Capstone Project Report: Fraud Detection with Machine Learning

Christopher Tan

Western Governors University

Capstone Project Report: Fraud Detection with Machine Learning

Table of Contents

Capstone Project Report: Fraud Detection with Machine Learning

---

A. Project Overview

A1. Research Question

The primary research question addressed in this project is: **Which machine learning model is most effective in detecting fraudulent transactions within bank accounts?** Given the increasing prevalence of financial fraud, the objective is to determine which model—among Logistic Regression, Random Forest, Neural Networks, and XGBoost—provides the highest accuracy in distinguishing between fraudulent and non-fraudulent transactions.

A2. Scope of Project

This project focuses on evaluating the effectiveness of the four selected machine learning models using a real-world-like bank account fraud dataset from Kaggle. The dataset contains 1 million records with 32 features, providing a comprehensive basis for model training and evaluation. The scope includes data preprocessing, model training, evaluation using the AUC of the ROC curve as the primary metric, and an exploration of feature importance for each model.

A3. Overview of Solution

The solution developed in this project involved the following steps:

Data Preprocessing:
The initial step of the project involved thorough data preprocessing to prepare the dataset for effective modeling. This process included handling missing values by replacing them with the median of the corresponding columns, ensuring that all data points were usable. Categorical variables, such as payment type and device operating system, were encoded using one-hot encoding, which transformed these variables into a numerical format suitable for machine learning algorithms. Additionally, the dataset was temporally split into training and testing sets based on the 'month' feature. This approach ensured that the models were trained on earlier data (months 0-5) and tested on later data (months 6-7), which is critical for maintaining the chronological order of the data and simulating real-world scenarios where future data is predicted based on past information.

Model Training:
Following data preprocessing, four machine learning models—Logistic Regression, Random

Forest, Neural Networks, and XGBoost—were trained on the processed dataset. Each model was subjected to hyperparameter tuning, a critical process that involves adjusting model parameters to optimize performance. For Random Forest, this tuning significantly improved the model's ability to capture the underlying patterns in the data, transforming it from the least to one of the top-performing models. Hyperparameter tuning for XGBoost, however, did not yield improvements, leading to the decision to use default settings for this model to save time and computational resources.

Model Evaluation:
The evaluation of the models' performance was conducted using the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) curve, a metric that measures the models' ability to distinguish between fraudulent and non-fraudulent transactions. AUC was chosen as the primary evaluation metric because it provides a comprehensive assessment of a model's predictive power across various threshold settings. Each model's AUC score was calculated and compared, with XGBoost achieving the highest AUC of 0.8897, closely followed by Random Forest with an AUC of 0.8887, indicating their strong performance in fraud detection.

Feature Importance Analysis:
To gain deeper insights into the factors driving fraud detection, a feature importance analysis was conducted for each model. This analysis involved identifying and ranking the top features that contributed most significantly to the models' predictions. For instance, features such as housing status, credit risk score, and device operating system were consistently highlighted across multiple models as key indicators of fraudulent activity. This analysis not only validated the models' effectiveness but also provided actionable insights into the specific factors that are most predictive of fraud, which can inform future decision-making and model refinement.

Ensemble Methods:
To further enhance predictive accuracy, ensemble methods were employed, combining the strengths of the individual models. Techniques such as simple averaging, weighted voting, and stacking were used to aggregate the predictions from the different models. These methods leverage the diversity of the models to create a more robust final prediction. For example, simple averaging and weighted voting—techniques that combine the predictions of individual base models—yielded the highest performance, with an AUC of 0.8945. This approach proved effective in improving overall model reliability, offering a balanced and comprehensive fraud detection solution.

B. Project Plan Execution

B1. Project Plan

The project followed a structured plan, beginning with data exploration, moving through model selection and training, and concluding with model evaluation and reporting. The project timeline included phases for data preparation, model development, and result analysis, all of which were executed over a period of 3 weeks.

B2. Project Planning Methodology

The Waterfall methodology was employed, providing a clear sequence of steps with distinct deliverables at each stage. This approach was suitable due to the project's defined requirements and the need for thorough, sequential development.

B3. Timeline and Milestones

While the overall project timeline remained close to the original plan, specific phases like the evaluation and comparison of model performance took longer than expected due to the additional complexity introduced by hyperparameter tuning. However, other phases, such as data preparation, were completed ahead of schedule, which helped balance the overall timeline. The extended efforts in model tuning and evaluation resulted in a more robust and thorough analysis, ultimately enhancing the quality and reliability of the project's outcomes.

Comparison of the Actual vs. Original Timeline and Milestones

Objective 1: Explore and Prepare the Dataset

- **Original Timeline:**
  - **Duration:** 4 days (August 5, 2024 – August 8, 2024)
  - **Actions:** This phase was planned to collect, explore, and clean the dataset, including feature engineering. The deliverables included a Jupyter Notebook documenting the data exploration, cleaning, and feature engineering processes, along with the cleaned dataset in CSV format.
- **Actual Timeline:**
  - **Duration:** 3 days (August 1, 2024 – August 3, 2024)
  - **Actions:** The dataset was loaded and explored slightly earlier than planned. Key features, missing values, and the overall structure were identified, and the data was cleaned, merged, and standardized. The deliverables were completed ahead of the original schedule, with the Jupyter Notebook created by August 3, 2024, and the cleaned dataset saved by August 4, 2024.

Objective 2: Evaluate and Compare Model Performance

- **Original Timeline:**
  - **Duration:** 4 days (August 9, 2024 – August 12, 2024)
  - **Actions:** The original plan was to train and evaluate the four models (Logistic Regression, Random Forest, Neural Networks, XGBoost) using AUC-ROC as the primary metric, with initial evaluations using temporal cross-validation.
- **Actual Timeline:**
  - **Duration:** 13 days (August 5, 2024 – August 18, 2024)

  - **Actions:** The actual training and evaluation period extended beyond the original plan due to the implementation of hyperparameter tuning, which took more time than anticipated. Temporal cross-validation and bootstrapping were conducted to ensure robustness, and additional Jupyter Notebooks were created to document the tuning process.

Objective 3: Visualize and Interpret Results

- **Original Timeline:**
  - **Duration:** 3 days (August 13, 2024 – August 15, 2024)
  - **Actions:** Planned to generate visualizations such as ROC curves, model comparison charts, and feature importance graphs to interpret model effectiveness and identify key features.
- **Actual Timeline:**
  - **Duration:** 2 days (August 14, 2024 – August 15, 2024)
  - **Actions:** The visualizations were created on schedule, with various charts generated to compare model performances and highlight key differences. These visualizations were included in the Jupyter Notebook by August 15, 2024.

