Capstone Project Presentation
17/1/2025

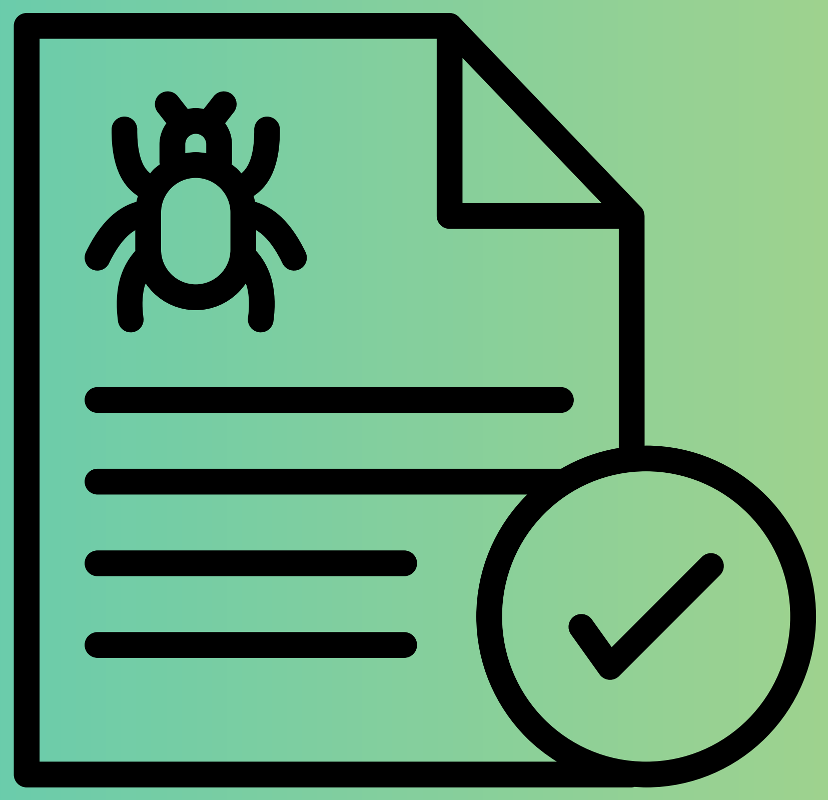
Shaping a Secure Digitalize Future: Predicting and Preventing Bank Fraud

By:
Michael Map (I.O.D Student - Data Science & A.I)

Bank Fraud intro:

Banking institutions face increasing challenges due to fraudulent activities that result in significant financial losses, erosion of customer trust, and reputational damage.

Fraudulent transactions can include unauthorized account access, money laundering, fake identities, and other deceptive practices. Detecting such activities in real-time or before significant damage occurs is paramount for financial institutions.



‘Tackling’ of Bank Fraud

How may ***predictive analytics*** be used
to **prevent** and **detect** bank fraud?

Leveraging Machine Learning to Secure Financial Transactions

Using **historical data** analytics and **user behaviors**(anyone opened up a bank a/c), **predictive models** discern patterns indicative of fraud.

For eg. **a model trained** on datasets marked with fraud instances **learns to recognize and flag** similar patterns in **new incidents**, whether they be in call-logs, **frequencies** or **unusual** user behaviors.

Agenda

- 1. Problem Statement
- 2. Dataset Overview
- 3. Methodology
- 4. Data Preprocessing
- 5. Model Development
- 6. Performance Evaluation
- 7. Business Recommendations
- 8. Data Limitations
- 9. Conclusion

