CISC 839:Topics in Data Analytics
Final Project
Fraud Detection

Group 9

Amr Sayed	20398048
Bilal Morsy	20398551
Hassan Ahmed	20399136
Omar Amer	20401064

Supervised by:

Dr. Yuan Tian

Outline

- Introduction and Motivation
- Dataset Statistics and EDA
- Methodology
- Experimental Results
- Discussion and Conclusion

Introduction & Motivation

In this section we define what is the problem as well as the motivation why solving this problem is important.

Introduction

- **Bank fraud:** A financial institution challenge.
- **Detecting fraud:** Key to success.
- **Imbalanced data**: Model development challenge.
- **Privacy-preserving**: Dataset features.
- **Reducing financial risks:** Fraud detection impact.
- **Enhancing customer trust:** Fraud prevention importance.

Motivation

- **Fraud detection:** Safeguarding customers and institutions.
- **Transaction security**: Maintaining financial safety.
- **Trust preservation**: Ensuring reliable services.
- **Loss minimization**: Reducing financial impacts.
- **Bias avoidance**: Upholding ethical AI practices.
- **Fairness in ML**: Ensuring equitable model outcomes.

Dataset and Statistics

In this section we discover the dataset structure and perform EDA.

Datasets

We used the introduced Bank Account Fraud (BAF) dataset, which is the first publicly available privacy-preserving, large-scale and realistic suite of tabular datasets. BAF suite of datasets has been published at NeurIPS 2022.

The suite's datasets were created using advanced Generative Adversarial Network (GAN) models to ensure the privacy of applicants, which is a growing concern in today's society and legal landscape.

The BAF dataset advantages:

- It mimics a real banking dataset
- It consists of 6 datasets (Base and 5 variants for comparison)
- Provides privacy-protected attributes
- Serves as a large-scale test set which helps in model generalization

Datasets (Cont'd)

Our BAF base dataset contains:

1 million instances

32 realistic features used in the fraud detection use-case

A column of "month", providing temporal information about the dataset

Protected attributes, (age group, employment status and % income).

Column Name
income
name_email_similarity
prev_address_months_count
current_address_months_count
customer_age
days_since_request
intended_balcon_amount
payment_type
zip_count_4w
velocity_6h
velocity_24h
velocity_4w
bank_branch_count_8w
date_of_birth_distinct_emails_4w
employment_status
credit_risk_score
email_is_free
housing_status
phone_home_valid
phone_mobile_valid
bank_months_count
has_other_cards
proposed_credit_limit
foreign_request
source
session_length_in_minutes
device_os
keep_alive_session
device_distinct_emails
device_fraud_count
month

Datasets (Cont'd): Variants

1st variant

- Introduces higher group size disparity in protective fields.

2nd variant

- Introduces higher prevalence disparity.

3rd variant

- Provides better separability for one of the demographic groups.

4th variant

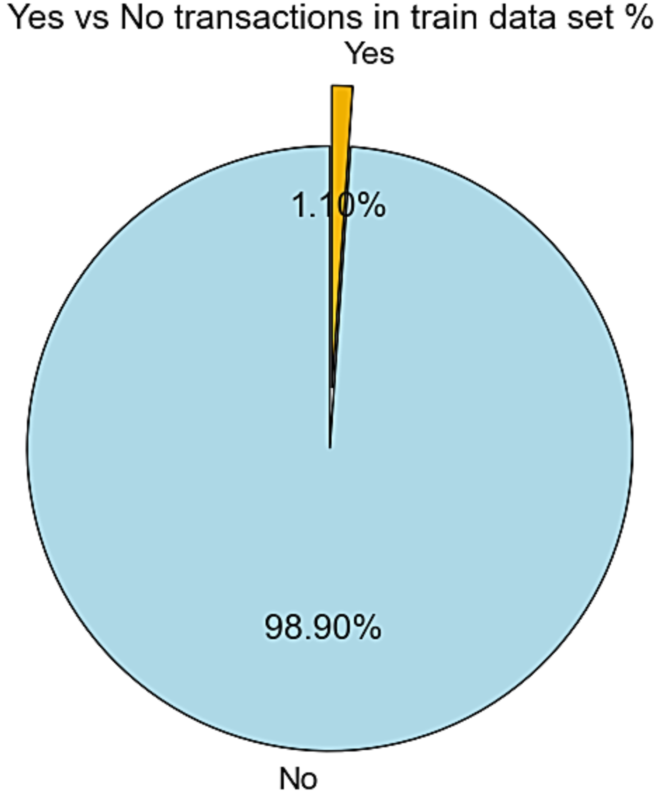
- Introduces higher prevalence disparity in the training dataset.

