Credit Card Fraud Detection, a Big Data Approach

Thanina Ait Ferhat, Kamilia Benlamara, Fendes Iness, Benkerrou Lynda
University of Paris Cité, France

April 2024

Abstract

This study delves into advanced machine learning techniques for detecting credit card fraud within a big data framework. We employ three sophisticated classifiers: Logistic Regression, Random Forest, and XGBoost, optimized through Random Oversampling and SMOTE to address significant class imbalances in a dataset from data.world, encompassing transactions from September 2013. Our methods are rigorously evaluated using metrics like ROC-AUC, F1-score, precision, and recall, with performance comparisons conducted to ascertain the most effective strategies. The outcomes underscore the efficacy of these machine learning approaches in enhancing fraud detection capabilities, thereby contributing to more secure financial transactions.

1 Introduction

Credit card fraud detection is crucial for safeguarding financial transactions and maintaining consumer trust within the financial industry. This challenge not only involves significant financial implications with projected global losses reaching up to $43 billion by 2025 [3, 7] but also showcases a complex big data problem where vast volumes of transactions are processed in real-time [6].The necessity to adapt to continuously evolving fraud tactics and diverse transaction data types adds further complexity to developing effective fraud detection systems.

The primary challenges include the highly imbalanced nature of transaction data, the realtime processing requirements, and the complexity of integrating and processing diverse data sources. Additionally, strict regulatory compliance and the high costs associated with detection errors (false positives and false negatives) make it imperative to strike a careful balance in fraud detection strategies [8].

To address these issues, sophisticated big data technologies and machine learning approaches are employed to enhance the detection capabilities and operational efficiency of fraud detection systems [5]. These technologies are crucial for developing models that can accurately identify fraudulent activities and adapt to new fraud patterns [6].

The objective of our project aims to utilize advanced machine learning techniques to effectively identify and prevent fraudulent credit card transactions. This involves classifying transactions into fraudulent or legitimate categories, thereby protecting consumers and ensuring the integrity of financial systems.

2 Related Work

Several research studies in the domain of Credit Card Fraud Detection have already made significant progress in this field. These efforts have predominantly centered on identifying fraudulent activities within credit card transactions, employing signal processing techniques and machine learning algorithms. In this section, we will conduct a review of existing literature in this field to gain insights into the methodologies utilized and the outcomes achieved.

Supervised Class Distribution Learning for GANs-Based Imbalanced Classification: The article introduces the SCDL-GAN method, which tackles class imbalance by combining Generative Adversarial Networks (GANs) with supervised class distribution learning. This approach utilizes a supervised Wasserstein auto-encoder to accurately capture class distributions. By generating diverse, high-quality artificial instances, it enhances classification performance by ensuring the classifier recognizes discriminative characteristics among different classes.

iEDeaL: A Deep Learning Framework for Detecting Highly Imbalanced Interictal Epileptiform Discharges: The article presents iEDeaL, a deep learning framework for detecting interictal epileptiform discharges (IEDs) in EEG data. This innovative framework can handle real and highly imbalanced datasets without requiring feature extraction. By employing a novel neural network architecture and loss function, iEDeaL optimizes the detection process, thereby improving accuracy and efficiency compared to existing methods.

Credit Card Fraud Detection Based on Improved Variational Autoencoder Generative Adversarial Network: This article presents an innovative approach to credit card fraud detection by improving the generator component of the Variational Autoencoder Generative Adversarial Network (VAEGAN). This method outperforms traditional oversampling techniques by producing more diverse synthetic data, enhancing detection performance across multiple metrics. The study provides a detailed framework for comparing classification and oversampling methods, highlighting the benefits of the enhanced VAEGAN approach in fraud detection scenarios.

An Ensemble Learning Approach for Anomaly Detection in Credit Card Data: The article introduces the Credit Card Anomaly Detection (CCAD) model, which combines base learner paradigms with meta-learning ensemble techniques to enhance credit card anomaly detection. By using outlier detection algorithms as base learners and XGBoost as the meta-learner, CCAD achieves superior performance in identifying anomalies, particularly from minority class instances, addressing challenges posed by data imbalance and overlapping class samples.