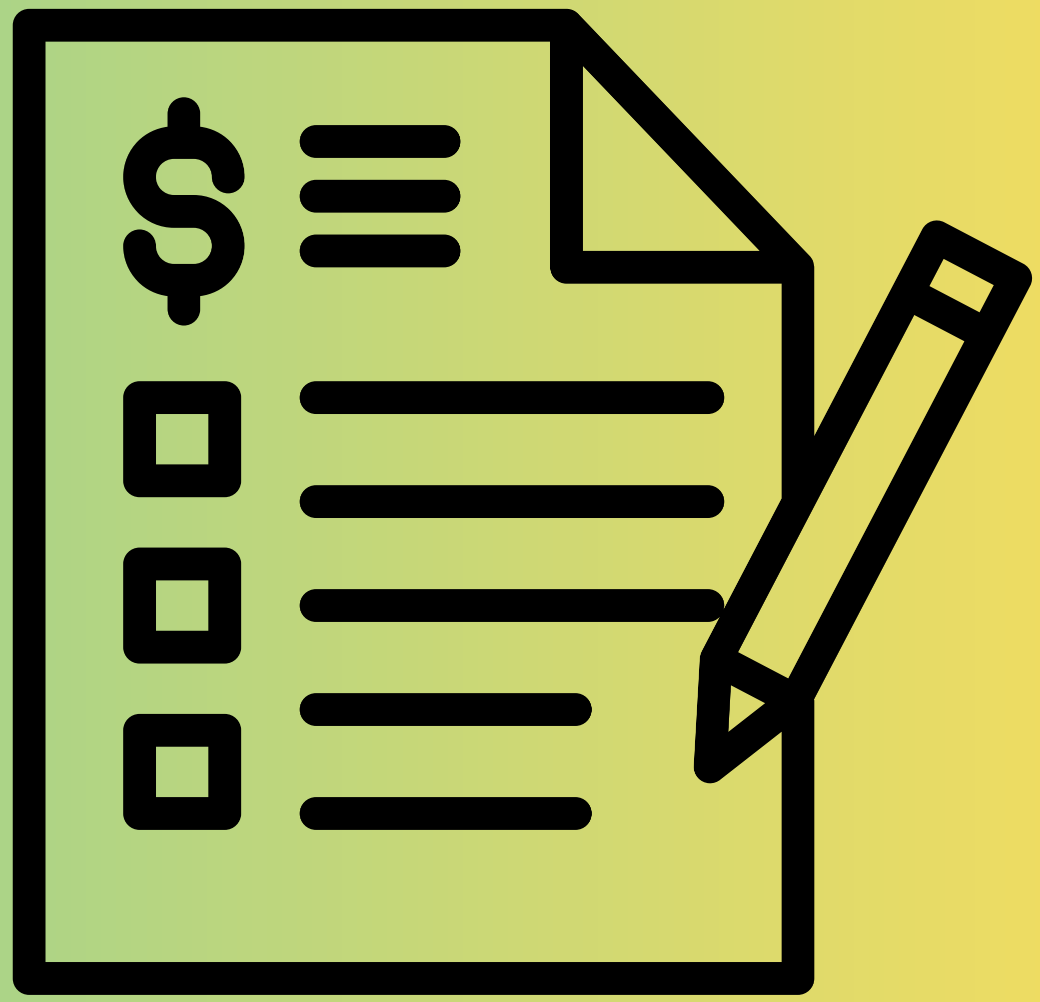
Problem Statement

- Fraudulent transactions significantly impact financial institutions.

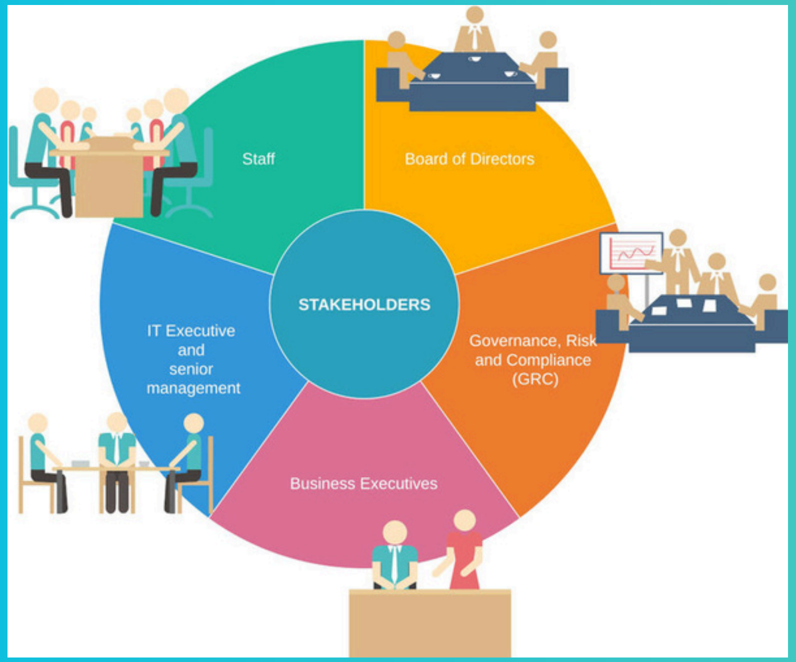
- Objectives:
- - Reduce false positives and negatives.
- - Enhance customer trust and security.

Challenges in Fraud Detection **Elements:**

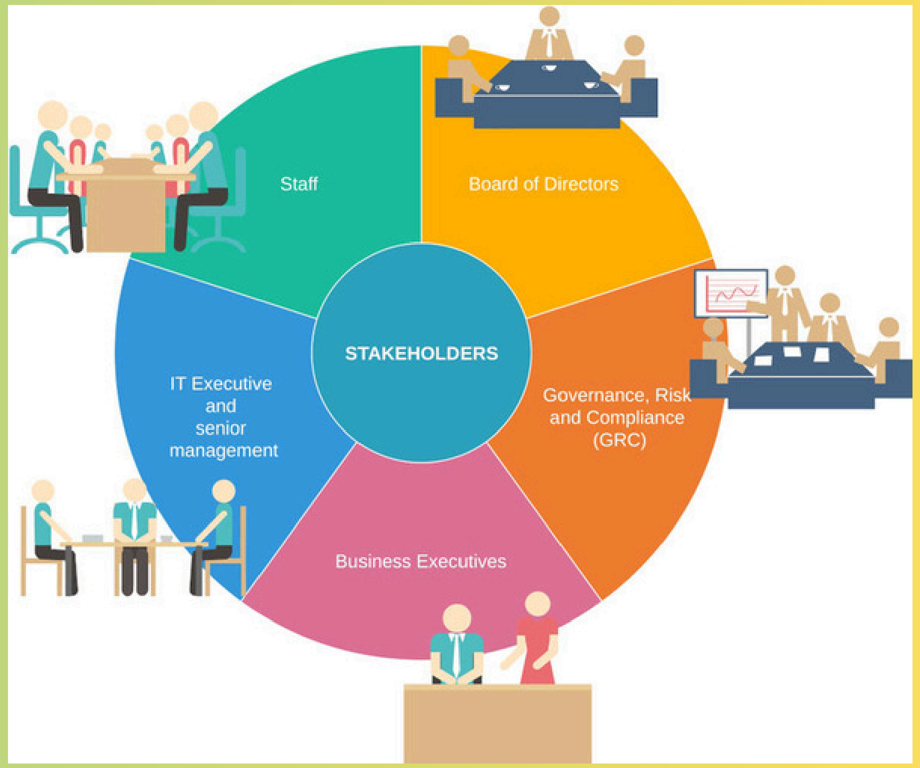
- High Volume of Transactions
- Sophisticated Fraud Tactics
- False Positives
- Imbalanced Data