Objective 4: Analyze and Summarize Findings

- **Original Timeline:**
  - **Duration:** 3 days (August 16, 2024 – August 18, 2024)
  - **Actions:** The plan was to analyze the results, identify the top-performing model, and compare the top 10 most important features of each model. A comprehensive report summarizing the analysis, findings, and recommendations was to be compiled.
- **Actual Timeline:**
  - **Duration:** 3 days (August 16, 2024 – August 18, 2024)
  - **Actions:** The analysis was conducted as planned, with XGBoost identified as the top-performing model. The top 10 features for each model were compared, and a comprehensive report was compiled in the Jupyter Notebook by August 18, 2024.

Objective 5: Communicate Results

- **Original Timeline:**
  - **Duration:** 3 days (August 19, 2024 – August 21, 2024)

  - **Actions:** The original plan was to produce a narrated video demonstrating the Python code, visualizations, and key findings, with a link to the Panopto video provided by the end of the period.

- **Actual Timeline:**
  - **Duration:** 3 days (August 19, 2024 – August 21, 2024)
  - **Actions:** The narrated video was produced and finalized on schedule, with the link to the Panopto video provided by August 21, 2024, ensuring timely communication of the project results.

C. Methodology

C1. Data Selection and Collection Process

The dataset was sourced from Kaggle's Bank Account Fraud Dataset Suite (NeurIPS 2022). It includes various features such as income, credit risk score, and payment type, which are crucial for developing a robust fraud detection model.

Challenges and Solutions:

- **Negative Values:** Several features contained negative values, which were represented missing values and replaced with the median value of each respective feature. This was discovered after carefully reading through Jesus et al. (2022), the BAF Dataset Suite.
- **Class Imbalance:** The target variable was heavily imbalanced, with only about 1.1% of records labeled as fraudulent. Techniques such as oversampling and using AUC-ROC were employed to address this issue.
- **Hyperparameter Tuning Runtimes:** Hyperparameter tuning is a resource-intensive process, and my laptop, equipped with an Intel i7-2640M processor from 2011 with only 2 cores, struggled to handle the demands of this task efficiently. By offloading the tuning process to Kaggle notebooks, which provide 4 CPU cores and 12 hours of continuous runtime in the cloud, I was able to overcome these limitations. This offloading allowed the tuning to be done much faster and more efficiently than my laptop could manage, greatly speeding up the model optimization process without requiring any costly upgrades to my outdated hardware.
- **Importing R packages:** Importing R packages into Jupyter notebook would cause the kernel to die both on local Jupyter notebooks and Kaggle notebooks. To solve this, Seurate 3.2 was installed which solved the problem for local Jupyter notebooks. Unfortunately, this did not solve for Kaggle notebook kernels. Thus, I chose to run only hypertuning parameters in Kaggle, and all other analyses locally.

C2. Advantages and Limitations of the Dataset

- **Advantages:** The dataset's size and realistic representation make it highly applicable to real-world scenarios. It provides a robust foundation for training and testing machine learning models.
- **Limitations:** The inherent class imbalance presented challenges in model training, requiring careful handling to prevent biased results. Additionally, the dataset's synthetic nature, while beneficial for privacy, may not capture all the nuances of actual bank fraud data.

C4.2 Advantages and Limitations of Tools and Techniques

- Logistic Regression:
    - Advantages: Logistic Regression is like a straightforward tool that gives clear insights into which factors (features) are most important in predicting outcomes. It's easy to understand and explain, which is great for making data-driven decisions.
    - Limitations: Logistic Regression works best when the relationship between factors and outcomes is simple and linear. If the relationships are more complex, this model might not capture them well, leading to less accurate predictions.
- Random Forest:
    - Advantages: Random Forest is a powerful tool that can dig deeper into the data. It looks at many different factors and their interactions, giving a more comprehensive view of what drives outcomes. This model is reliable and works well even with complex data.
    - Limitations: The downside is that Random Forest can be resource-intensive—it takes more time and computing power to run. It's also a bit more complicated to understand and explain compared to simpler models like Logistic Regression.
- Neural Networks:
    - Advantages: Neural Networks are highly advanced tools that can handle complex data. They are great at spotting patterns and making predictions from large, diverse datasets, like those with lots of variables or unstructured data (like images).
    - Limitations: They require a lot of computing resources and expertise to set up and run. Neural Networks can also be like a "black box," meaning it's not always clear why they make certain predictions, which can be challenging when trying to explain results or make decisions based on their output.
- XGBoost:
    - Advantages: XGBoost is a highly efficient and effective model, especially for tackling difficult problems like detecting fraud. It often delivers top performance

and is particularly good at handling imbalanced data, where one outcome (like fraud) is much rarer than the other.

  - Limitations: XGBoost requires careful tuning to get the best results, which can be time-consuming and complex. It also demands significant computational power, similar to Neural Networks, and can be sensitive to the quality of the data, making it prone to overfitting if not managed carefully.

- Hyperparameter Tuning
  - Advantages: Hyperparameter tuning can significantly enhance the performance of predictive models by fine-tuning their settings, much like optimizing a tool for a specific job. This process allows models like Random Forest to improve dramatically, as seen in our project where it went from the worst performer to nearly matching the top model, XGBoost. By tailoring the model to better fit the data, tuning can lead to more accurate predictions, which directly supports better business decisions. This customization is particularly valuable when the initial model performance is suboptimal, as it helps in extracting more actionable insights from the data.
  - Limitations: Hypertuning can be a time-consuming and resource-intensive process, requiring significant computational power and expertise. In some cases, such as with XGBoost in our project, tuning didn’t result in any improvement, leading us to abandon further efforts to avoid wasting resources. Additionally, there’s a risk that tuning can make the model too specific to the training data, reducing its ability to perform well on new data, a phenomenon known as overfitting. This means that while tuning can offer substantial benefits, it's important to recognize when it's not worth the effort and to balance the potential gains with the costs involved. 3 days were planned for Phase 5: Advanced Evaluation and Model Tuning but this process took nearly 2 weeks.