Overall, each article presents a unique approach to tackle class imbalance in different domains, contributing to improved classification and anomaly detection performance. They utilize various techniques, including deep learning frameworks, GANs, and ensemble learning methods, to handle imbalanced datasets effectively and enhance model performance.

FAST: A ROC-based Feature Selection Metric for Small Samples and Imbalanced Data Classification Problems: This paper introduces the FAST method to enhance classification on datasets with small sample sizes and class imbalances, particularly in applications like credit card fraud detection. FAST evaluates features using ROC curves and AUC, outperforming traditional methods. It effectively handles imbalanced datasets and small sample sizes, as demonstrated through empirical validation.

Improving SVM Classification on Imbalanced Data Sets in Distance Spaces: This article addresses Support Vector Machine's (SVM) challenges in classifying rare events in imbalanced datasets, especially in non-Euclidean distance spaces like time series data. It introduces "ghost points" to augment minority class representation directly in distance space, improving SVM performance. The method integrates dynamic time warping (DTW) and optimal subsequence bijection (OSB) as distance measures for handling time series data. Experimental results demonstrate enhanced SVM classification accuracy, especially on the minority class.

SPO: Structure Preserving Oversampling for Imbalanced Time Series Classification: This paper presents the SPO technique for handling imbalanced time series data in classification tasks. SPO generates synthetic samples using a multivariate Gaussian distribution, preserving the covariance structure of the minority class. Experimental results demonstrate superior classification performance compared to existing oversampling methods. SPO is particularly beneficial for time series data applications, ensuring synthetic data supports the minority class while respecting the underlying data structure, thereby enhancing overall model generalization.

T-SMOTE: Temporal-oriented Synthetic Minority Oversampling Technique for Imbalanced Time Series Classification: This article introduces T-SMOTE, a technique tailored for addressing class imbalance in time-series data. Unlike traditional oversampling methods, T-SMOTE considers the inherent temporal properties of time-series data, especially for minority classes. It not only improves classification performance but also enhances early prediction capabilities in time-series applications, particularly in industries like healthcare where early predictions are crucial.

The articles offer a variety of innovative techniques to enhance credit card fraud detection, with a particular emphasis on managing class imbalances and optimizing classification models. Some articles, such as SCDL-GAN and Improved VAEGAN, focus on using GANs to generate more diverse and higher quality synthetic data, while others, like iEDeaL and FAST, explore feature selection methods and deep learning to improve detection accuracy. Additionally, approaches such as CCAD and Improving SVM Classification highlight the effectiveness of ensemble learning techniques and SVM enhancement for detecting anomalies, including frauds in credit card data. Furthermore, oversampling techniques specifically designed for temporal data, such as SPO and T-SMOTE, offer promising avenues to enhance the robustness of fraud detection models in contexts where temporal patterns play a crucial role. By integrating these approaches and techniques into your own work on credit card fraud detection, it is possible to enhance model accuracy and reliability, thereby reinforcing financial security and user trust.

3 Experimentation

3.1 The database

We used a dataset from data.world Credit Data Card Fraud Detection wich contains transactions logged in September 2013. The dataset encompasses a total of 284,807 transactions, with 492 samples flagged as fraudulent. This substantial class imbalance, where fraudulent transactions represent only 0.172% of the total, emphasizes the necessity of careful class balancing handling techniques before constructing any models. These transactions occurred over a two-day period. For confidentiality reasons, most variables are anonymized, and the features have been transformed using Principal Component Analysis in the given dataset. Additionally, it should be noted that this dataset is one of the rare ones available for fraud detection purposes. Moreover, the cardholder identifier information is not provided.

- There is only numerical input features resulting from the PCA transormations (features V1 to V28).
- Only features "Time" and "Amount" have not been transformed with PCA. The features "Time" indi-

cates the seconds elapsed between each transaction and the first one in the dataset, while "Amount represents the transaction amount.

- The "Class" feature serves as the response variable, with a value of 1 denoting fraud and 0 otherwise.