These **elements** frame the problem in a way that highlights its technical, operational, and contextual complexities, which are essential for defining objectives and guiding solution development. Including these in the problem statement ensures a comprehensive understanding of the challenges at hand.



Business Stakeholders



Bank Management:
Decision-makers who prioritize fraud prevention strategies and allocate resources for system implementation.

Fraud Detection Teams:
Analysts and investigators responsible for monitoring flagged transactions and taking corrective actions.

Compliance and Risk Departments:
Ensure the system aligns with legal and regulatory requirements while mitigating risks associated with financial crimes.

Value Proposition

This project offers a reliable and efficient solution for identifying fraudulent transactions in banking systems.

Key benefits include:

Enhanced Security

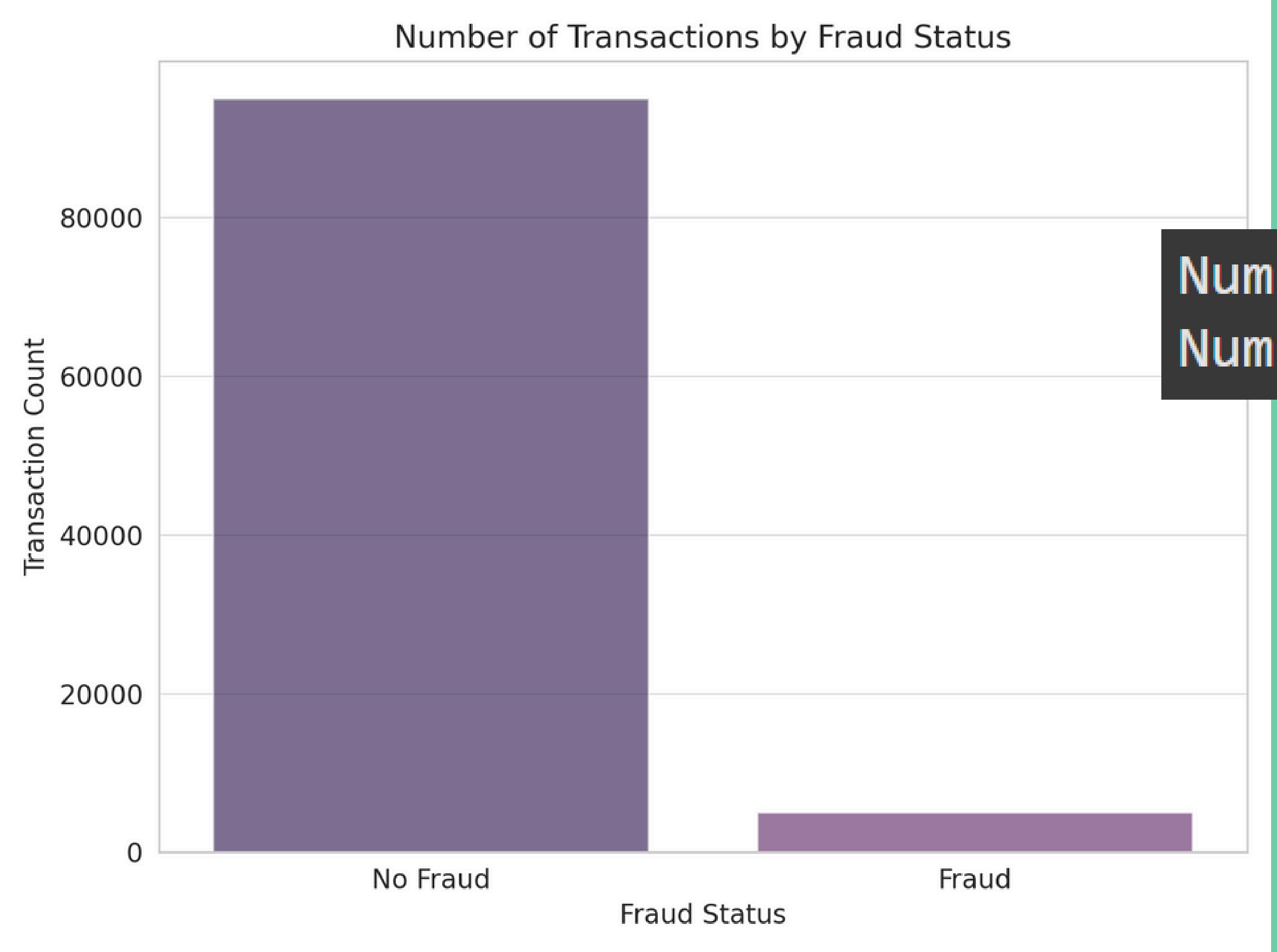
Scalability

Actionable Insights

Ease of Deployment

Dataset Overview

- Source: Kaggle Dataset – NeurIPS 2022
  https://www.kaggle.com/datasets/sgpjesus/bank-account-fraud-dataset-neurips-2022/data
- Key Features:
- - Transaction Amount
- - Customer Demographics
- - Transaction Time
- - Merchant Type
- Target Variable: **Fraudulent** (**Yes/No**)




```
Number of rows: 1000000
Number of columns: 32
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000000 entries, 0 to 999999
Data columns (total 32 columns):
 #   Column                            Non-Null Count    Dtype
---  ------                            --------------    -----
 0   fraud_bool                        1000000 non-null  int64
 1   income                            1000000 non-null  float64
 2   name_email_similarity             1000000 non-null  float64
 3   prev_address_months_count         1000000 non-null  int64
 4   current_address_months_count      1000000 non-null  int64
 5   customer_age                      1000000 non-null  int64
 6   days_since_request                1000000 non-null  float64
 7   intended_balcon_amount            1000000 non-null  float64
 8   payment_type                      1000000 non-null  object
 9   zip_count_4w                      1000000 non-null  int64
 10  velocity_6h                       1000000 non-null  float64
 11  velocity_24h                      1000000 non-null  float64
 12  velocity_4w                       1000000 non-null  float64
 13  bank_branch_count_8w              1000000 non-null  int64
 14  date_of_birth_distinct_emails_4w  1000000 non-null  int64
 15  employment_status                 1000000 non-null  object
 16  credit_risk_score                 1000000 non-null  int64
 17  email_is_free                     1000000 non-null  int64
 18  housing_status                    1000000 non-null  object
 19  phone_home_valid                  1000000 non-null  int64
