Financial Fraud Detection using Machine Learning (April 2019)

Aamenah A. Bashir, 20801620*, University of Waterloo*, Shafaq Z. Iqbal, 20697578, University of Waterloo, and Sulaiman Olabiyi, 20690635, University of Waterloo

***Abstract*— Machine learning is an immensely growing field having applications in almost every field. The need of artificial intelligence is growing day by day and there are various machine learning algorithms that help in accomplishing this task. In this project, we will be focusing on one of such issues, fraud detection. Fraud detection is difficult because it usually involves class imbalanced data, since there are significantly fewer fraudulent transactions. We will be applying different oversampling and undersampling methods to deal with the imbalanced nature of the data. Subsequently, different machine learning algorithms are tested on both imbalanced dataset and different versions of the balanced data. This would result in the identification of the best combinations of sampling techniques and machine learning algorithms, which generalize very well for datasets with imbalanced nature. We also identify which algorithms perform consistently over a wider range of imbalanced datasets.**

***Index Terms*— Imbalanced Data, SMOTE, ADASYN, Support Vector Machine, Random Forest, Decision Tree, Logistic Regression, Neural Networks, Cohen-Kappa**

I. Introduction

FRAUD is one of the fast growing issues in today's world especially when it is considered in the context of money. It has become a billion-dollar business and will keep on increasing every year. The global economic crime survey done by PwC in 2018 depicts that half i.e. 49 percent of the total 7,200 companies that they surveyed experienced some kind of fraud. This was found to be more that than 2016 study done by PwC where a greater number of companies participated, out of which 36% experienced fraud.

Technology, where it plays an important role for benefits of the world, at the same time it also has a negative impact. The crimes like frauds co-evolve with technology, especially Information Technology. Traditional and manual ways of data analysis to detect frauds is a very complex and tedious task especially when transaction per day has increased drastically over the past years. To find out one fraud out of a thousand or more transactions manually is nearly impossible.

II. Problem Definition

There are many other applications other than fraud detection that have to deal with imbalanced nature of the data. This is one of the major issues faced in the field of machine learning that how to deal with imbalanced data. Instead of applying algorithms directly one has to be completely aware of the information the classes of the dataset contain.

Fraud detection on the other is also one of the biggest issues in any domain of knowledge and it's necessary to have control over this crime for the benefit of businesses and people. Various machine learning algorithms helps in detecting these frauds in a quicker and smarter way. Thus, by applying these algorithms we can achieve better and interesting results that makes it very easy to detect frauds of any sort so that actions can be taken against it.

III. Dataset

The dataset used for this project is Credit Card Fraud Detection data from the Machine Learning Group at Université Libre de Bruxelles. The dataset contains information about 284,807 transactions over the period of two days. These transaction samples are classified into two classes, i.e. 0 (non-fraudulent) and 1 (fraudulent) transactions. It has

total 30 features, out of which 28 are unnamed due to the sensitive nature of the data. These 28 features are the result of PCA task on the data. The two unscaled features are 'Time' and 'Amount'. The time feature is redundant and was removed.

