Comparative study of Resampling and Machine Learning on Fraud Detection

1st Shashankk Shekar Chaturvedi
Department of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, USA
schaturv1@stevens.edu

2nd Shahaji Deshmukh
Department of Mathematical Sciences
Stevens Institute of Technology
Hoboken, USA
sdeshmuk1@stevens.edu

3rd Anubhav Jaiswal
Department of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, USA
ajaiswal1@stevens.edu

Abstract—In response to the escalating challenges of credit card fraud in online transactions, our project rigorously explores the optimal Machine Learning (ML) algorithms, which including Logistic Regression, k-Nearest Neighbors (KNN), Decision Trees, Random Forest,Support Vector Machine (SVM), and XGBoost, for fraud detection. Focusing on the critical issue of class imbalance within fraud detection datasets, we employ Random Oversampling and SMOTE as essential techniques. Through this comparative analysis, we aim to provide valuable insights into the strengths and weaknesses of each algorithm, contributing to the ongoing enhancement of ML-based solutions for more secure online credit card transactions.

I. Introduction

”According to the Nilson Report in October 2023, global losses from credit card fraud are expected to soar over $397.4 billion in the next 10 years, with $165.1 billion of those losses happening in the U.S. alone. The significance of credit card fraud detection has never been more pronounced. The ramifications of these escalating costs reverberate across the entire payment lifecycle, impacting banks, credit card companies, consumers, and businesses alike. A crucial aspect is understanding the nuances of credit card fraud to develop effective strategies.

Credit card fraud can be categorized into two forms: Card Present Fraud, which relies on physically stolen or duplicated cards and employs tactics like theft, pickpocketing, and card skimmers to capture and replicate details at payment points; and Card-Not-Present Fraud, where fraudsters operate without physical possession of the card, employing account takeover techniques to manipulate account information, authenticate transactions, and evade fraud detection tools. In recent years, we have seen a surge in the second category of credit card fraud, i.e., Card-Not-Present Fraud.

In the realm of credit card fraud detection, Machine Learning (ML) models play a pivotal role. Among the models available in ML, we compare two approaches, namely: Predictive Models and Outlier Models. Predictive models like Logistic Regression stand out as pragmatic choices. Leveraging a binary classification approach, they are easy to implement and computationally efficient, although they assume a linear relationship between features. Another widely used predictive model is Decision Trees, known for their intuitive nature and capacity to capture complex non-linear relationships.

On the outlier modeling front, XGBoost stands out as a powerful choice. As a gradient boosting algorithm, XGBoost is well-suited for anomaly detection tasks, showcasing robust performance and adaptability. It effectively identifies outliers while offering advantages such as scalability and the ability to handle large datasets. Moreover, XGBoost provides flexibility in handling complex relationships within high-dimensional spaces. However, it’s important to note that the performance of XGBoost in outlier detection may be influenced by the appropriate tuning of hyperparameters. Careful consideration and optimization of hyperparameters will be crucial to ensuring the effectiveness of XGBoost in this context.

While assessing these models, evaluation metrics such as precision, recall, F1 score, and the area under the ROC curve (AUC-ROC) are employed. Furthermore, careful data preprocessing, including strategies for handling imbalanced datasets, and hyperparameter tuning are essential for optimal performance. A figure describing the dataset is provided for reference in Fig. 1

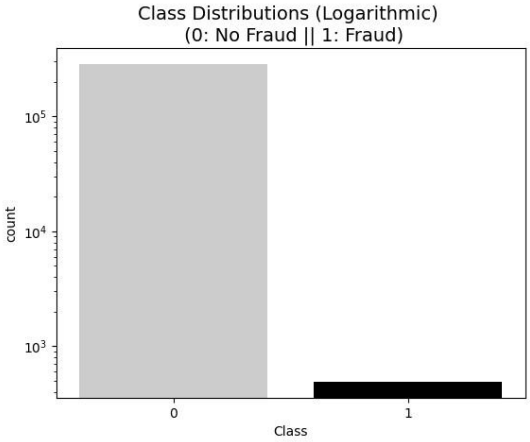

Fig. 1: Logarthmic Class Distribution Plot

II. Related Work

According to ”Analysis of Machine Learning Techniques for Credit Card Fraud Detection” a study by Abrar H. Nadim et al. [?] addresses the rise in credit card fraud, leveraging Without Handling Imbalance

