AY2024/2025 Semester 2

DBA3803 Predictive Analytics in Business

Fraudulent Bank Account Prediction

Class Section: SA1

Group: 6

Name	Matriculation Number
Zhou Yujie	A0234133U
Nie Xinyi	A0288395H
Chou Tzu Hsuan	A0240437L
Khoo Jordon	A0252285A
Tran Lam Vy	A0245204R

Table of Contents

1. Abstract

This project aims to develop a prediction model to help financial institutions detect bank account fraud. Regression and tree-based models were used on a high-quality synthetic dataset from Kaggle. Through testing the various models, we have concluded that the XGBoost model worked the best, achieving a recall rate for fraud of 0.99 and an ROC-AUC score of 0.86. The model identified "device_os_windows" as the crucial indicator of fraud, with a likelihood of fraud occurring greatly increasing when this feature is present. The XGBoost model, with its strong predictive performance, will allow financial institutions to detect fraud with a higher success rate and prevent the opening of fraudulent bank accounts, hence reducing the amount of illegal activities like money laundering taking place within the financial ecosystem.

2. Introduction

2.1 Background and Business Problem

With the rapid growth of digital transactions in the modern age, the opening of fraudulent bank accounts has become a huge problem for financial institutions and businesses. This problem has only worsened in recent times with the rise of artificial intelligence (AI), where fraudsters are now able to use stolen or synthetic identities to establish fraudulent bank accounts. They also make use of sophisticated tactics such as the use of mobile emulators to mimic legitimate devices, allowing them to create even more fraudulent accounts with the use of just one device (Ben-dov, 2023). The dangers of fraudulent bank account openings mainly come from the use of these bank accounts to facilitate money laundering. Money laundering is a major issue in the world, as they are not only a criminal activity, but it is also used to finance other criminal activities taking place globally. It is estimated that between 2 to 5% of the global gross domestic product (GDP) is being laundered each year, which would be equivalent to SGD 1.07 to 2.87 trillion (Europol, 2022) In 2023, Singapore had also just uncovered its largest money laundering case to date involving assets worth around SGD 3 billion, and the police had to incur over SGD 600,000 in costs for the operation in 2023 (Ministry of Home Affairs, 2025).

2.2 Problem Statement

To help financial institutions proactively identify fraudulent accounts, this project intends to develop a prediction model utilising machine learning. A number of prediction techniques will be assessed in order to discover the best model for precisely recognising fraudulent applications and pinpointing the main causes of fraudulent activity.

2.3 Data Used

The dataset utilised in this project is a synthetic dataset obtained from Kaggle, titled "Bank Account Fraud Dataset Suite", comprising detailed information on various features of bank accounts and their application information. Due to the confidentiality of banking transactions and account application data, real-world datasets tend to be heavily redacted to protect sensitive information. In most cases, the feature details and background information are removed, making it difficult for us to understand the relationships between variables and conduct meaningful analysis. As such, we decided to use a synthetic dataset instead. This dataset closely mirrors real-world scenarios and provides realistic features that are relevant to detecting fraudulent bank accounts, allowing us to build and evaluate predictive models more effectively (Kaggle, 2024).

2.4 Methods Used

In our project, we used a range of predictive modelling techniques, including Logistic Regression, Decision Trees, Random Forests, Gradient Boosting, and XGBoost. To address the class imbalance in the dataset, undersampling was used. We also explored the introduction of class weights to further penalise the wrong classification of fraudulent cases. The models were all evaluated with the same set of performance metrics so that we could make fair comparisons between them, and the best model was then chosen primarily based on recall rate and ROC-AUC, prioritising low false negatives and strong overall performance.

3. Dataset Overview

3.1 Dataset Structure

The dataset contains 1,000,000 observations and 32 features, capturing customer details, application information, and bank account-related information. Our target variable for the dataset will be the "fraud_bool" feature, which is a binary variable where 1 represents a fraudulent bank account and 0 represents a non-fraudulent bank account.

3.2 Data Cleaning

3.2.1 Handling missing values and duplicates

A bulk of our data cleaning consisted of handling missing values, with most of the analysis being included in our data preparation notebook. We first conducted an initial assessment to check for any missing or duplicated values in the dataset, and found that there were none. However, through understanding the features and the feature description, we realised that there were some features where a negative value in the dataset represents a missing value. As such, we then proceeded to search for negative values in features that should not have negative values, and found 7 features with negative values. We then proceeded to handle missing values for each feature in a way that is appropriate for each of them:

intended_balcon_amount: This feature represents the initial transferred amount into a bank account during an application. There were 742,523 missing values for this feature, making it the highest among all the features with missing values. Since this would make up almost three-quarters of our dataset, it would not be ideal for us to drop all rows with missing values, as it would result in a huge loss of data. Instead, we decided to replace the missing values with the median amount instead.

bank_months_count: This feature represents the number of months the applicant had their previous bank account for. As such, we deduced that applicants who did not have a previous bank account with the bank would have missing values in this feature. This deduction was further supported when we checked the distribution of values for this feature, and there are no rows that indicate a value of 0. As such, we stuck with our assumption and converted the negative values to 0.

prev_address_months_count: This feature represents the number of months in the previous registered address of the applicant. Based on the same logic as "bank_months_count", we checked the distribution of values for this feature and also found that there were no rows that indicate a value of 0. As such, we converted the negative values to 0.

current_address_months_count, session_length_in_minutes, device_distinct_emails_missing, velocity_6h_missing: For this group of features, we compared the missing rows against the rest of the rows to see if there was a significantly higher proportion of fraud counts in the missing rows. This was to check if the missing rows are meaningful for our analysis. After comparing the missing rows and the rest, we concluded that the missing rows were not meaningful in affecting the fraud count rate. Furthermore, the combined number of rows that were missing made up only 0.67% of our total dataset. As such, we felt that it was appropriate for us to drop the rows with missing values for these four features.