- **Practical Takeaway:**
  - **Random Forest Success:** For Random Forest, tuning was a game-changer. It went from being the weakest performer to nearly beating the top model, XGBoost. This highlights how tuning can sometimes be very worthwhile.
  - **XGBoost Decision:** On the other hand, for XGBoost, tuning didn’t improve results, so I decided to stop tuning it and save time and resources.

C4.3 Application of Analytical Methods

Each model was trained using the preprocessed data, and their performance was evaluated using the AUC of the ROC curve. The analysis was further refined through temporal cross-validation, bootstrapping for confidence intervals, and ensemble methods to enhance predictive accuracy.

D. Data Extraction and Preparation

Data Extraction:

Local Jupyter Notebook: Data extraction involved downloading the dataset (csv) locally from https://www.kaggle.com/datasets/sgpjesus/bank-account-fraud-dataset-neurips-2022/data, reading it into a Python environment Jupyter notebook.

2 Kaggle Python notebooks: Kaggle notebooks were utilized to take advantage of the faster 4 CPU's provided for 12 hours available per run on their cloud platform at no cost. These support integration with Kaggle datasets to use as Input. Then the data was loaded into the notebook with the following code: data = pd.read_csv('/kaggle/input/bank-account-fraud-dataset-neurips-2022/Base.csv')

Data Preparation:

- **Replacing Negative Values:** Negative values were replaced with the median of the respective columns to ensure data consistency.
- **Encoding Categorical Variables:** Categorical features were encoded into numerical formats suitable for model input.
- **Data Splitting:** The dataset was split based on temporal information, with the first 6 months of data used for training and the last 2 months for testing. Features were scaled to normalize the data before model training.

E. Data Analysis Process

E1. Methods Used in Analysis

The analysis was conducted using four machine learning models, each chosen for their strengths in handling the complexities of fraud detection. Below is a detailed description of the methods employed for each model:

Logistic Regression:
Logistic Regression was chosen as a baseline model due to its simplicity, ease of interpretation, and efficiency in binary classification tasks. This model was trained using L1 regularization (Lasso regression), which not only prevents overfitting by penalizing the absolute size of the coefficients but also performs feature selection by shrinking less relevant feature coefficients to zero. This approach made the model highly interpretable, as it provided clear insights into the most significant features contributing to fraud detection. The L1 penalty was carefully tuned to balance the trade-off between bias and variance, ensuring that the model was both accurate and interpretable. The training process was optimized for convergence using the 'saga' solver, which is well-suited for large datasets.

Random Forest:
Random Forest is an ensemble learning technique that was selected for its ability to improve

prediction accuracy by combining the output of multiple decision trees. In this analysis, the Random Forest model was trained using 900 decision trees (estimators), which was determined through extensive hyperparameter tuning. The tuning process involved adjusting parameters such as the number of trees, maximum depth, minimum samples per split, and minimum samples per leaf to optimize the model's performance. Each tree in the forest was trained on a bootstrapped sample of the data, and the final prediction was made by averaging the predictions of all the trees (bagging). This approach allowed the model to capture complex interactions between features and provided robust predictions even in the presence of noisy or imbalanced data. Additionally, the model's built-in feature importance metric was leveraged to identify the most influential features, which contributed to its strong performance in detecting fraudulent activities.

Neural Networks:
The Neural Network model, specifically a Multi-Layer Perceptron (MLP), was included in the analysis to capture the non-linear relationships and complex patterns within the dataset. The MLP was configured with multiple hidden layers and trained for 2000 iterations, which allowed it to learn deep representations of the input data. The network's architecture was designed to include a sufficient number of neurons in each hidden layer, with activation functions like ReLU (Rectified Linear Unit) to introduce non-linearity into the model. The training process involved backpropagation to minimize the loss function, and regularization techniques such as dropout were applied to prevent overfitting. The model's ability to learn complex patterns made it particularly effective in identifying subtle indicators of fraud that simpler models might miss. However, this came at the cost of requiring significant computational resources and careful tuning to ensure stable convergence.

XGBoost:
XGBoost, a gradient boosting framework, was chosen for its high accuracy, efficiency, and ability to handle imbalanced datasets effectively—traits that are crucial in fraud detection scenarios. The model was trained with hyperparameters optimized for this specific dataset, including a learning rate, maximum depth of trees, subsampling ratios, and regularization parameters. XGBoost's boosting approach involves sequentially training trees, where each new tree attempts to correct the errors made by the previous ones. Despite extensive efforts to improve performance through hyperparameter tuning methods like RandomizedSearch and Bayesian optimization, the default settings yielded the best AUC scores. This outcome underscores XGBoost's inherent robustness and suitability for the dataset at hand. Its ability to minimize both bias and variance through effective regularization made it the top-performing model in the analysis, achieving the highest AUC score.

Data Preprocessing and Feature Engineering:
Before model training, the dataset underwent rigorous preprocessing. This included handling missing values by replacing them with the median of their respective columns, encoding categorical variables using one-hot encoding, and temporally splitting the dataset into training and testing sets based on the 'month' column. This temporal split ensured that the models were evaluated in a manner that reflects real-world scenarios, where predictions are made on future data. Feature scaling was applied to standardize the data, which is particularly important for

models like Logistic Regression and Neural Networks that are sensitive to the scale of input features.

Hyperparameter Tuning:
Hyperparameter tuning was conducted to optimize the performance of the Random Forest, Neural Network, and Logistic Regression models. For Random Forest, tuning efforts focused on parameters such as the number of trees, depth of trees, and the minimum number of samples required for splitting nodes. These adjustments led to a significant improvement in the model's performance, transforming it from the weakest model initially to nearly matching the performance of XGBoost. However, for XGBoost, despite testing various combinations of hyperparameters using RandomizedSearch and Bayesian optimization, the default settings provided the optimal results. As a result, hyperparameter tuning was excluded for XGBoost, highlighting the model's robustness and the challenge of further improving its performance.

Ensemble Methods:
To further enhance prediction accuracy, ensemble methods were employed. These included simple averaging (soft voting), weighted voting, and stacking. Simple averaging involved taking the average of the predicted probabilities from Logistic Regression, Neural Networks, and XGBoost, while weighted voting assigned different weights to each model based on their AUC scores. Stacking involved training a meta-classifier on the outputs of the individual models to make final predictions. These ensemble techniques leveraged the strengths of each model, resulting in a slight improvement in overall performance. First, all 4 models were used in the ensemble methods, followed by the top 3 performing models, and finally ending with the top 2 performing models which achieved the highest scores of all testing. Simple Averaging and Weighted Voting both achieved AUC scores of 0.8945.