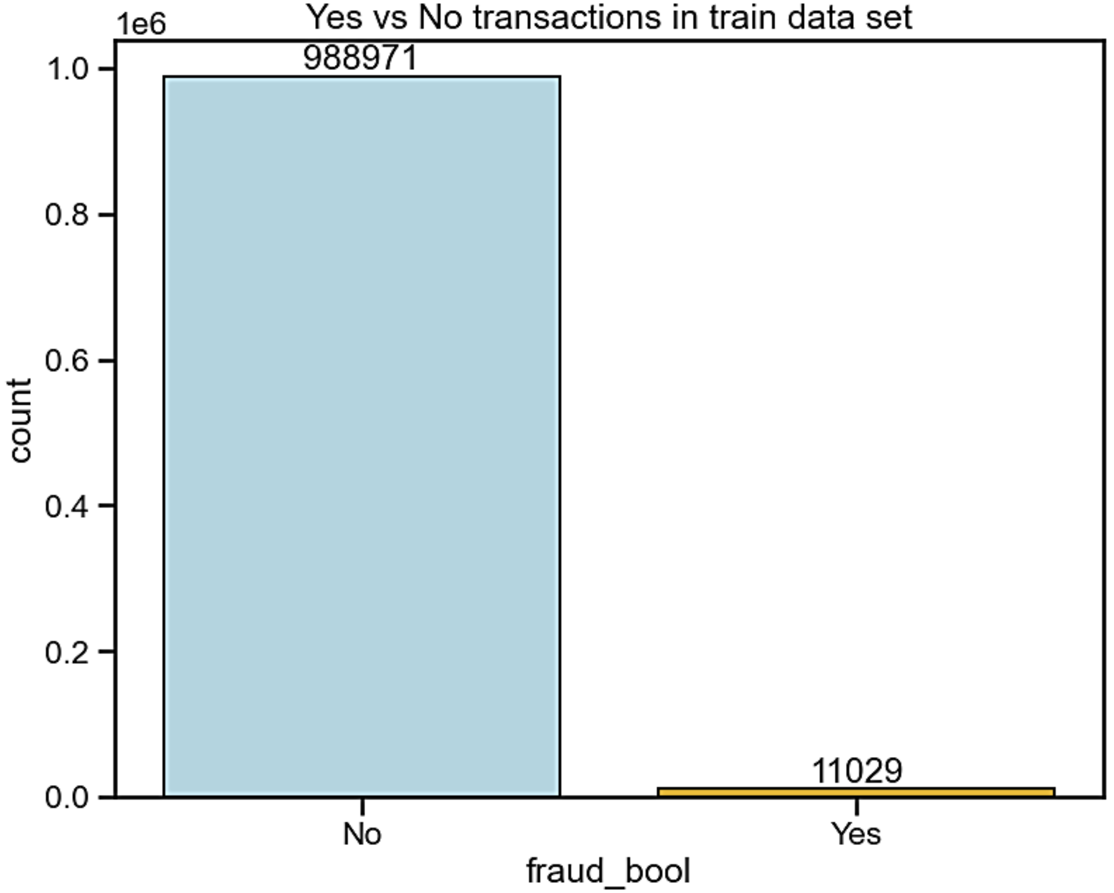
5th variant

- Provides better separability in the training dataset for one of the demographic groups.

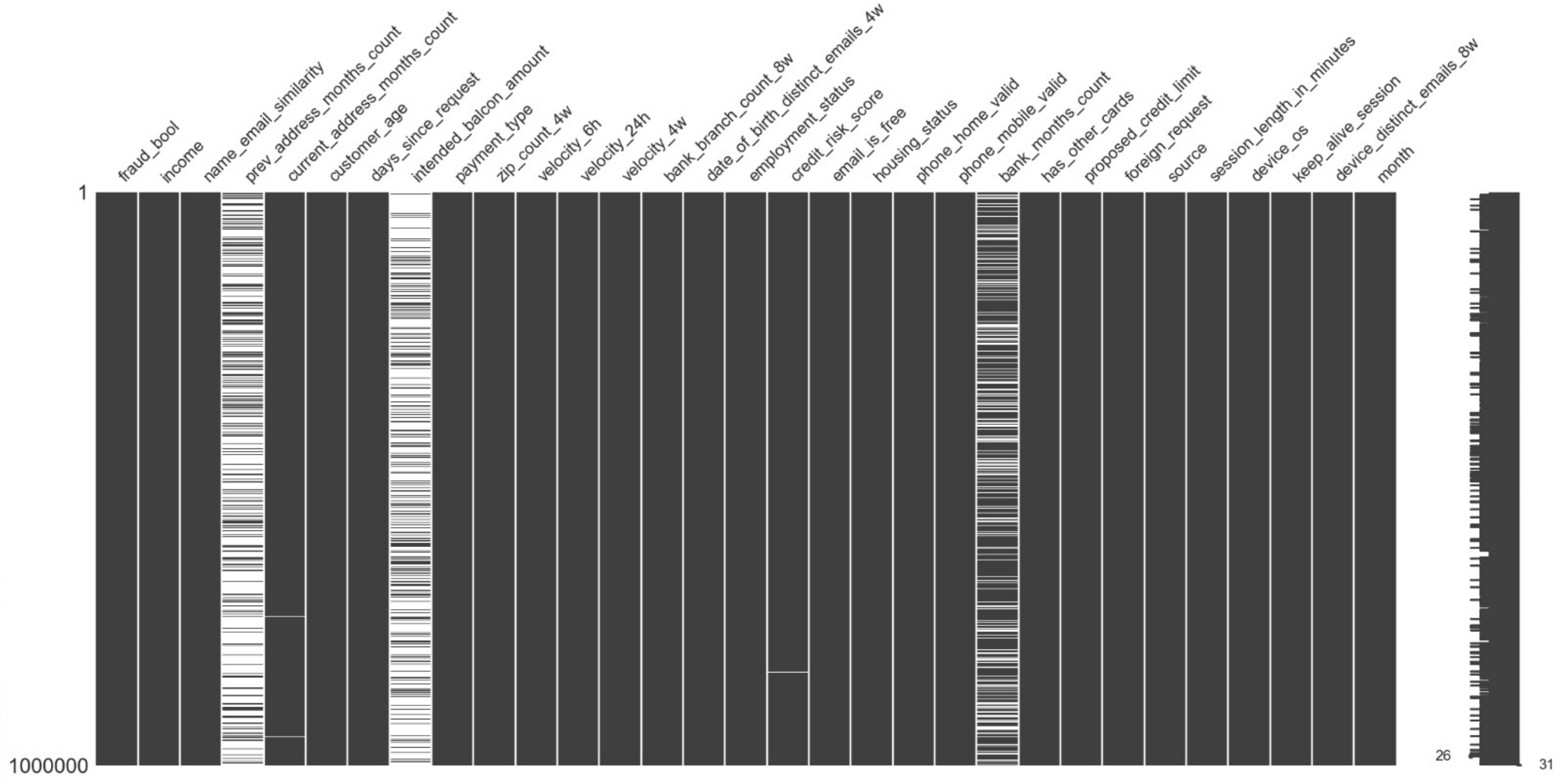
Base Dataset Statistics: Imbalance

We have 98.90% of non-fraud (988971) and only 1.103% (11029) of fraud transactions!



