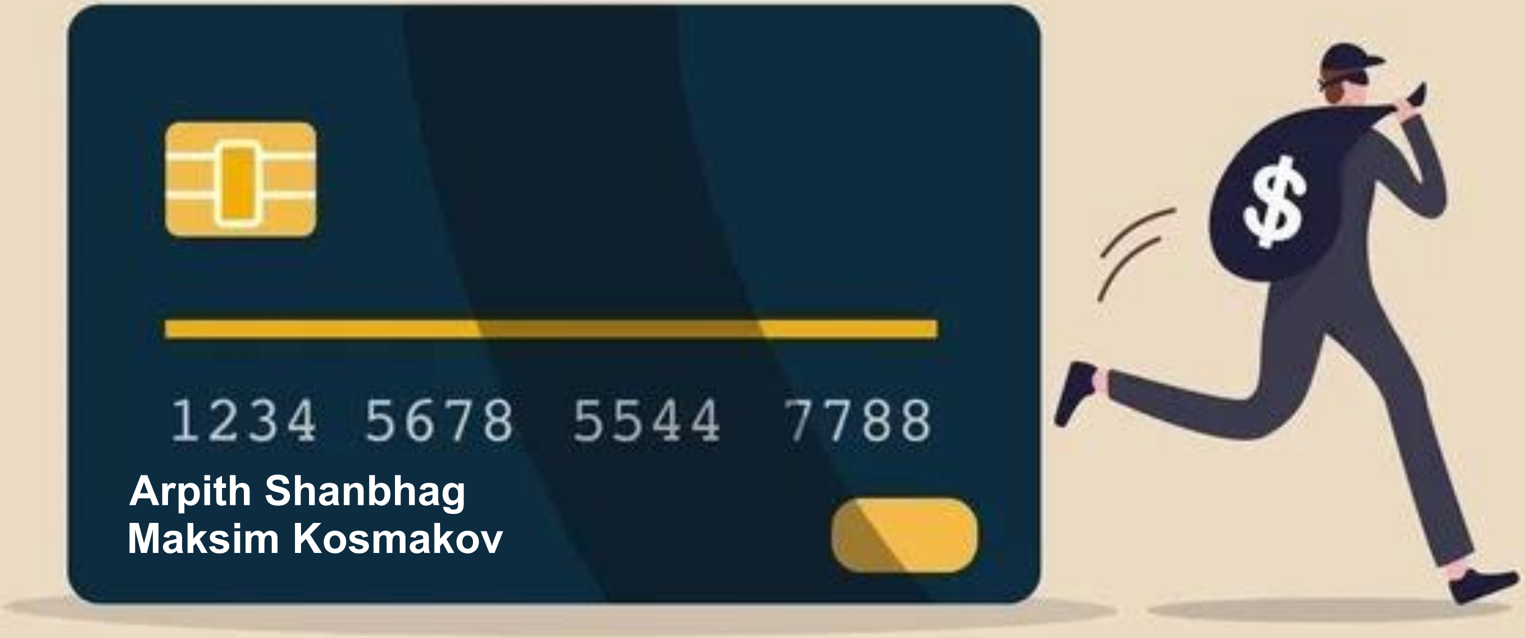
Detecting Bank Fraud



Arpith Shanbhag
Maksim Kosmakov

Project description and Stakeholders

We aim to develop a machine learning model to accurately detect fraudulent bank account opening activity using the NeurIPS 2022 dataset, which is highly imbalanced and contains anonymized customer features.

This project is valuable to multiple stakeholders, including:

- **Financial Institutions & Banks:** To improve fraud detection and reduce financial losses.
- **Credit Card Companies**: To prevent fraudulent transactions and enhance security.
- **Consumers:** To protect customers from unauthorized transactions and fraudulent charges.
- **Regulatory Agencies:** To ensure compliance with financial fraud prevention standards.

Data

- The Bank Account Fraud (BAF) suite of datasets has been published at **NeurIPS 2022.**

- Each instance in the suite of datasets represents a synthetic, feature-engineered bank account opening application in tabular format. These were generated using a **CTGAN** trained on a **real-world anonymized dataset** for bank account opening fraud.