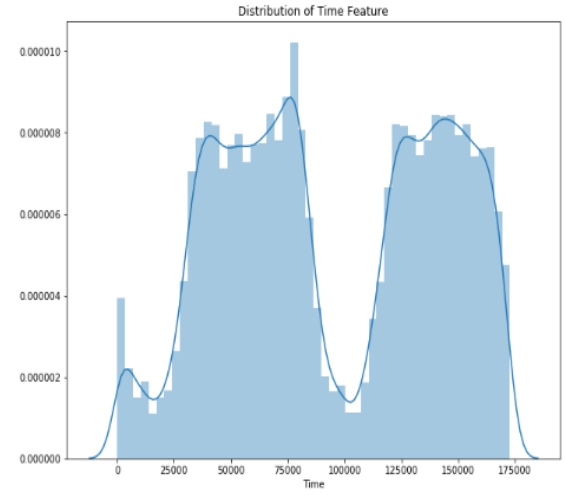

Figure 1a

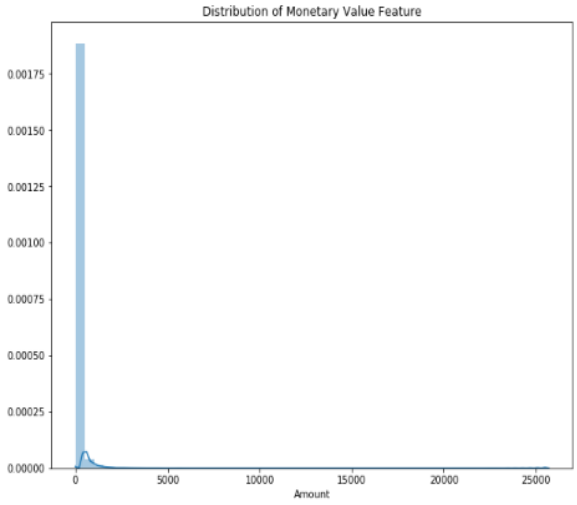

Figure 1b

Figure 1a shows the plot of time of transactions vs number of transactions. Since the data was collected over the time period of two days, the graph is bimodal which means that the dips show the night time where very few transactions took place.

Figure 1b shows the monetary values of transactions vs no of transactions. The graph shows that there are very few transactions with high monetary values and the majority have small amount of transactions.

All these applications where classes are known come under supervised learning

A. Imbalanced Data

The dataset is used in this project is an example of supervised learning as the data is classified into binary class i.e. 0 and 1. The classification algorithms that will be applied will learn from the already provided classification and the predict for any new sample given to it based on its learning. Although the data is already classified as binary class, the distribution of zeroes and ones is not uniform or near to uniform. The number of ones i.e. the number of frauds is 492 which makes 0.17% of the whole data. Whereas, the negative class i.e. the legit transactions sum up to 284315 which is 99.8% of the whole data. This situation of data distribution leads to a state known as 'Imbalanced' state in any dataset. Figure 2 shows the imbalanced property of the dataset using histogram.

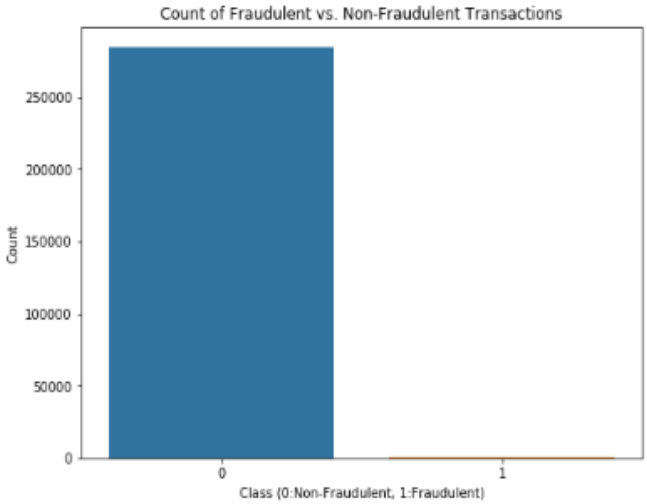

Figure 2

The main problem because of the imbalanced data arise when we apply various machine learning algorithms like Neural Networks or Random Forest, the result we get is highly biased towards the majority class. The algorithms assume that the data is distributed equally and is trained based on that. As a result, the minority class is not classified as expected. Figure 3 shows how data points are distributed in an imbalanced case where the red points show minority class and blue points show the majority class. In context of fraud detection, the goal is to find the fraudulent transactions. Since majority of the transactions are not fraudulent, it causes model to predict the fraudulent transactions as valid which is in fact unwanted result. This gives rise to accuracy paradox which means that the overall accuracy of such model will be very good because of the bias of the majority class. Some real-world examples of imbalanced data include oil spill detection, network intrusion detection, fraud detection and rare diseases [1][2][3].