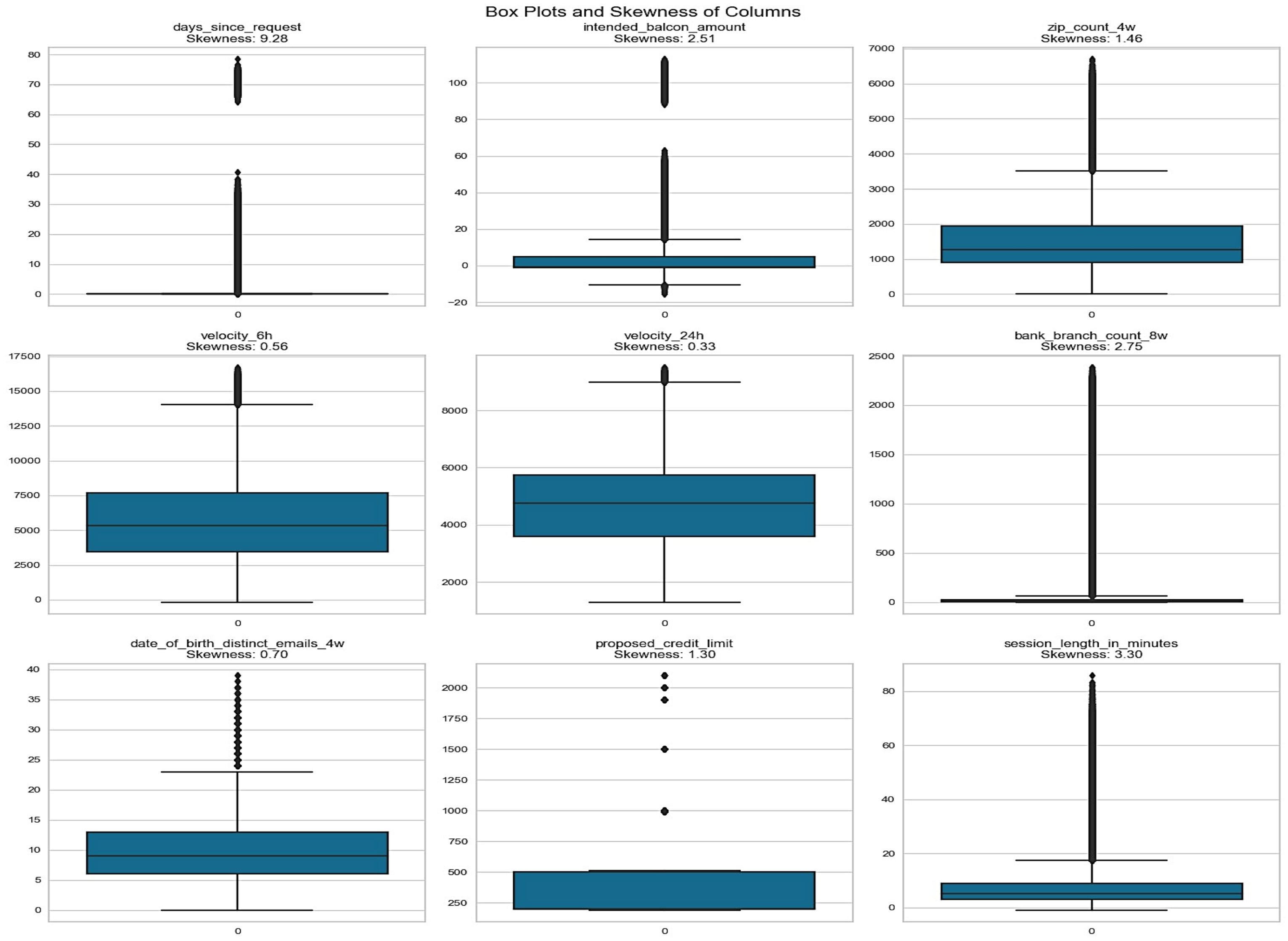


Base Dataset Statistics (Cont'd): Nulls



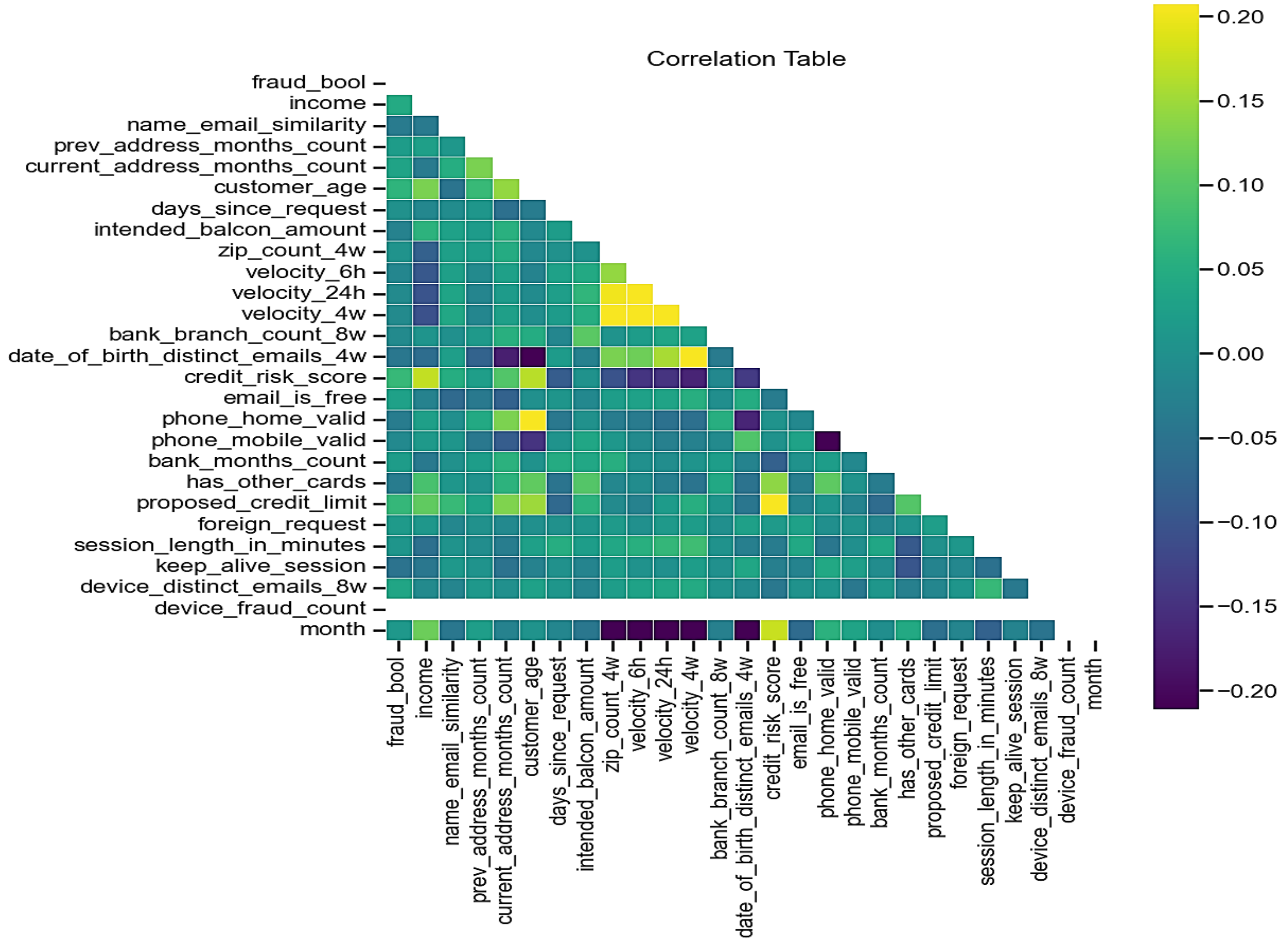
Features	Nulls %
prev_address_months_count	71.2920
intended_balcon_amount	74.2523
current_address_months_count	0.4254
credit_risk_score	0.0488
bank_months_count	25.3635
session_length_in_minutes	0.2015
device_distinct_emails_8w	0.0359

