTE-Resampled Training Set.

Precision is reduced from 0.97 to 0.90, while *Recall* is increased from 0.78 to 0.84. Relying on *Accuracy* alone does not capture the entire picture when handling Class Imbalances. Also, AUC tends to correlate with better model performance – but was not aware when oversampling with Logistic Regression led to overfitting.

During the observations taken in this essay, Overfitting was best noted by the combination of two factors: 1.) A significant increase in AP (From oversampling) coupled with, 2.) Less than 0.01 change in both *Recall* and *Precision*.

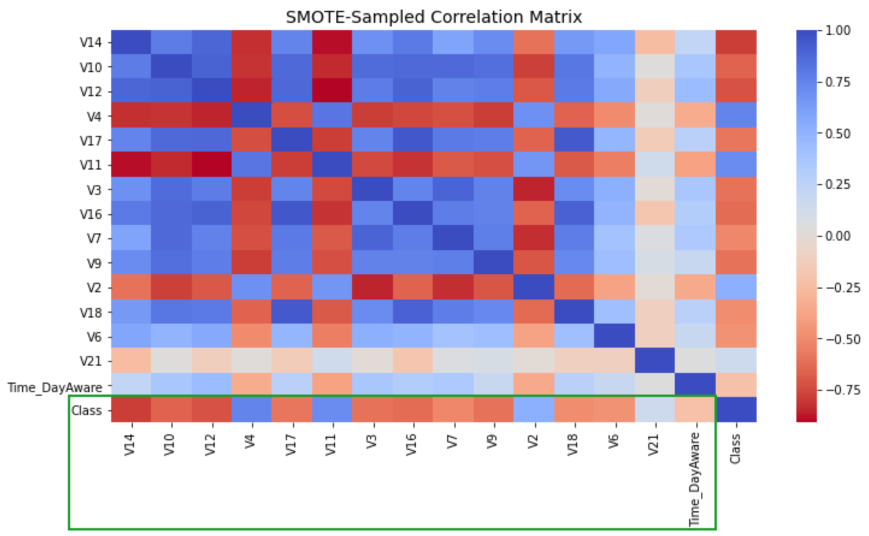
Because the damages from fraud activity rely on a combination of 1.) Accurately identifying fraud (TP); 2.) Reducing the fraud 'Misses' (FN); and 3.) Reducing 'False Alarms' (FP); it is advantageous to use AUC and PRC to explore both the Positive Prediction of a model, and its *sensitivity* and *specificity*. The latter two can be more context dependent. For example, a credit-card issuer might shift the 2nd and 3rd priority to reduce excess pressure on the customer base for a short time. The delta across *Precision* and *Recall* can be used to tune model performance in this case.

Looking at the actual detection rates, the RFC with an unscaled SMOTE input performed the best in these preliminary observations with 116 fraud transactions accurately identified, 22 fraud transactions misclassified as genuine, and 13 genuine transactions misclassified as fraudulent – from a total of 85,443 transactions tested.

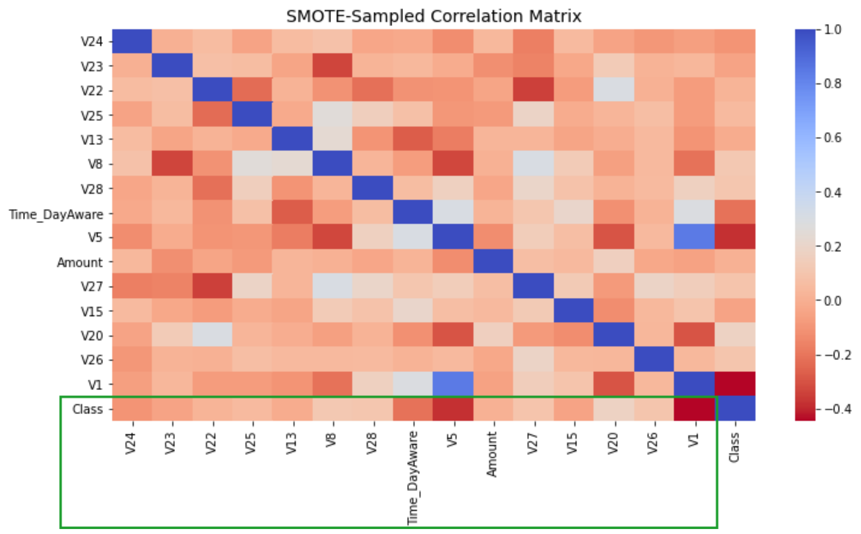
Feature Selection

Feature Selection is a process in Machine Learning where features are selected from the feature set that contribute more to the predicted variable than other features. This can be done manually or using algorithms, and can benefit modeling by reducing overfitting, improving model performance, and reducing the model's training time. The correlation matrix above is an example of a *tool* that is sometimes used to select features.