 20  phone_mobile_valid                1000000 non-null  int64
 21  bank_months_count                 1000000 non-null  int64
 22  has_other_cards                   1000000 non-null  int64
 23  proposed_credit_limit             1000000 non-null  float64
 24  foreign_request                   1000000 non-null  int64
 25  source                            1000000 non-null  object
 26  session_length_in_minutes         1000000 non-null  float64
 27  device_os                         1000000 non-null  object
 28  keep_alive_session                1000000 non-null  int64
 29  device_distinct_emails_8w         1000000 non-null  int64
 30  device_fraud_count                1000000 non-null  int64
 31  month                             1000000 non-null  int64
dtypes: float64(9), int64(18), object(5)
memory usage: 244.1+ MB
```

Each instance is a synthetic feature-engineered bank account application with the following fields:

- **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 pass
Ranges between $[0, 79]$ days.
- **intended_balcon_amount** (numeric): Initial transf
Ranges between $[-16, 114]$ (negatives are missing valu
- **payment_type** (categorical): Credit payment plan ty
values.
- **zip_count_4w** (numeric): Number of applications withi
Ranges between $[1, 6830]$.
- **velocity_6h** (numeric): Velocity of total applications m
number of applications per hour in the last 6 hours. Ra
- **velocity_24h** (numeric): Velocity of total applications m
number of applications per hour in the last 24 hours. R
- **velocity_4w** (numeric): Velocity of total applications m
number of applications per hour in the last 4 weeks. R
- **bank_branch_count_8w** (numeric): Number of total a
branch in last 8 weeks. Ranges between $[0, 2404]$.
- **date_of_birth_distinct_emails_4w** (numeric): Numb
same date of birth in last 4 weeks. Ranges between $[0,$

- **employment_status** (categorical): Employment status of the applicant. 7 possible (annonymized) 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** (categorical): Current residential status for applicant. 7 possible (annonymized) 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).

- **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** (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** (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 $[0, 7]$.
- **fraud_bool** (binary): If the application is fraudulent or not.

Methodology

Workflow:
1. Data Exploration
2. Feature Engineering
3. Model Training
4. Evaluation and Optimization

Algorithms:
- Random Forest Classifier
- LightGBM

Tools: Python (Scikit-learn, LightGBM)

Data Import

Step 1: Read the Data

Exploratory Data Analysis of Bank Accounts Application

Step 2.1: Explore and Clean the Data(where applicable)

Number of Transactions by Fraud Status

Step 2.2: Prepare the Data

Missing Values of Features by Fraud Status (Crucial)

