Mule Account Detection

1.1 Dataset

- The Bank Account Fraud (BAF) suite of datasets has been published at ***NeurIPS 2022*** ***(2022 Conference on Neural Information Processing Systems held in New Orleans, Louisiana)***.
- It comprises a total of 6 different synthetic bank account fraud tabular datasets.
- BAF is a ***realistic***, complete, and robust test bed to evaluate novel and existing methods in ML and fair ML, and the first of its kind!

1.2 About the dataset

- Evaluating new techniques on realistic datasets plays a crucial role in the development of ML research and its broader adoption.
- In recent years, there has been a significant increase of publicly available unstructured data resources for NLP tasks.
- However, tabular data — which is prevalent in many high-stakes domains — has been lagging behind. To bridge this gap, ***Bank Account Fraud (BAF), the first publicly available privacy-preserving, large-scale, realistic suite*** of tabular datasets was presented.
- The suite was ***generated by applying state-of-the-art tabular data generation techniques on an anonymized, real-world bank account opening fraud detection dataset***.
- This setting carries a set of challenges that are commonplace in real-world applications, including temporal dynamics and significant class imbalance.
- Additionally, to allow practitioners to stress test both performance and fairness of ML methods, each dataset variant of BAF contains specific types of data bias.

1.3 About the dataset

<u>This suite of datasets is:</u>

- ***Realistic***, based on a present-day real-world dataset for fraud detection.
- ***Biased***, each dataset has distinct controlled types of bias.
- ***Imbalanced***, this setting presents a extremely low prevalence of positive class.
- ***Dynamic***, with temporal data and observed distribution shifts.
- ***Privacy preserving***, to protect the identity of potential applicants we have applied differential privacy techniques (noise addition), feature encoding and trained a generative model (CTGAN).

1.4 BAF suite variants

- Set of 6 challenges for the Machine Learning methods.

Dataset	Description
Base	Sampled to best represent original dataset.
Variant I	Has higher group size disparity than base.
Variant II	Has higher prevalence disparity than base.
Variant III	Has better separability for one of the groups.
Variant IV	Has higher prevalence disparity in train.
Variant V	Has better separability in train for one of the groups.

Each variant provides a unique, realistic challenge for performance, fairness, and robustness of ML methods.

1.5 BAF suite generation

How was BAF generated?

Feature Selection
Features were selected based on 5 LightGBM models feature importance.
Used features are composed mostly of aggregations (PII data was not included).

Noise Mechanisms
We added Laplacian noise to the data before the generative process.
Categorical features were changed according to the prior distribution.
Additionally, features of Applicant Age and Income were binned.

Generative Model
We used a GAN architecture adapted to the tabular dataset domain (CTGAN).
A total of 70 GANs were tested, with random sampling of hyperparameters.

Filtering and Transformations
Generated datasets were sampled to not contain repeated instances w.r.t. the original dataset.
Transformations were applied to maintain original observed behaviours (*e.g.* number of decimal places).

2.1 Features of the dataset (1)

Each dataset is composed of:

- 1 million (***10 lakh***) accounts.
- ***30 realistic features*** used in the fraud detection use-case.
- ***Protected attributes***: age group, employment status and income.

2.2 Realistic features

- 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 (annonymized) 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].

2.3 Realistic features

- 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 (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.

2.4 Realistic features

- 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 (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 account is fraudulent or not.

3.1 Data pre-processing

1. Analysing fraud bools:

- Non-Frauds : 988971
- Frauds : 11029

2. Split data into features and target:

- Features : All columns except fraud_bool
- Target : fraud_bool

3. One-hot encode categorical columns:

- Vectorize the categorical columns.

4. Using standard scaler :

- Ensures that features with different units or scales are standardized, preventing any one feature from dominating the learning process due to its scale.