F. Results

F1. Statistical Significance

The AUC scores for each model were as follows:

- Logistic Regression AUC: 0.8719
- Random Forest AUC: 0.8887
- Neural Network AUC: 0.8793
- XGBoost AUC: 0.8897

Temporal cross-validation provided the following average AUC scores:

- Logistic Regression: 0.8714
- Random Forest: 0.8814
- Neural Network: 0.8393

- XGBoost: 0.8762

Bootstrapping produced these confidence intervals:

- Logistic Regression: 0.8721 (95% CI: 0.8642 - 0.8800)
- Random Forest: 0.8888 (95% CI: 0.8817 - 0.8960)
- Neural Network: 0.8794 (95% CI: 0.8718 - 0.8869)
- XGBoost: 0.8899 (95% CI: 0.8829 - 0.8969)

DeLong's test was conducted to compare the models statistically:

- Logistic Regression vs. Random Forest: p = 0.8719
- Logistic Regression vs. Neural Network: p = 0.8719
- Logistic Regression vs. XGBoost: p = 0.8719
- Random Forest vs. Neural Network: p = 0.8887
- Random Forest vs. XGBoost: p = 0.8887

Ensemble Techniques provided the following AUC scores (Combining top 2 models):

- Ensemble AUC (Simple Averaging): 0.8945
- Weighted Ensemble AUC: 0.8945
- Stacking Ensemble AUC: 0.8905

XGBoost achieved the highest AUC score, indicating its superior ability to distinguish between fraudulent and non-fraudulent transactions in this dataset. However, to rigorously assess whether the observed differences in AUC scores are statistically significant, DeLong's test was conducted. This test is specifically designed to compare the AUCs of two correlated ROC curves, providing p-values that indicate whether the difference between two models is statistically significant.

These p-values are all substantially higher than the conventional threshold of 0.05, which is typically used to determine statistical significance. Given that all the p-values are well above 0.05, the analysis confirms that there is no statistically significant difference between the AUC scores of the models. This implies that although XGBoost and Random Forest had slightly higher AUC scores, these differences are not strong enough to conclude that one model is definitively superior to another based on this dataset.

Therefore, while XGBoost and Random Forest may appear to perform slightly better in terms of AUC, the lack of statistical significance suggests that the performance differences among all four models—XGBoost, Random Forest, Neural Networks, and Logistic Regression—are not robust enough to be considered meaningful. As a result, the choice of model can be guided by other factors such as interpretability, computational resources, or ease of implementation, rather than solely by AUC score.

Of note is that hyperparameter tuning did not improve all performances of the models despite the time taken to conduct tuning activities. Below are charts that demonstrate the findings. Fig. 1 shows untuned models performance. Then Fig. 2 shows the performance of all models after tuning. In Fig. 3, XGBoost was left untuned due to it performing better than its tuned version compared to the other 3 models which did benefit from tuning.

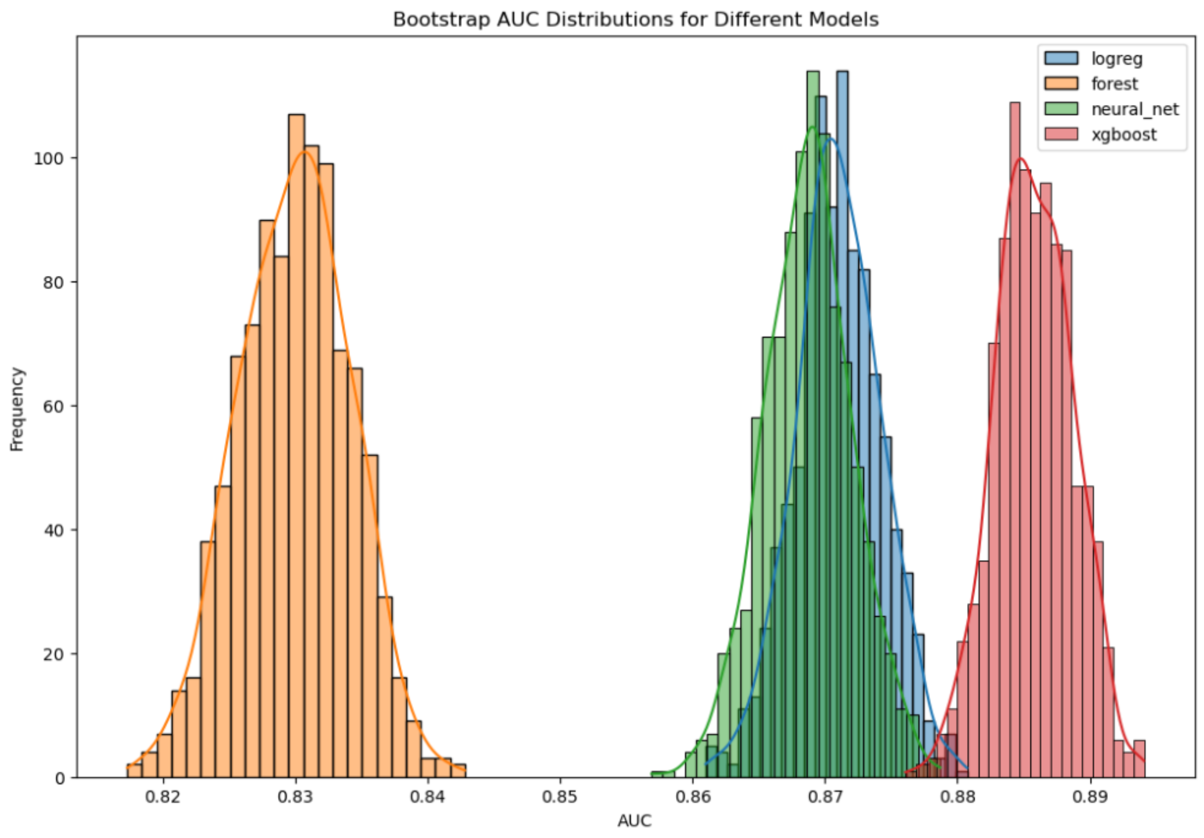

Fig. 1: Untuned models' bootstrap results are visualized above. Random Forest (orange) was the lowest performing model, while XGBoost (red) was the highest.