Correlation Matrix (CM). The Correlation Matrix visualizes correlation across the variable set – giving insight into which features align with (and against) the Class variable. The color bar to the right of the CM Heatmap provides a key to interpret the variable squares and the range across the correlation matrix. The CM across these cases shows darker shades of blue for variables that correlate with fraudulent activity, and darker shades of red for variables with high anti-correlation. Independent variables with higher correlation (or anti-correlation) with the dependent variable also should be evaluated for their correlation with other independent variables. High correlation that occurs across multiple independent variables and the dependent variable might mean one of the independent variables is caused by another correlated independent variable (and not the dependent variable directly). *Multicollinearity* like that should be noted since it can lead to a decrease in model performance.



The Correlation Matrix for the Top 15 Important Features from Mean Decrease in Impurity.

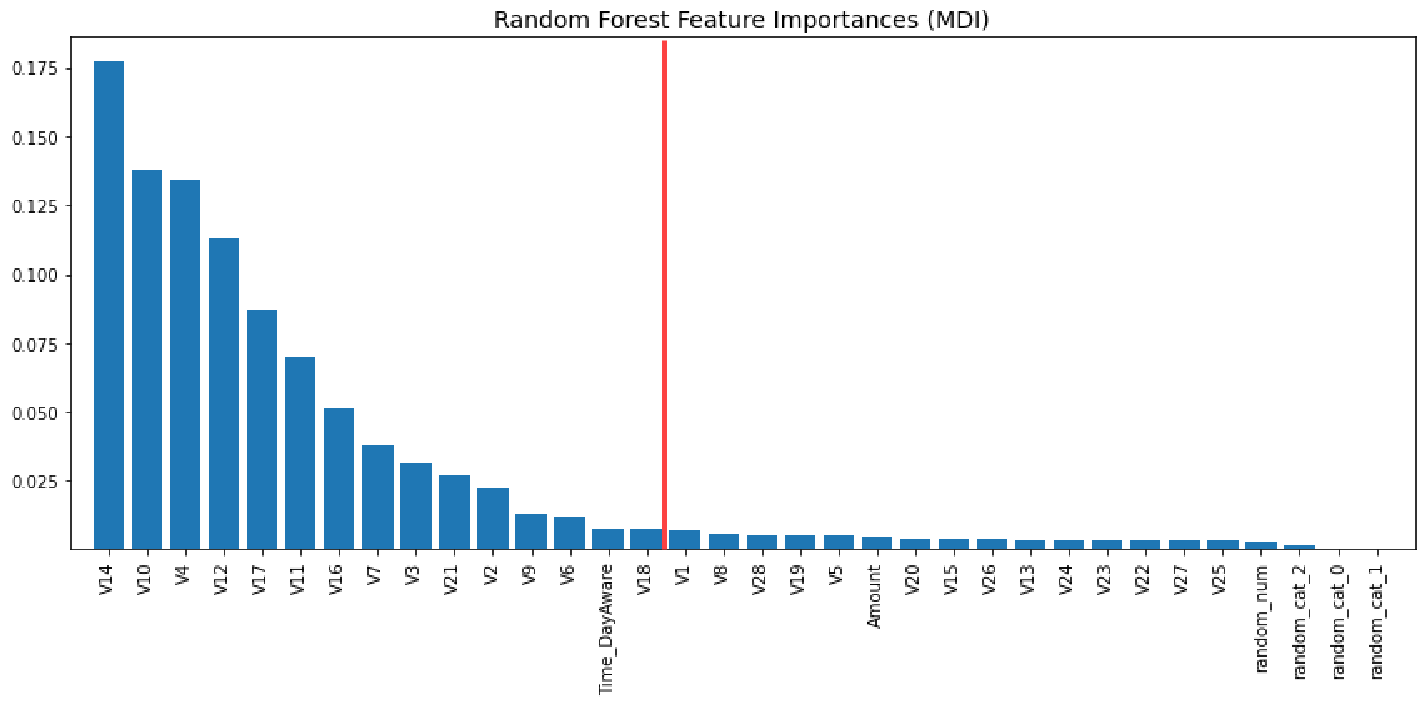


The Correlation Matrix for the Bottom 15 Important Features from Mean Decrease in Impurity.