	Precision	Recall	F1-Score
Normal	1.00	1.00	1.00
Fraud	0.77	0.65	0.70

Handling Imbalance by RuS (ratio < 0.04)			
	Precision	Recall	F1-Score
Normal	1.00	1.00	1.00
Fraud	0.60	0.80	0.68

Handling Imbalance by RoS (ratio < 0.04)			
	Precision	Recall	F1-Score
Normal	1.00	1.00	1.00
Fraud	0.63	0.80	0.71

Handling Imbalance by SMoTE (ratio < 0.04) Precision			
	Recall	F1-Score	
Normal	1.00	1.00	1.00
Fraud	0.65	0.81	0.72

TABLE I: Performance Metrics for Logistic regression with l2 regularization.

machine learning algorithms. It evaluates Logistic Regression, Random Forest, Decision Tree, and SVM, focusing on accuracy, sensitivity, specificity, and precision. The dataset's severe right skewness is tackled through undersampling and oversampling, and the system is implemented in Python.

In Iain Brown and Christophe Mues' study, "An Experimental Comparison of Classification Algorithms for Imbalanced Credit Scoring Data Sets," [?] the authors evaluate techniques for analyzing imbalanced credit scoring data. Assessing traditional methods alongside gradient boosting, support vector machines, and random forests, they discover that random forest and gradient boosting classifiers excel in credit scoring, effectively managing class imbalances.

The paper by Abdulghani, UCAN, and Alheeti [?], presented at the 2021 14th International Conference on Developments in eSystems Engineering (DeSE), focuses on "Credit Card Fraud Detection Using XGBoost Algorithm." The study explores the application of the XGBoost algorithm for enhancing credit card fraud detection. By addressing this crucial issue within eSystems, the authors contribute valuable insights into the effectiveness of XGBoost in bolstering fraud detection mechanisms, providing a relevant foundation for related works in the field.

We intend to use basic oversampling and under-sampling techniques along with SMOTE to handle the class imbalance. We intend to examine most of the algorithms mentioned above. While specifically testing the Accuracy and F1-Score of (XGBoost).

III. Our Solution

Our Data Pr-Processing Step was straight forward, we had two features not scaled namely 'Amount' and 'Time', We used "StandardScaler" to scale the features in situ. There is no need to Perform a PCA Transformation as the dataset is also We endeavour to test the metrics described above on K-nearest neighbours, Decision Trees, Random Forest, and SVM along with an ensemble method of Extreme Gradient Boosting. To provide a comparative analysis on the above algorithms along with the affects of re-sampling strategies.

A. Description of Dataset

In our preliminary analysis of the data, we noted that, apart from the 'Transaction' and 'Amount' columns, the specifics of other features remain undisclosed due to privacy considerations, although they have undergone scaling. The 'Amount', 'Time' and 'Class' columns were not scaled. More Importantly, there are no missing values in the dataset. The vast majority of transactions (99.83%) are categorized as Non-Fraud or Normal, with Fraudulent transactions comprising only 0.17% of the total entries as shown in Table II. The dataset, representing credit card transactions by European cardholders in September 2013 over a two-day period with 284,807 transactions, includes 492 identified frauds. The features are anonymized in accordance with GDPR guidelines. Documentation indicates that, except for 'Time' and 'Amount', all features underwent PCA transformation for dimensionality reduction, with the 'V' features presumably scaled. These insights lay the groundwork for more in-depth analyses and model development in the realm of fraud detection.

TABLE II: Tabular Class Sample Count

Class	Count	Percentage
0	284315	99.83
1	492	0.17

Time: is described as the difference between the first transaction in the data set with the i^{th} transaction. V1-V28: Anatomized features representing various transaction attributes

(e.g., time, location, etc.) Amount: The transaction amount, Class: Binary label indicating whether the transaction is fraudulent (1) or not (0)

B. Machine Learning Algorithms

1) Logistic Regression

Logistic regression stands out as the most widely utilized classification model within machine learning. It computes the probability of each class to make predictions, combining input variables with corresponding weights. If we consider x as an independent variable and y as a dependent variable, the linear regression representation is given by $y=a_0+a_1x$, where a_0 represents the bias term and a_1 is the weight for the input variable x. In the context of logistic regression we employ a

logistic function which forecasts the probability of each class and utilizing a sigmoid function.

