Exploring Class Imbalance with Fraud Detection

James Bush

Springboard

Capstone II

2021-03-10

TABLE OF CONTENTS

Fraud Detection and Imbalance

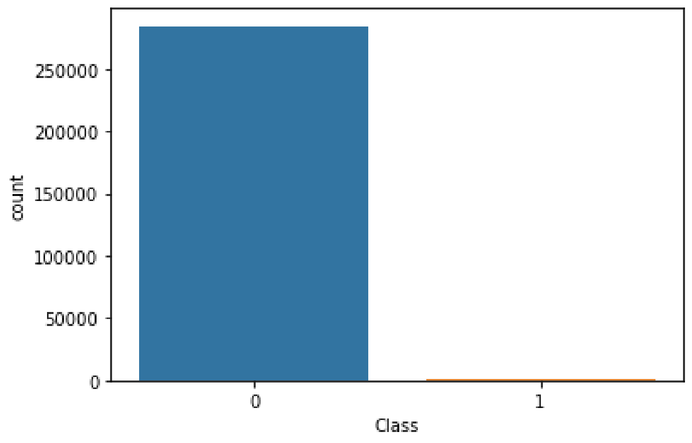
The ability to capture fraudulent activity across credit card transactions is an immediate boon for any credit card provider – proving both security and comfort to customers. Because of the nature of credit card fraud activity, datasets for training fraud classifiers will almost always be *imbalanced.* Another similar problem is detecting rare diseases, where the population without the disease would greatly outnumber those with the disease. Imbalanced datasets provide unique problems for classification models in Machine Learning. A significantly skewed dataset influences sampling because the features from the larger group will have a heavier weight, due to their higher sample rate. Also, data imbalance influences how models are scored and selected.

Consider a dataset with 10,000 credit card transaction records. A *balanced* dataset might have 5,200 authorized transactions and 4,800 fraudulent ones (Though, any company with that dataset should be *more concerned with filing bankruptcy* than develop fraud detection). A more realistic representation of fraudulent data in the real-world use case would be an *imbalanced* dataset – where 9,970 credit card transactions are authorized and only 30 are fraudulent.

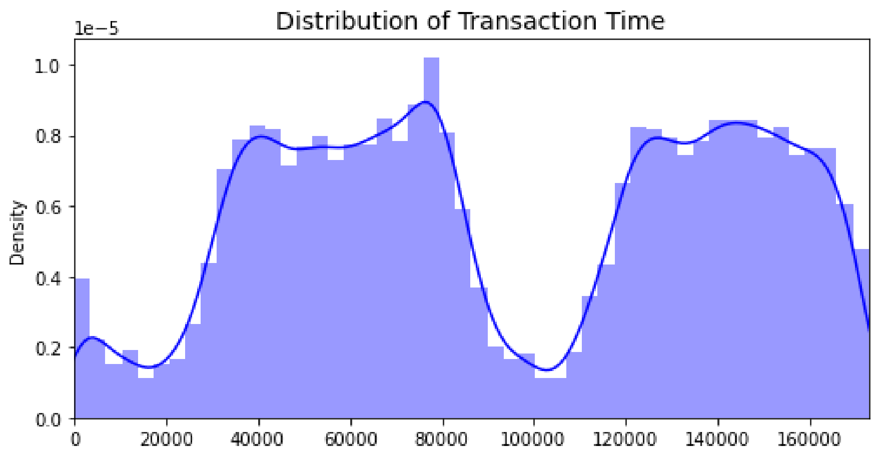
Problems related to imbalanced classification tend to have high-risk conditions. For example, the ratio of fraudulent transactions to non-fraudulent transactions is significantly low. However, the cost of undetected fraudulent activity for a business is significantly high. With rare diseases, while the occurrence of a rare disease is scarce, they tend to cause debilitating conditions or death.