In the Correlation Matrices above, the latter CM represents the less important features. Across the squares related to the Class variable, few of the variables appear to exceed the 0.2 to -0.2 range for correlation (and none above the 0.4 to -0.4 range). The top CM, containing *more important* features, has variables that range from 0.75 to -0.75. It is important to note that the above correlation matrices are associated with *one method of determining Feature Importance* (Mean Decrease in Impurity). Other methods also exist, and an alternative method to Mean Decrease in Impurity will be discussed later.

With Logistic Regression, *feature coefficients* can be returned from the decision function and evaluated for importance. Below, two methods of Feature Selection are explored for the Random Forest Classifier. During Feature Selection, 2 random features are added to the dataset: *random_num* (a randomized numerical feature), and *random_cat* (a randomized categorical feature consisting of 3 possible categories). The *random_cat* feature is represented in the first case below as *random_cat_0, random_cat_1*, and *random_cat_2* because it is transformed into a numerical representation during preprocessing for the model; and the feature importance is returned as an attribute of the model's fit. In the second case, the feature importance is performed separate from the model.

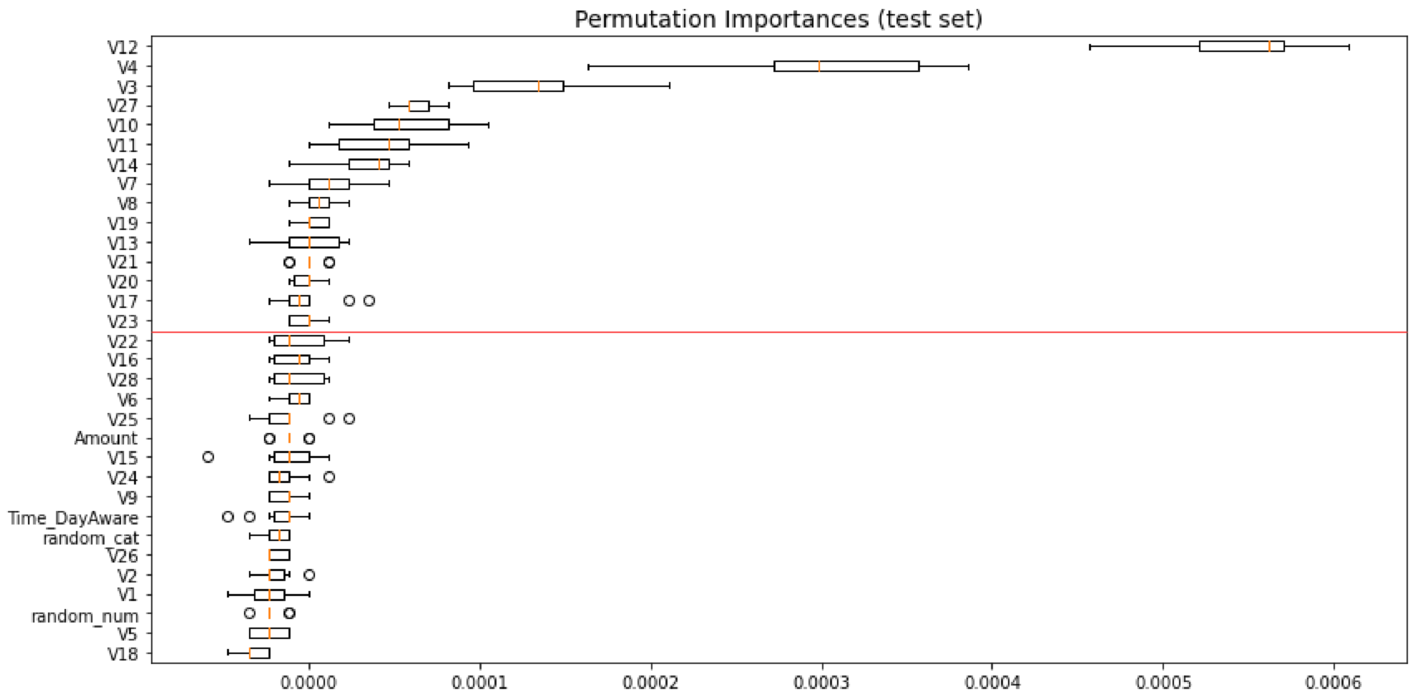
Mean Decrease in Impurity (MDI). For the above Random Forest Classifier, the scikit-learn library has a built-in attribute (*feature_importances_*) that returns *impurity-based* feature importance. This method is called on a fit of the SMOTE-resampled data with the Random Forest Classifier, and visualized here:



Gini Importance Returns Feature Importance from the SMOTE RFC Fit.

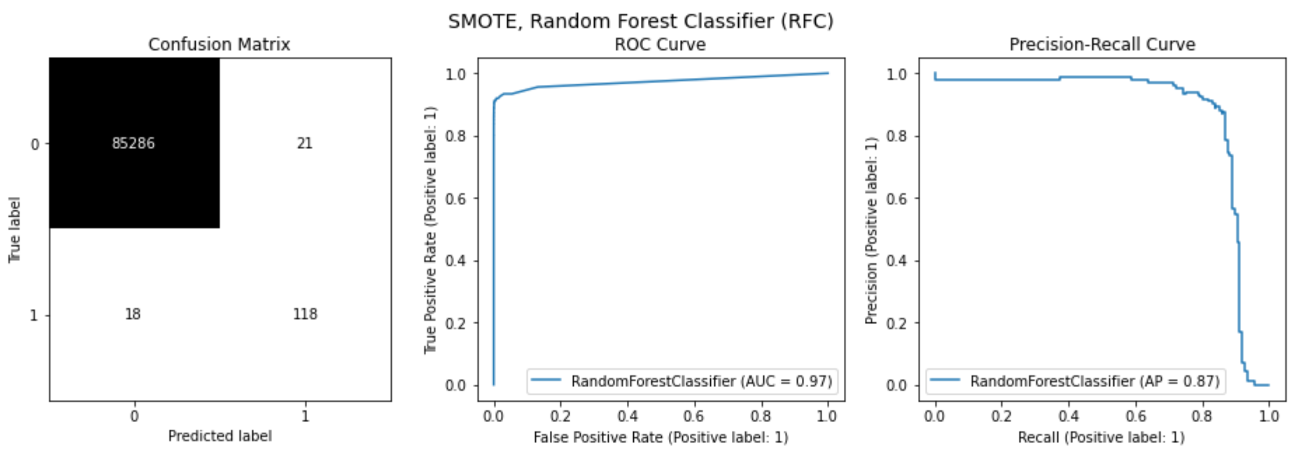
Also called *Gini Importance*, MDI calculates feature importance as the sum over the number of splits that include the feature – compared to the number of samples it splits. This is done across all trees.

Permutation Importance. Permutation Importance (or *Mean Decrease in Accuracy*) is another method available for feature importance. Permutation Importance randomly permutes the values of the features and evaluates the resulting increase in error. This is seen as removing the association between the feature and the target. MDI can inflate the importance of numerical features, and because it is computed on statistics derived from the training dataset, the importance can be even higher for features that are not predictive of the target variable if the model has the capacity to use them to overfit (scikit-learn, 2020). Permutation Importance is offered as a better alternative for evaluating feature importance for this reason (scikit-learn, 2020).