For optimal result we have used a Logistic Regression Algorithm with l2 regularization, as L2 is more Robust against outliers in the dataset, After trying performing grid search of parameters our algorithms performed sub-optimaly until we selected the best parameters suggested by Mohbey, Krishna Kumar, et al [?]. Performing KFold Cross Validtion of 5-folds. *2) K-Nearest Neighbours*

KNN is a simple and intuitive algorithm that classifies a data point based on the majority class of its k-nearest neighbors in the feature space. It makes predictions by identifying the class that is most common among its nearest neighbors. Important hyperparameters for KNN include nearest neighbors count, which specifies the number of neighbors to consider, weights for determining the weighting scheme for neighbors (uniform or distance-based), and p, the power parameter for the Minkowski distance metric. KNN is effective in identifying patterns based on the similarity of data points. In fraud detection, fraudulent transactions may exhibit similar characteristics, making KNN a suitable algorithm to detect anomalies by considering the proximity of transactions in the feature space. Decision trees are capable of capturing complex decision boundaries and relationships within the data. This is beneficial in fraud detection scenarios where fraudulent activities may be associated with specific patterns or sequences of events.

3) Stochastic Linear Support Vector Machine

Support Vector Machines aim to find the hyperplane that best separates data into different classes. Stochastic Linear SVM is a variant that uses a subset of the training data for each iteration, making it computationally more efficient for large datasets. Linear SVM uses a linear kernel function. SMV also uses regularizations terms, which mainly are L1, L2 and Elasticnet. SVMs, especially the stochastic variant, are well-suited for high-dimensional data, making them applicable when dealing with datasets resulting from techniques like PCA. SVMs can find optimal hyperplanes to separate different classes and can be adjusted to handle imbalanced datasets by setting appropriate class weights.

4) Decision Tree

A decision tree is a tree-like model where each internal node represents a decision based on a particular feature, and each leaf node represents the predicted class. It recursively splits the data into subsets based on the most informative feature at each step. A Decision Tree is a tree-like model where each internal node represents a decision based on a particular feature, and each leaf node represents the predicted class. Hyperparameters such as max depth control the maximum depth of the tree, minimum samples split set the minimum number of samples required to split an internal node, minimum samples leaf dictate the minimum number of samples required to be at a leaf node, and criterion specifies the function to measure the quality of a split (e.g., ”gini” for Gini impurity or ”entropy” for information gain). Decision trees are capable of capturing complex decision boundaries and relationships within the data. This is beneficial in fraud detection scenarios where fraudulent activities may be associated with specific patterns or sequences of events.

5) Random Forest Classifier

Random Forest is an ensemble learning method that builds multiple decision trees and combines their predictions. It introduces randomness during the tree-building process by considering random subsets of features and data points, which helps to reduce overfitting. Random Forest is an ensemble learning method that builds multiple decision trees and combines their predictions. In addition to the hyperparameters of Decision Trees, Random Forest introduces parameters like number of estimators (number of trees in the forest), maximum features, maximum number of features considered for splitting a node, and bootstraping. Random Forests build multiple decision trees and combine their predictions, providing robustness and reducing overfitting. In fraud detection, where the dataset may be imbalanced with a majority of genuine transactions, Random Forests can effectively handle this imbalance and identify patterns associated with fraudulent activities.

6) Xgboost

XGBoost (Extreme Gradient Boosting) is a gradient boosting algorithm that builds an ensemble of weak learners, typically decision trees. It optimizes a loss function by adding trees sequentially, with each new tree correcting errors made by the previous ones. It is known for its speed and high performance. Hyperparameters such as number of estimators (number of boosting rounds), learning rate, step size shrinkage to prevent overfitting, maximum depth of a tree, subsample, fraction of samples used for fitting trees, colsample bytree, fraction of features used for fitting trees, L1 regularization term on weights, and L2 regularization term on weights. Tuning these parameters is essential for achieving optimal model performance. XGBoost is known for its high performance and efficiency. It can capture complex relationships, non-linear patterns, and interactions within the data. Its ability to handle imbalanced datasets, along with features like regularization and boosting, makes it a strong candidate for fraud detection where the fraudulent class is often a minority.