Figure 3 [4] – Imbalanced Class Distribution

IV. Imbalanced Data – Methods and Approach

There are several ways and techniques to deal with imbalanced nature of any dataset. The most common of which is resampling the data which includes two major techniques; oversampling and undersampling. The former is used to add more minority class and the latter removes the data points from the majority class to balance the skewness in the class distribution. Application of these techniques on the data set is the part of the preprocessing of the data

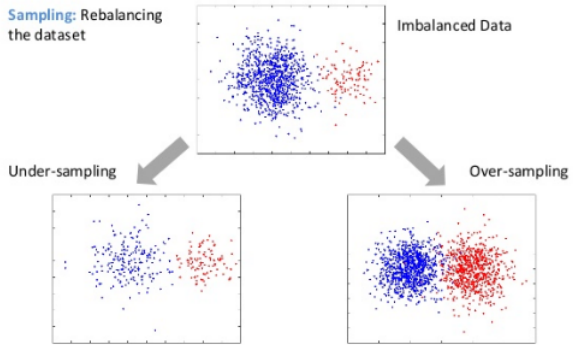

Figure 4 [5]

A. Oversampling Techniques

Oversampling, also known as over sampling is used to increase the minority class to balance out the class distribution. In this project the number of frauds which belongs to class 1 are increased by using different over sampling techniques such a SMOTE and ADASYN.

1. SMOTE (Synthetic Minority Over Sampling Techniques):

SMOTE technique adds data points by using n-neighbors technique. It finds n-nearest neighbors in the minority class for every sample and draws a line between all the neighbors after which it generates points at random position depending on the parameter telling how many more points are needed. Figure 5 shows the generation of data points using SMOTE taking 5 nearest neighbors. For all 5 neighbors, it draws line and place points on the line where it is a minority class. [6]

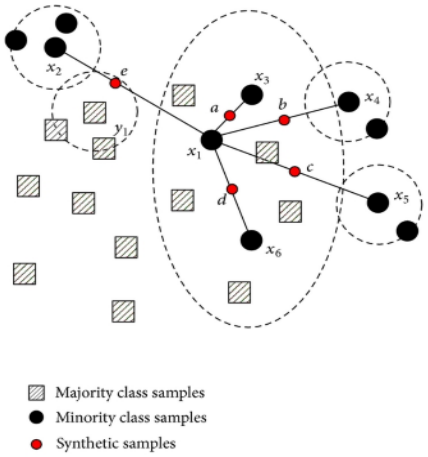

Figure 5 [6]

2. ADASYN (Adaptive Synthetic Sampling Technique):

ADASYN is an improvement of SMOTE. The samples added in this technique has more variance making it more realistic whereas in SMOTE the samples are linearly correlated to the parent values. [6]

B. Undersampling Techniques

Undersampling is an efficient method for balancing the classes while imbalance learning. Undersampling involves using a subset of the majority class to train the classifier. Since many majority class samples are ignored, the training set becomes more balanced and the training process becomes faster.

Two types of undersampling techniques were utilized:

1. Random Undersampling

Random undersampling is a non-heuristic method that aims to balance class distribution through the random elimination of majority class examples. Using random undersampling it is possible to achieve a desired proportion of the classes as it is a random selection with replacement of the subsets of the desired class. Three proportions were considered to study the impact of the proportions.

2. Informative Undersampling

Informative or cleaning undersampling techniques were also used where the dataset is cleaned using specific heuristics. Using this technique, it is not possible to specify the desired proportion of the classes and therefore the classes can still end up being imbalanced. Possible heuristics used for this method are listed below:

i. Edited Nearest Neighbor (ENN) - ENN removes any example whose class label differs from the class of at least two of its three nearest neighbors. [7]

ii. Condensed Nearest Neighbor (CNN) - Hart's Condensed Nearest Neighbor Rule (CNN) is

used to find a consistent subset of examples. [7]

iii. Neighborhood Cleaning Rule (NCL) - Neighborhood Cleaning Rule (NCL) uses the Wilson's Edited Nearest Neighbor Rule (ENN) to remove majority class examples. [7]

iv. One Sided Selection (OSS) - One-sided selection (OSS) is an undersampling method resulting from the application of Tomek links followed by the application of CNN. [7]

V. Machine Learning Algorithms

Support Vector Machine - The aim of this technique is to fit a hyperplane between the available data points in space while ensuring that the samples are separated by the largest gap possible. Following this the data points are classified by checking which side of the plane the data points fall on. SVM typically involves using a binary linear classifier [8]. For the project, the Python "scikit-learn" library was used for the SVM algorithm.