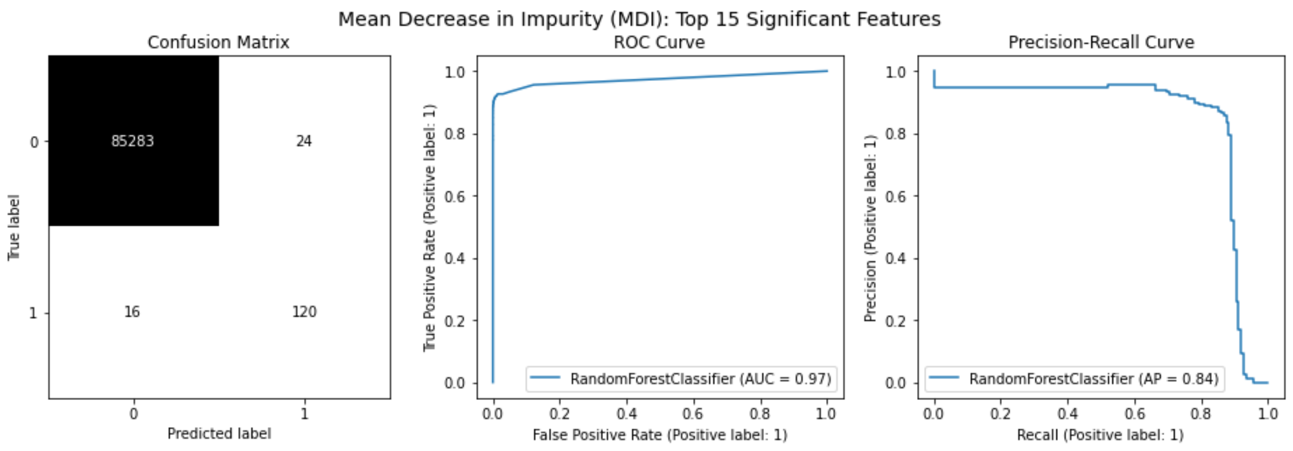


Finalizing the Model

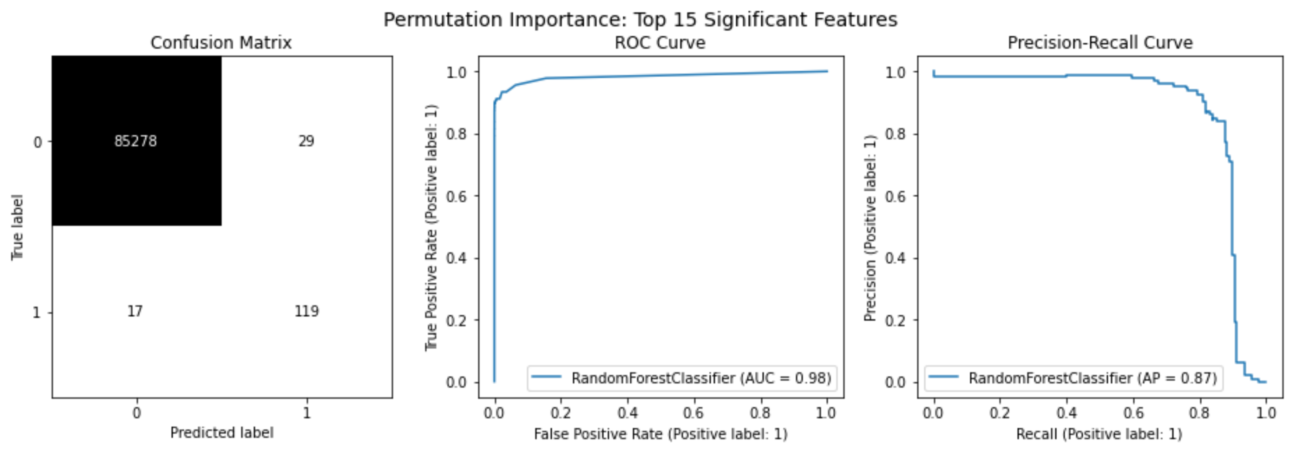
To evaluate the different feature importance considerations, a SMOTE-resampled dataset is used. In each iteration, the ‘important features’ subset is isolated in the training set and fit with a Random Forest Classifier. Below, 5 iterations of the above parameters are run using the following four ‘feature importance’ cases: A *Control Fit* with no Feature Selection, *Top* 15 MDI Important Features, Top 15 Permutation Importance Important Features, A User-Selected Composite of 12 Features, and a Composite Bottom-Half of Features.



```
Accuracy        = 1.00
Precision       = 0.85
Recall          = 0.87
F1 Score        = 0.86
```



```
Accuracy        = 1.00
Precision       = 0.83
Recall          = 0.88
F1 Score        = 0.86
```



```
Accuracy        = 1.00
Precision       = 0.80
Recall          = 0.88
F1 Score        = 0.84
```

Summary

For American Express in 2018, 2 days of transactions averages to 51.4 million transactions, for a value of $4.6 billion. A fraud detection model helps businesses 1.) Accurately Identify Fraud; 2.) Reduce the number of ‘Misses’ in Fraud Detection; and 3.) Reduce the number of ‘False Alarms’ that interrupt the flow of business.