Exploring the Data

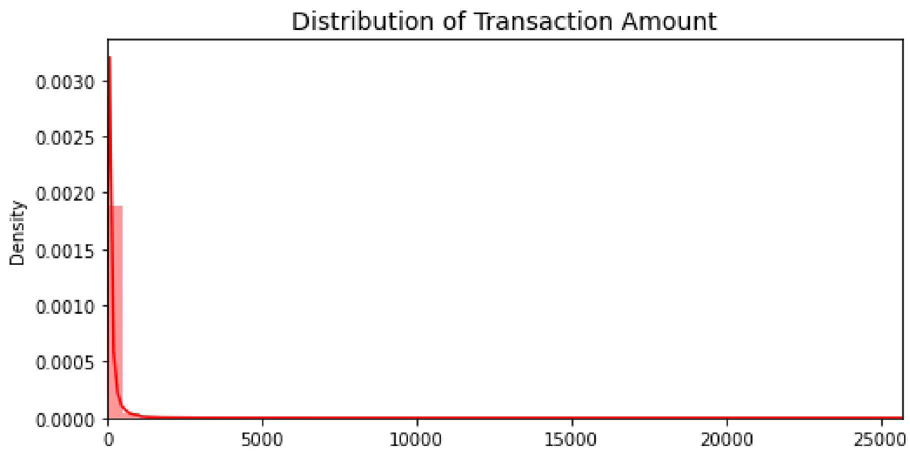
The data used for this essay is titled, "Credit Card Fraud Detection. Anonymized credit card transactions labeled as fraudulent or genuine," compiled by the Machine Learning Group of ULB and retrieved from kaggle.com. The dataset was collected and analyzed as a part of a research collaboration of Worldline and the Machine Learning Group of ULB on big data mining and fraud detection. In this dataset, there are 284,315 records in the majority class (I.e., 'Not fraudulent'), and 492 records in the minority class (I.e., 'Fraudulent'). The majority class is represented as 0, and the minority as:



Due to confidentiality issues, 28 of the 30 features of the dataset (V1-V28) have been replaced with numerical input variables transformed by PCA. The only features not transformed in this way are *Time* and *Amount*:



Time represents the number of seconds elapsed between each transaction and the first transaction in the dataset. Amount represents the transaction amount.

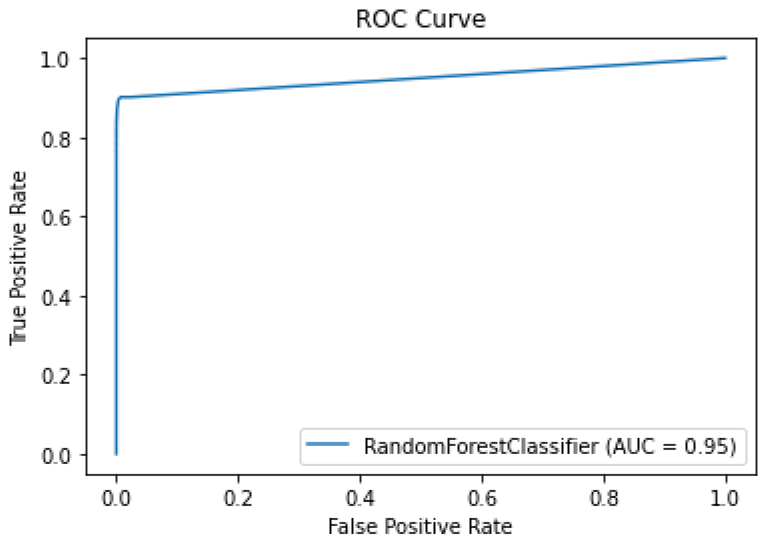


As seen in the visuals above, values in the Time feature series extend above 160,000, and values in the Amount feature series exceed $25,000. To reduce the noise of Time and Amount when processing across the entire feature set, the two features are pre-processed with the Sklearn Robust Scaler method. Robust Scaler scales the data according to quantile range (Rather than median), making it more robust to outliers than the Standard Scaler.

Fit the Data

The original fit is a 70-30 test-train split input to a *Random Forest Classifier* (RFC). The first 4 values: *Accuracy, Precision, Recall,* and *F1* Score, are used for scoring different aspects of model performance. The *Receiver Operating Characteristic Curve* (ROC) provides points across the range of classifier thresholds. The *Area Under Curve* (AUC) represents the space under the ROC. “AUC provides an aggregate measure of performance across all possible classification thresholds (Google Developers, 2021).” It can be used for overall model performance evaluation and for model performance ranking because it considers the entire space under the models’ threshold boundary.

```
Accuracy =  0.9995435553526912
Precision =  0.924812030075188
Recall =  0.8092105263157895
F1 Score =  0.863157894736842
```