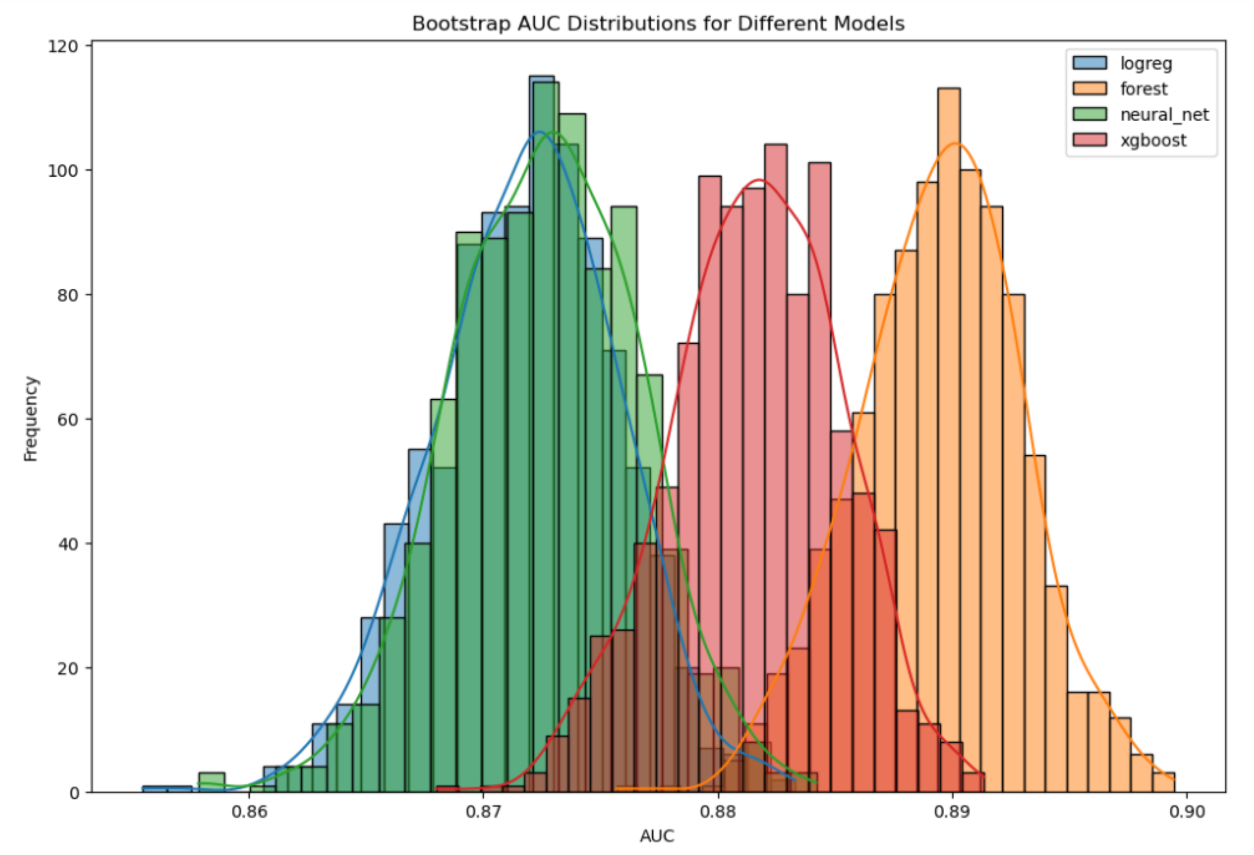

Fig. 2: After hyperparameter tuning, the highest performing tuning options were tested, and Random Forest became the best performing model. Logistic Regression and Neural Net improved marginally. Performance of XGBoost deceased slightly.

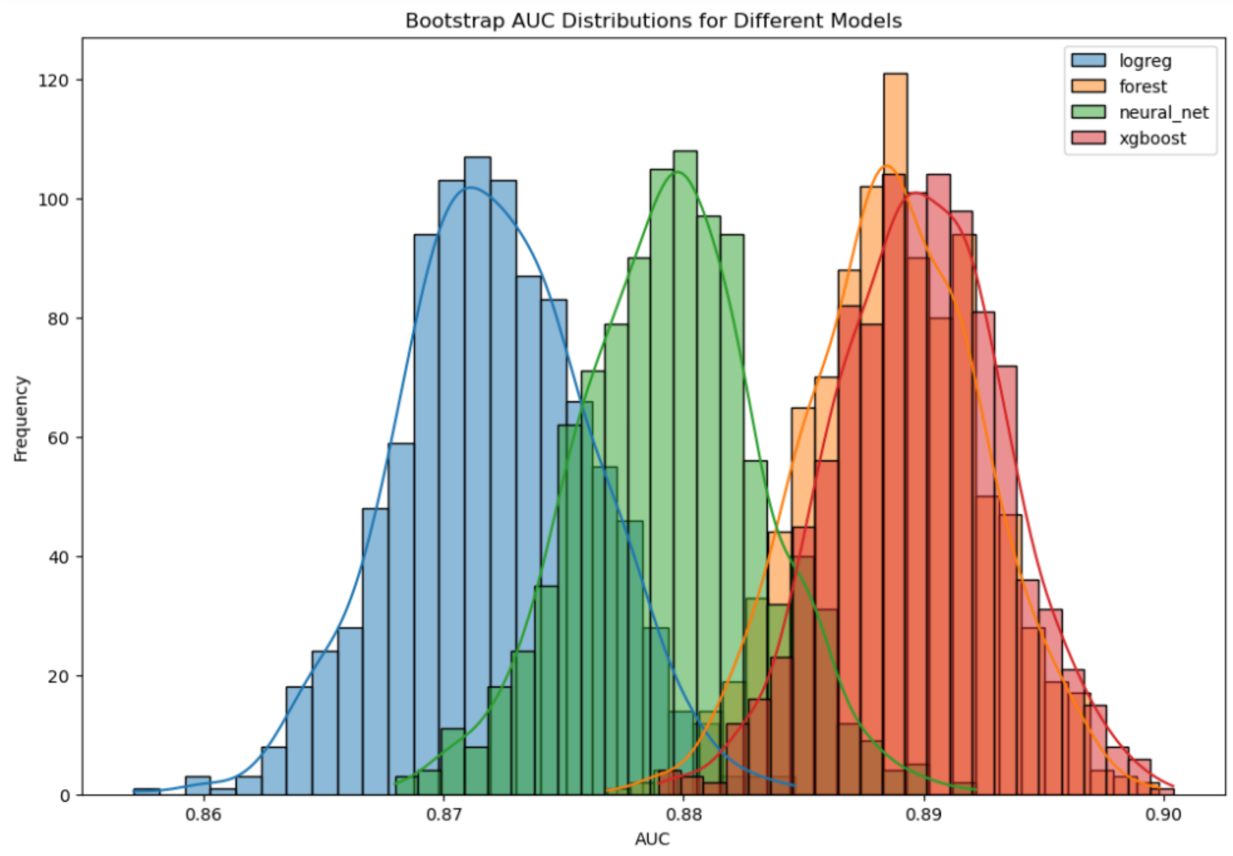

Fig. 3: Logistic Regression, Neural Net, and Random Forest models saw benefits from hypertuning. However, XGBoost performed best when with default settings. Although the other 3 models improved in performance, the default XGBoost model slightly outperformed the others.