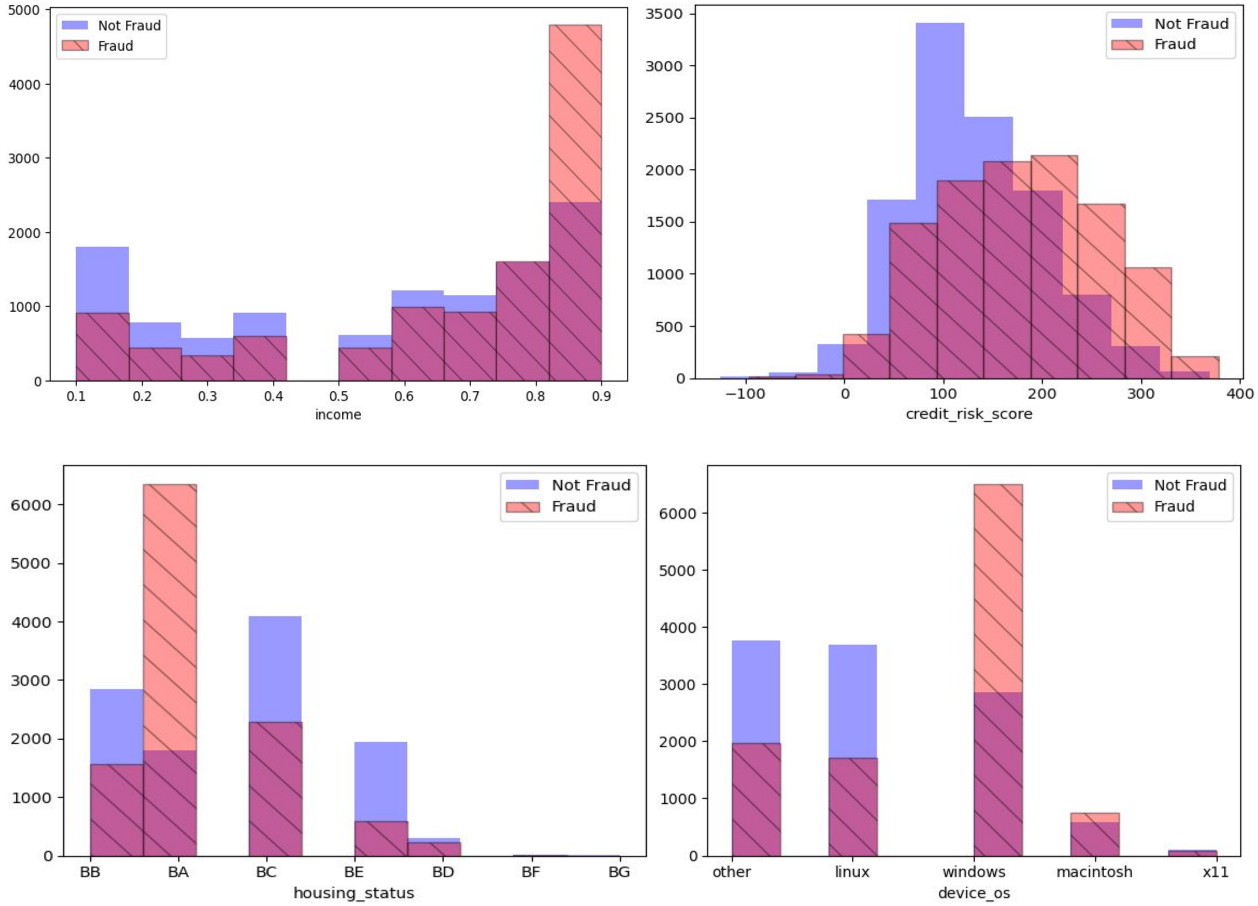
- The BAF dataset contains **1 million** synthetic bank account opening applications. Each row is an application, labeled by is_fraud (1=fraud, 0=legitimate)

- It includes **30 realistic features** (e.g., customer age, personal income, employment status, credit history metrics) plus a month column (time of application).

- Highly unbalanced dataset (1% of fraud cases).