4.1

Scenario - 1

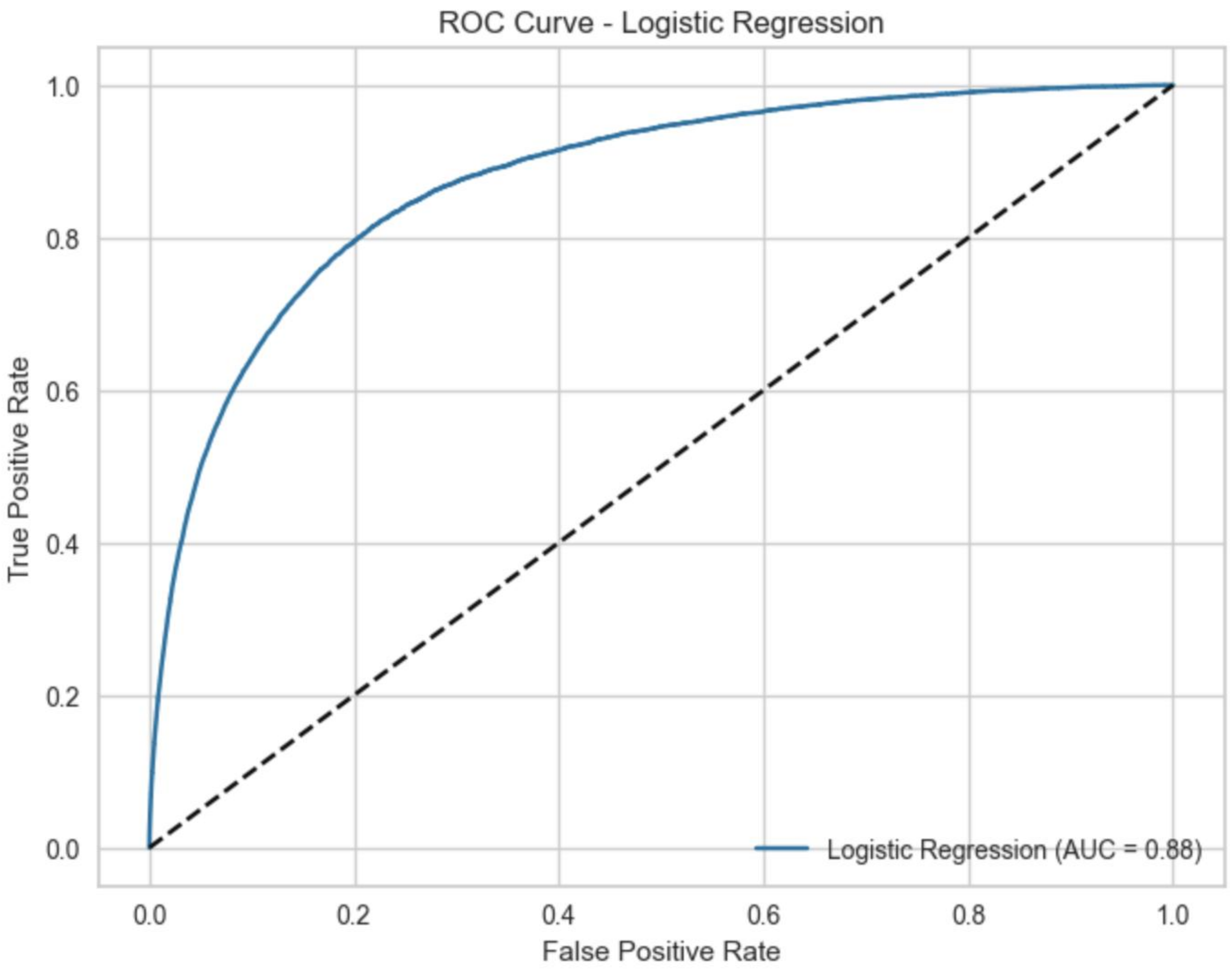
Training data: Account numbers 1-10,00,000 (complete dataset)

Testing data: Account numbers 1-10,00,000 (same complete dataset)

- Testing data is ***seen*** by the models.