IV. Implementation Details

A. Prepossessing

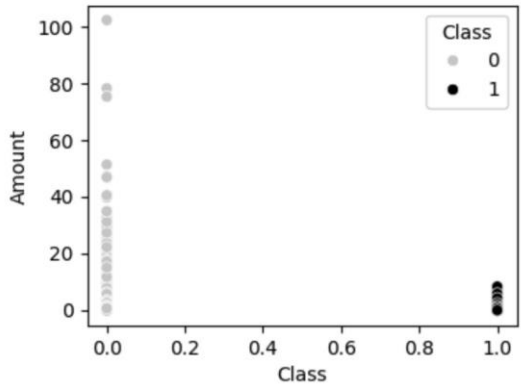

Fig. 2: Amount For Fraud Scatterplot

The dataset is result of PCA and includes a time feature and an amount feature. The 'Time' column does not provide any crucial information, it represents the time that has elapsed from the first transaction, linearly increasing. There aren't any time dependant features which need to be time scaled, therefore we drop the time column. The 'Amount' column represents the amount for each transaction, upon visualization, we realize that most of the fradulent cases are limited to under 20 euros. However, this is not guaranteed, since our dataset was collected over 3 days, the fradulent transactions were limited to under 20 euros, this might not happen in the future, therefore we must make fraud detection agnostic of amount.

Since PCA has been performed on the dataset we do not need to scale it and we can move forward to splitting the dataset for training and testing.

B. Test Train Split

Since our dataset is imbalanced, we will be employing sampling techniques while evaluating the best classifier. It is generally recommended to split your data into training and testing sets before applying any sampling techniques. This ensures that the evaluation of your model's performance is evaluated on unseen data, providing a more realistic assessment of its generalization ability. We use 'sklearn.model selection.train test split' with a test size of 0.2 to generate training and testing data sets.

We perform a two-step split of your main dataset into a minor set, and then further splitting the minor set into training and testing subsets. The train _test split function is used to split the main training dataset into a minor set. The split is performed with a test size of 20%, meaning that 20% of the data is designated for the minor set. The stratify parameter ensures that the class distribution is preserved in the split, which is important for imbalanced data sets. We further split the minor set in a training and test set. We do this perform grid search on classifiers faster.

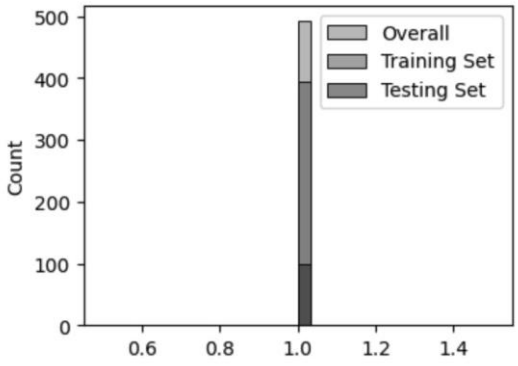

Fig. 3: Test Train Split for Fradulent cases

C. Resampling

Using resampling techniques in fraud detection is crucial because class imbalance is a common challenge in fraud datasets, where instances of fraudulent activities are typically much less frequent than legitimate transactions. Class imbalance can significantly impact the performance of machine learning models, and employing sampling techniques helps address these issues. Since our dataset is imbalanced as pointed out in table 2, we will explore sampling techniques such as a Random Under Sampling(RUS), Random Over Sampling(ROS), Synthetic Minority Over-sampling Technique(SMOTE) and Adaptive Synthetic Sampling(ADASYN).

1) Random Under Sampling (RUS)

Undersampling focuses on reducing the imbalance ratio by removing samples from the majority class. Samples may be simply randomly removed, as in random undersampling (RUS). RUS is a fast and easy way to balance a dataset and is therefore widely used. With fewer samples in the majority class, training times can be significantly reduced. This is especially beneficial when dealing with large datasets, as the model has less data to process during training. By balancing the class distribution, random undersampling can help prevent the model from being biased towards the majority class. This bias is common in imbalanced datasets, where the model might favor the class with more instances. In the case of fraud RUS will help reduce bias by training on a balanced dataset containing genuine and fraudulent transactions. However, Removing samples from the majority class may result in a loss of valuable information, and the model might not generalize well to the original distribution. With a reduced amount of data, there's a higher risk of overfitting, where the model may memorize the training data rather than learning generalizable patterns. The level of undersampling can impact results. If too many samples are removed, the model might struggle to capture the complexity of the majority class.