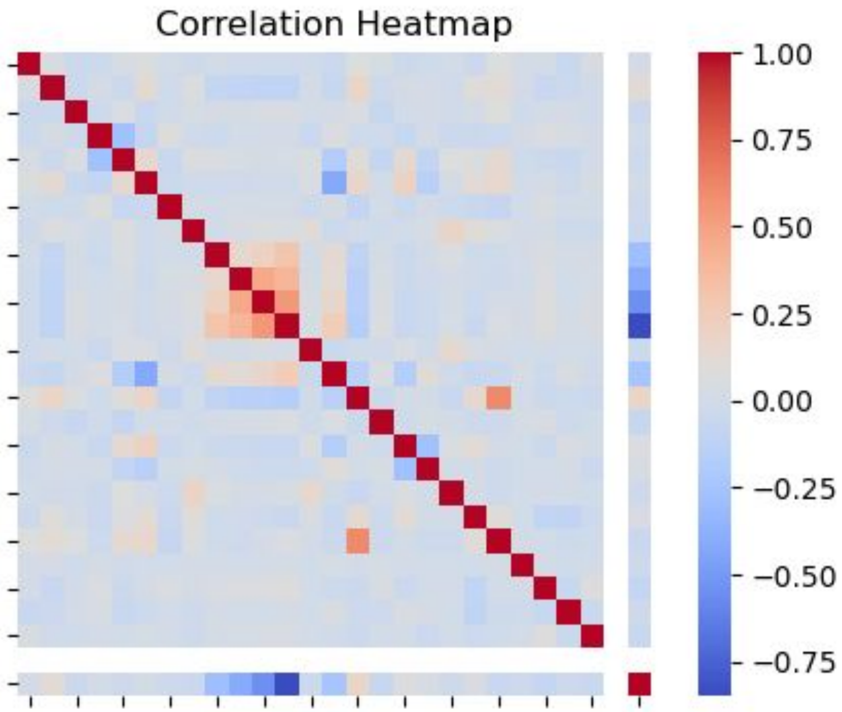
EDA

Feature distribution:



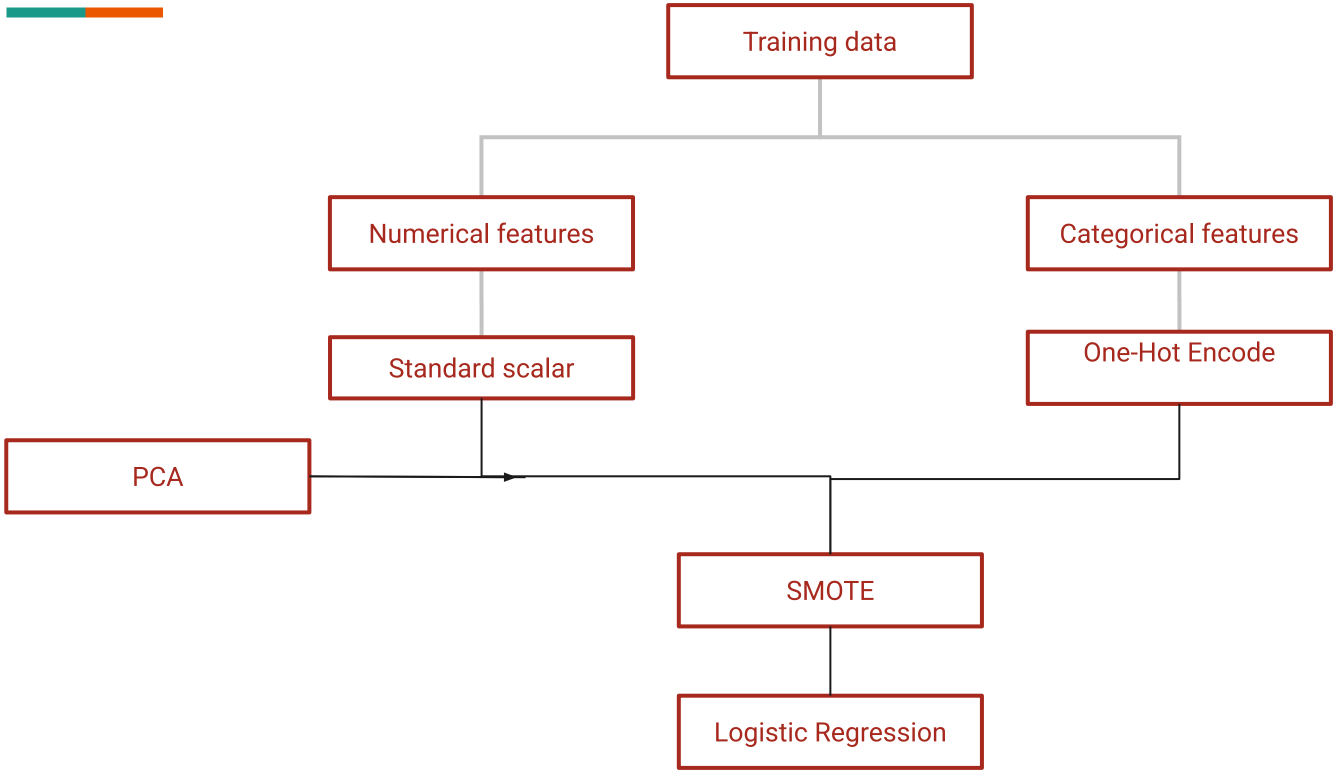
EDA

Correlations:

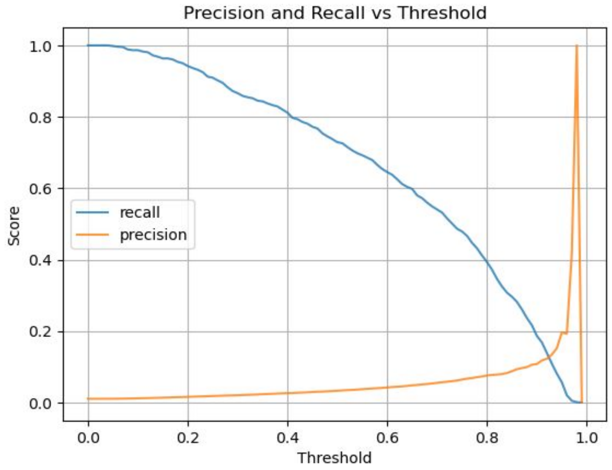


- Low correlations between most features
- Some correlation between credit score and proposed credit limit