Distribution and Outliers of Features by Fraud Status

Feature Engineering: Fraud Detection of Bank Account Applications

Train-Test Split

Step 3.1: Split the Data

Data Transformation

COMPARISON OF ENCODERS

Step 4.1: Min-Max Scaling for Numerical Features

Step 4.2: Pearson Correlation Test for Multicollinearity

Step 4.3: Label Encoding

Step 4.4: Resampling of Imbalanced Dataset

Modelling ~

Step 5.1: Define each of a Model

Step 5.2: Fit each of a Model

Evaluation ~

Data Preprocessing

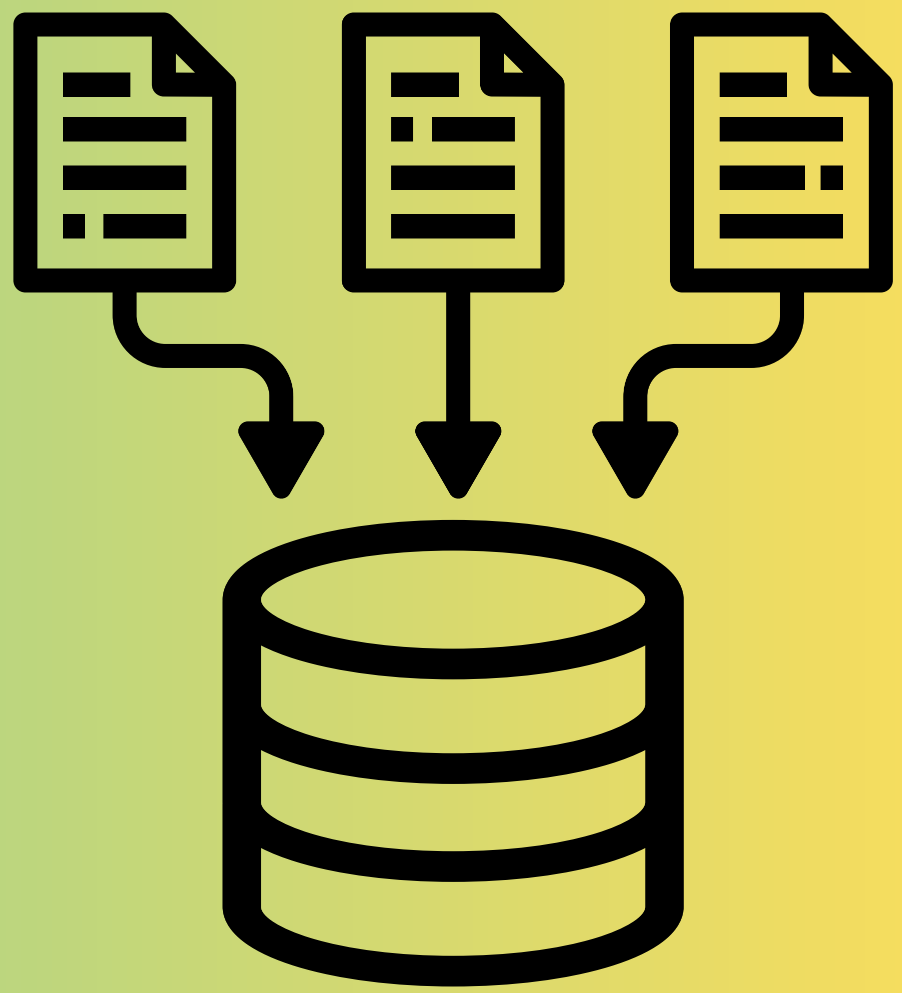
Steps:

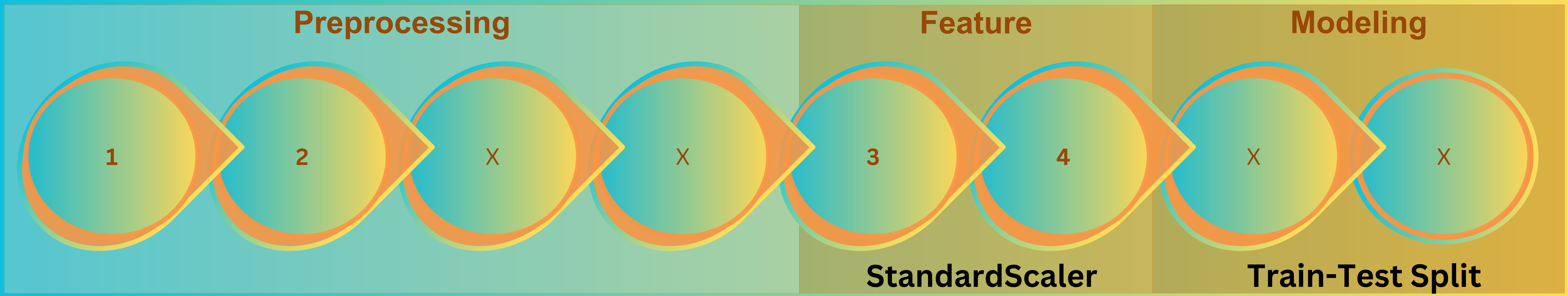
1. Handling Missing Values

2. Encoding Categorical Variables

3. Feature Scaling using StandardScaler

4. Train-Test Split (80-20)





Model Developments

- **Algorithms:**
- - Random Forest Classifier **(RF)**
-
- - LightGBM **(LGB)**

- **Metrics:**
- - Accuracy
- - Precision, Recall, F1-Score
- - ROC-AUC

- **Classification Report:**
- Base on Test Modellings

```
[ ]  rf_clf.score(X_test_scaled, y_test)

     0.986415

[ ]  print(metrics.classification_report(y_test, y_pred))

                   precision    recall  f1-score   support

                0       0.99      1.00      0.99    197891
                1       0.19      0.09      0.12      2109

         accuracy                           0.99    200000
        macro avg       0.59      0.54      0.56    200000
     weighted avg       0.98      0.99      0.98    200000

[ ]  roc_auc_score(y_test, rf_clf.predict_proba(X_test_scaled)[:, 1])

     0.8482447168310652
```

```
[ ]  lgb_clf.score(X_test_scaled, y_test)

     0.98747

     print(metrics.classification_report(y_test, y_pred))

                   precision    recall  f1-score   support

                0       0.99      1.00      0.99    197891
                1       0.25      0.10      0.14      2109

         accuracy                           0.99    200000
        macro avg       0.62      0.55      0.57    200000
     weighted avg       0.98      0.99      0.98    200000

[ ]  roc_auc_score(y_test, lgb_clf.predict_proba(X_test_scaled)[:, 1])

     0.8738768929073056
```