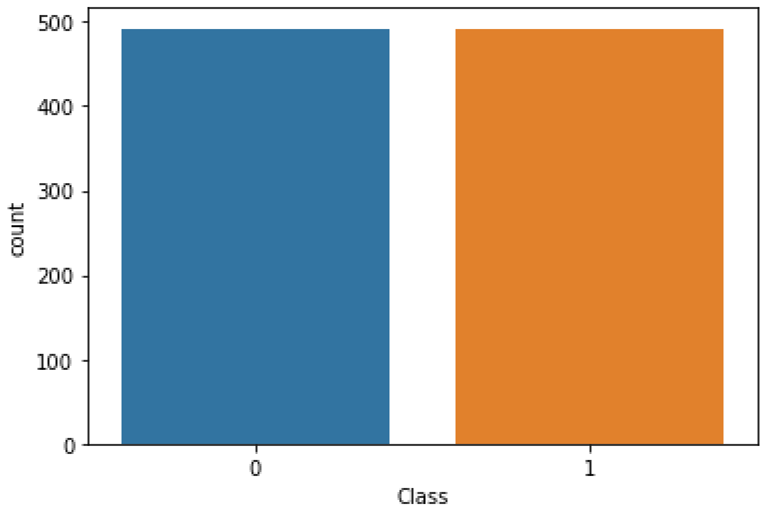


With the initial fit, a 99.96% accuracy score is returned. However, an imbalanced dataset will always return an artificially high Accuracy score when the minority class dependent variable is underrepresented enough.

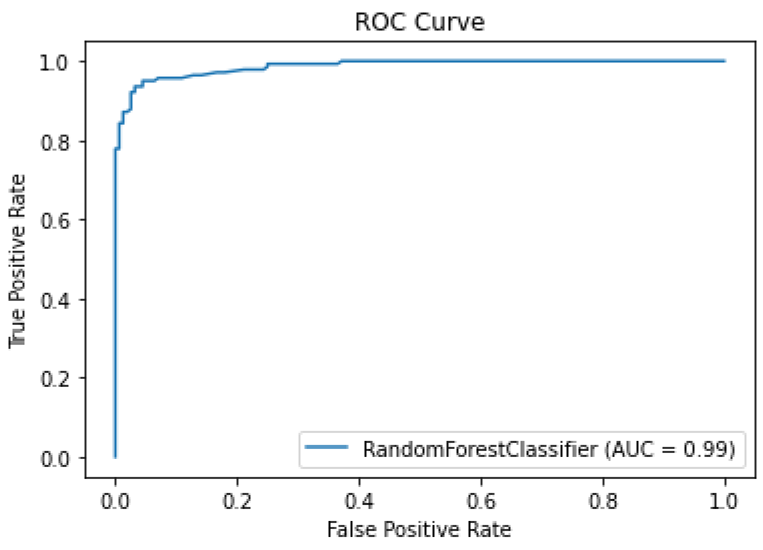
Sampling Methods with Imbalanced Datasets

Sampling is the process of gathering observations so predictions can be made about a population. Sampling methods introduce bias but can also be used to improve other factors. In machine learning, independent variables (Features) from those samples input into a model and drive the classification process. When a dataset is imbalanced, features of the majority class have a higher chance of being selected in the sample – and features in the minority class have a lower chance. This results in an under-representation of the minority class in the features (I.e., Prediction variables), as well as the outcome (I.e., Dependent variable). Some sampling methods to reduce the imbalance across the dataset include *undersampling*, *oversampling*, and *synthetic sampling.*

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. Below, we've reduced the number of samples in the majority class to 492, with a total sum of 984 samples – resulting in a 1:1 ratio of *not-fraud* to *fraud* transactions.

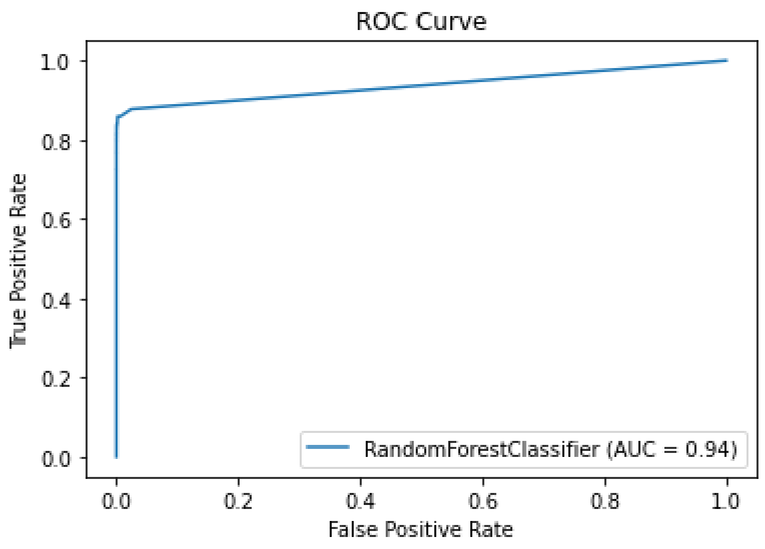


Fitting the undersampled set to a RFC returns an AUC of 0.99 on the ROC Curve.