Base Dataset Statistics (Cont'd): Outliers



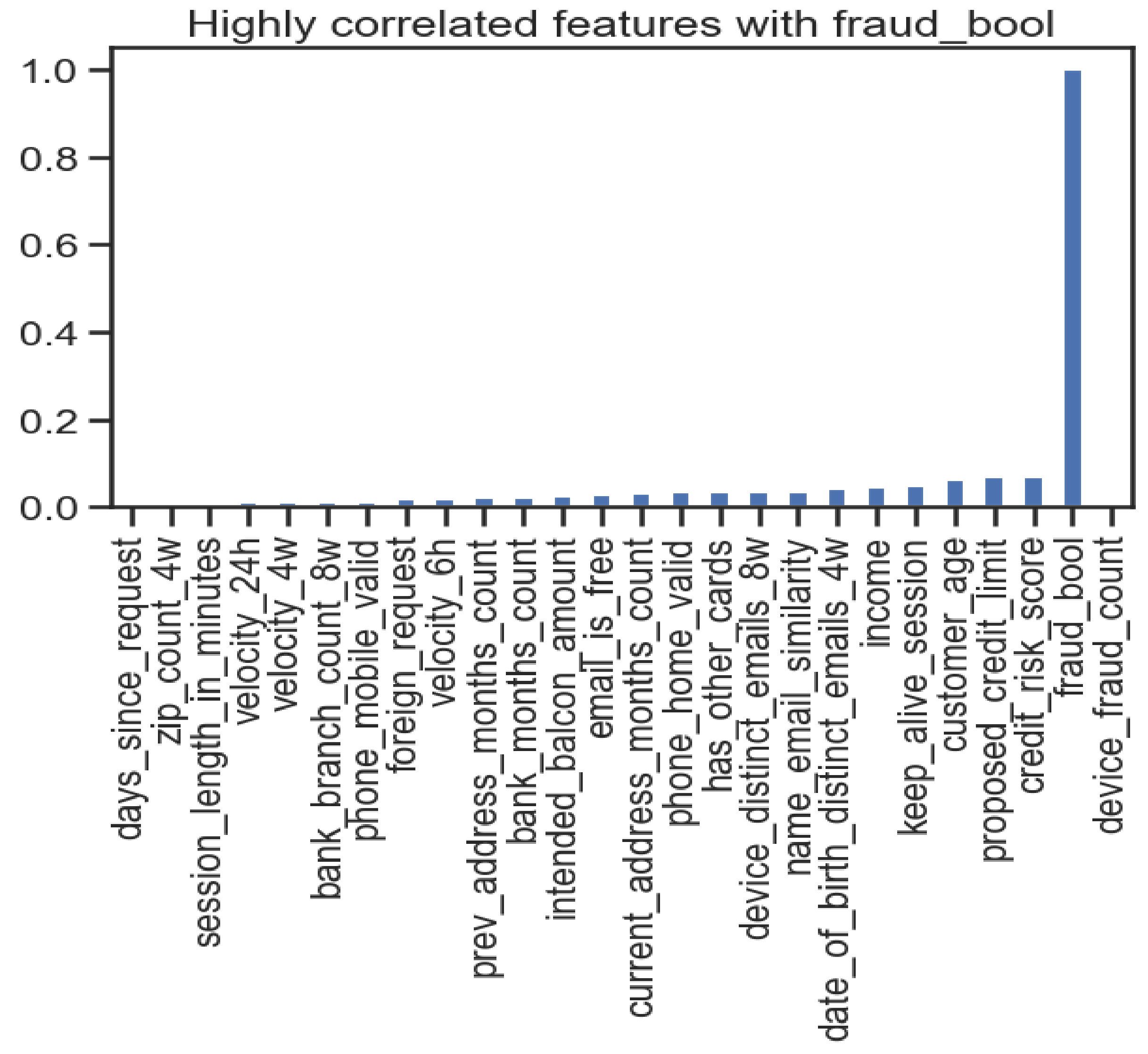
Features Having Outliers
days_since_request
intended_balcon_amount
zip_count_4w
velocity_6h
velocity_24h
bank_branch_count_8w
date_of_birth_distinct_emails_4w
proposed_credit_limit
session_length_in_minutes

Base Dataset Statistics (Cont'd): Correlation



No features exceeded 20% correlation. Thus, there's a little multi-co-linearity.