Performance Evaluation

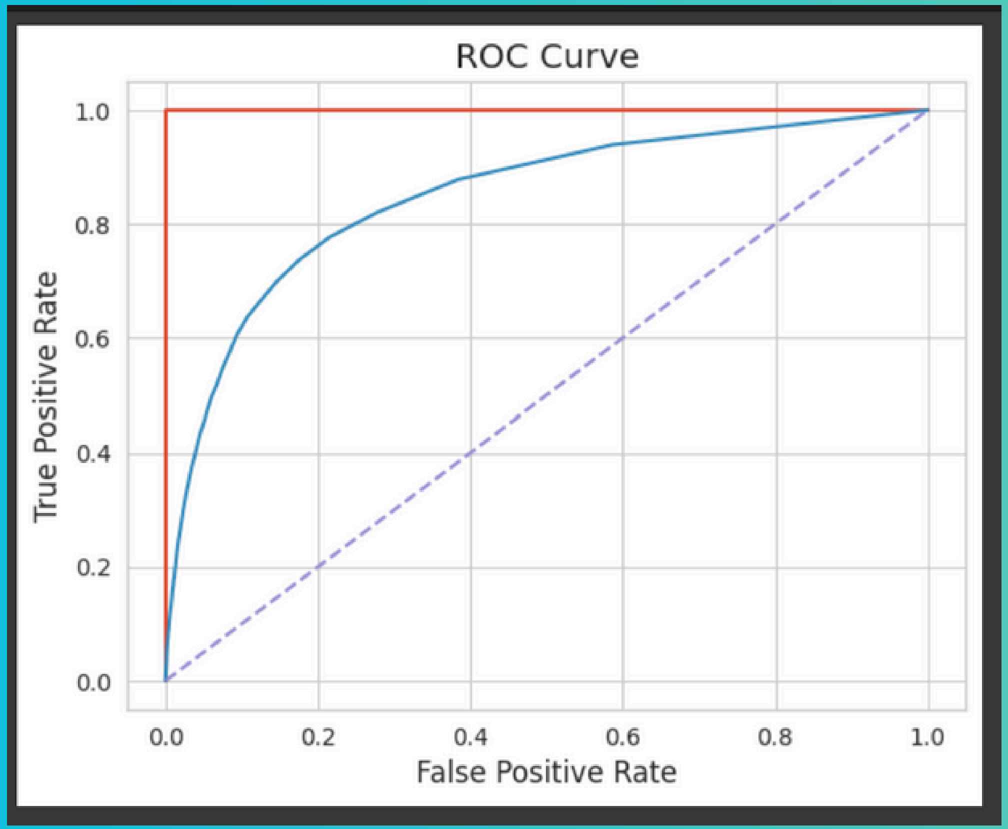
- *Confusion Matrix:*
- - Highlight false positives and negatives >>>
-
- Accuracy Scoring >>> **RandomForest** VS **LightGBM**

6B1. Random Forest

```
[ ]  y_pred = rf_clf.predict(X_test_scaled)
     confusion_matrix(y_test, y_pred)

     array([[197095,    796],
            [  1921,    188]])

[ ]  rf_clf.score(X_test_scaled, y_test)

     0.986415
```