4.2 Training using ML models

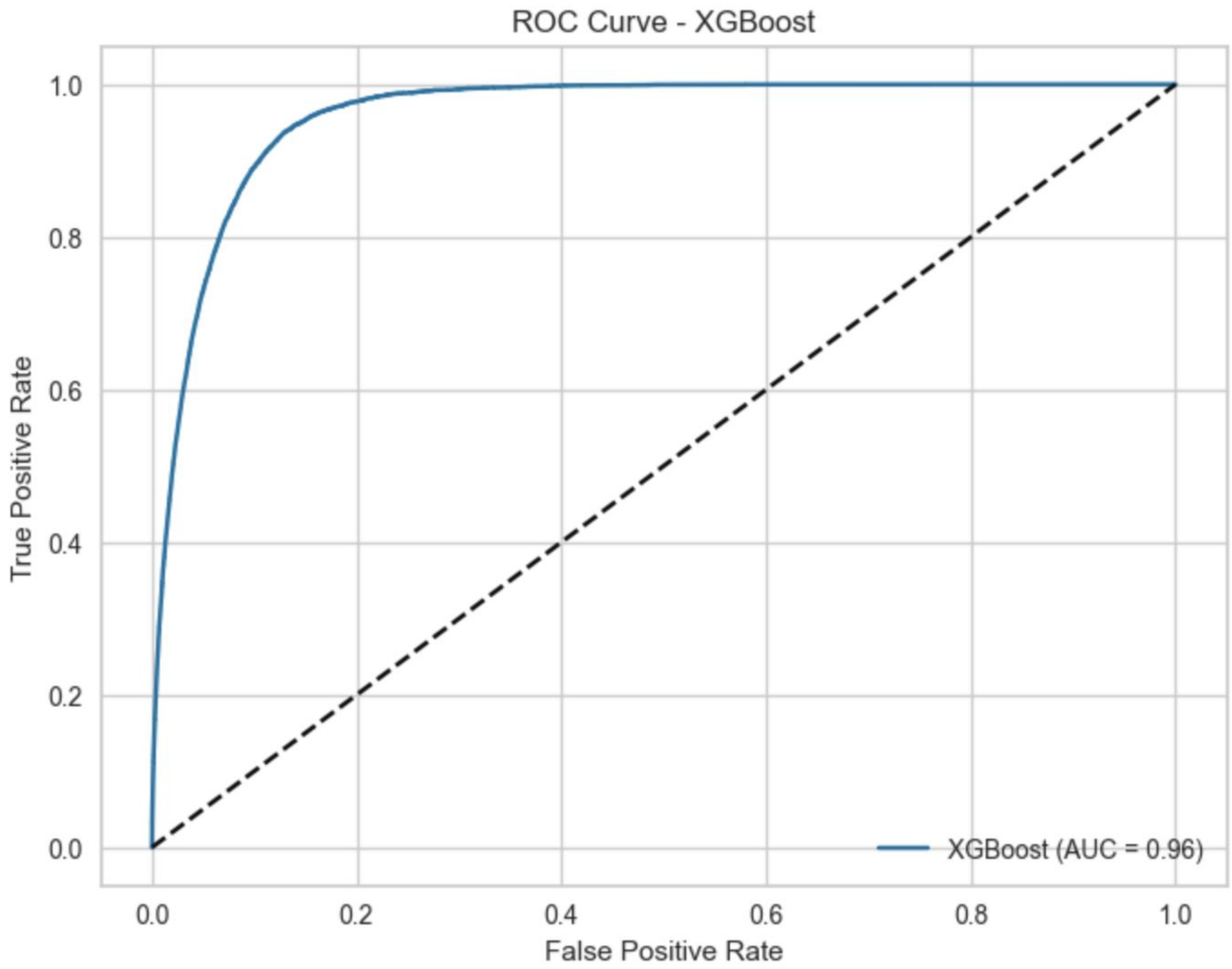
1. Logistic Regression:



- ❖ With an AUC of 0.88, the model correctly distinguishes between positive and negative instances approximately 88% of the time.