Base Dataset Statistics (Cont'd): Correlation

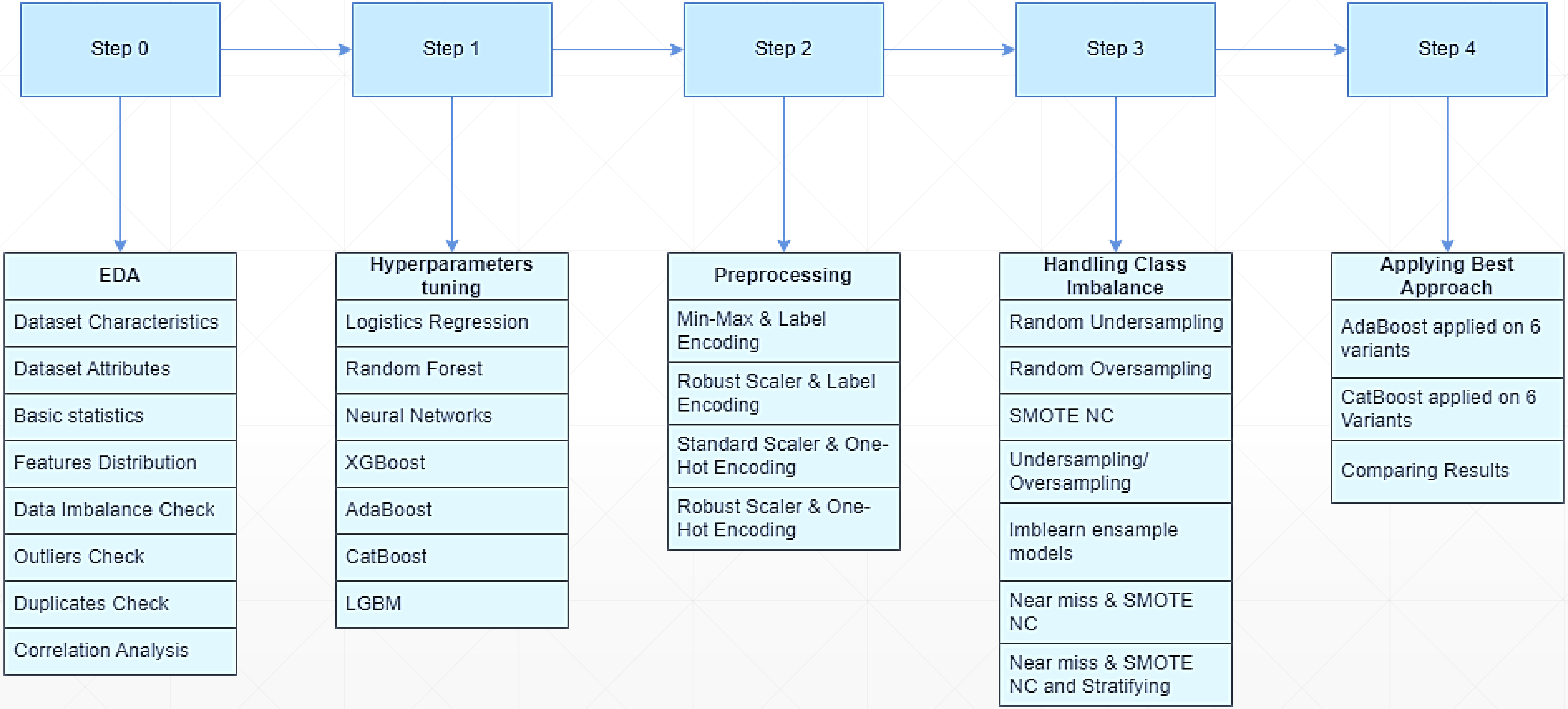


No features correlated with the fraud bool.

Methodology

In this section we describe our modeling strategy as well as our evaluation metrics.

Methodology: Pipelines

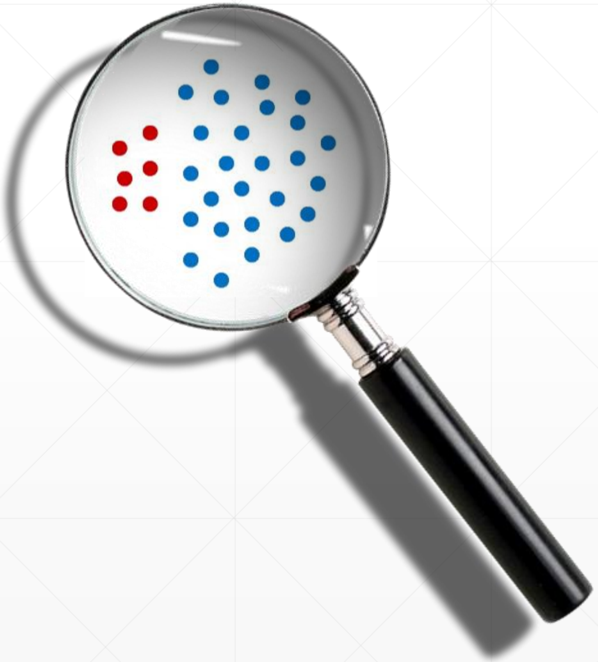


Methodology (Cont’d): Evaluation Metrics

Performance Metric: Recall @ 5% FPR (False Positive Rate)

Fairness Metric: Predictive Equality (FPR Balance)

AUC - Area Under the Curve



Experimental Results

In this section we go through the performed pipelines and their results from Step 1 till Step 4.

Step 1

Hyperparameters Tuning for the Baseline Models.

Experiments Results: Step 1

Model	AUC	TPR	Predictive equality
Logistic Regression	0.877	49.69%	89.52%
Random Forest	0.805	33.39%	34.14%
Neural Network	0.884	51.88%	84.36%
XGBoost	0.867	46.63%	76.07%

Baseline Models

Model	AUC	TPR	Predictive equality
Logistic Regression	0.879	49.65%	88.42%
Random Forest	0.872	48.68%	96.23%
Neural Network	0.884	52.19%	99.25%
AdaBoost	0.893	52.40%	100.00%
XGBoost	0.886	54.66%	88.81%
CatBoost	0.895	55.14%	86.27%
LGBM	0.886	51.91%	79.99%

Results after Hyperparameters Tuning

Model	AUC	TPR	Predictive equality
Logistic Regression	0.002	-0.0004	-0.011
Random Forest	0.067	0.1529	0.6209
Neural Network	0	0.0031	0.1489
XGBoost	0.019	0.0803	0.1274

Deviation of results from baseline models

Step 2 – 4 Pipelines

Experimenting with Different Preprocessing Pipelines.