3.2.2 Drop any unnecessary columns

When analysing the features, we realised that the distribution of values for the count of device fraud ("device_fraud_count") was 100% of 0. As such, we decided to drop this entire column as it did not make any meaningful contribution to our analysis.

3.3 Feature Engineering

Next, we conducted feature engineering by creating new features that we believed might help to identify underlying patterns in the dataset better. These are the new features that we created using some of the available features in the dataset:

1. Income-to-Age Ratio: Created by dividing the income for each applicant by the customer's age. We felt that this would be a meaningful feature to have, as it can assess the earnings potential of an individual relative to their age to determine their financial stability.
2. Income to Credit Risk Ratio: Created by dividing the income for each applicant by their credit risk. This would help the banks to measure income sufficiency against credit risk, evaluating if the applicant's income would be sufficient to cover their potential financial risks.
3. Credit Risk to Age Ratio: The credit risk to age ratio was created to identify credit risk trends across different age groups, potentially highlighting potential financial behaviour patterns with age.
4. Income Group: Lastly, the income group column was created by grouping applicants' income into three distinct categories, "low", "medium", and "high". This was done by dividing the income data into three bins according to its distribution. We created this column to analyse applicants through broader income groups, since it might allow us to better identify patterns and trends that may not be as obvious when considering individual income values separately.

3.4 Final Dataset: final_dataset.csv

The final dataset consists of 995,343 records and 38 attributes (Appendix Fig. 1), and is now fully prepared for predictive analysis.

3.5 Exploratory Data Analysis

To get a better understanding of the dataset, we conducted exploratory data analysis, with key visualisations included in the Appendix.

As expected, we found that the target variable was highly imbalanced, with fraudulent transactions accounting for less than 1% of all records (Appendix Fig. 2). Income data showed a right-skewed distribution (Appendix Fig. 3), and when binned into categories, most customers fell into the low to medium income brackets (Appendix Fig. 4).

A boxplot comparing credit risk scores across income groups revealed that individuals in lower-income tiers generally had higher credit risk (Appendix Fig. 5). Lastly, a correlation heatmap highlighted moderate relationships between fraud and several numerical features (Appendix Fig. 6).

These insights helped guide our feature selection for the modelling phase.

4. Methods

4.1 Standardised Modelling Procedures and Evaluation Methods

To ensure methodological consistency across all models, a set of standardised procedures was implemented during our project model development process.

4.1.1 One-hot encoding categorical variables

We performed one-hot encoding on the categorical variables in the dataset to convert them into numerical variables for modelling.

4.1.2 Train-Test Splitting and Feature Scaling

To build and evaluate the predictive models in a manner that simulates real-world deployments, we divided the entire dataset into a training subset and a testing subset using a 70:30 split ratio. Scikit-learn's train_test_split() function is used, applying the stratify parameter to the target variable fraud_bool.

To prevent data leakage, all preprocessing steps are performed after splitting sets, which include:

1. Feature Scaling: This step is crucial for logistic regression, which assumes comparable scaling of input features. Though scaling is not absolutely necessary for tree-based models due to their nonlinear nature, a consistent method is still applied for comparison purposes.
2. Pipeline Separation: The scaler is applied only to the training set, ensuring that information from the test set does not leak into the model training process.

4.1.3 Handling Class Imbalance with Undersampling

One of the key challenges in our fraud detection process is the severe imbalance in class distribution, where legitimate transactions greatly outnumber fraudulent ones. This imbalance can lead models to be biased toward the majority class, thereby failing to detect actual fraud; therefore, to mitigate this issue, we implemented undersampling as a data-level solution to address class imbalance.

In our analysis, undersampling was applied strictly to the training subset after the train-test split to ensure no information from the test set leaks into the training process. Due to the large size of our initial dataset (995,343 rows), we opted to randomly sample a subset of the majority class (non-fraud) for training. Undersampling was applied only to the training set after the train-test split to prevent data leakage. We used RandomUnderSampler to create a balanced training subset with a 1:1 ratio between fraud and non-fraud cases, resulting in 15,408 rows (7,704 per class).

4.1.4 Cross-Validation and Hyperparameter Tuning

To estimate model performance and prevent overfitting, our models were evaluated using K-Fold Cross-Validation with stratification enabled, ensuring that each fold of the training data maintained the same class distribution as the original dataset. To maintain the consistency of our model analysis, we adopted a 5-fold stratified cross-validation as a standard across models. In this section, we also tuned the models with GridSearchCV and RandomizedSearchCV, a cross-validation setup that helps to reliably evaluate hyperparameter combinations without biasing them towards the majority of categories.

4.1.5 Evaluation Metrics

Relying on accuracy alone can be misleading; therefore, we focused on the following evaluation metrics to increase credibility for our model:

Recall	Measures how many actual fraud cases were correctly identified.
ROC-AUC Score	Provides a threshold-independent measure of model discrimination capability that helps compare models across different probability thresholds.

Recall was prioritised amongst other metrics as the costs of predicting a fraudulent account as non-fraudulent is much greater than wrongly flagging a normal account as fraudulent. Meanwhile, F1-score and the confusion matrix are also used as secondary metrics throughout model development.

4.2 Logistic Regression

Logistic regression was chosen as a baseline model for predicting fraudulent transactions (fraud_bool). It is a commonly used method for binary classification due to its interpretability and efficiency. However, it is sensitive to issues like class imbalance and unscaled features, so we incorporated preprocessing and resampling strategies to address those challenges.

4.2.1 Data Preprocessing

The dataset was highly imbalanced, with fraud cases making up less than 1% of all transactions. To address this, we first cleaned the data by removing rows with missing or infinite values. Categorical variables were one-hot encoded using drop_first=True to reduce multicollinearity.

To handle class imbalance, we followed the approach outlined in Section 4.1.3. Specifically, we applied random undersampling to the training set to achieve a balanced 1:1 ratio between fraud and non-fraud cases. Features were then scaled using StandardScaler to ensure numerical stability during model training.