4.3 Training using ML models

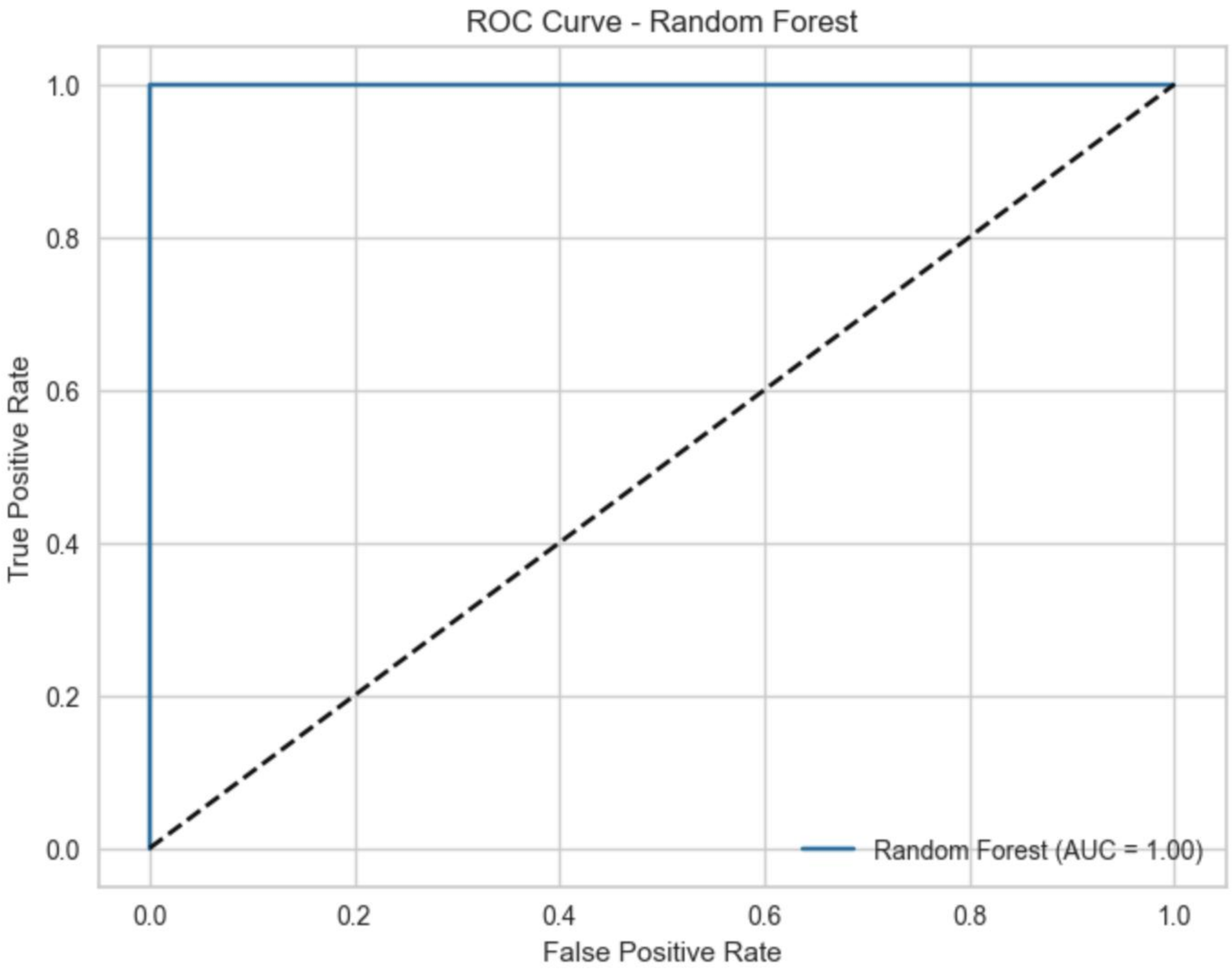
2. XGBoost:



- With an AUC of 0.96, the model correctly distinguishes between positive and negative instances approximately 96% of the time.

4.4 Training using ML models

3. Random Forest:



- ❖ With an AUC of 1.00, the model correctly distinguishes between positive and negative instances approximately 100% of the time.

4.5 Testing the models

Method 1:

using average of all models and adjusting the threshold

- Threshold > 91.8%

Result:

- Fraudulent Accounts: 103
- Accuracy: 100%

Method 2:

using each model separately and then intersecting

- Threshold > 50%

Result (LR):

- Fraudulent Accounts: 2,02,596

Result (XGB):

- Fraudulent Accounts: 1,33,818

Result (RF):

- Fraudulent Accounts: 11,005

Result (Intersected):

- Common account numbers: 8,603
- Accuracy: 100%

4.6

Observations

- When the testing data is seen all fraud accounts are successfully detected.