Step 2 - Pipeline 1

Step 1
- Imputing the missing values by Mean, Mode

Step 2
- Detect outliers
- Clip outliers

Step 3
- Using MinMax Scaler
- Using Label encoding

Step 4
- Model selection(using the same models of step 1)

Step 5
- Tuning the best model(CatBoost)

Step 2 - Pipeline 1 Results

Model	AUC	TPR	Predictive equality
Logistic Regression	0.867	46.11%	89.01%
Random Forest	0.801	30.33%	33.84%
Neural Network	0.872	47.88%	93.63%
XGBoost	0.888	52.99%	94.11%
AdaBoost	0.8893	53.47%	100.00%
CatBoost(after tuning again)	0.893	54.93%	85.81%
LGBM	0.882	50.38%	79.88%

Pipeline results

Model	AUC	TPR	Predictive equality
Logistic Regression	-0.012	-0.035	0.0059
Random Forest	-0.071	-0.184	-0.62391
Neural Network	-0.012	-0.043	-0.0562
XGBoost	0.002	-0.017	0.053
AdaBoost	-0.004	0.0107	0
CatBoost	-0.002	-0.002	-0.0046
LGBM	-0.004	-0.015	-0.0011

Deviation from step 1 results

Step 2 - Pipeline 2

Step 1
- Imputing the missing values by Mean, Mode
- Deleted 'prev_address_months_count','bank_months_count'

Step 2
- Didn't handle the outliers.

Step 3
- Using Robust Scaler
- Using Label encoding

Step 4
- Model selection (using the same models of step 1)

Step 5
- Tuning the best model (CatBoost)

Step 2 - Pipeline 2 Results

Model	AUC	TPR	Predictive equality
Logistic Regression	0.859	44.89%	92.65%
Random Forest	0.875	48.19%	98.92%
Neural Network	0.8749	48.26%	92.91%
XGBoost	0.884	51.11%	80.26%
AdaBoost	0.885	52.61%	100.00%
CatBoost(after tuning again)	0.893	55.35%	87.51%
LGBM	0.882	50.42%	79.69%

Pipeline results

Model	AUC	TPR	Predictive equality
Logistic Regression	-0.02	-0.0476	0.0423
Random Forest	0.003	-0.0049	0.0269
Neural Network	-0.0091	-0.0393	-0.0634
XGBoost	-0.002	-0.0355	-0.08551
AdaBoost	-0.008	0.0021	0
CatBoost	-0.002	0.0021	0.0124
LGBM	-0.004	-0.0149	-0.003

Deviation from step 1 results

Step 2 - Pipeline 3

Step 1
- Imputing the missing values by Mean, Mode
- Deleted 'prev_address_months_count','bank_months_count'

Step 2
- Deleted the outliers.

Step 3
- Using Standard Scaler
- Using One Hot Encoding

Step 4
- Model selection (using the same models of step 1)

Step 5
- Tuning the best model (CatBoost)

Step 2 - Pipeline 3 Results

Model	AUC	TPR	Predictive equality
Logistic Regression	0.877	48.55%	87.48%
Random Forest	0.872	47.00%	99.53%
Neural Network	0.878	49.81%	86.79%
XGBoost	0.883	52.08%	91.38%
AdaBoost	0.884	52.03%	100.00%
CatBoost(after tuning again)	0.885	51.94%	91.97%
LGBM	0.871	46.90%	82.56%

Pipeline results

Model	AUC	TPR	Predictive equality
Logistic Regression	-0.002	-0.011	-0.0094
Random Forest	0	-0.0168	0.033
Neural Network	-0.006	-0.0238	-0.1246
XGBoost	-0.003	-0.0258	0.0257
AdaBoost	-0.009	-0.0037	0
CatBoost	-0.01	-0.032	0.057
LGBM	-0.015	-0.0501	0.0257

Deviation from step 1 results

Step 2 - Pipeline 4

Step 1
- Imputing the missing values by Mean, Mode
- Deleted 'prev_address_months_count'

Step 2
- Didn't handle the outliers.

Step 3
- Using Robust Scaler
- Using One Hot Encoding
- .And deleted the last column created for each feature after one hot encoding manually

Step 4
- Model selection (using the same models of step 1)

Step 5
- Tuning the best model (CatBoost)

Step 2 - Pipeline 4 Results

Model	AUC	TPR	Predictive equality
Logistic Regression	0.8646	45.97%	90.09%
Random Forest	0.8724	48.05%	99.77%
Neural Network	0.884	51.18%	93.76%
XGBoost	0.8863	52.22%	82.13%
AdaBoost	0.8874	52.81%	100.00%
CatBoost(after tuning again)	0.8933	54.86%	86.46%
LGBM	0.8789	49.79%	80.52%

Pipeline results

Model	AUC	TPR	Predictive equality
Logistic Regression	-0.0144	-0.0368	0.0167
Random Forest	0.0004	-0.0063	0.0354
Neural Network	0	-0.0101	-0.0549
XGBoost	0.0003	-0.0244	-0.0668
AdaBoost	-0.0056	0.0041	0
CatBoost	-0.0017	-0.0028	0.0019
LGBM	-0.0071	-0.0212	0.0053

Deviation from step 1 results

Step 3 – 7 Pipelines

Experimenting with Different SOTA techniques to Handle the Class Imbalance Issue.