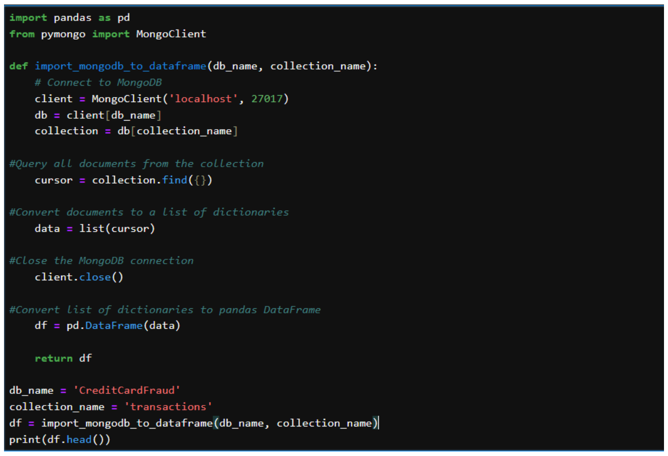
The dataset was given under one CSV file format.

3.2 The Database Management System

We opted for MongoDB as database management system. This choice is justified by multiple reasons. MongoDB is known for :

1. **Performance**: In fraud detection for bank transactions, rapid read and write operations are crucial for timely processing of data. MongoDB's performance optimization ensures that transaction data can be analyzed swiftly, allowing banks to make timely decisions to prevent fraudulent activities.
2. **Scalability**: Scalability is a crucial aspect in fraud detection for bank transactions, especially considering the ever-increasing volume of transactions and the need to process them in real-time. MongoDB's horizontal scalability feature aligns well with this requirement, as it allows for the seamless addition of more servers to accommodate growing data volumes and increasing transaction traffic. With MongoDB's scalability capabilities, banks can effectively handle the influx of transaction data, ensuring that their fraud detection systems remain responsive and efficient even as transaction volumes grow over time.
3. **High Availability**: Continuous availability of the system is essential in fraud detection to ensure that transaction monitoring is not interrupted. MongoDB's built-in support for replication and automatic failover ensures constant data availability, even in the face of hardware failures or network disruptions. This feature minimizes downtime and enables banks to continuously monitor transactions for fraudulent activities without interruption.
4. **Flexibility**: Unlike traditional SQL databases, MongoDB is a NoSQL database, meaning it doesn't require a fixed schema. This allows for the storage and management of unstructured or semi-structured data, which is common in modern applications, without the need for predefined schemas.

To import the CSV file to MongoDB, we used Compass GUI.



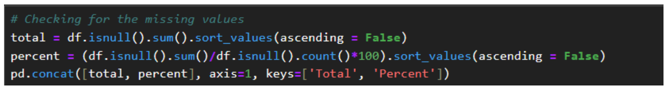
```python
import pandas as pd
from pymongo import MongoClient

def import_mongodb_to_dataframe(db_name, collection_name):
    # Connect to MongoDB
    client = MongoClient('localhost', 27017)
    db = client[db_name]
    collection = db[collection_name]

#Query all documents from the collection
    cursor = collection.find({})

#Convert documents to a list of dictionaries
    data = list(cursor)

#Close the MongoDB connection
    client.close()

#Convert list of dictionaries to pandas DataFrame
    df = pd.DataFrame(data)

    return df

db_name = 'CreditCardFraud'
collection_name = 'transactions'
df = import_mongodb_to_dataframe(db_name, collection_name)
print(df.head())
```

Figure 1: Connection to MongoDB

3.3 Methods and Preprocessing

3.3.1 Preprocessing Techniques

Checking for missing values :there are no missing values in the dataset



```python
# Checking for the missing values
total = df.isnull().sum().sort_values(ascending = False)
percent = (df.isnull().sum()/df.isnull().count()*100).sort_values(ascending = False)
pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
```

Figure 2: Checking for missing values

Data visualization: bar plot to visualize the distribution of the class in the dataset.