4.2.2 Baseline Model Performance

Our initial model used the liblinear solver, which supports both L1 and L2 penalties and performs well on small to medium datasets. To account for class imbalance, we set class_weight='balanced' and used compute_sample_weight to give more weight to minority-class samples during training. This setup provided a fair and interpretable benchmark for evaluating the impact of later tuning and regularisation.

4.2.3 Regularisation Techniques and Hyperparameter Tuning

To improve generalisation and prevent overfitting, we applied L1 (Lasso) and L2 (Ridge) regularisation. L1 helps by zeroing out unimportant features, effectively performing feature selection, while L2 shrinks coefficients to stabilise the model.

We tuned both models using GridSearchCV with 5-fold cross-validation and tested multiple values for the regularisation strength C (0.01, 0.05, 0.1, 1, 10) using recall as the scoring metric. This allowed us to identify the best configuration for each regularisation type and strike a balance between model complexity and practical effectiveness in catching fraudulent transactions.

4.3 Decision Tree, Random Forests, Gradient Boosting & XG Boost

4.3.1 Data Preprocessing

To handle the class imbalance, we used random undersampling on the training sets for the majority class (non-fraud) to have the same number as the minority class (fraud) as we are prioritising being

able to detect future fraud cases. Class weights were also applied to the various models to penalise wrongly classifying fraudulent accounts more. For the Decision Tree and Random Forest classifiers, class_weight_parameter was tweaked to address class imbalance, while sample weights were used for Gradient Boosting and scale_pos_weight for XGBoost.

For hyperparameter tuning, we used StratifiedKFold for all models, ensuring that each fold maintained the proportion of fraud and non-fraud cases. This was crucial to preserving the class distribution in each fold and ensuring the model's generalisability. We also employed RandomizedSearchCV for all methods to efficiently search the hyperparameter space, reducing training time while maintaining model performance, which is particularly essential when it comes to our huge dataset. This approach is more efficient than GridSearchCV, while still ensuring that the best-performing configurations are selected. A table of optimal parameters for each model can be found in the appendix.

4.3.2 Decision Trees

We considered the use of decision tree models due to their ease of interpretability, even without domain knowledge. Interpretability is important in our context as it allows us to identify which variables are most significant in predicting fraudulent bank account applications. By understanding these key factors, we can make informed recommendations to financial institutions on how to better detect and prevent fraudulent accounts, thereby improving their fraud detection strategies.

4.3.3 Random Forest

Random Forest is an ensemble learning method that combines multiple decision trees to improve model accuracy. Its unique feature also comes in the form of randomly sampling features for each sample, hence decorrelating the trees and preventing over-reliance on any one feature. Due to this, Random Forest can capture complex, non-linear relationships between features more effectively than a single decision tree. This could potentially lead to improved results, especially in tasks such as fraud detection, where complex interactions between features are critical. However, the trade-off for this increased accuracy is reduced interpretability compared to simpler models like Decision Trees.

4.3.4 Gradient Boosting

Gradient Boosting is a powerful model known for its ability to iteratively improve performance by fitting new predictors to the errors made by previous predictors. This process allows Gradient Boosting to capture complex relationships between features, enhancing the model's ability to detect subtle patterns in the data. The adaptive learning rate further contributes to its effectiveness in refining the model over time. Given its capacity to model complex interactions, Gradient Boosting is a strong choice for tasks like fraud detection, where identifying intricate relationships between variables is crucial for accurate predictions.

4.3.5 XGBoost

XGBoost, an enhanced and more regularised version of Gradient Boosting, offers faster training times and improved accuracy, making it an excellent choice for scaling with larger datasets. By incorporating both L1 (Lasso) and L2 (Ridge) regularisation, XGBoost effectively reduces overfitting, which may be a concern in our context as we have 54 feature variables after one-hot encoding, leading to more robust models. (Khandelwal, 2020)

5. Results

5.1 Results of Logistic Regression

5.1.1 Hyperparameter Tuning Analysis

Tuning the hyperparameters and applying L1 regularisation contributed meaningfully to refining our logistic regression model, even though the performance gains were not statistically huge. Using grid search with 5-fold cross-validation, we found that setting the regularisation strength C=1 and max_iteration = 1000 gave the best results for both L1 and L2 models, suggesting that a moderate level of penalty struck the right balance between underfitting and overfitting.

For the L1 model in particular, this tuning was especially valuable. It not only improved the model's performance slightly but also acted as a form of feature selection by pushing less important coefficients to zero. This helped simplify the model, reduce noise, and make the results easier to interpret.

While the improvements over the baseline model were incremental, they were still meaningful, especially given the dataset's high dimensionality and class imbalance. In fraud detection, even small boosts in sensitivity can have a big impact, so this step added real value.

5.1.2 Threshold Analysis

While our models were initially trained and evaluated using the standard classification threshold of 0.5, we went a step further to explore how changing this threshold would affect recall. Since logistic regression produces probability scores, adjusting the threshold changes how aggressively the model flags transactions as fraud.

We tested a range of thresholds from 0.0 to 1.0 and plotted the corresponding recall values. As expected, recall dropped steadily as the threshold increased. Meanwhile, the ROC-AUC stayed constant, which confirmed that the model's ability to rank predictions was not affected by the threshold, and that was only its decision boundary.