Random Forest - Random forest is a technique that constructs multiple decision trees and within RF, ensemble methods generally use several weak classifiers to achieve a better performance than any one algorithm on its own. The random forest technique has the advantage of performing relatively well on small amounts of data and can perform efficiently on large amounts of data [9]. The RF algorithm used is from Python's "scikit-learn" library [10].

Random forests are a group of classification or regression trees that were trained on bootstrap samples of the training data using random feature selection in the process. Each tree votes for the most popular class once the trees have been generated and these tree voting processes are collectively defined as random forests. [11]

Decision Tree - Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. [12] The DT algorithm used is from Python's "scikit-learn" library which uses the CART algorithm. CART (Classification and Regression Trees) is very similar to C4.5, but it differs in that it supports numerical target variables (regression) and does not compute rule sets. CART constructs binary trees using the feature and threshold that yield the largest information gain at each node.

Neural Network (Multi-Layer Perceptron) - Neural networks (NN) are mathematical representations based on the processes of the human brain. The advantage of a NN is its flexibility in modelling non-linear datasets. For our project we used the most widely used type of NN, the multilayer perceptron. A multi-layered perceptron is usually composed of an input layer (consisting of neurons for all input variables), a hidden layer (consisting of any number of hidden neurons), and an output layer (in our case, one neuron). Each neuron processes its inputs and transmits its output value to the neurons in the subsequent layer. Each connection between neurons is assigned a weight during training [13].

During modelling, the weights of the network are first randomly initialized and then iteratively adjusted to minimize an objective function, e.g., the sum of squared errors. This iterative procedure can be based on simple gradient descent learning or more advanced optimization methods. [13]

Logistic Regression - Logistic regression is a statistical method for predicting binary classes. The outcome or target variable is dichotomous in nature meaning there are only two possible classes. It computes the probability of an event occurrence and is a special case of linear regression where the target variable is categorical in nature. It uses a log of odds as the dependent variable. For the project, the Python sklearn.linear_model library was used for Logistic Regression. [14]

VI. Evaluation Metrics

For Fraud Detection, different evaluation metrics can be prioritized when selecting the optimal combination of sampling technique and machine learning algorithm. The specific importance placed by the bank on detection of a fraudulent transaction also impacts which evaluation metric is important. If identifying both fraudulent and non-fraudulent transactions correctly is important to the bank, then popular metrics like overall accuracy, log-loss, F1-score, and area under the ROC curve are ideal. If either one of the two transaction types are more important to the bank, appropriate metrics can include precision and recall. Another relevant metric is the Cohen's Kappa statistic. It is an under-utilized metric that focuses on evaluating the overall performance of the classifier for an imbalanced data. We discuss the relevant evaluation metrics considered in this project below.

VII. Accuracy

Accuracy is basically the total percentage of the correct predictions out of all the predictions made by the classifier. The formula is given as

$$(TP + TN)/ (TP + FP + TN + FN)$$

where

TP = True Positives: the total number of accurate predictions for the positive class

TN = True Negatives: the total number of accurate predictions for the negative class

FP = False Positives: the total number of inaccurate predictions for the positive class

FN = False Negatives: the total number of inaccurate predictions for the negative class

Accuracy is useful when both classes are important but can be inadequate for imbalanced datasets

VIII. RECALL

Recall evaluates how well the classifier predicts the positive

class based on the total number of actual positives in the data. It is also called sensitivity and the formula is given as:

TP / (TP + FN)

Recall neglects the performance of the classifier for the negative class. Depending on the objective of the classification task, it can be useful and combined with another metric, or relatively inadequate.

IX. PRECISION

Precisions evaluates how well the classifier predicts the positive class based on the total number of positives predicted by the classifier. The formula is given as:

TP / (TP + FP)

Precision also neglects the performance of the classifier for the negative class. It is mostly useful when combined with another more comprehensive metric.