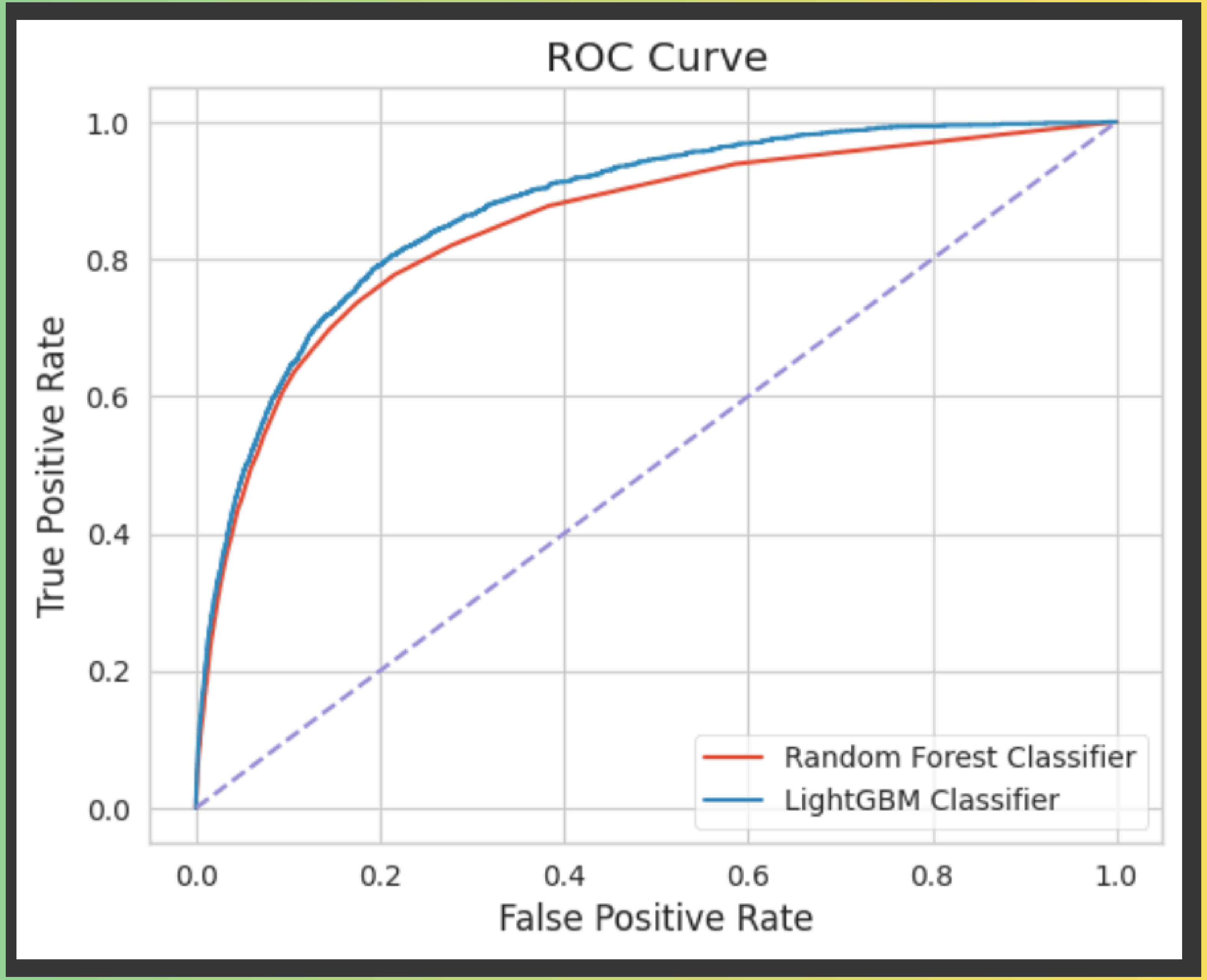
6B2. LightGBM

```
[ ]  y_pred = lgb_clf.predict(X_test_scaled)
     confusion_matrix(y_test, y_pred)

     array([[197290,    601],
            [  1905,    204]])

[ ]  lgb_clf.score(X_test_scaled, y_test)

     0.98747
```

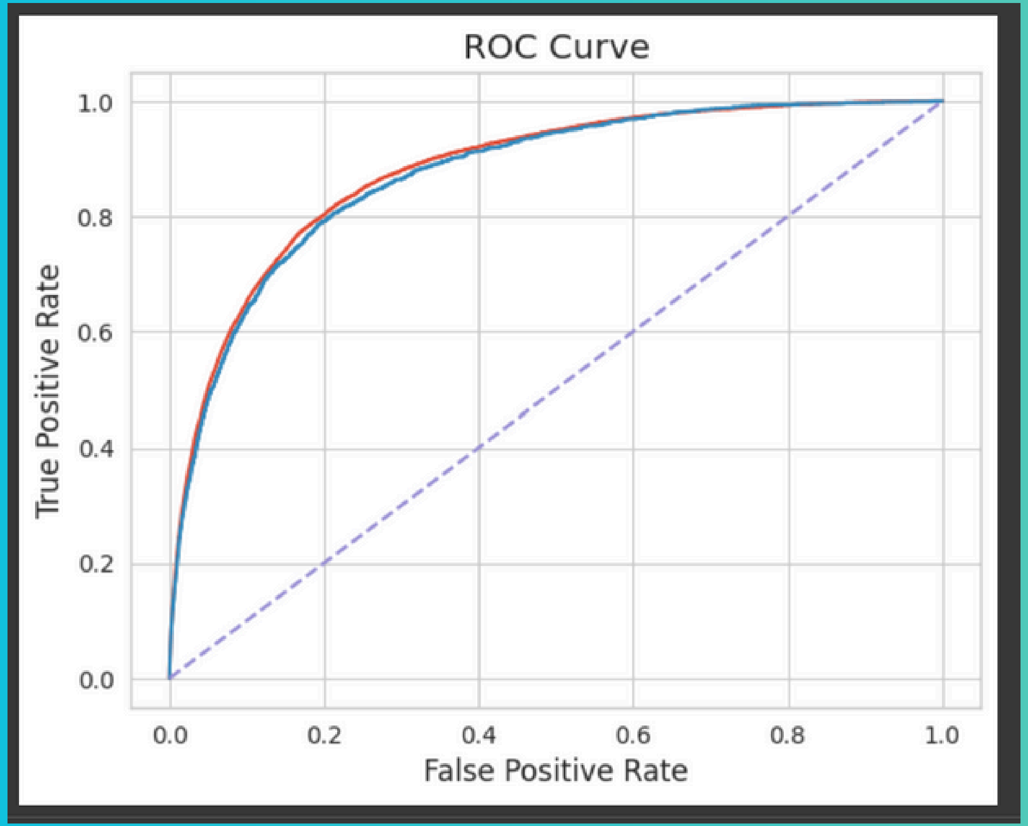


Key Results **RandomForest** Classifier model:

- **Accuracy: [0.99%] Rounded up 0.986%**
- **ROC-AUC: [0.85%] Rounded up 0.848%**



VS



Key Results **LightGBM** Classifier model:

- **Accuracy: [0.99%] Rounded up 0.987%**
- **ROC-AUC: [0.87%] Rounded up 0.873%**

The **performance metrics** as follow are base on `Macro Average`:

Test Models	ROC-AUC	Precision	Recall	F1-Score	Accuracy
Random Forest	`0.8482`	0.59	0.54	0.56	0.986
LightGBM	`0.8738`	0.62	0.55	0.57	0.987

Business Recommendations

- **Fraud Prevention Measures:**
- - Implement real-time fraud detection systems.
- - Focus on high-risk transaction patterns.