2) Random Over Sampling (ROS)

Oversampling consists in artificially increasing the proportion of samples from the minority class. The most naive approach is random oversampling (ROS), in which samples from the minority class are randomly duplicated. By duplicating samples from the minority class, you provide the model with more examples to learn from. This can be especially beneficial when the minority class is underrepresented, and the model might struggle to identify patterns within it. Random oversampling can help prevent bias towards the majority class. It ensures that the model is exposed to a more balanced representation of both classes during training. In situations where the minority class, such as fraud detection, contains important and informative samples, random oversampling can lead to better model performance. It allows the model to focus on learning diverse patterns from the minority class. ROS also has some drawbacks, duplicating samples from the minority class may increase the risk of overfitting, where the model memorizes the training data rather than learning generalizable patterns. There is an underlying risk introducing noise if the minority class contains noisy or irrelevant samples, duplicating them may introduce noise into the training set, impacting model performance. Sensitivity to sampling rate for ROS might also create new issues, a higher sampling rate might create bias towards the minority class, affecting the ability of the model to generalize.

3) Synthetic Minority Over Saampling Technique (SMOTE)

SMOTE is a resampling technique designed to address class imbalance in machine learning datasets, especially in binary classification problems where one class is significantly underrepresented compared to the other. SMOTE generates synthetic instances for the minority class by interpolating between existing instances. A combination of over-sampling the minority (abnormal) class and under-sampling the majority (normal) class can achieve better classifier performance (in ROC space) than only under-sampling the majority class[Chawla ref, https://arxiv.org/pdf/1106.1813.pdf]. Fraudulent transactions are typically a minority class, while genuine transactions constitute the majority. SMOTE helps balance the class distribution, ensuring that the machine learning model is exposed to sufficient instances of fraudulent transactions during training. Unlike simple over-sampling methods that duplicate existing instances, SMOTE generates synthetic instances. This helps mitigate overfitting, where the model may memorize the training data rather than learning general patterns. Fraud detection often requires a high level of sensitivity (recall) to ensure that as many fraudulent transactions as possible are correctly identified. SMOTE aids in achieving higher sensitivity by providing the model with more examples of the minority class. However, as is the case with training synthetic data, noise in the training data might be echoed, therefore, careful consideration of the impact on model performance is essential.

4) Adaptive Synthetic Sampling (ADASYN)

ADASYN is designed to adaptively generate synthetic samples for the minority class based on the local density of existing instances. This adaptability can be beneficial when the distribution of the minority class is not uniform throughout the feature space. ADASYN tends to focus more on generating synthetic samples for instances that are difficult for the model to learn. This can help improve the model's ability to capture the intricate patterns within the minority class. Compared to simple oversampling, ADASYN's adaptive approach may reduce the risk of overfitting by generating synthetic samples in regions where the minority class is underrepresented, rather than uniformly across the entire feature space. ADASYN can be computationally more expensive than simpler techniques due to its adaptive nature and the need to consider local density. As with any synthetic data generation technique, there is a risk of introducing noise if the synthetic samples do not accurately represent the underlying data distribution.

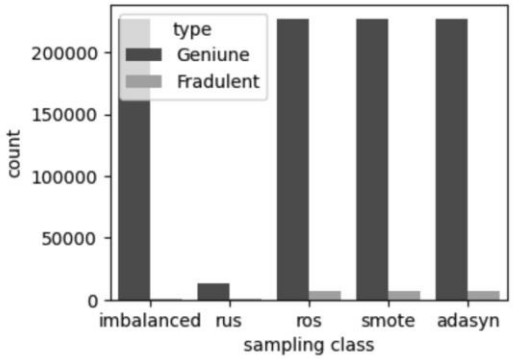

Fig. 4: Sampling Techniques

D. Machine Learning Algorithm Tuning

In this section, we will be discussing the implementation and hyper-paramter tuning of various machine learning algorithms mentioned above. We will then evaluate their performance based on Precision, Recall, Macro F1 Score and AUC.

We perform a grid search for every machine learning algorithm against every sampling technique. For every sampling technique we run a grid search and compare those results.

1) Logistic Regression

The logistic regression parameters are configured for a binary classification task. The regularization strength, controlled by the inverse of C, is set to 1.0, indicating moderate regularization. The optimization algorithm, specified by the 'liblinear' solver, is suitable for small to medium-sized datasets. The regularization type 'l2' penalizes the square of coefficients, preventing overfitting. The maximum number of iterations (max iter) is set to 100, ensuring efficient convergence. The 'auto' option for multi class enables automatic selection of the most suitable strategy. The tolerance for convergence (tol) is

set to 0.0001, defining the acceptable change for convergence during training.