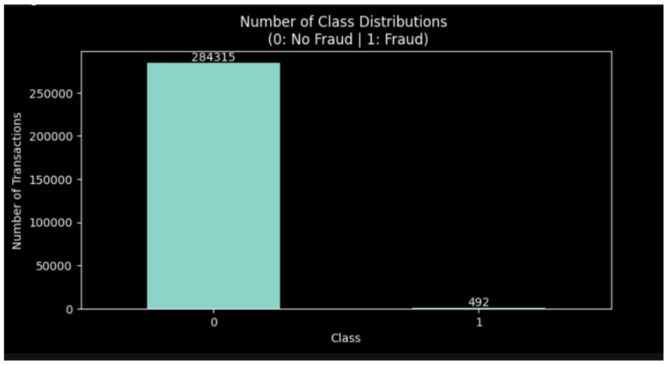
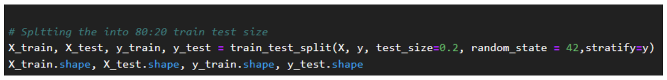


Figure 3: Data visualization

Outliers handling: as the whole dataset is transformed with PCA, we assume that the outliers are already treated.

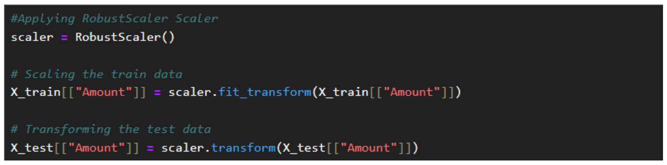
Splitting the data into train & test data : we used the train_test_split() function from scikitlearn to split the dataset X and its corresponding labels y into training and testing sets. The split is done with a 80/20 ratio for the test set and the random state is set to 42 for reproducibility. The training set is used to fit the model while the testing set is used to evaluate the model's performance on new, unseen data.



```
# Spltting the into 80:20 train test size
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state = 42,stratify=y)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Figure 4: Train-Test split

Feature Scaling :we chose to employ the RobustScaler scaler from scikit-learn as it is particularly robust to outliers, as these can significantly skew the mean and standard deviation used by other scaling techniques like StandardScaler and MinMaxScaler. Since PCA is already performed on the dataset from V1 to V28 features, we are scaling only the **Amount** field.



```
#Applying RobustScaler Scaler
scaler = RobustScaler()