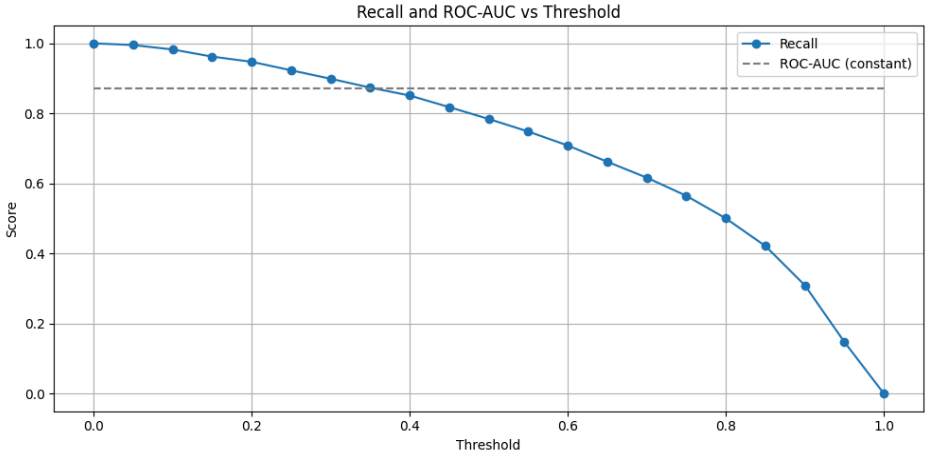

Fig.1: Recall and ROC-AUC vs. Classification Threshold (Line Plot)

From the plot, a threshold around 0.3 stood out as a sweet spot. It preserved a high recall around 0.9 without triggering the sharp performance decline we saw beyond 0.4. This analysis showed that even

after training, we can fine-tune the model's behaviour to better align with business goals, especially when deciding how much risk we're willing to take on false positives.

5.1.3 Evaluation of Logistic Regression Model Results

All three logistic regression models (base, L1, and L2) performed similarly across most evaluation metrics. However, the L1 model consistently showed a slight edge.

Evaluation Metrics Across Different Logistic Regression Models				
	Recall (y = 1)	Accuracy	ROC-AUC	Precision (y = 1)
Baseline	0.7841	0.8026	0.8719	0.04
L1	0.7853	0.8042	0.8722	0.04
L2	0.7841	0.8026	0.8719	0.04

Fig.2: Evaluation Metrics Comparison Across Logistic Regression Variants (Table)

While these differences may seem marginal, they carry weight in a fraud detection setting, where even minor gains in recall can translate into catching more fraudulent activity and therefore reducing financial losses. Precision remained low at 0.04 for all models, which is expected given the class imbalance and the emphasis on recall.

5.1.4 Best Logistic Regression Model Selection

Based on the evaluation results, the L1-regularised logistic regression model is selected as the strongest performer. It recorded the highest recall of 0.7853, accuracy of 0.8042, and ROC-AUC of 0.8722 among the three models tested. While the improvements were modest, they were consistent across all key metrics, indicating that L1 regularisation enhanced the model's ability to detect fraud while also reducing noise by filtering out less relevant features.

Its combination of strong performance, built-in feature selection, and interpretability made the L1 model not only effective but also practical. Altogether, it offered the most balanced and deployment-ready solution for fraud detection in this context.

5.2 Results of the Decision Tree, Random Forest, Gradient Boosting & XGBoost

After deciding on the best hyperparameters for each tree model with hyperparameter tuning and cross-validation, we trained the model with the best hyperparameters on the sampled training data and used the trained model to predict the test data.

Best Parameters for Tree-based Methods									
	n_estimators	min_samples_split	min_samples_leaf	max_depth	class_weight	criterion	learning_rate	subsample	colsample_bytree
Tuned Decision Tree	-	1602	101	2	{0: 1, 1: 2}	entropy	-	-	-
Random Forest	200	50	1	5	{0: 1, 1: 2}	-	-	-	-
Gradient Boosting	200	50	-	-	-	-	0.2	-	-
XGBoost	100	-	-	5	3 (scale_pos_weight)	-	0.01	0.8	0.8

Fig.3: Best Parameters Comparison for Tree-Based Methods

5.2.1 Evaluation of Tree-based Model Results

We proceeded to compare the results of each model on the test set and evaluated their performance based on our decided set of primary metrics, including recall at 1 and AUC score, along with other secondary metrics like precision at 1 and accuracy. Without any tuning done, the base decision tree model already achieves a relatively high recall rate of 0.79 and a decent AUC score of 0.77. This shows that even the base decision tree can already work well in predicting the opening of fraudulent bank accounts. However, after improving the model by using ensemble methods and gradient boosting

methods, the performance of tree-based methods greatly improved. When tested on the test set, both random forest and XGBoost models managed to achieve a recall rate that is higher than 0.90, indicating their ability to correctly identify almost all fraudulent account openings within the test set.

5.2.2 Best Tree-based Model Selection

After comparing all the tree-based models based on our primary metrics, we concluded that the XGBoost model performed the best out of the tree-based models, mainly due to it having the highest recall score at 1 of 0.99, coupled with a high AUC score of 0.86.

Tree-based Methods				
	Recall (y = 1)	ROC-AUC Score	Accuracy	Precision (y = 1)
Base Decision Tree	0.79	0.77	0.61	0.02
Tuned Decision Tree	0.97	0.71	0.23	0.01
Random Forest	0.95	0.86	0.44	0.02
Gradient Boosting	0.8	0.88	0.81	0.04
XGBoost	0.99	0.86	0.25	0.01

Fig.4: Tree-Based Method Score Evaluation

5.2.3 Feature Importance

When analysing the top 10 features for each tree-based model based on their importance, the feature "device_os_windows" was consistently listed as the most important feature for all the tree-based models. This indicates that the model can detect fraud based on what kind of operating system the application of the bank account was made on, and such information would be meaningful in helping banks to implement safeguarding policies in the future aimed at specific features like this.

Other features that were consistently listed as one of the most important include "prev_address_months_count" and "current_address_months_count".