X. F1-SCORE

F1-Score is a comprehensive metric that evaluates the classifier by considering the false positives and false negatives performance. It is usually better than accuracy for uneven dataset. It is basically a weighted average of precision and recall and thus given by the formula

2 * (Precision * Recall) / (Precision + Recall)

XI. AREA UNDER THE ROC CURVE

ROC stands for receiver operating characteristic and it is curve that shows the classifier performance at different classification thresholds. The ROC curve is constructed using the recall (also known as true positive rate) and the false positive rate (FP / (FP + TN). It also depicts the trade-off between the true positive rate and the false positive rate. The area under the ROC curve (AUC) is the entire area underneath a two-dimensional ROC curve.

XII. COHEN'S KAPPA

Cohen's kappa evaluates how well the classifier performs compared to the performance obtained from a classifier that simply guesses at random using class frequency distribution. The formula is given as:

$$(po - pe) / (1 - pe) \quad [15]$$

where

po = observed agreement of the classifier for the two classes

pe = expected agreement of a hypothetical classifier that uses chance

The calculated value can be interpreted in different ways, but one standardized way is provided by Landis and Koch [16]. It is shown as:

Kappa	Agreement
< 0	Less than chance agreement
0.01 – 0.20	Slight agreement
0.21 – 0.40	Fair agreement
0.41 – 0.60	Moderate agreement
0.61 – 0.80	Substantial agreement
0.81 – 0.99	Almost perfect agreement

XIII. EVALUATION METRIC SELECTION

For metric selection for our project, we consider the perspective of a bank that places much more emphasis on identifying fraudulent transactions but still want to correctly identify other transactions as correctly as possible. This can be tricky for an imbalanced dataset. Considering the descriptions of the metrics above, the positive class in our project are the fraudulent transactions and the negative class are the non-fraudulent transactions. Thus, true positives are the correctly identified fraudulent transactions, false positives are non-fraudulent transactions identified as fraudulent, true negatives are non-fraudulent transactions identified correctly and false negatives are fraudulent transactions identified as non-fraudulent. Consequently, true positives and false negatives are very important to the bank

Ideally, recall seems like a good metric for such a scenario and it is used by other practitioners that are working on fraud detection. However, for an imbalanced dataset, some combinations of sampling technique and algorithm give very good recall scores while having weak overall performance. In order to combat this, we decided to use recall along with one of the other more comprehensive metrics. Such metric can combine both precision and recall or evaluate the overall performance of the classifier with emphasis on the positive class.

Accuracy while comprehensive gives equal importance to both types of transactions. F1-Score combines precision and recall equally to calculate a harmonic mean, but it does not take into consideration true negatives and can give significantly different optima for stepwise changes in the sampling technique for imbalanced data or machine learning algorithm hyperparameters. It also does not have a very good intuitive explanation. Area under the ROC curve (AUC) is a good candidate for supporting the recall score but it summarizes the classifier performance over regions of the ROC space that are not important. Consequently, it is not easy to combine with the recall score for classifier selection in an imbalanced dataset because most classifiers perform well over large regions of the ROC.

XIV. EVALUATION METRIC CRITERIA (SOMETHING NEW)

While F1-measure, accuracy and AUC can still be used as supporting metric to a reasonable extent, we decided to use Cohen's Kappa instead because it works very well for an imbalanced dataset without sacrificing overall performance. Compared to accuracy, F1-Score and AUC, Cohen's Kappa informs about how much better a classifier is performing over the performance of a classifier that simply guesses at random using class distribution. Combining the recall scores and Cohen's Kappa results in a high selectivity among the different combinations of sampling technique and machine learning algorithm. Cohen's Kappa can help us identify and eliminate sampling and classifiers combinations that have high recall scores and an overall performance that is mostly from

the effect of the imbalanced data. Such classifiers overfit on the positive class and use the imbalanced nature of the data to mask their actual overall effectiveness. Subsequently, we end up with sampling and classifier combinations that have good recall and an overall performance that is propped up by the class imbalance. Additionally, we have included the accuracy of a sampling and classifier combination to support our final decisions

When discussing the possible evaluation metrics, we emphasize combinations of sampling technique and machine learning algorithms because it was observed that classifiers performed different when trained on data resulting from different sampling techniques. This is due to the class imbalance of the data, which has a significant effect on the learning algorithms.