Parameter	Value
C	1.0
max_iter	100
multi_class	auto
penalty	l2
solver	liblinear
tol	0.0001

TABLE III: Logistic Regression Hyperparameters

2) K-Nearest Neighbour

The k-Nearest Neighbors (KNN) algorithm is configured with parameters for effective classification by utlizing grid search on the minor dataset delinated in the section above.. The distance metric 'cosine' measures the cosine of the angle between data points, suitable for high-dimensional spaces. 'n _neighbors' is set to 16, determining the number of neighbors considered during classification. The distance weightage is governed by 'weights', favoring closer neighbors with 'distance'. Parallel processing is enabled with 'n jobs' set to 16, enhancing computational efficiency by utilizing 16 processor cores. Together, these parameters define a robust and scalable KNN classification model.

Parameter	Value
metric	cosine
n_jobs	16
n_neighbors	16
p	1
weights	distance

TABLE IV: K-Nearest Neighbors (KNN) Configuration

3) Decision Tree

The decision tree is configured with specific parameters for effective classification using grid search. The 'criterion' is set to 'entropy,' indicating the use of information gain as the impurity measure for node splitting. With a maximum depth of 5, the tree is constrained to prevent overfitting. 'min _samples leaf' is set to 4, ensuring each leaf node contains at least 4 samples to improve robustness. The 'min samples split' of 2 sets the minimum number of samples required to split an internal node. Together, these parameters guide the decision tree's structure, balancing complexity, and accuracy for optimal predictive performance.

Parameter	Value
criterion	entropy
max_depth	5
min_samples_leaf	4
min_samples_split	2

TABLE V: Decision Tree Hyperparameters

4) Random Forest Classifier

The Random Forest consists of 100 decision trees, each with a maximum depth of 5, a minimum of 4 samples per leaf, and a minimum of 2 samples to split internal nodes, all using entropy as the criterion for node splitting. This ensemble approach enhances predictive performance by aggregating the results of multiple decision trees. We reach these hyper parameters by using grid search.

Parameter	Value
criterion	entropy
max_depth	5
min_samples_leaf	4
min_samples_split	2
n_estimators	100

TABLE VI: Random Forest Hyperparameters

5) SVM

Since our dataset has several thousand rows. It is best to use Stochastic Gradient Descent to train SVM models due to a time and resource constraint. We use Scikit learns SDGClassifier with the loss mode to set 'hinge'. This gives a Stochastic Linear SVM. We tune hyper parameters by conducting a grid search consisting of a regularization penalties, L1, L2 and Elasticnet. We also utilize alpha, a constant that multiplies the regularization term, greater alpha leads to stronger regularization. Since our dataset is imbalanced, we can employ class weights to tune model. FOr different sampling techniques we have different class weights for the Stochastic SVM.

Parameter	Value
alpha	0.01
class_weight	{0: 0.025}
loss	hinge
max_iter	10000
penalty	l2

TABLE VII: Linear SVM Hyperparameters

6) Xgboost

XGBoost, an ensemble learning algorithm, utilizes various hyperparameters to optimize model performance. In the provided XGBoost configuration, the colsample bytree parameter controls the fraction of features randomly sampled for each tree. The learning _rate influences the step size during optimization. max depth sets the maximum depth of each tree, and n estimators determines the number of boosting rounds. The scale pos weight parameter addresses class imbalance by assigning weights to positive class instances. This combination of parameters, when appropriately tuned, empowers XGBoost

to build a robust ensemble of decision trees, effectively capturing complex patterns and relationships within the data for improved predictive accuracy.

Parameter	Value
colsample_bytree	texttt0.8
learning_rate	texttt0.1
max_depth	texttt16
n_estimators	texttt50
scale_pos_weight	texttt50

TABLE VIII: XGBoost Configuration 1

V. Evaluation

A fraud detection problem is in nature a cost-sensitive problem: missing a fraudulent transaction is usually considered more costly than raising a false alert on a legitimate transaction. In the former case, losses not only include the amount of the fraudulent transaction. In an imbalanced fraud dataset, where the majority of transactions are non-fraudulent, utilizing precision, recall, and F1 score becomes crucial for assessing the performance of a machine learning model, especially in fraud detection. Precision is important when we want to minimize false positives. In fraud detection, precision measures the accuracy of identifying actual fraud among the predicted fraud cases. High precision means fewer non-fraudulent transactions are mistakenly classified as fraud. Recall is crucial for ensuring that most actual fraud cases are detected. It measures the ability of the model to capture all instances of fraud. In fraud detection, high recall indicates a low rate of false negatives, meaning fewer actual fraud cases are missed. The F1 score is particularly useful when there is an imbalance between the classes. It helps find a balance between precision and recall. A higher F1 score indicates a model that performs well in both minimizing false positives and false negatives.