Step 3 - Pipeline 1

Step 1

- Used Step 2 – Pipeline 3 Preprocessing

Step 2

- Handled Class Imbalance using “imblearn” ensemble models

Imblearn Ensemble models Benefits

- No need for manual class balancing
- Reduced Overfitting
- Provides both Bagging and Boosting Models

Step 3 - Pipeline 1 Results

Model	AUC	TPR	Predictive equality
Balanced Bagging Classifier	0.8777	49.42%	95.37%
Balanced Random Forest	0.8542	41.29%	100.00%
RUS Boosting Classifier	0.8774	48.98%	100.00%
Easy Ensemble Classifier	0.8861	52.08%	100.00%

Pipeline Results

Model	AUC	TPR	Predictive equality
Logistic Regression	0.877	48.55%	87.48%
Random Forest	0.872	47.00%	99.53%
Neural Network	0.878	49.81%	86.79%
XGBoost	0.883	52.08%	91.38%
AdaBoost	0.884	52.03%	100.00%
CatBoost(after tuning again)	0.885	51.94%	91.97%
LGBM	0.871	46.90%	82.56%

Step 2 Pipeline 3 Results

Model	Compared With	AUC	TPR	Predictive equality
Balanced Bagging Classifier	Logistic Regression	0.0007	0.87%	7.89%
Balanced Random Forest	Random Forest	-0.018	-5.71%	0.47%
RUS Boosting Classifier	AdaBoost	-0.007	-3.05%	0.00%
Easy Ensemble Classifier	AdaBoost	0.0021	0.05%	0.00%

Deviation from Step 2 Pipeline 3 Results

Step 3 - Pipeline 2

Step 1
- Used Step 2 – Pipeline 4 Preprocessing

Step 2
- Handled Class Imbalance using Random Undersampling

Random Undersampling Benefits
- Balancing the dataset
- Reduce Overfitting
- Faster Training

Step 3 - Pipeline 2 Results

Model	AUC	TPR	Predictive equality
Logistic Regression	0.8697	47.12%	95.99%
Random Forest	0.8701	48.33%	99.96%
Neural Network	0.8858	52.08%	85.67%
XGBoost	0.8819	50.49%	63.85%
AdaBoost	0.8827	51.77%	100.00%
CatBoost(after tuning again)	0.8921	54.10%	89.29%
LGBM	0.8889	52.74%	87.27%

Pipeline Results

Model	AUC	TPR	Predictive equality
Logistic Regression	0.0051	0.0115	0.059
Random Forest	-0.0023	0.0028	0.0019
Neural Network	0.0018	0.009	-0.0809
XGBoost	-0.0044	-0.0173	-0.1828
AdaBoost	-0.0047	-0.0104	0
CatBoost	-0.0012	-0.0076	0.0283
LGBM	0.01	0.0295	0.0675

Deviation from Step 2 Pipeline Results

Step 3 - Pipeline 3

Step 1

- Used Step 2 – Pipeline 4 Preprocessing

Step 2

- Handled Class Imbalance using Random Oversampling

Random Oversampling Benefits

- Improve Minority Class Representation
- Prevent Overfitting on Majority Class
- Avoid Information Loss

Step 3 - Pipeline 3 Results

Model	AUC	TPR	Predictive equality
Logistic Regression	0.8652	46.35%	92.96%
Random Forest	0.8724	48.12%	99.79%
Neural Network	0.8863	53.27%	88.72%
XGBoost	0.8845	53.27%	78.64%
AdaBoost	0.8874	53.02%	100.00%
CatBoost(after tuning again)	0.8937	54.76%	85.90%
LGBM	0.8916	54.03%	83.86%

Pipeline Results

Model	AUC	TPR	Predictive equality
Logistic Regression	0.0006	0.0038	0.0287
Random Forest	0	0.0007	0.0002
Neural Network	0.0023	0.0209	-0.0504
XGBoost	-0.0018	0.0105	-0.0349
AdaBoost	0	0.0021	0
CatBoost	0.0004	-0.001	-0.0056
LGBM	0.0127	0.0424	0.0334

Deviation from Step 2 Pipeline Results

Step 3 - Pipeline 4

Step 1

- Used Step 2 – Pipeline 4 Preprocessing

Step 2

- Handled Class Imbalance using SMOTE-NC Oversampling

SMOTE-NC Benefits

- Handling Continuous and Categorical Features
- Enhancing Generalization
- Avoiding Data Loss

Step 3 - Pipeline 4 Results

Model	AUC	TPR	Predictive equality
Logistic Regression	0.8384	40.58%	80.36%
Random Forest	0.8582	44.48%	100.00%
Neural Network	0.8443	42.81%	100.00%
XGBoost	0.8393	38.60%	61.05%
AdaBoost	0.8467	42.77%	100.00%
CatBoost(after tuning again)	0.8567	45.59%	55.84%
LGBM	0.8462	43.71%	65.23%

Pipeline Results

Model	AUC	TPR	Predictive equality
Logistic Regression	-0.0262	-0.0539	-0.0973
Random Forest	-0.0142	-0.0357	0.0023
Neural Network	-0.0397	-0.0837	0.0624
XGBoost	-0.047	-0.1362	-0.2108
AdaBoost	-0.0407	-0.1004	0
CatBoost	-0.0366	-0.0927	-0.3062
LGBM	-0.0327	-0.0608	-0.1529

Deviation from Step 2 Pipeline Results

Step 3 - Pipeline 5

Step 1

- Used Step 2 – Pipeline 4 Preprocessing

Step 2

- Handled Class Imbalance using Random Undersampling followed by Random Oversampling

Undersampling then Oversampling Benefits

- Improved Generalization
- Reduced Overfitting
- Computationally Efficient

Step 3 - Pipeline 5 Results

Model	AUC	TPR	Predictive equality
Logistic Regression	0.8509	43.29%	74.09%
Random Forest	0.8663	45.21%	90.91%
Neural Network	0.8742	49.76%	57.10%
XGBoost	0.8648	45.21%	54.34%
AdaBoost	0.8727	48.85%	100.00%
CatBoost(after tuning again)	0.8791	51.18%	60.85%
LGBM	0.8754	50.24%	54.95%

Pipeline Results

Model	AUC	TPR	Predictive equality
Logistic Regression	-0.0137	-0.0268	-0.16
Random Forest	-0.0061	-0.0284	-0.0886
Neural Network	-0.0098	-0.0142	-0.3666
XGBoost	-0.0215	-0.0701	-0.2779
AdaBoost	-0.0147	-0.0396	0
CatBoost	-0.0142	-0.0368	-0.2561
LGBM	-0.0035	0.0045	-0.2557

Deviation from Step 2 Pipeline Results

Step 3 - Pipeline 6

Step 1

- Used Step 2 – Pipeline 4 Preprocessing

Step 2

- Handled Class Imbalance using Near Miss Undersampling then SOMTE-NC Oversampling

Near Miss Undersampling then SMOTE-NC Oversampling Benefits

- Improved Generalization
- Reduced Overfitting
- Computationally Efficient
- Complementary Effects
- Handling Continuous and Categorical Features