5.1

Scenario - 2

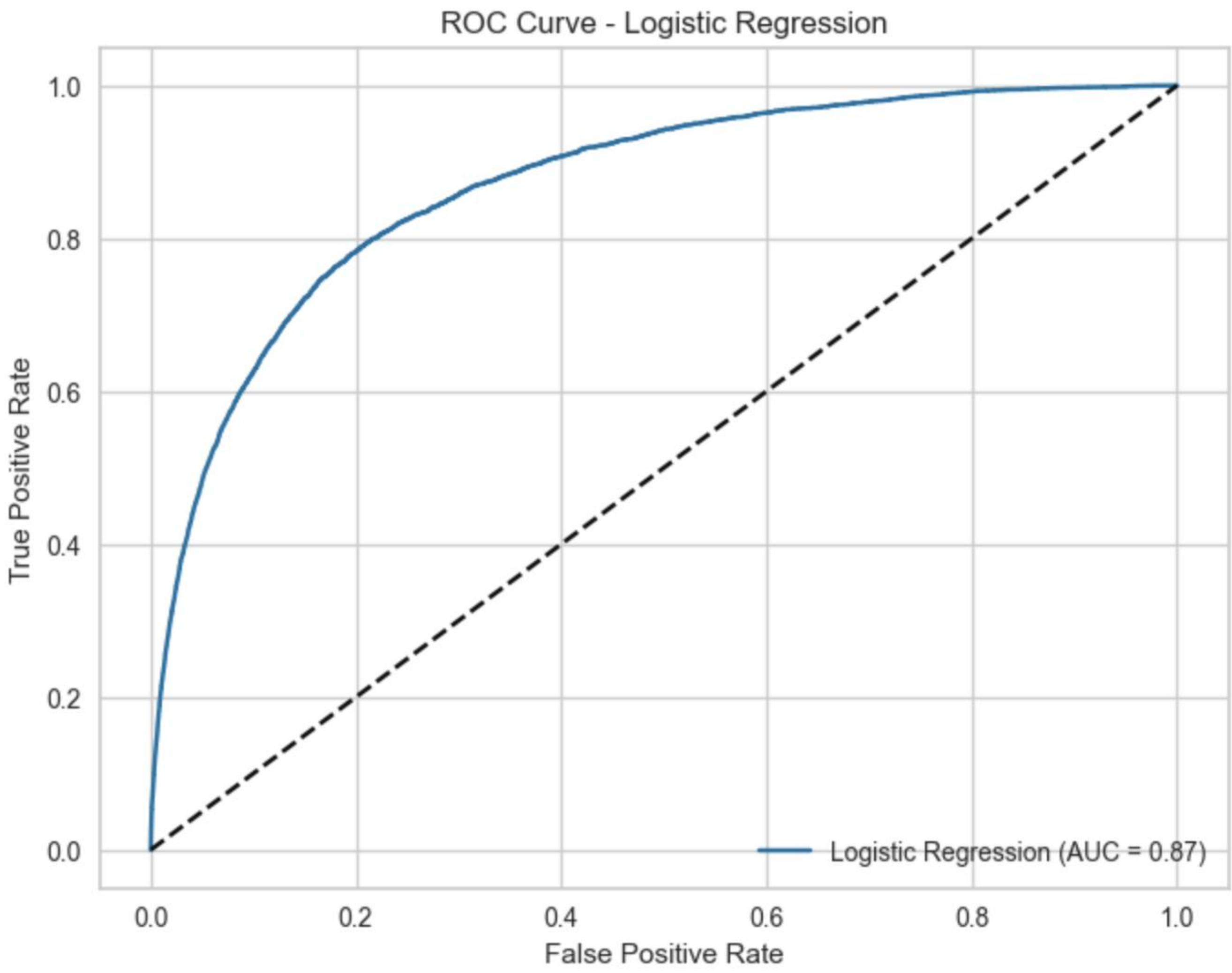
Training data: Account numbers 1-5,00,000 (first 5 lakh a/c)

Testing data: Account numbers 5,00,001-10,00,000 (next 5 lakh a/c)

❖ Testing data is ***unseen*** by the models.