Logistic regression pipeline for the selected columns

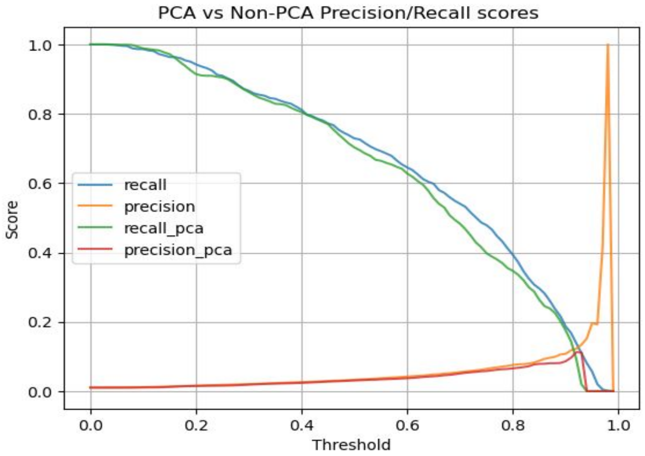


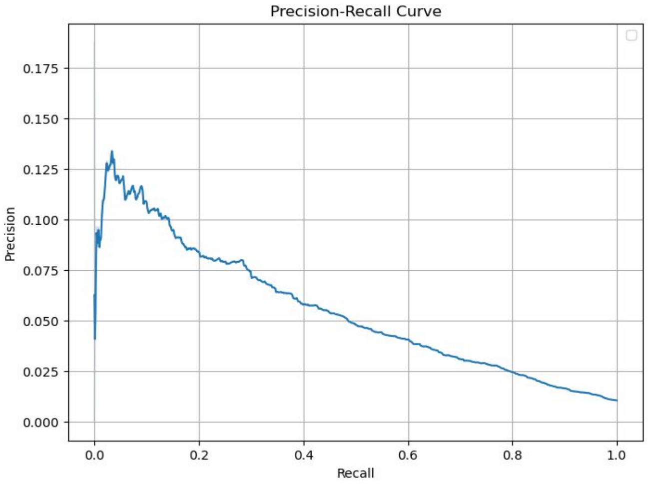
Logistic Regression



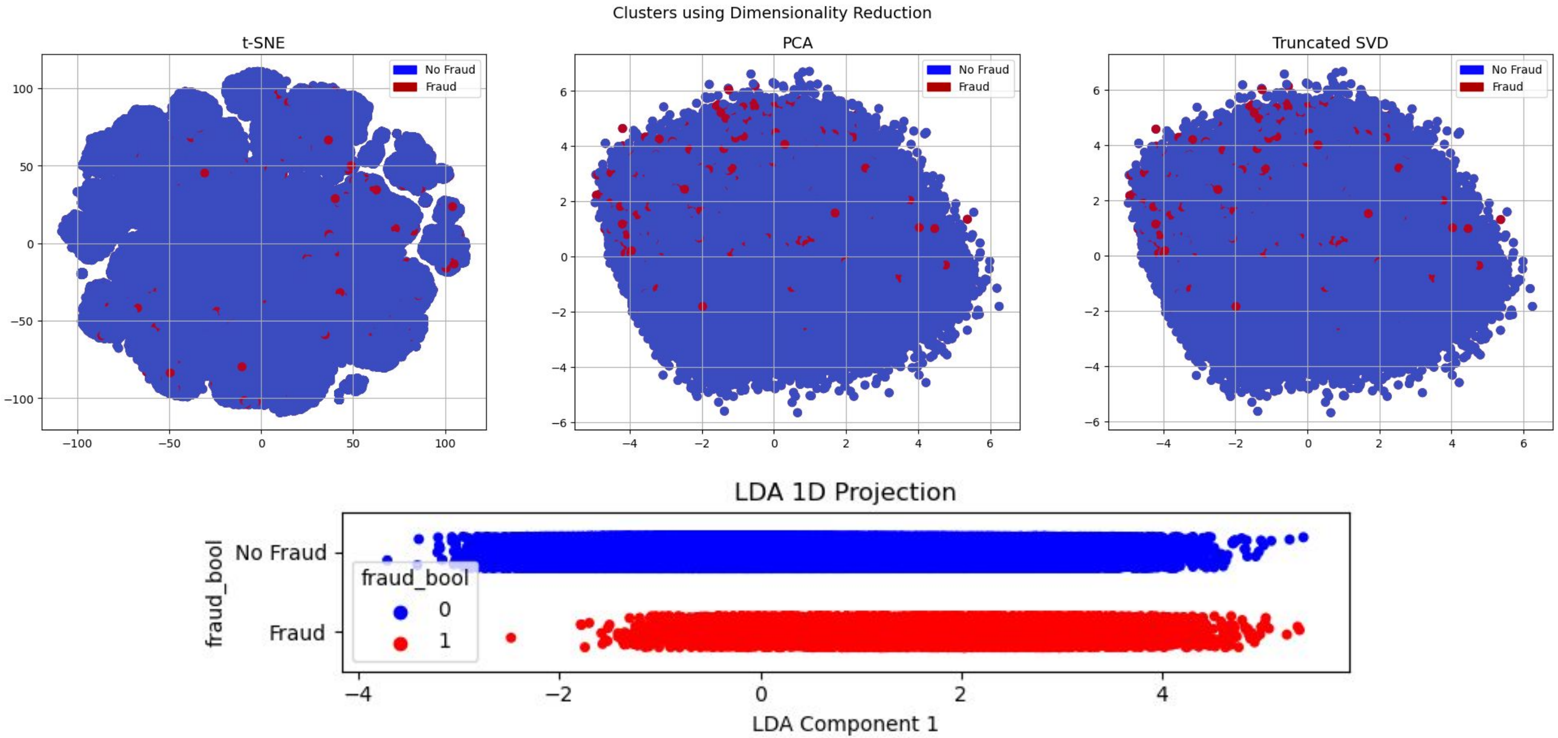
- Logistic regression with numerical features like income, age, credit risk and categorical features like housing status and device OS