XV. Results & Discussion

We have considered multiple combinations of sampling techniques, machine learning algorithms and algorithm hyperparameters. While the hyperparameters can be finetuned to further and marginally improve the accuracies of the classifiers, the selection of sampling technique and machine learning algorithm is much more important. Out of the six different machine learning approaches used, hyperparameter tuning was most important for four of them, namely: SVM, NLP and Random Forest. Hence, we carried only carried out hyperparameter tuning for the aforementioned learning algorithms on the original dataset. The obtained optimal parameters shown in the table below, were then used throughout the rest of the project.

Algorithm	**Optimal Hyperparameters**
SVM	C = 1
Random Forest	number of estimators = 25
Neural Network (MLP)	solver = adam, hidden layer sizes = (15, 15), alpha = 0.0001, activation = tanh

Using both Recall and Cohen's Kappa for primary evaluation of the different combinations, the machine learning algorithm performance results for the actual imbalanced data, six oversampled datasets and six undersampled datasets are given in the table below.

ML Algorithms	Sampling Methodology		Recall	Kappa	Accuracy
SVM	Imbalanced		0.6	0.71	0.999
	Over-sampling	SMOTE 1	0.84	0.60	0.995
		SMOTE 2	0.87	0.37	0.993
		SMOTE 3	0.87	0.29	0.993
		ADASYN 1	0.86	0.30	0.985
		ADASYN 2	0.87	0.19	0.971
		ADASYN 3	0.88	0.14	0.957
	Under-sampling	Random - 50% Underclass	0.91	0.09	0.98
		Random - 10% Underclass	0.84	0.73	0.997
		Random - 1% Underclass	0.80	0.82	0.998
		ENN Under-sampling	0.65	0.75	0.999
		CNN Under-sampling	0.74	0.47	0.999
		NCL Under-sampling	0.71	0.79	0.999
Gaussian Naïve Bayes	Imbalanced		0.84	0.11	0.978
	Over-sampling	SMOTE 1	0.86	0.12	0.983
		SMOTE 2	0.86	0.11	0.982
		SMOTE 3	0.86	0.11	0.981
		ADASYN 1	0.87	0.08	0.969
		ADASYN 2	0.87	0.08	0.967
		ADASYN 3	0.87	0.08	0.967
	Under-sampling	Random - 50% Underclass	0.85	0.07	0.961
		Random - 10% Underclass	0.85	0.12	0.978
		Random - 1% Underclass	0.84	0.11	0.978
		ENN Under-sampling	0.84	0.11	0.978
		CNN Under-sampling	0.67	0.26	0.994
		NCL Under-sampling	0.84	0.11	0.978
Neural Network (MLP)	Imbalanced		0.69	0.77	0.999
	Over-sampling	SMOTE 1	0.86	0.50	0.990
		SMOTE 2	0.82	0.65	0.992
		SMOTE 3	0.82	0.53	0.983
		ADASYN 1	0.84	0.60	0.945
		ADASYN 2	0.80	0.56	0.962
		ADASYN 3	0.80	0.47	0.997
	Under-sampling	Random - 50% Underclass	0.94	0.05	0.942
		Random - 10% Underclass	0.88	0.38	0.995