Step 3 - Pipeline 6 Results

Model	AUC	TPR	Predictive equality
Logistic Regression	0.8946	57.44%	68.26%
Random Forest	0.9138	62.30%	91.18%
Neural Network	0.9372	73.31%	57.96%
XGBoost	0.8648	45.21%	54.34%
AdaBoost	0.91	63.17%	100.00%
CatBoost(after tuning again)	0.9414	75.30%	60.79%
LGBM	0.9343	71.58%	62.57%

Pipeline Results

Model	AUC	TPR	Predictive equality
Logistic Regression	0.03	0.1147	-0.2183
Random Forest	0.0414	0.1425	-0.0859
Neural Network	0.0532	0.2213	-0.358
XGBoost	-0.0215	-0.0701	-0.2779
AdaBoost	0.0226	0.1036	0
CatBoost	0.0481	0.2044	-0.2567
LGBM	0.0554	0.2179	-0.1795

Deviation from Step 2 Pipeline Results

Step 3 - Pipeline 7

Step 1

- Used Step 2 – Pipeline 4 Preprocessing

Step 2

- Handled Class Imbalance using Near Miss Undersampling then SOMTE-NC Oversampling and Stratifying

Near Miss Undersampling then SMOTE-NC Oversampling and Stratifying Benefits

- Improved Generalization
- Reduced Overfitting
- Computationally Efficient
- Complementary Effects
- Handling Continuous and Categorical Features
- Maintain the distribution of classes across the train and test datasets

Step 3 - Pipeline 7 Results

Model	AUC	TPR	Predictive equality
Logistic Regression	0.8932	56.98%	81.77%
Random Forest	0.921	65.50%	94.18%
Neural Network	0.9451	75.70%	65.07%
XGBoost	0.9018	58.57%	67.81%
AdaBoost	0.9285	69.49%	100.00%
CatBoost(after tuning again)	0.9546	79.69%	55.26%
LGBM	0.9489	77.70%	56.87%

Pipeline Results

Model	AUC	TPR	Predictive equality
Logistic Regression	0.0286	0.1101	-0.0832
Random Forest	0.0486	0.1745	-0.0559
Neural Network	0.0611	0.2452	-0.2869
XGBoost	0.0155	0.0635	-0.1432
AdaBoost	0.0411	0.1668	0
CatBoost	0.0613	0.2483	-0.312
LGBM	0.07	0.2791	-0.2365

Deviation from Step 2 Pipeline Results

Step 4

Applying Our Best Approach to the Different Variants of the Dataset.

Experiments Results: Step 4

- Since the last experiment with NearMiss for undersampling followed by stratifying while creating the train and test sets then apply SMOTE-NC on the train data get the highest results. we use the same technique with the preprocessing done in experiment 7 in step 3 on all the variants of the dataset and here are the results.

Dataset	AUC	TPR	Predictive equality
Base AdaBoost	0.9346	72.08%	100.00%
Variant 1 AdaBoost	0.9324	69.63%	100.00%
Variant 2 AdaBoost	0.9378	71.21%	100.00%
Variant 3 AdaBoost	0.9329	70.67%	100.00%
Variant 4 AdaBoost	0.9383	72.21%	100.00%
Variant 5 AdaBoost	0.931	69.58%	100.00%

Dataset	AUC	TPR	Predictive equality
CatBoost Base	0.9535	80.37%	58.40%
CatBoost Variant 1	0.9538	77.88%	98.03%
CatBoost Variant 2	0.9549	79.37%	55.36%
CatBoost Variant 3	0.9534	78.47%	92.92%
CatBoost Variant 4	0.9561	80.05%	63.03%
CatBoost Variant 5	0.952	78.15%	87.26%

If we’re concerned about the fairness and final output we could use AdaBoost

If we’re concerned only about the fraud detection then we could use CatBoost

Discussion & Conclusion

Discussion

- **Limitations**
  - Slow training time prevented the usage of cross validation.
  - Using Stratified splitting.
  - Apply insufficient number of hyperparameter combinations in hyperparameter tuning process.
  - Although we mentioned in the proposal that we will use SVM algorithm, but we don’t use it since the kernel trick needs high resources.

Conclusion

1. Class Imbalance Matters: Addressing class imbalance is crucial for better model performance, fairness, and generalization in fraud detection.

2. Fairness Consideration: We evaluated fairness using Predictive Equality. Ensemble models like BalancedRandomForestClassifier and AdaBoost achieved good fairness results.

3. Fraud Detection Performance: CatBoost consistently performed well in detecting fraud, making it a strong candidate for fraud detection applications.

4. Ensemble Models Shine: Models from the imblearn library automatically handle class imbalance and improve both performance and fairness.

5. Model Variants: AdaBoost variants showed similar fairness results. CatBoost variants had varying fairness and fraud detection performance.

6. Data Preprocessing Matters: Proper data preprocessing, including handling missing values and scaling, contributed to better model results.

7. Choose Wisely: The final model choice depends on your priorities—prioritize fairness.

Work Load

Classes		
	Data Cleaning	Hassan Ahmed
	Data Preprocessing	Amr Sayed
	Modeling	Bilal Morsy
	NN Modeling	Omar Amer
Step 0		
	EDA	Bilal Morsy
	Comparing Variance Datasets	Amr Sayed
Step 1		
	LGBM & Random Forest	Amr Sayed
	XGBoost and AdaBoost	Hassan Ahmed
	Logistic Regression and Neural Networks	Omar Amer
	CatBoost	Bilal Morsy
Step 2		
	Pipeline 1	Amr Sayed
	Pipeline 2	Bilal Morsy
	Pipeline 3	Omar Amer
	Pipeline 4	Hassan Ahmed
Step 3		
	Under Sampling	Amr Sayed
	Over Sampling	Amr Sayed
	Under Sampling then Over Sampling	Bilal Morsy
	SMOTE - NC	Bilal Morsy
	Imblearn ensemble	Omar Amer
	Near miss and SMOTE - NC	Hassan Ahmed
	Near miss and SMOTE – NC and Stratifying	Omar Amer
Step 4		
	(CatBoost and AdaBoost and comparison)	Hassan Ahmed
Readme		
	Readme	Amr Sayed & Hassan Ahmed

Thank You