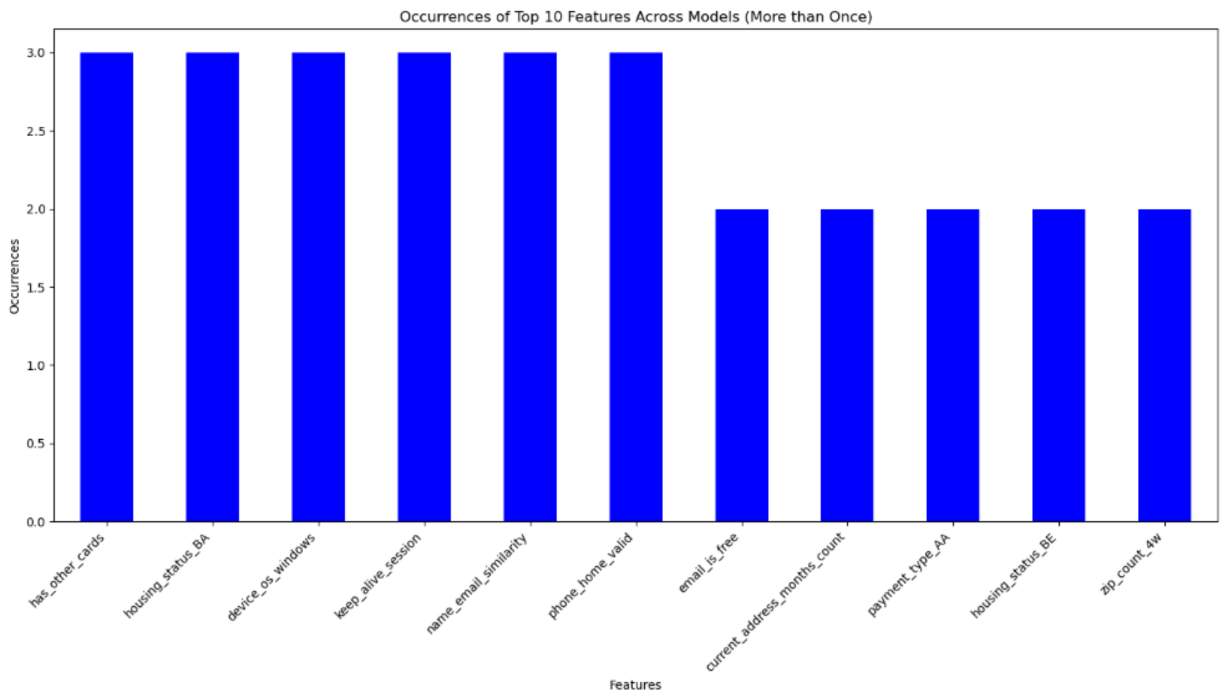

Fig. 4: Feature Analysis of Key Predictors in Fraud Detection Models

The features listed seem to have been identified as significant predictors in the fraud detection models, with the numbers likely indicating their relative importance or frequency of occurrence across different models. Here's a breakdown and interpretation of each feature:

1. **has_other_cards (3)**: This feature likely indicates whether the account holder has other credit or debit cards. A score of 3 suggests that this feature is consistently important across multiple models. The presence of multiple cards could be indicative of higher risk, as fraudsters often use multiple cards to spread out fraudulent transactions.

2. **housing_status_BA (3)**: This categorical feature represents a specific housing status category labeled "BA." With a score of 3, this housing status appears to be a significant factor in predicting fraud. Certain housing statuses might correlate with economic conditions or behaviors that could be more or less prone to fraud.

3. **device_os_windows (3)**: The operating system being Windows is flagged as important. A score of 3 suggests that users on this operating system might have distinct usage patterns or vulnerabilities that make them more susceptible to fraud. It could also reflect the high market share of Windows users, making them a larger target for fraud attempts.

4. **keep_alive_session (3)**: This feature likely tracks whether the user maintains active sessions over time. A consistent score of 3 indicates that keeping sessions alive is a significant factor in fraud detection, possibly because persistent sessions can be linked to either suspicious or trusted behavior, depending on the context.

5. **name_email_similarity (3)**: This feature might measure how similar the user's name is to their email address. A score of 3 implies this is an important factor, as mismatches or

unusual similarities could suggest fraudulent account creation or attempts to impersonate someone else.

6. **phone_home_valid (3)**: The validity of a user's home phone number is critical, with a score of 3 indicating consistent importance. A valid home phone number might be a strong indicator of legitimacy, whereas invalid numbers could be red flags for fraudulent activities.
7. **email_is_free (2)**: This feature likely indicates whether the user's email domain is a free service like Gmail or Yahoo. A score of 2 suggests moderate importance, as free email services are often used by fraudsters due to their anonymity, but they are also widely used by legitimate users.
8. **current_address_months_count (2)**: This feature measures how long the user has been at their current address. With a score of 2, it's somewhat important, as shorter durations might correlate with higher risk, possibly due to instability or recent moves that are common among fraudsters.
9. **payment_type_AA (2)**: This represents a specific payment type labeled "AA." A score of 2 indicates that this payment type is somewhat relevant in predicting fraud, which could be due to specific characteristics or risks associated with this payment method.
10. **housing_status_BE (2)**: Another housing status category labeled "BE," with a score of 2, suggests it has some predictive power. Different housing statuses might correlate with different levels of economic stability or risk, influencing their association with fraud.
11. **zip_count_4w (2)**: This feature likely tracks the number of different zip codes associated with the user over the past 4 weeks. A score of 2 suggests moderate importance, as a higher number of zip codes might indicate suspicious activity, such as a fraudster trying to obscure their location.