- At **0.5 threshold,** the model has a recall of around **75%**, but the precision is around **3%**





Dimensionality reduction

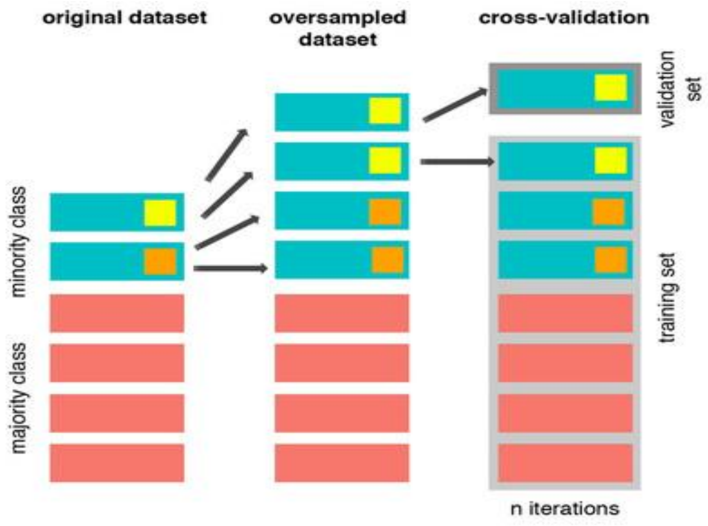


Undersampling and model selection

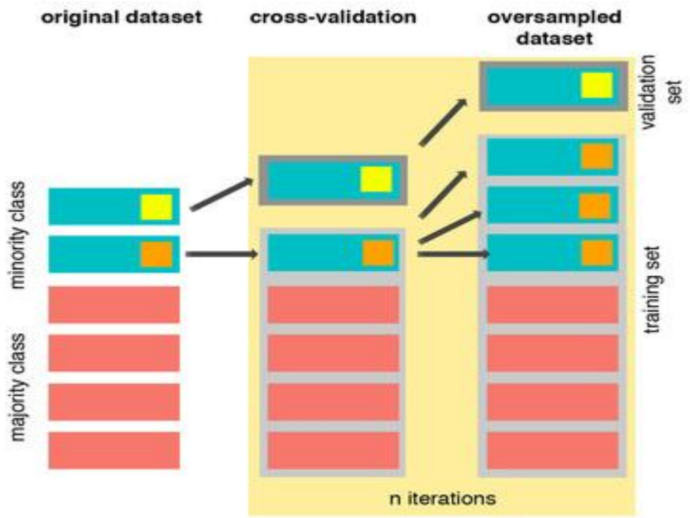
Model	Recall	Precision	f1	pr_auc
LogisticRegression (C=0.001, solver='liblinear')	0.793	**0.040**	**0.076**	**0.134**
KNeighborsClassifier (n_neighbors=3)	0.670	0.030	0.056	0.028
DecisionTreeClassifier (criterion='entropy', max_depth=3, min_samples_leaf=5)	**0.863**	0.019	0.04	0.03
Hybrid model (threshold= 0.89)	0.24	0.19	**0.210**	0.134

Oversampling using SMOTE

Wrong way !



Right way



Oversampling (results)

Model	Recall	Precision	f1	pr_auc
LogisticRegression	**0.628**	0.047	**0.088**	**0.094**
KNeighborsClassifier	0.342	0.048	0.084	0.030
DecisionTreeClassifier	0.133	**0.061**	0.084	0.018

Conclusion

- If you want a model that catches **as many frauds as possible**, take the one with the **highest recall (0.863)**
  DecisionTreeClassifier trained on undersamped data

- If you want a model that makes **as few false fraud predictions as possible**,
  take the one with the **highest precision (0.06)**
  DecisionTreeClassifier trained on oversampled data

- If you want a model that **balances recall and precision**, take the one with the **highest F1 score (0.088)**
  Logistic regression with a threshold= 0.89 trained on oversampled data

- If you want a model that performs well across **different thresholds**,
  take the one with the highest **PR AUC (0.134)**
  Logistic regression trained on undersampled data