Most Important Features (Tree-based Methods)			
	1	2	3
Tuned Decision Tree	device_os_windows	prev_address_months_count	current_address_months_count
Random Forest	device_os_windows	prev_address_months_count	current_address_months_count
Gradient Boosting	device_os_windows	current_address_months_count	name_email_similarity
XGBoost	device_os_windows	prev_address_months_count	housing_status_BE

Fig.5: Most Important Features for Each Tree-based Method

5.3 Best model selected

Best Model Selection				
	Recall (y = 1)	ROC-AUC Score	Accuracy	Precision (y = 1)
LogReg	0.79	0.87	0.80	0.04
XGBoost	0.99	0.86	0.25	0.01

Fig.6: Score Comparison between Two Models

After comparing the results of the best logistic regression as well as the best tree-based model, i.e., XGBoost, we determined XGBoost is the most effective model for fraud detection. XGBoost's high recall score outperforms the logistic regression score of 0.61, indicating its effectiveness in identifying fraudulent transactions in terms of the positive class. On the other hand, XGBoost also attained a higher ROC-AUC score, further demonstrating its ability to distinguish across various thresholds.

6. Discussion

Based on our analysis, we chose XGBoost as our preferred model for detecting fraudulent bank account openings due to its high recall score and ROC-AUC score. To further understand how the model predicts between fraud and no fraud, we can look at the most important features that it considers in its prediction and check if it would be in line with industry practices or benchmarks. The top features used by the model include checking the operating system of the device used for applications, the number of months the applicant stayed in their previous address, and the applicant's housing status.

We can see that there are some strengths and weaknesses with the prediction model. Surprisingly, one strength of the model is that its identification of the operating system as the most important feature is in line with industry practices. Fraud detection systems usually look for signs such as outdated operating systems to identify fraud, and this is usually because malware can be easily deployed on outdated operating systems. Banking malware has also been known to target Windows and other operating systems, with 70 percent of malware samples targeting the Windows operating system (Black, Gondal, Layton, 2018).

However, the other features identified as important by the model have some discrepancies with common industry practices. Based on common industry practice, banks tend to focus more on features such as the source of funding, the address of the applicant, and the location of receiving counterparty (Monetary Authority of Singapore, 2019). The features that are identified by the model do not have much relation to the features that banks tend to look out for when trying to detect and prevent fraud.

One main limitation of our analysis is the lack of real banking data available to train our predictive model. As we explained in the beginning, since real banking data is confidential and hard to come by, we are not able to train our model to its best capability since we are limited by the amount of data and information that we have. With the availability of real banking data, there might be other interesting and more accurate insights that our model would be able to gain that would not be possible with our current dataset.

On the other hand, there are still some takeaways from our training approach that would address our research objectives. Financial institutions can still benefit from our predictive model training approach, as they can couple our training approach with their confidential banking data to create a more accurate and robust prediction model for predicting fraudulent bank accounts.

7. Conclusion

Overall, our selected model would be beneficial to financial institutions as it provides a logical approach and process to train a prediction model that would be suitable to detect fraudulent bank account openings. The use of XGBoost as our selected model offers high predictive power, as seen in our recall rate on test data. Additionally, it also identifies feature importance during its training, enabling financial institutions to gain insights into the factors driving fraud detection outcomes.

However, our model also has its limitations in accuracy due to the lack of real-world data available for our training, which can cause our takeaways from the model to be less reliable. This can be improved by the financial institutions, which can provide their own real data when training their models to detect fraud.

Despite these constraints, our modelling approach remains a flexible and valuable blueprint for building a more tailored and impactful fraud detection system. Importantly, these models must be continuously monitored to ensure they remain fair and transparent, supporting financial institutions in preventing fraud without compromising customers' trust.

8. Appendix

Feature Name	Description
fraud_bool (binary)	If the application is fraudulent or not. (fraud_bool = 1 represents a fraudulent bank account and fraud_bool = 0 represents a legitimate bank account application).
income (numeric)	Annual income of the applicant (in decile form). Ranges between [0.1, 0.9].
name_email_similarity (numeric)	Metric of similarity between email and applicant's name. Higher values represent higher similarity. Ranges between [0, 1].
prev_address_months_count (numeric)	Number of months in previous registered address of the applicant, i.e. the applicant's previous residence, if applicable. Ranges between [–1, 380] months (–1 is a missing value).
current_address_months_count (numeric)	Months in currently registered address of the applicant. Ranges between [–1, 429] months (–1 is a missing value).
customer_age (numeric)	Applicant's age in years, rounded to the decade. Ranges between [10, 90] years.
days_since_request (numeric)	Number of days passed since application was done. Ranges between [0, 79] days.
intended_balcon_amount (numeric)	Initial transferred amount for application. Ranges between [–16, 114] (negatives are missing values).
payment_type (categorical)	Credit payment plan type. 5 possible (anonymized) values.

zip_count_4w (numeric)	Number of applications within same zip code in last 4 weeks. Ranges between [1, 6830].
velocity_6h (numeric)	Velocity of total applications made in last 6 hours, i.e., average number of applications per hour in the last 6 hours. Ranges between [–175, 16818].
velocity_24h (numeric)	Velocity of total applications made in last 24 hours, i.e., average number of applications per hour in the last 24 hours. Ranges between [1297, 9586].
velocity_4w (numeric)	Velocity of total applications made in last 4 weeks, i.e., average number of applications per hour in the last 4 weeks. Ranges between [2825, 7020].
bank_branch_count_8w (numeric)	Number of total applications in the selected bank branch in last 8 weeks. Ranges between [0, 2404].
date_of_birth_distinct_emails_4w (numeric)	Number of emails for applicants with same date of birth in last 4 weeks. Ranges between [0, 39].
employment_status (categorical)	Employment status of the applicant. 7 possible (anonymized) values.
credit_risk_score (numeric)	Internal score of application risk. Ranges between [–191, 389].
email_is_free (binary)	Domain of application email (either free or paid).
housing_status_BE (categorical)	Current residential status of applicant. 7 possible (anonymized) values.
phone_home_valid (binary)	Validity of provided home phone.