- **Customer Impact:**
- - Minimize disruptions for genuine customers.
- - Increase trust in banking services.

Adopt LightGBM:

Use LightGBM as the core model due to its superior performance in distinguishing fraudulent transactions (ROC-AUC: 0.8738).

Optimize Decision Thresholds:

Adjust thresholds to balance precision and recall based on the bank’s priorities (e.g., higher recall for fraud prevention).

Monitor Key Features:
Focus on **important predictors** (**transaction time, amount, merchant type**) to design targeted fraud detection rules.

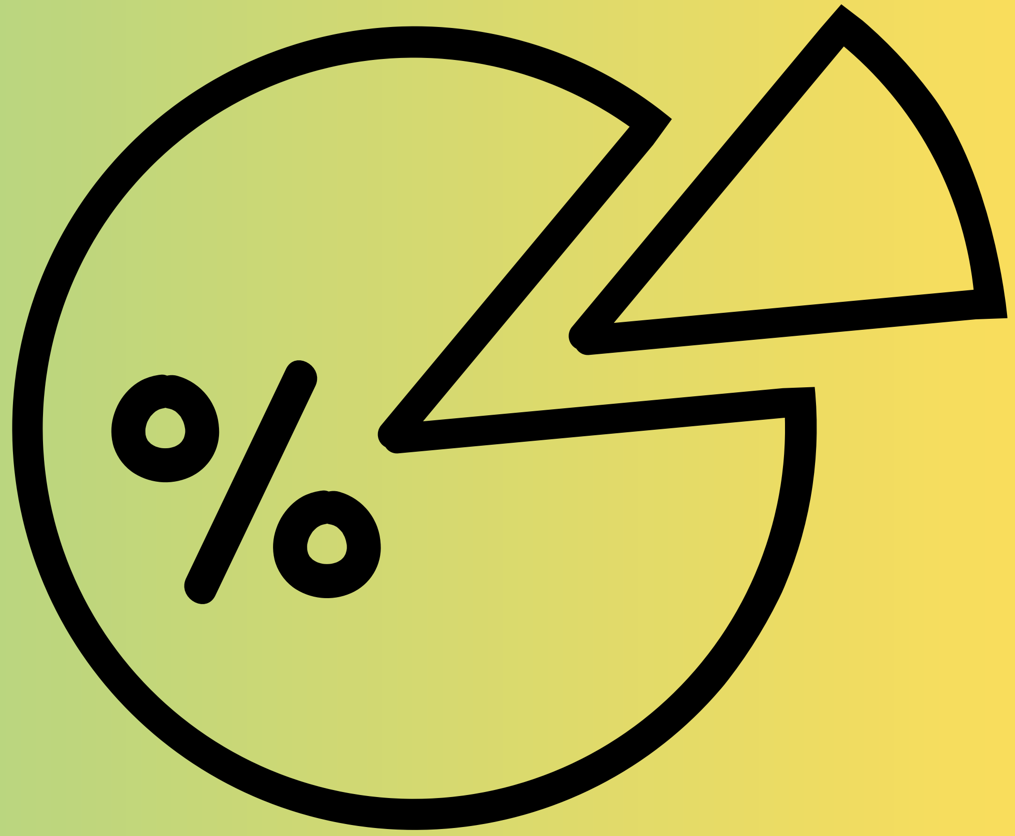
Use Explainability Tools:
Incorporate tools like **SHAP**(SHapley Additive exPlanations) for transparent fraud detection insights, **improving stakeholder trusts.**

Establish Evaluation Dashboards:
Track metrics like ROC-AUC, precision, and recall in real-time to ensure model effectiveness.

These steps will not only improve fraud detection accuracy, reduce financial losses, but as well as enhance customer trusts!

Data Limitations

- **Imbalanced** Dataset
- Limited Feature Diversity
- Lack of Temporal Context
- **Absence** of External Data
- Feature Granularity



Suggestions for Additional Data

- **Behavioral** Data
- **Historical** Data
- **Geographic** and **Demographic** Data
- Temporal Features
- External Risk Indicators
- Social and Economic Data

Conclusion

- Machine learning effectively detects fraudulent transactions.
- Strategic deployment can significantly reduce financial losses.

Implementing a fraud detection system using the LightGBM model provides significant advantages for identifying and preventing fraudulent transactions in real-time.

The model's superior performance metrics, particularly its high ROC-AUC score (0.8738), making it an excellent choice! for balancing precision and recall in fraud detection.

This approach not only **reduces financial losses** but also **strengthens customer confidence!** in the **bank's ability** to **safeguard** their assets.

Q&A

Thank You!

Questions and Feedback Welcome!