5.2 Training using ML models

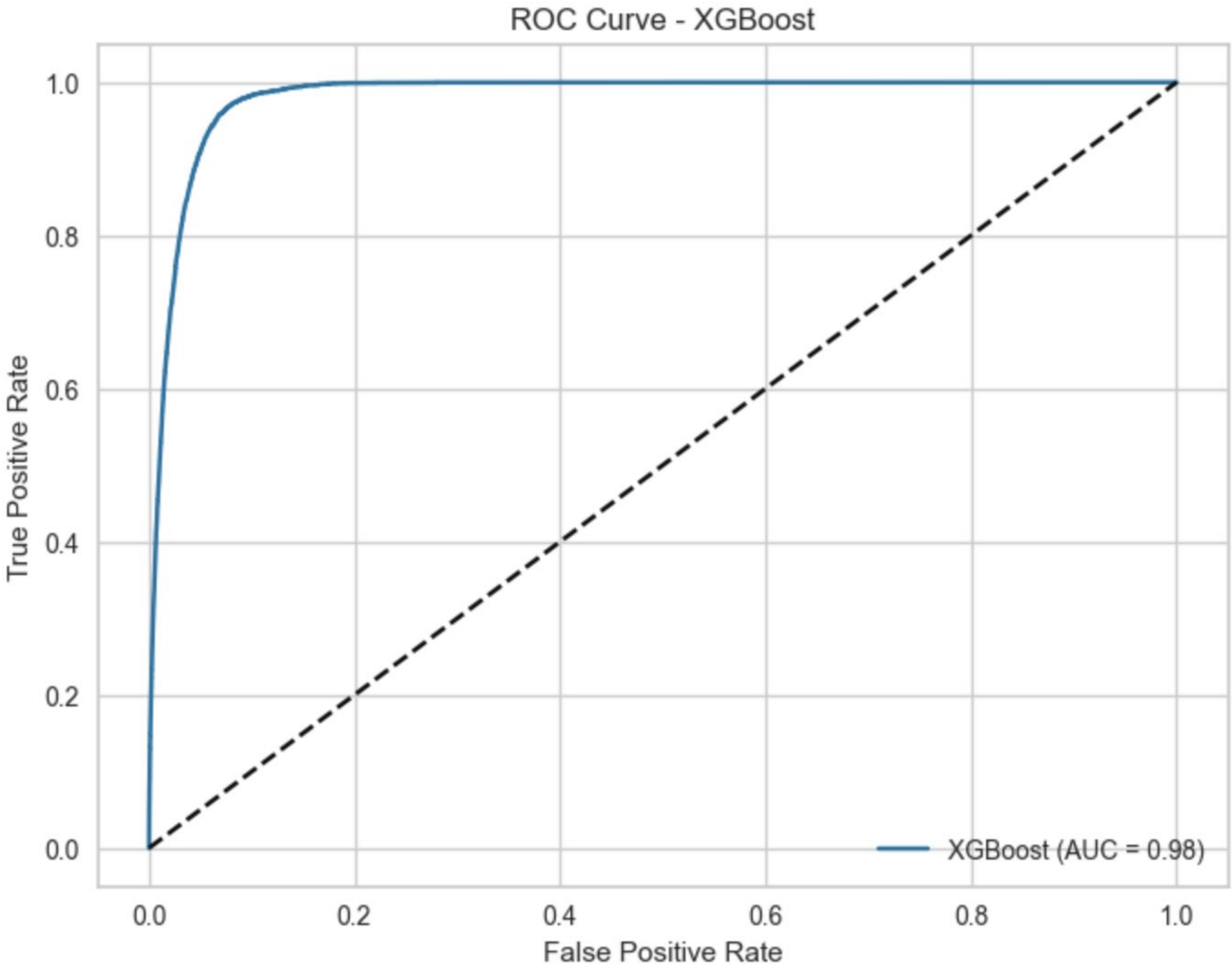
1. Logistic Regression:



- With an AUC of 0.87, the model correctly distinguishes between positive and negative instances approximately 87% of the time.