phone_mobile_valid (binary)	Validity of provided mobile phone.
bank_months_count (numeric)	How old is previous account (if held) in months. Ranges between [-1, 32] months (–1 is a missing value).
has_other_cards (binary)	If applicant has other cards from the same banking company.
proposed_credit_limit (numeric)	Applicant's proposed credit limit. Ranges between [200, 2000].
foreign_request (binary)	If origin country of request is different from bank's country.
source (categorical)	Online source of application. Either browser (INTERNET) or app (TELEAPP).
session_length_in_minutes (numeric)	Length of user session in banking website in minutes. Ranges between [–1, 107] minutes (–1 is a missing value).
device_os_windows (categorical)	Operative system of device that made request. Possible values are: Windows, macOS, Linux, X11, or other.
keep_alive_session (binary)	User option on session logout.
device_distinct_emails_8w (numeric)	Number of distinct emails in banking website from the used device in last 8 weeks. Ranges between [–1, 2] emails (–1 is a missing value).
device_fraud_count (numeric)	Number of fraudulent applications with used device. Ranges between [0, 1].
month (numeric)	Month where the application was made. Ranges between [0, 7].

Fig. 1: List of Features and Description

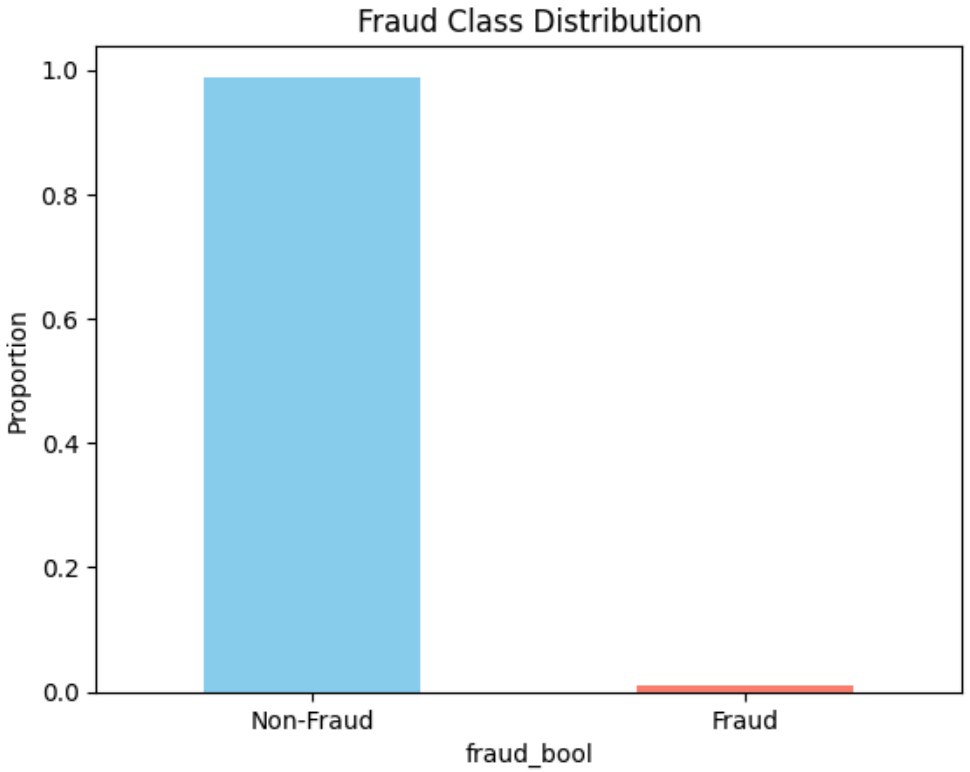

Fig. 2. Fraud Class Distribution (Bar Plot)

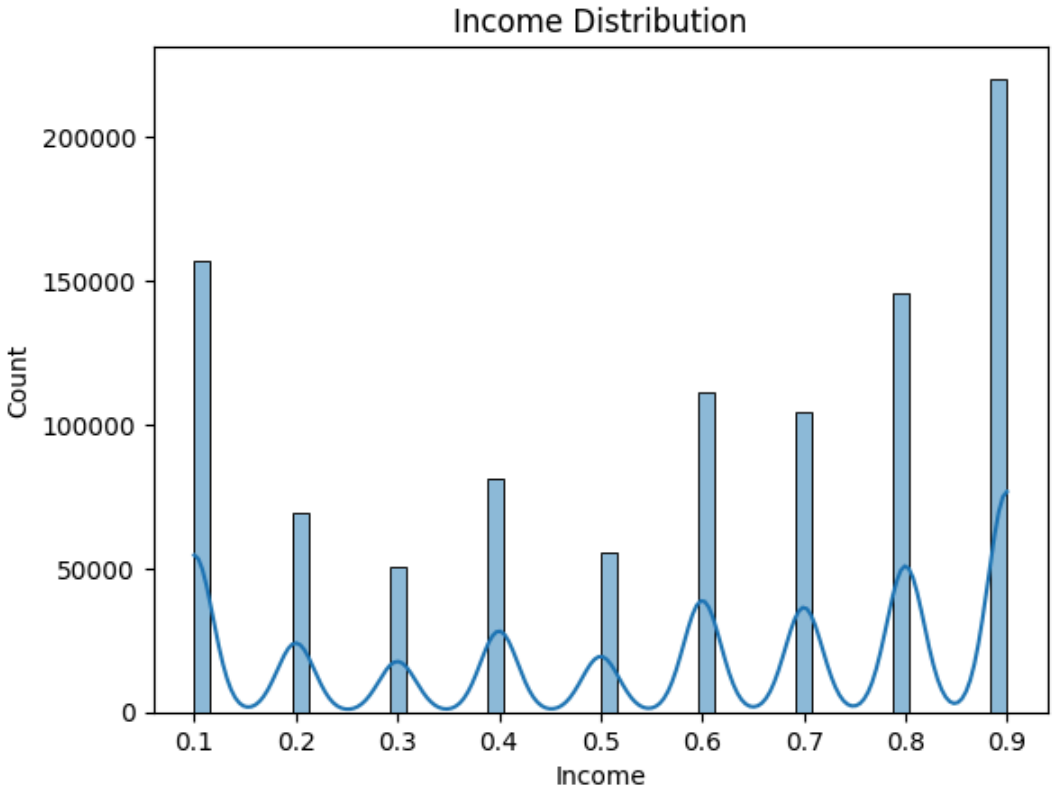

Fig. 3. Income Distribution (Histogram with KDE)