# Scaling the train data
X_train[["Amount"]] = scaler.fit_transform(X_train[["Amount"]])

# Transforming the test data
X_test[["Amount"]] = scaler.transform(X_test[["Amount"]])
```

Figure 5: Feature Scaling

Checking and handling Skewness :initially, histograms were plotted for all features to visually identify any skewness. The plots revealed that many features were indeed highly skewed.

Subsequently, the skewness of each feature was quantified using the skew() method from Pandas. Given the presence of skewness, we opted to employ the PowerTransformer from scikit-learn's preprocessing suite. This tool uses the Yeo-Johnson transformation, which is effective in normalizing skewed data and accommodates both positive and negative values, thereby making distributions more Gaussian.
To ensure proper application, the PowerTransformer was first fitted on the training data. Following this, it was used to transform both the training and test datasets, aligning them more closely with the assumptions of Gaussian distribution needed for many statistical models and machine learning algorithms.

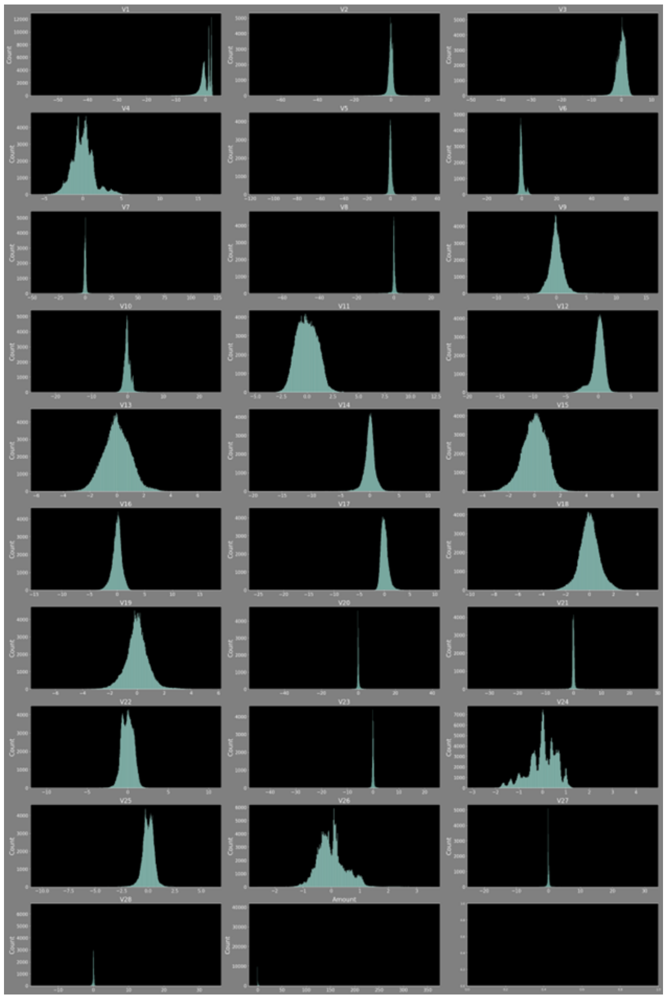
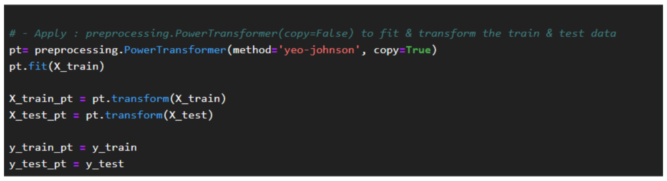


Figure 6: Histograms of the variables distributions'



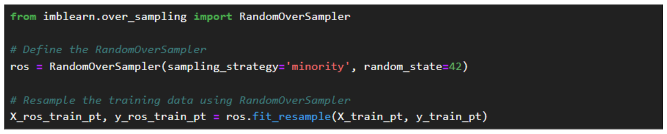
```
# - Apply : preprocessing.PowerTransformer(copy=False) to fit & transform the train & test data
pt= preprocessing.PowerTransformer(method='yeo-johnson', copy=True)
pt.fit(X_train)

X_train_pt = pt.transform(X_train)
X_test_pt = pt.transform(X_test)

y_train_pt = y_train
y_test_pt = y_test
```

Figure 7: Handling skewness

To tackle the significant class imbalance between fraudulent and legitimate transactions, we utilized two key preprocessing techniques:

1. **Random Oversampling:** this method increases the number of instances in the minority class (fraudulent transactions) by replicating them until both classes are approximately equal in size. It aims to enhance the model's learning from the minority class [4].

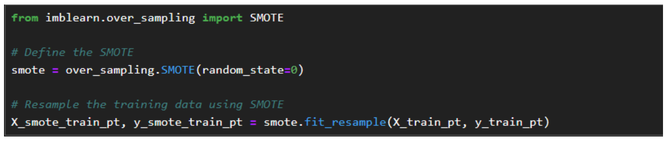
2. **Synthetic Minority Over-sampling Technique (SMOTE):** SMOTE generates synthetic samples from the minority class by interpolating between ex-



```
from imblearn.over_sampling import RandomOverSampler

# Define the RandomOverSampler
ros = RandomOverSampler(sampling_strategy='minority', random_state=42)

# Resample the training data using RandomOverSampler
X_ros_train_pt, y_ros_train_pt = ros.fit_resample(X_train_pt, y_train_pt)
```

Figure 8: Random oversampling

isting samples in the feature space. This approach helps to create a more general model and avoids the overfitting issues typically associated with Random Oversampling [1].