Significant Observations. From the above experimentation, two processes were observed to have a positive & significant impact on modeling the Class Imbalance dataset: 1.) Implementing the Random Forest Classifier (over the Logistic Regression Classifier; and 2.) Synthetic resampling. No significant advantage was observed from the feature selection based on MDI & PI Feature Importance (Additional cases are included in notebook *08_remodel_withimportantfeatures*).

Ensemble Methods like the Random Forest Classifier have many advantages in Machine Learning. The RFC is not sensitive to scale or imbalance – suggesting that some improvements might be made to the preprocessing steps that preceded the LRC fit. Switching the fit from the LRC to the RFC increased the *Precision* score from 0.05 to 0.96 – decreasing the number of *False Positives* from 2751 to 5. *Recall* was observed to decrease from 0.95 to 0.79 – increasing the number of *False Negatives* from 8 to 28.

When fitting the RFC, SMOTE resampling was observed to improve the *True Positive* and *False Negative* counts (108, 28 to 118, 18 respectively) with a slight increase in the *False Positives* (5 to 21). Since *True Positives* (Correctly Identifying Fraud) and *False Negatives* (Missing Fraud) are arguably more important than *False Positives* (False Alarms) – this improvement is seen as highly favorable to the practicality of the model’s business application.

A model that could predict fraudulent activity nearly 87% of the time while triggering false alarms less than 1% could be sufficient to solve a problem created by Class Imbalance fraudulent activity in a business setting. Also, though it was not covered in this essay, an unscaled input into an Ensemble Classifier returned a 0.93 *Recall* score (an 9% increase) in exchange for an increase in ‘False Alarms’. However, further valuation is needed to determine whether 3620 additional *False Alarm* occurrences is worth a reduction of the *Missed Fraud* cases by 10.

Machine Learning vs. Learning Fraudsters. Fraudsters are a dynamic actor. As opportunities and penalties for fraud arise, Fraudsters’ behavior will change. The frequency and availability of training data will also influence model selection. While Logistic Regression Classifiers might offer more flexibility or long-term accuracy from hyperparameter fine-tuning, Random Forest Classifiers can be quickly trained thanks to less input data restrictions. On the other hand, Random Forest Classifiers become bogged down with large input datasets, so other model options might be more robust when dealing with the magnitude of transaction traffic a business maintains.

References

Batista, G. E. A. P. A., Bazzan, A. L. C., Mondard, M. C. (2003). *Balancing Training Data for Automated Annotation of Keywords: a Case Study.* Instituto de Ciencias Matematicas e de Computacao, USP Caixa Postal 668, 13560-970, Sao Carlos, Brazil.

Board of Governors of the Federal Reserve System. (2020). *The 2019 Federal Reserve payments Study*. Retrieved from https://www.federalreserve.gov/paymentsystems/2019-December-The-Federal-Reserve-Payments-Study.htm

Google.com. (2021). *Classification: ROC Curve and AUC.* Retrieved from https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Kaggle Inc. (2021). *Credit Card Fraud Detection. Anonymized credit card transactions labeled as fraudulent or genuine*. Retrieved from https://www.kaggle.com/mlg-ulb/creditcardfraud

Midi, H., Sarkar, S. K., Sohel, R. (2010). Collinearity diagnostics of binary logistic regression model. *Journal of Interdisciplinary Mathematics*. Vol. 13 (2010), No. 3, pp. 253-267.

Nilson Report. (2021). *Market Shares (%) of Purchase Volume for Top U.S. Credit Cards.* Issue 1170. Retrieved from https://nilsonreport.com/publication_newsletter_archive_issue.php?issue=1170

scikit-learn. (2020). *Permutation Importance vs Random Forest Feature Importance (MDI).* Retrieved from https://scikit-learn.org/stable/auto_examples/inspection/plot_permutation_importance.html