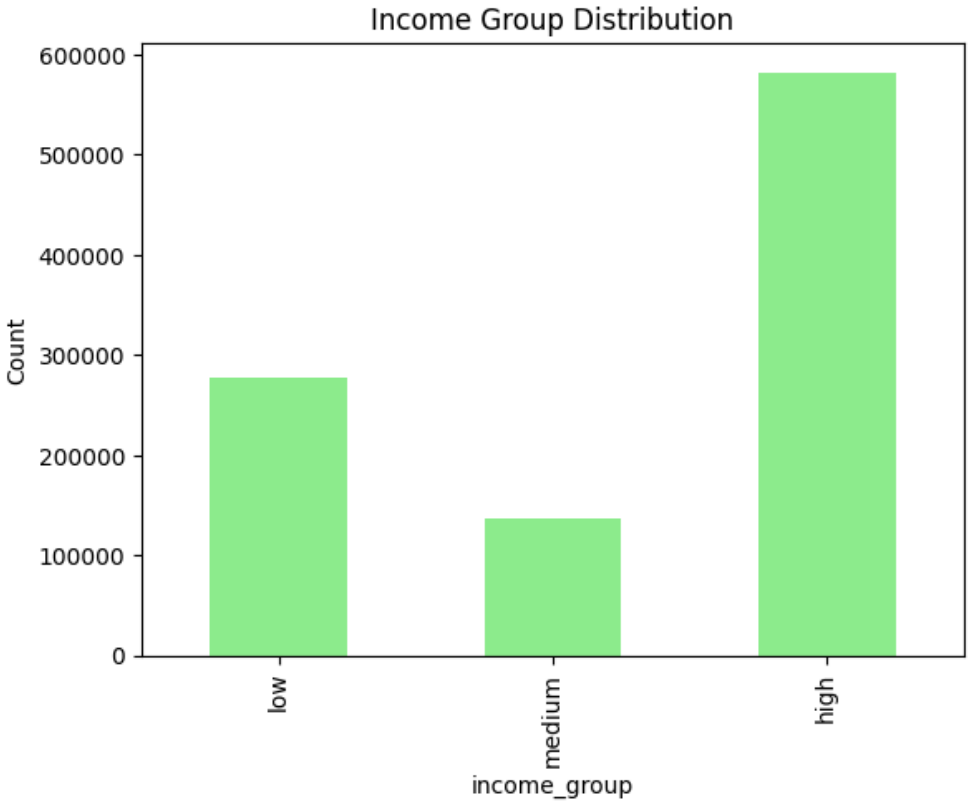

Fig. 4. Income Group Distribution (Bar Plot)

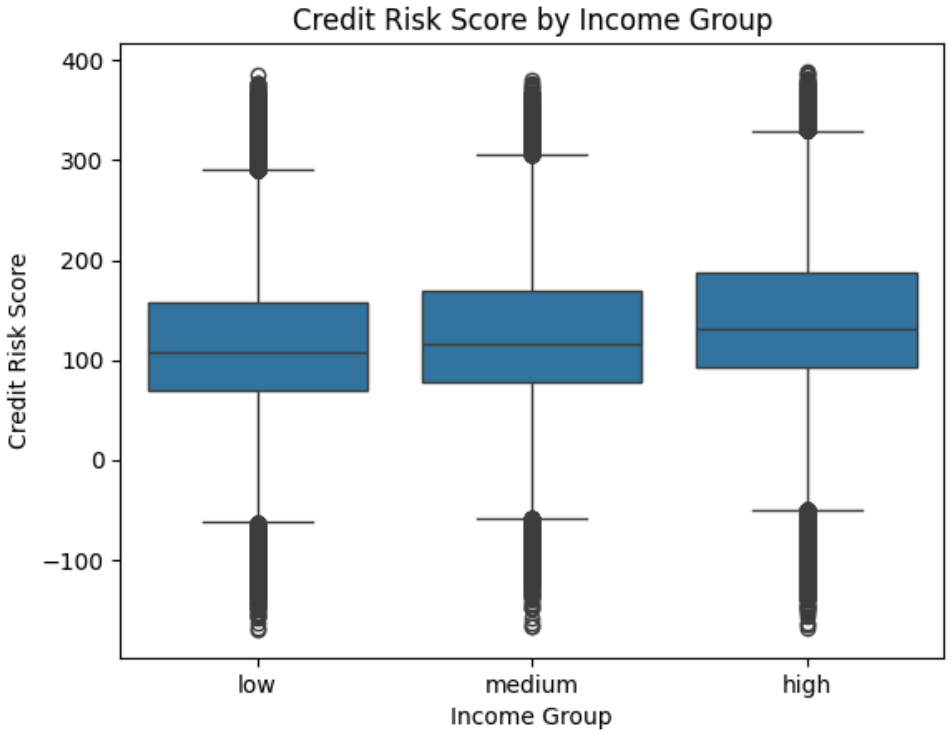

Fig. 5. Credit Risk Score by Income Group (Boxplot)