```
from imblearn.over_sampling import SMOTE

# Define the SMOTE
smote = over_sampling.SMOTE(random_state=0)

# Resample the training data using SMOTE
X_smote_train_pt, y_smote_train_pt = smote.fit_resample(X_train_pt, y_train_pt)
```

Figure 9: Synthetic Minority Over-sampling Technique

3.3.2 Classification Methods

In order to address our problem we have used three different classification methods.

1. Logistic Regression: Logistic regression is a widely used discriminative learning algorithm that uses a hypothesis of the form

$$h_\theta(\mathbf{x}) = \frac{1}{1+e^{-\theta^T \mathbf{x}}} \tag{1}$$

$$h_\theta(\mathbf{x}) = \frac{1}{1+e^{-\theta^T \mathbf{x}}}$$

In its simplistic form, logistic regression fits a linear decision boundary in the feature space to model a binary dependent variable. Parameters are fit by maximum likelihood, where the log likelihood is given by:

$$\ell(\theta) = \sum_{i=1}^{m} y^{(i)} \log h_\theta(\mathbf{x}^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(\mathbf{x}^{(i)})) \tag{2}$$

$$\ell(\theta) = \sum_{i=1}^{m} y^{(i)} \log h_\theta(\mathbf{x}^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(\mathbf{x}^{(i)}))$$

This approach aims to find the set of parameters θ that maximizes the likelihood of the observed data. By adjusting the parameters, logistic regression learns to predict the probability that a given input sample belongs to a particular class .

2. Random Forest Random Forest is a supervised machine learning algorithm that addresses regression and classification problems effectively. It constructs multiple decision trees during training and uses a majority voting system to improve prediction accuracy and reliability. Techniques like Bootstrap aggregation and entropy criteria are applied to enhance precision.

The following equations represent the Gini impurity calculations used to determine the quality of splits in the model:

$$IG(N_p, a) = Gini(N_p) - \sum_{i=1}^{c} \frac{|N_i|}{|N_p|} Gini(N_i) \tag{3}$$

$$Gini(N_p) = 1 - \sum_{j=1}^{m} P_j^2 \tag{4}$$

where N_p is the total data points at node N_p, $|N_i|$ indicates the data at node N_i, c is the number of unique labels, and P_j is the proportion of the jth label at node N_p [9].

The **min_samples_split** hyperparameter controls the minimum number of samples required for a node split, reducing complexity and overfitting.

The **n_estimators** hyperparameter sets the number of trees, influencing model variance and stability but increasing computational demands. It is recommended to adjust these parameters based on dataset characteristics and computational resources.

3. XGBoost XGBoost (Extreme Gradient Boosting) is a highly efficient and scalable implementation of gradient boosting, a powerful ensemble machine learning technique. It combines multiple weak decision tree models to create a strong predictive model by iteratively adding new trees to the ensemble to improve overall prediction accuracy.

The objective function that XGBoost aims to optimize can be formalized as:

$$L(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{5}$$

Where:

- $l(y_i, \hat{y}_i)$ is the loss function measuring the difference between the true label y_i and the predicted label $\hat{y}_i$.
- $\Omega(f_k)$ is the regularization term that penalizes the complexity of individual decision trees f_k.
- K is the number of decision trees in the ensemble [2].

We optimized the XGBoost model by fine-tuning parameters such as the learning rate, maximum tree depth, and subsample ratio.

The learning rate controls step size shrinkage, preventing overfitting. Maximum tree depth regulates the complexity of individual trees, while the subsample ratio determines the fraction of samples used for each tree. This optimization process enhanced the model's performance and predictive accuracy.

3.4 Methods comparison

This subsection aims to compare the three applied methods using the two oversampling types ROS and SMOTE. We'll detail the obtained performance of each model.

3.4.1 Evaluation Metrics

For models evaluation, we used the metrics cited above:

- **Precision:** is a performance metric used to assess the accuracy of classification or prediction models. It represents the ratio of correctly predicted positive cases (true positives) to the total predicted positive cases, which includes both true positives and false positives (incorrectly predicted positive cases). In essence, precision measures the proportion of predicted positive cases that are actually true positives. It is calculated by :

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{6}$$