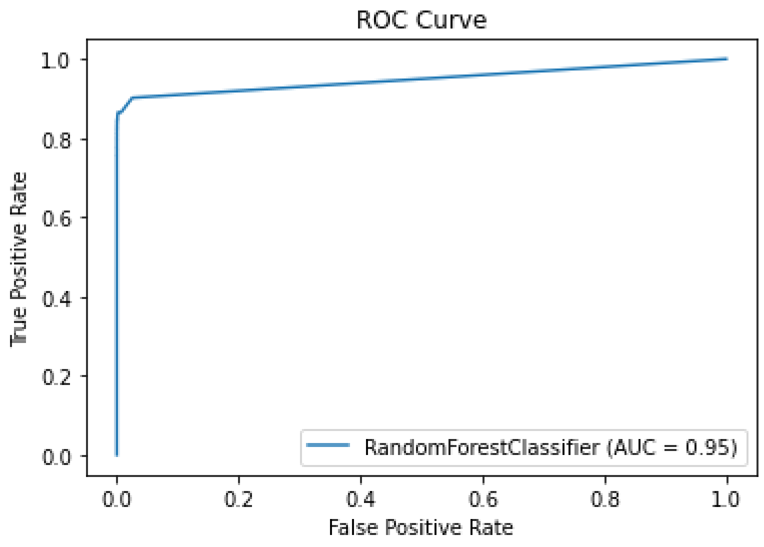
However, the undersampled set represents less than 1% of the entire data sample – resulting in over 99% of the dataset going unused. While the undersampled dataset has a 1:1 class ratio, it creates under-representation of the majority class *features* (I.e., Predictor variables) since less than 1% of the majority's features are being considered as predictors for the majority class.

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

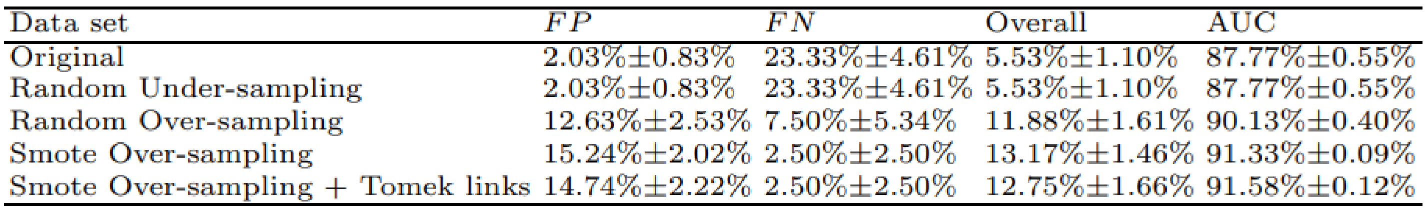


Fitting the oversampled sample to the RFC returns an AUC of 0.93 on the ROC Curve this time. In general, oversampling can help reduce the problem of under-representation of the minority class in the outcome – while also preserving the representation of the majority class features. However, oversampling can lead to overfitting.

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. In this example, *imblearn.combine.SMOTETomek* is used. SMOTETomek leverages Tomek links to develop the sample. Tomek links combine random under-sampling that removes majority class examples to help balance sample distribution. In the example, SMOTETomek is set to a ratio of 3:4 minority samples for each majority samples.



The SMOTETomek model returns an AUC of 0.95. In Batista, Bazzan, & Monard (2003), various sampling methods were applied to an imbalanced dataset and evaluated with AUC; false-positive rate (FP) and fasle-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).