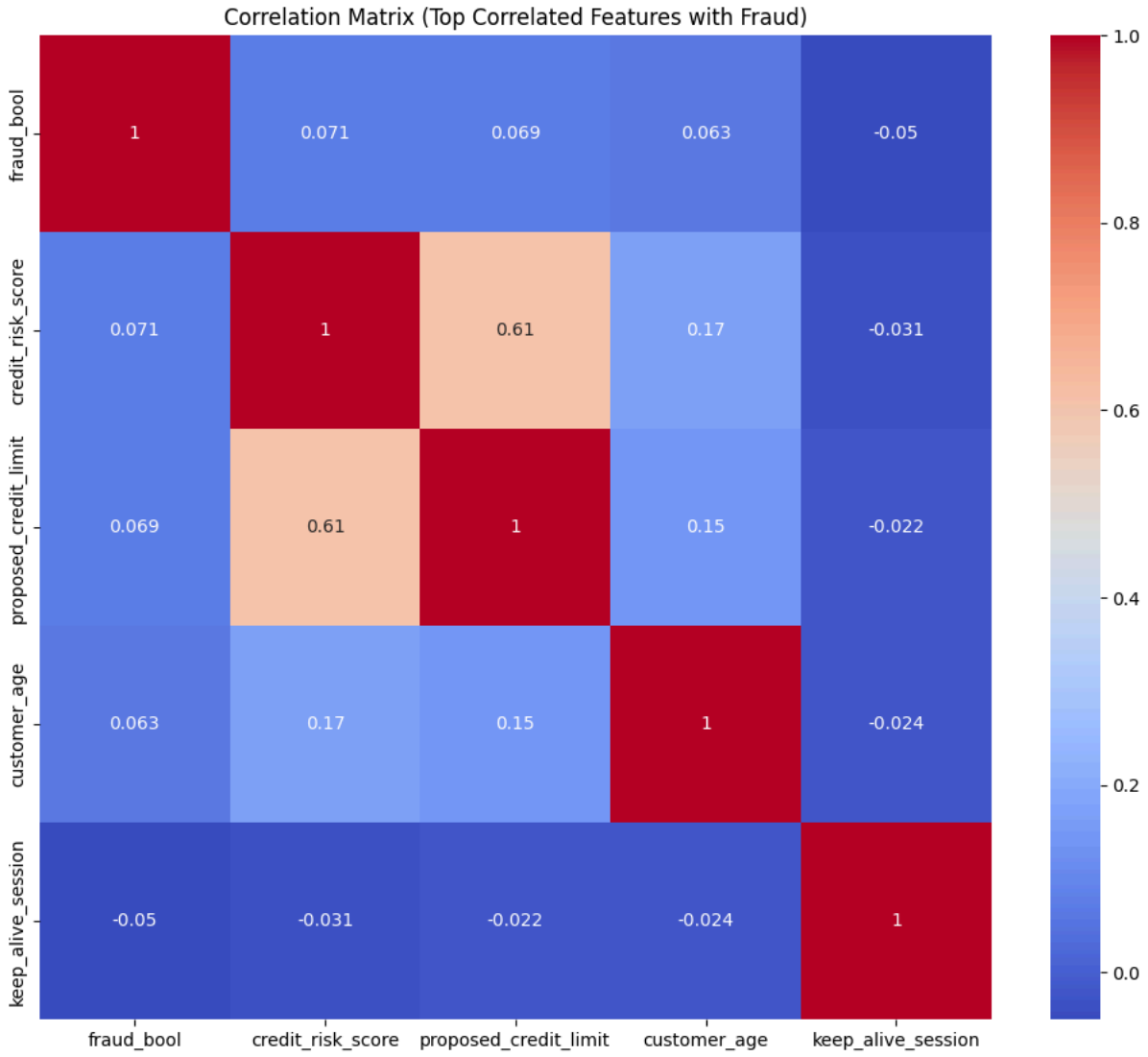

Fig. 6. Correlation Matrix of Top Fraud-Related Features (Heatmap)

Codes

The codes for our split into the following notebooks for different uses:

- Data Preprocessing and Cleaning: Step1_Data_Cleaning.ipynb
- Exploratory Data Analysis: Step2_EDA.ipynb
- Logistic Regression Models: Step3a_LogisticRegression_Model.ipynb
- Tree-based Models: Step3b_Tree_Based_Models.ipynb

9. References

Black, Gondal, & Layton. (2017, October 9). *A survey of similarities in banking malware behaviours*. Computers & Security. https://www.sciencedirect.com/science/article/abs/pii/S016740481730202X

Effective-practices-to-detect-and-mitigate-the-risk-from- ... (2019). https://www.mas.gov.sg/-/media/MAS/Regulations-and-Financial-Stability/Regulatory-and-Supervisory-Framework/Anti_Money-Laundering_Countering-the-Financing-of-Terrorism/Effective-Practices-to-Detect-and-Mitigate-the-Risk-from-Misuse-of-Legal-Persons-June-2019.pdf

Feki, R. (2022, January 3). Imbalanced data: best practices. A guide to deliver great results on ML. Medium. Retrieved November 2, 2024, from https://rihab-feki.medium.com/imbalanced-data-best-practices-f3b6d0999f38

Jesus, S. (2023, November 29). *Bank Account Fraud Dataset Suite (neurips 2022)*. Kaggle. https://www.kaggle.com/datasets/sgpjesus/bank-account-fraud-dataset-neurips-2022/data

Khandelwal, N. (2020, July 7). *A Brief Introduction to XGBoost. Extreme Gradient Boosting with XGBoost! | by Neetika Khandelwal*. Towards Data Science. Retrieved November 2, 2024, from https://towardsdatascience.com/a-brief-introduction-to-xgboost-3eaee2e3e5d6

Machine-learning techniques for detecting new account fraud. Transmit Security. (2023, August 23). https://transmitsecurity.com/blog/machine-learning-techniques-for-detecting-new-account-fraud

Money laundering. Europol. (2022). https://www.europol.europa.eu/crime-areas/economic-crime/money-laundering

Total value of assets surrendered to the state in connection with the $3 billion money laundering case as of 31 December 2024. Ministry of Home Affairs. (2025).

https://www.mha.gov.sg/mediaroom/parliamentary/total-value-of-assets-surrendered-to-the-state-in-connection-with-the-3-billion-money-laundering-case-as-of-31-december-2024/