  A high precision score indicates that the model accurately identifies positive cases, with few false positives in its predictions. Conversely, a low precision score suggests a higher rate of false positives, which can lead to inaccuracies or misleading results.
- **Recall:** a key metric in classification or prediction models, measures how comprehensively the model identifies positive cases. It's calculated by dividing the number of true positives (correctly predicted positive cases) by the sum of true positives and false negatives (missed positive cases).

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{7}$$

  Essentially, recall tells us the proportion of actual positive cases that the model correctly detects.
  A high recall value signifies that the model effectively captures positive cases, minimizing the instances it misses. Conversely, a low recall indicates a higher rate of missed positive cases, potentially leading to inaccurate or incomplete results.
- **Roc-AUC Score (Receiver Operating Characteristic - Area Under the Curve) score:** serves as a fundamental metric for assessing the effectiveness of binary classifiers. It quantifies a model's capacity to differentiate between positive and negative classes by computing the area beneath the receiver operating characteristic curve.
- F1 Score : is a commonly used metric for evaluating the performance of binary classifiers. It combines precision and recall into a single score and provides a measure of the overall accuracy of the model. It is calculated by :

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{8}$$

3.4.2 Analysis of Results

The following tables summarize the various metrics used to evaluate the best models for each method (best hyperparameters). Additionally, the results pertain to the two oversampling applied techniques.

1. **LogisticRegression :** The parameter C have a value of 0.01 and penalty value is 12.

Over-Sampling	ROC-AUC Score	F1-Score	Precision	Recall
ROS	0.971	0.932	0.925	0.938
SMOTE	0.969	0.921	0.9113	0.931

2. **RandomForestClassifier:** the values of min_samples_split and n_estimators were respectively 5 and 500.

Over-Sampling	ROC-AUC Score	F1-Score	Precision	Recall
ROS	0.963	0.825	0.805	0.846
SMOTE	0.978	0.825	0.805	0.846

3. **XGBoost Classifier :**

Over-Sampling	Parameter	ROC-AUC Score	F1-Score	Precision	Recall
ROS	learning_rate: 0.6 max_depth: 5 subsample: 0.7	0.977	0.936	0.988	0.890
SM -OTE	learning_rate: 0.8 max_depth' 5 subsample: 0.9	0.992	0.924	0.998	0.861

We chose ROC-AUC Score for this comparision because it is widely accepted metric for comparing binary classification models, including those used in fraud detection, due to its sensitivity to class imbalance, threshold agnosticism, and ability to capture the trade-off between sensitivity and specificity.
Best scores of each model are as following:

- Logistic Regression: the best performed model achieved a score of 0.971 using ROS method as oversampling type.
- Random Forest Classifier: the best performed model achieved a score of 0.978 using SMOTE method as oversampling type.
- XGBoost Classifier: the best performed model achieved a score of 0.992 using both used oversampling types.

We can clearly observe that the best model was the XGBoost Classifier.

4 Conclusion

References

[1] Nitesh V Chawla et al. "SMOTE: synthetic minority over-sampling technique". In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[2] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (2016), pp. 785–794.

[3] Syed Kamaruddin, Md Kamal Hasan, and Md Mustafizur Rahman. "Credit card fraud detection using big data analytics". In: *Proceedings of the International Conference on Computing Advancements.* 2018, pp. 1–6.

[4] Saeed Mirshekari. *Dealing with Imbalanced Data: Oversampling Techniques.* 2023. URL: `https://saeedmirshekari.com/blog/dealing-with-imbalanced-data-oversampling-techniques/`.

[5] Namrata Pandey et al. "Credit card fraud detection using big data framework". In: *International Journal of Creative Research Thoughts (IJCRT)* 6.2 (2018), pp. 523–526.

[6] Ravi Patidar and Lokesh Sharma. "Challenges and Complexities in Machine Learning based Credit Card Fraud Detection". In: *arXiv preprint arXiv:2208.10943* (2022).

[7] M Sathyapriya and V Thiagarasu. "Big data analytics techniques for credit card fraud detection: A review". In: *International Journal of Science and Research (IJSR)* 6.5 (2017), pp. 1–5.

[8] SEON. *Credit Card Fraud Detection: Best Methods in 2023.* `https://seon.io/resources/credit-card-fraud-detection/`. Accessed: 2024-04-22. 2023.

[9] Emilija Strelcenia and Simant Prakoonwit. "Improving Classification Performance in Credit Card Fraud Detection by Using New Data Augmentation". In: (2024).