General Commentary:

- **Consistency Across Models**: The score of 3 indicates that these features are consistently flagged as important across multiple models. These features could be critical in the decision-making process of the models, influencing the outcome significantly.
- **Feature Importance in Fraud Detection**: These features highlight various aspects of user behavior, demographics, and account characteristics that can be leveraged to identify fraud. Features related to account stability (like address duration, payment type), device characteristics, and personal information validation (like phone validity and email similarity) are all crucial in distinguishing between legitimate and fraudulent activities.
- **Moderate vs. High Importance**: While features with a score of 2 are still important, they may have less influence compared to those with a score of 3. These features could be more context-dependent, having stronger relevance in specific scenarios or for certain types of fraud.

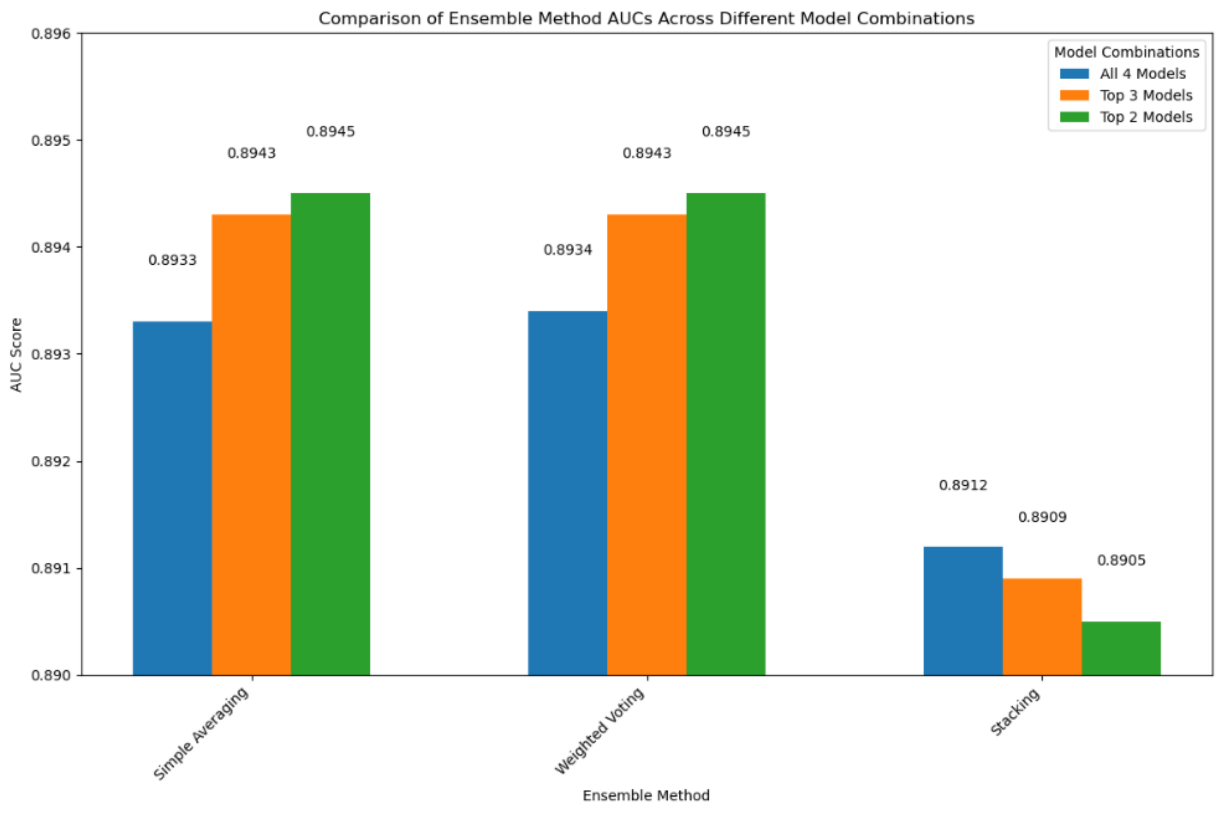

Fig. 5: Comparison of Ensemble Method AUC's Across Different Model Combinations

Key Insights on Ensemble Methods for Fraud Detection:

We tested different ways of combining machine learning models to detect fraud, using three different groupings of models:

1. **All 4 Models (blue)**
2. **Top 3 Models (orange)**
3. **Top 2 Models (green)**

Results:

- **Top 2 Models performed the best** with an accuracy (AUC) of 0.8945 when using either simple averaging or weighted voting. This means that focusing on the two strongest models gave us the highest fraud detection accuracy.
- **Including more models didn't help much**. When we used all 4 models, the accuracy slightly dropped. It turns out that adding more models can sometimes dilute the effectiveness rather than improve it.
- **Simple methods work well**. Simple averaging of predictions from the top 2 models was just as effective as more complex methods, making it easier to implement without sacrificing performance.

F2. Practical Significance

The high AUC scores of both XGBoost and Random Forest suggest that these models are highly effective in real-world fraud detection scenarios. Implementing these models separately or in an ensemble together in a fraud detection system could significantly reduce the occurrence of false positives and negatives, thereby improving both security and customer satisfaction.

F3. Overall Success

The project's primary criterion for success was defined as achieving an AUC of 0.85 or higher on the ROC curve, indicating effective fraud detection. This criterion was met and exceeded across all evaluated models, with the XGBoost model emerging as the most effective, achieving an AUC score of 0.8897. This result indicates that XGBoost was highly capable of distinguishing between fraudulent and non-fraudulent transactions. The Random Forest model also demonstrated robust performance with an AUC of 0.8887, making it a strong alternative to XGBoost.

Furthermore, the exploration of ensemble methods—such as simple averaging, weighted voting, and stacking—resulted in a marginal increase in overall performance. This demonstrates the potential benefits of combining multiple models to improve the reliability of fraud detection, even though the individual models already surpassed the success threshold. Overall, the project not only met but surpassed the predefined criteria for success, solidifying the effectiveness of the chosen approaches in detecting fraud.