		Random - 1% Underclass	0.81	0.80	0.999
		ENN Under-sampling	0.82	0.84	0.999
		CNN Under-sampling	0.80	0.75	0.999
		NCL Under-sampling	0.80	0.82	0.999
Random Forest	Imbalanced		0.78	0.84	0.999
	Over-sampling	SMOTE 1	0.84	0.84	0.999
		SMOTE 2	0.82	0.84	0.999
		SMOTE 3	0.82	0.53	0.999
		ADASYN 1	0.84	0.60	0.999
		ADASYN 2	0.80	0.56	0.999
		ADASYN 3	0.80	0.47	0.999
	Under-sampling	Random - 50% Underclass	0.90	0.10	0.976
		Random - 10% Underclass	0.85	0.66	0.999
		Random - 1% Underclass	0.80	0.81	0.999
		ENN Under-sampling	0.76	0.82	0.999
		CNN Under-sampling	0.84	0.64	0.998
		NCL Under-sampling	0.76	0.83	0.999
Decision Tree	Imbalanced		0.78	0.83	0.999
	Over-sampling	SMOTE 1	0.84	0.85	0.999
		SMOTE 2	0.84	0.85	0.999
		SMOTE 3	0.84	0.85	0.999
		ADASYN 1	0.81	0.84	0.999
		ADASYN 2	0.81	0.84	0.999
		ADASYN 3	0.81	0.84	0.999
	Under-sampling	Random - 50% Underclass	0.89	0.04	0.924
		Random - 10% Underclass	0.85	0.17	0.986
		Random - 1% Underclass	0.82	0.57	0.998
		ENN Under-sampling	0.79	0.79	0.999
		CNN Under-sampling	0.85	0.10	0.975
		NCL Under-sampling	0.76	0.83	0.999
Logistic Regression	Imbalanced		0.62	0.70	0.999
	Over-sampling	SMOTE 1	0.86	0.51	0.993
		SMOTE 2	0.86	0.35	0.992
		SMOTE 3	0.88	0.29	0.988
		ADASYN 1	0.87	0.27	0.980
		ADASYN 2	0.88	0.17	0.964
		ADASYN 3	0.90	0.12	0.949
	Under-sampling	Random - 50% Underclass	0.91	0.08	0.966
		Random - 10% Underclass	0.85	0.56	0.998
		Random - 1% Underclass	0.82	0.81	0.999
		ENN Under-sampling	0.66	0.75	0.999
		CNN Under-sampling	0.84	0.22	0.999
		NCL Under-sampling	0.71	0.79	0.999

We identify the best generalizing models as the models that combines a top precision score with a solid Kappa score and above average accuracy. Those criteria guide the sampling and learning algorithm selections made below.

For oversampled data, the three best combination are:

- SMOTE 1, Random Forest (Recall: 0.84,Kappa = 0.84)
- SMOTE 1, Decision Tree (Recall: 0.84, Kappa = 0.85)
- SMOTE 3, Decision Tree (Recall: 0.84, Kappa = 0.85)

For undersampled data, the three best combination are:

- Random Undersampling (10%) with Random Forest (Recall: 0.85, Kappa = 0.66)
- CNN Undersampling with Random Forest (Recall: 0.84, Kappa = 0.64)
- Random Undersampling (1%) with Logistic Regression (Recall: 0.82, Kappa = 0.81)

Average Recall values for the imbalanced data is about 0.76 (Kappa = 0.83) but can be as high as 0.81 after parameter tuning. Hence, there can be an argument to be made against any form of sampling. Estimated recall values from other practitioners that use the same dataset and also avoid overfitting for Recall is about 0.84. Many models from other practitioners for this dataset give good recall but low Kappa or low accuracy. Other models focus on ROC and neglect recall

optimization.

The minimum Recall for our selected models is 0.82 and the minimum Kappa is 0.61 which implies substantial agreement for the model. The highest Recall value obtained is 0.85 from random forest using 10% random undersampling. This is interesting 10% random undersampling results in significantly smaller dataset compared to the original data and oversampling approaches. This could be because the 10% random undersampled data contains the exact same fraudulent transactions in the original dataset and we are prioritizing classification of fraudulent transactions. Oversampled approaches don't necessarily vary the critical features of the minority class (fraudulent transactions in this case) so it is possible a classifier can learn from a 10% random undersampled data the same rules from an oversampled version of the data.

XVI. Conclusion

Independent of sampling technique, Random Forrest and Decision Trees algorithms are recommended for fraud detection applications with class imbalance of this nature. While Logistic Regression and Support Vector Machines gives very good performance for a specific sampling technique at random, both Random Forrest and Decision Trees perform consistently over a larger range of different sampling techniques. The arguments for oversampling or undersampling are not always clear, as it was observed the performance for some of the sampling and algorithm combinations were just about the performance expected using the imbalanced data as it is. However, if sampling has been decided for, then SMOTE and Random Undersampling (range of 1-10% for the minority class) are the suggested sampling techniques given their better performance over a range of algorithms. Random undersampled data despite using significantly smaller datasets can be the best option when the primary goal of the classification is identification of the minority class (fraudulent transactions in this case).