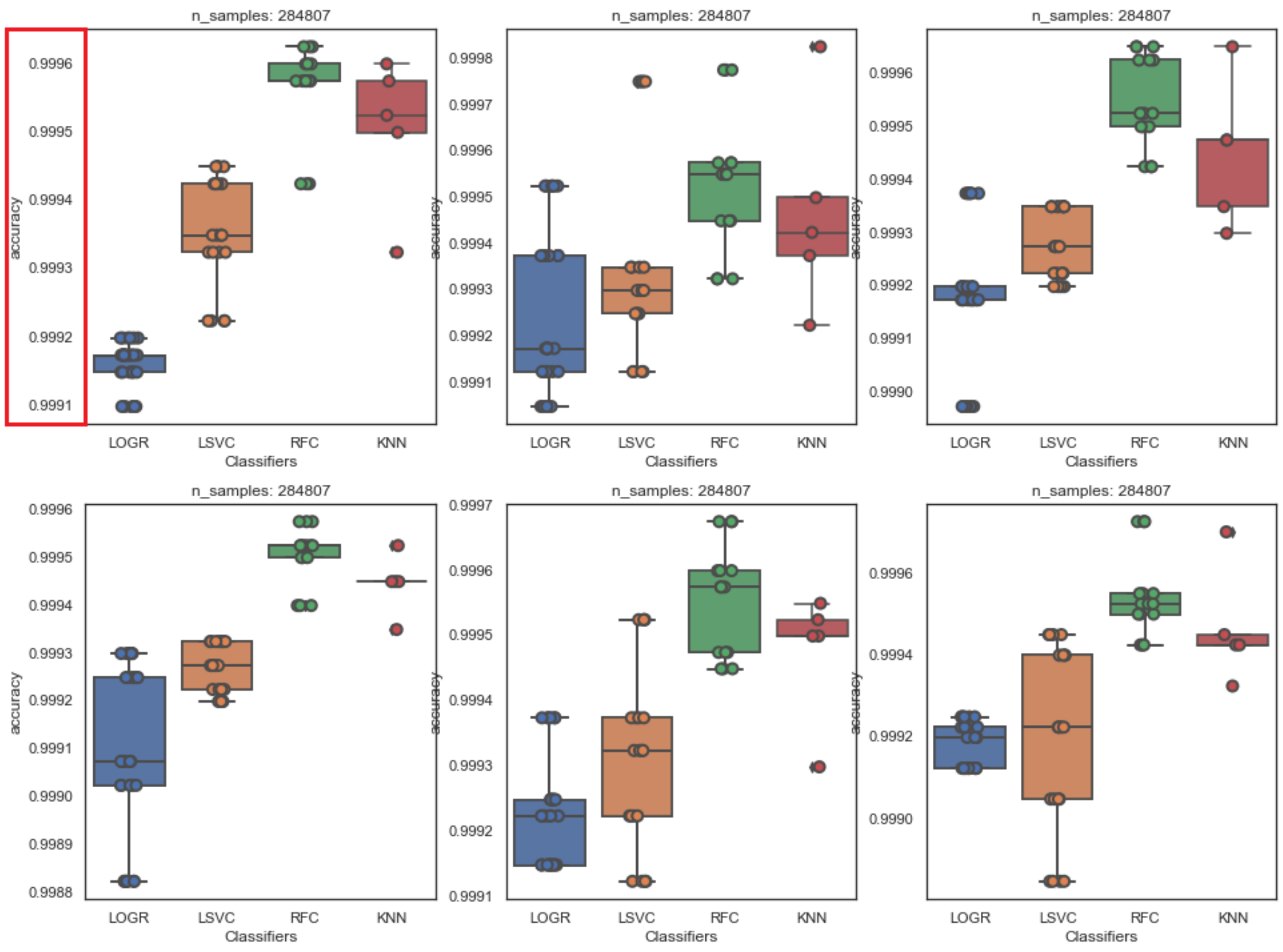
Barista et al. (2003) observed a decrease in FN when applying random oversampling, and again when applying SMOTE. Accompanying the decrease in FN was an increase in FP. Finally, improvements in AUC across the sampling methods were observed when applying oversampling, SMOTE, and again with SMOTETomek.

Evaluating Model Performance with Imbalanced Datasets

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.

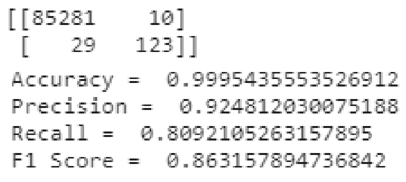
$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn}$$

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 we passed our majority class as 0 and our minority class as 1 for the dependent variables, Sklearn orients the confusion matrix as:

TN	FP
FN	TP



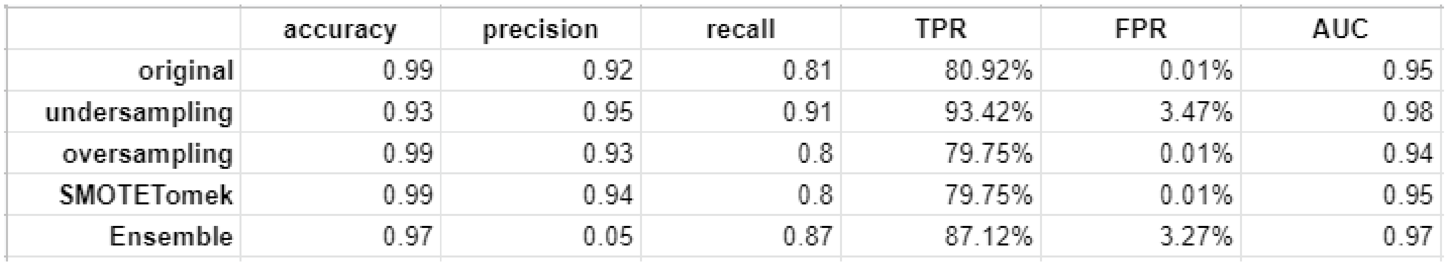
```
[[85281    10]
 [   29   123]]
Accuracy =  0.9995435553526912
Precision =  0.924812030075188
Recall =  0.8092105263157895
F1 Score =  0.863157894736842
```

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 TRP 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.

Detecting Fraudulent Activity

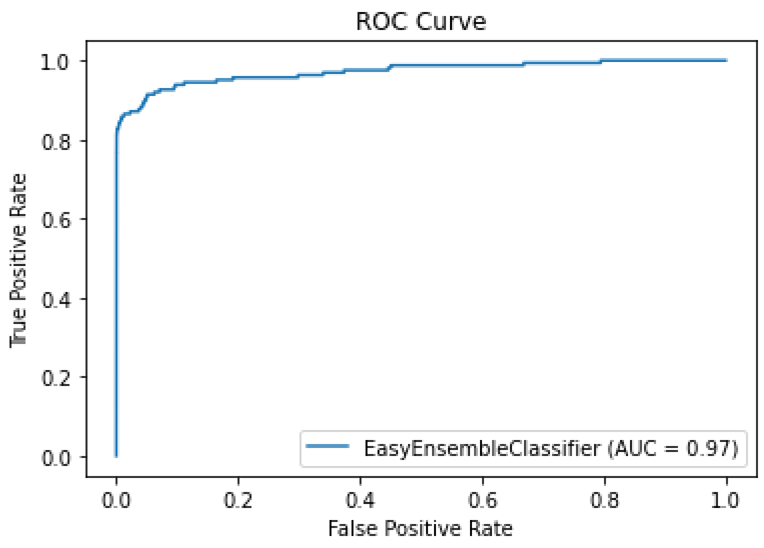
To construct a model for fraud detection, the same oversampling procedure will be used to help balance the dataset. Oversampling was chosen over SMOTETomek to simulate an enterprise environment, where tradeoffs for the development efficacy of oversampling might equate to the <1% improvement in AUC.



	accuracy	precision	recall	TPR	FPR	AUC
original	0.99	0.92	0.81	80.92%	0.01%	0.95
undersampling	0.93	0.95	0.91	93.42%	3.47%	0.98
oversampling	0.99	0.93	0.8	79.75%	0.01%	0.94
SMOTETomek	0.99	0.94	0.8	79.75%	0.01%	0.95
Ensemble	0.97	0.05	0.87	87.12%	3.27%	0.97

The last suggestion for handling imbalanced data in this essay will be the implementation of Ensemble Techniques. This model uses the *imblearn.ensemble EasyEnsembleClassifier,* and was trained and fit without hyperparameter tuning or other modeling optimization techniques.

When dealing with fraudulent activity, a business has a priority to detect fraud (TP) and prevent misclassifying fraud (FN), since these two actions create the largest cost for the customer. Using the AUC as reference, this model can be interpreted as having a ~97% probability to detect fraud (TPR) over accidentally misclassifying authorized activity (FPR).



Relying on the AUC alone doesn’t provide information related to misclassifying fraud (FN). *Recall*, another scoring metric for model performance is calculated as TP/ FN+TP. Also, it is important to acknowledge the significant change in *Precision*. Precision is calculated as TP/TP+FP. With respect to fraudulent activity, predicting fraud when a transaction is actually authorized (FP) isn’t as costly as the FN – so some leeway should be allowed conceptually if a model can be adjusted to reduce FN as the expense of FP. However, this magnitude of change warrants a further review.

Considering Recall in context of AUC helps expand the scope of the model evaluation process, so it can be evaluated with respect to its ability to detect fraud – and its ability to minimize misclassifying fraud as authorized activity; both dimensions of controlling costs that relate to the occurrence of fraudulent transactions.

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.

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