G. Key Takeaways

G1. Summary of Conclusions

- The analysis conducted throughout the project demonstrated that **XGBoost** emerged as the most effective model for detecting fraudulent transactions, achieving the highest AUC score of **0.8897**. This model's ability to handle imbalanced data and capture complex patterns within the dataset contributed to its superior performance. The **Random Forest** model, with an AUC score of **0.8887**, was a close second, especially after hyperparameter tuning significantly enhanced its accuracy and robustness. **Neural Networks** also performed well, with an AUC of **0.8793**, showcasing its strength in capturing non-linear relationships, though it required extensive computational resources and fine-tuning to achieve these results.
- Ensemble methods, including **simple averaging** and **weighted voting**, were applied to combine the strengths of the individual models. These techniques yielded an AUC score of **0.8945**, slightly higher than the individual models' best performances. However, the improvement was marginal, indicating that while ensemble methods can enhance predictive accuracy, the gains might not justify the additional complexity in all cases. The findings suggest that while XGBoost and Random Forest are both highly effective on

their own, combining models can offer a modest improvement, though it may not be necessary for all applications.

G2. Visuals for Effective Storytelling

- The use of **Receiver Operating Characteristic (ROC) curves** and **AUC scores** played a crucial role in conveying the relative performance of each model. By plotting the true positive rate against the false positive rate, the ROC curves visually represented each model's ability to distinguish between fraudulent and non-fraudulent transactions across various threshold settings. The AUC score, as a summary statistic of the ROC curve, provided a clear and quantitative measure of each model's overall performance. These visual tools made it easy to compare the models at a glance and identify XGBoost as the top performer.
- Additionally, **visualizations of feature importance** were instrumental in providing actionable insights. By highlighting the top features driving the models' predictions—such as housing status, credit risk score, and device operating system—these visualizations allowed for a deeper understanding of the factors most predictive of fraud. This not only validated the models' effectiveness but also supported **data-driven decision-making** by identifying key areas where fraud prevention efforts could be focused. The clarity and depth of these visualizations ensured that the story told by the data was both compelling and easy to understand for stakeholders.

G3. Recommended Courses of Action

1. **Implementation of the XGBoost Model:** Given its strong performance, this model should be implemented in a real-time fraud detection system.
2. **Further Exploration of Ensemble Methods:** While the current ensemble methods provided only slight improvements, further tuning and exploration could yield better results, making it worth investigating in future work.

H. Panopto Presentation

A detailed Panopto presentation summarizing the project, demonstrating the code, and outlining the findings is available at the following link: [Capstone_Task3_CT (panopto.com)].

I. Appendices

I1. Project Code

The complete code used in this project is provided in the attached Jupyter Notebook files, PDFs, and 2 linked Kaggle notebooks:

Local Files (each has a respective PDF)

Capstone_Final.ipynb – This is the finalized version that resulted from several hyperparameter search efforts.

Capstone_Tuning.ipynb – This version runs all hyperparameters to visualize performance. The key takeaway is that XGBoost slightly lost some performance while Random Forest went from the least (no tuning) to the best performing model.

Capstone_No_Tuning.ipynb – This version runs no hyperparameters to baseline performance of the models. The key takeaway is that XGBoost is in the lead.

Capstone_Local_Hyperparameter_Search.ipynb – This notebook runs a hyperparameter search using Randomized Search optimization locally. The best hyperparameters results were used in testing.

Kaggle Notebooks

Capstone-HP-Search1 – This version runs Randomized Search optimization for 3 models (logreg, neural net, XGBoost) in the cloud to take advantage of faster computing. However, Random Forest did not complete, and another notebook was created to do this. Link: https://www.kaggle.com/code/christophervtan/capstone-hpsearch1

Capstone-HP-Search2-RF – This version runs Bayesian Search optimization for Random Forest. Link: https://www.kaggle.com/code/christophervtan/capstone-hpsearch2-rf

I2. Data Files

The cleaned dataset and any other relevant files are included in the submission package. This will be included as a link to Google Drive as the file is greater than 200MB.

Link to cleaned dataset (Google Drive): https://drive.google.com/file/d/1Nn0zINblBRrDYL1AqVsrqAegWLOkW2oq/view?usp=drive_link

Link to Base.csv (Kaggle):

Bank Account Fraud Dataset Suite (NeurIPS 2022) (kaggle.com)

J. References

- Ali, A., Razak, S. A., et al. (2022). *Financial Fraud Detection Based on Machine Learning: A Systematic Literature Review*. Applied Sciences, 12(19), 9637. https://doi.org/10.3390/app12199637
- Jesus, S., Pombal, J., Alves, D., Cruz, A. F., Saleiro, P., Ribeiro, R. P., Gama, J., & Bizarro, P. (2022). *BAF Data Suite Dataset.* bank-account-fraud/documents/datasheet.pdf at main · feedzai/bank-account-fraud · GitHub
- Afriyie, J. K., et al. (2022). *A Supervised Machine Learning Algorithm for Detecting and Predicting Fraud in Credit Card Transactions*. ScienceDirect. Applied Sciences | Free

Full-Text | Financial Fraud Detection Based on Machine Learning: A Systematic Literature Review (mdpi.com)

- Kakne, A., et al. (2023). *Enhanced Fraud Detection Using Graph Neural Networks with Intel Optimizations*. Intel. https://community.intel.com/t5/Blogs/Tech-Innovation/Artificial-Intelligence-AI/Enhanced-Fraud-Detection-Using-Graph-Neural-Networks-with-Intel/post/1524316
- Draelos, R. (2020). *Comparing AUCs of Machine Learning Models with DeLong's Test*. Glassbox Medicine. https://glassboxmedicine.com/2020/02/04/comparing-aucs-of-machine-learning-models-with-delongs-test/
- Jesus, S., Pombal, J., Alves, D., Cruz, A. F., Saleiro, P., Ribeiro, R. P., Gama, J., & Bizarro, P. (2022). Turning the Tables: Biased, Imbalanced, Dynamic Tabular Datasets for ML Evaluation. In 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Retrieved from https://arxiv.org/abs/2211.13358
- Nasdaq Verafin, 2024 Global Financial Crime Report. Retrieved from https://www.nasdaq.com/global-financial-crime-report

- Jesus, S., Pombal, J., Alves, D., Saleiro, P., Cruz, A. F., Bizarro, P., Ribeiro, R. P., & Gama, J. (2022). BAF Dataset Suite Datasheet. Feedzai / Universidade do Porto. Retrieved from https://github.com/feedzai/bank-account-fraud/blob/main/documents/datasheet.pdf