XVII. Recommendations

The following recommendations are offered after the completion of the project:

- Hyperparameters for the machine learning algorithms can be finetuned further to marginally increase the obtained accuracies for each combination of sampling technique and learning algorithm
- Additional sampling techniques can be trialed and evaluated. Sampling techniques not attempted include Cluster Centroids, Tomek Links, Instance Hardness Threshold, One Sided Selection, Repeated Edited Nearest Neighbors, SMOTEENN (combines oversampling and undersampling) and Keras Balanced Batch Generator.
- Class imbalance of fraud detection data can be an ideal use case for General Adversarial Networks (GAN). GANs can be trialed to generate synthetic fraudulent transactions with more variety and effectiveness. This can improve the performance of the learning algorithms
- Selected combinations of sampling technique and learning algorithms can be trialed in a different industrial context with similar objectives and class imbalance. A suitable example of such application is identification of fraudulent insurance claims from in the insurance industry.
- Feature extraction and selection were not carried out significantly due to the nature (PCA data) of the available data for this project. Both operations can be analyzed in detail further and used to perform techniques like cost-based learning when more information about the features is available.

References

[1] "On the Class Imbalance Problem", Xinjian, Guo, Yilong, Yin, Cailing, Dong, Gongping, Yang, Guangtong, Zhou, School of Computer Science and Technology, Shandong University, Jinan, 250101, China.

[2] Kotsiantis, D. Kanellopoulos and P. Pintelas, "Handling imbalanced datasets: A review", GESTS International Transactions on Computer Science and Engineering 30 (1) , 2006, pp. 25-36.

[3] Visa and A. Ralescu, "Issues in Mining Imbalanced Data Sets-A Review Paper", in Proceedings of the Sixteen Midwest Artificial Intelligence and Cognitive Science Conference, MAICS-2005, Dayton, 2005, pp. 67-73.

[4] https://www.datascience.com/blog/imbalanced-data

[5] http://api.ning.com/files/vvHEZw33BGqEUW8aBYm4epYJWOfSeUBPVQAsgz7aWaNe0pmDBsjgggBxsyq*8VU1FdBshuTDdL2-bp2ALs0E-0kpCV5kVdwu/imbdata.png

[6] I. Bhattacharyya, "SMOTE and ADASYN (Handling Imbalanced Data Set)", https://medium.com/coinmonks/smote-and-adasyn-handling-imbalanced-data-set-34f5223e167

[7] G. Batista et al., "A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data" Volume 6 Issue 1

[8] Y. Obeidi, "Unbalanced Credit Card Fraud Detection", 2017 https://github.com/yazanobeidi/fraud-detection/blob/master/credit_card_fraud_detection_yazan_obeidi.pdf

[9] D. Storey, "Random Forests, Decision Trees, and Ensemble Methods Explained", 2018 https://www.datascience.com/blog/random-forests-decision-trees-ensemble-methods

[10] Pedregosa, et. al., "Scikit-learn: Machine Learning in Python" in JMLR 12, pp. 2825-2830, 2011. [Online] Available: http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

[11] L. Breiman, A Cutler. "Random Forests", 2001, https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

[12] "1.10. Decision Trees", https://scikit-learn.org/stable/modules/tree.html

[13] I. Brown et al. "An experimental comparison of classification algorithms for imbalanced credit scoring data sets", Expert Systems with Applications, Volume 39, Issue 3, 15 February 2012, Pages 3446-3453, https://www.sciencedirect.com/science/article/pii/S095741741101342X#b0020

[14] "Logistic Regression", https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[15] Cohen J. "A coefficient of agreement for nominal scales", 1960, Educational and Psychological Measurement, 20(1), 37-46

[16] Landis, J. R. et al. "The measurement of observer agreement for categorical data", 1977. Biometrics 33 (1) 159-174,