Precision, recall, and F1 score are essential metrics for evaluating the performance of a machine learning model in fraud detection scenarios, where class imbalance is a common challenge. Ultimately, we should use the metric which helps us resolve our final goal. If we want to utilize this dataset and the best performing classifier to accurate predict fraud, then we should consider using a precision as our primary metric. If the ultimate goal is to increase the detection of fraudulent cases, then we should use recall as our primary metric. A cost matrix can help us generalize and create a weighted F-1 score to better analyze and evaluate the best models performance.

TABLE IX: Classifier/Sampling Performance

Strategy	Model	F-1	Precision	Recall	AU-PRC	AU$_R OC$	Accuracy
ADASYN	RF	0.933559	0.933559	0.933559	0.752519	0.985241	0.999544
ADASYN	XGBOOST	0.838289	0.777663	0.933076	0.482088	0.981344	0.998578
RUS	KNN	0.834276	0.772321	0.933049	0.472821	0.971456	0.998525
Imbalanced	XGBOOST	0.927712	0.932160	0.923355	0.732512	0.983158	0.999508
SMOTE	XGBOOST	0.919051	0.914868	0.923320	0.703223	0.988241	0.999438
SMOTE	RF	0.936065	0.955415	0.918297	0.762639	0.992469	0.999579
ROS	XGBOOST	0.923956	0.935334	0.913160	0.720180	0.981320	0.999491
RUS	RF	0.888965	0.876023	0.902833	0.606845	0.938967	0.999210
Imbalanced	SVM	0.903982	0.910350	0.897810	0.653842	NaN	0.999350
Imbalanced	KNN	0.917451	0.952188	0.887685	0.702038	0.948716	0.999473

VI. Future Directions

- Unsupervised Learning - Implementing clustering algorithms and conducting a T-SNE.
- Improve the performance of existing Stochastic SVM in this study. Using a Batch SVM will definitely increase precision, accuracy and f1 scores further.
- Explore ways to create a cost matrix to further fine tune models. Whether they should lean towards better Precision or Recall.
- Explore the feasibility of real-time fraud detection systems. Investigate streaming algorithms and techniques that enable the timely identification and prevention of fraudulent transactions as they occur.

VIIC. onclusion

In conclusion, the application of machine learning in fraud detection represents a powerful and evolving paradigm that holds immense promise for enhancing security in financial transactions. Our study has explored various machine learning algorithms, including Logistic Regression, k-Nearest Neighbours, Decision Trees, Random Forests, Stochastic SVM and Xgboost, and evaluated their effectiveness in identifying fraudulent activities. Our study has coupled these algorithms with several Re-sampling techniques to handle the inherent

imbalance in fraudulent transaction data. The results indicate ADASYN with Random Forest, showcasing the potential of these models in mitigating financial risks. While our work has made strides in improving fraud detection accuracy, it is crucial to acknowledge the dynamic nature of fraud schemes and the continuous need for model refinement. Ongoing research in adaptive algorithms, feature engineering, and anomaly detection will play a pivotal role in staying ahead of emerging fraud tactics. The cost-sensitive nature of fraud detection, as evidenced by the importance of precision, recall, and F1 score, underscores the need for a balanced approach that minimizes false positives without compromising the identification of actual fraud cases. As we move forward, collaborations between data scientists, industry experts, and policymakers will be essential to navigate the ethical considerations and legal implications associated with implementing machine learning in real-world financial systems. In essence, the intersection of machine learning and fraud detection holds the promise of creating more robust and resilient systems, safeguarding financial assets and fostering trust in digital transactions.

REFERENCES

- Hands on Machine Learning with Scikit-Learn & TensorFlow by Aurelien G´ eron (O'Reilly). CopyRight´ 2017 Aurelien G´ eron´
- Machine Learning Over& Undersampling Python Scikit ScikitImblearn by CodingManiac
- auprc, 5-fold c-v, and resampling methods by Jeremy Lane (Kaggle Notebook)
- Fraud Detection Handbook. Chapter 6: Imbalanced Learning. Cost Sensitive Learning. Retrieved from [URL]