5.3 Training using ML models

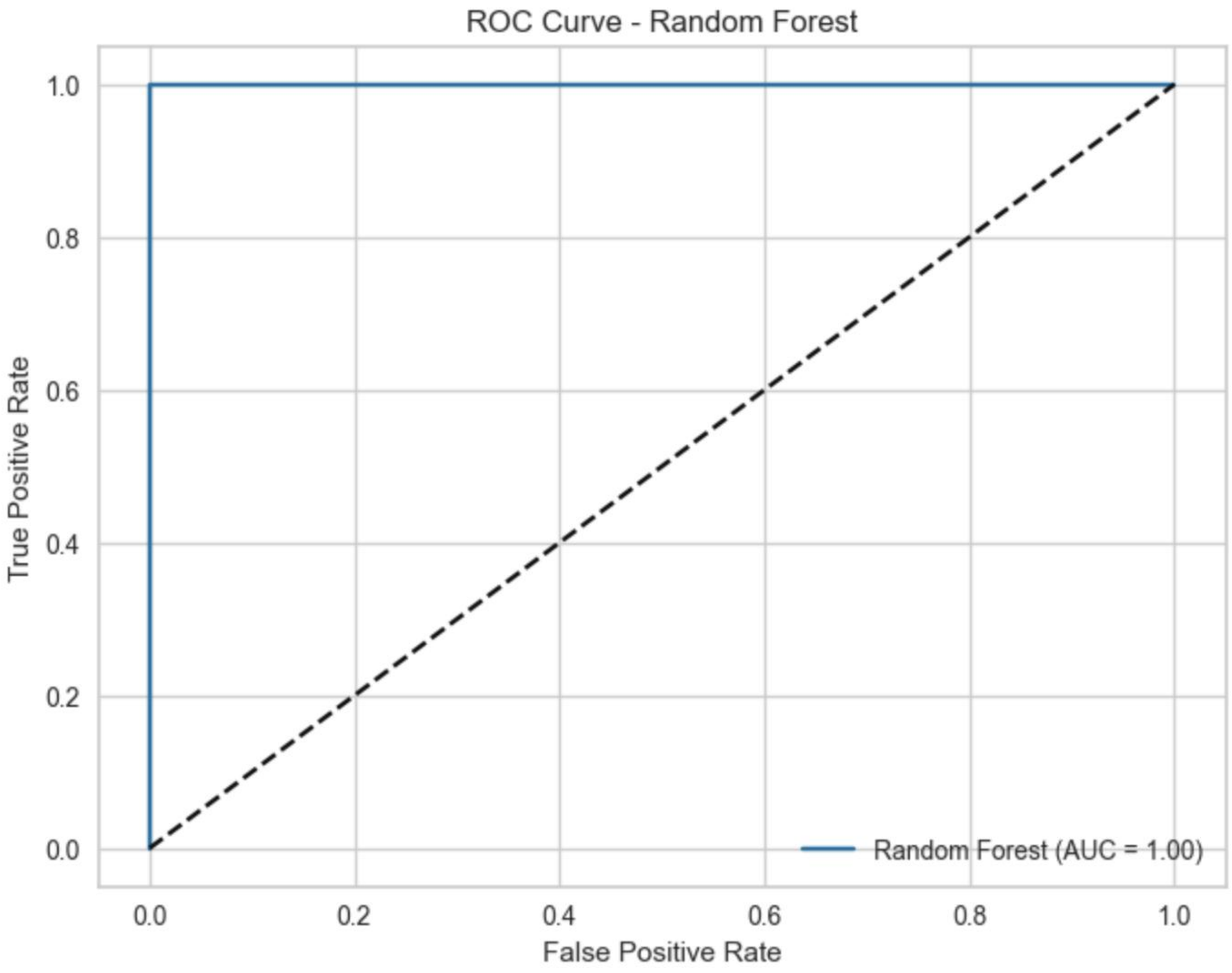
2. XGBoost:



- ❖ With an AUC of 0.98, the model correctly distinguishes between positive and negative instances approximately 98% of the time.

5.4 Training using ML models

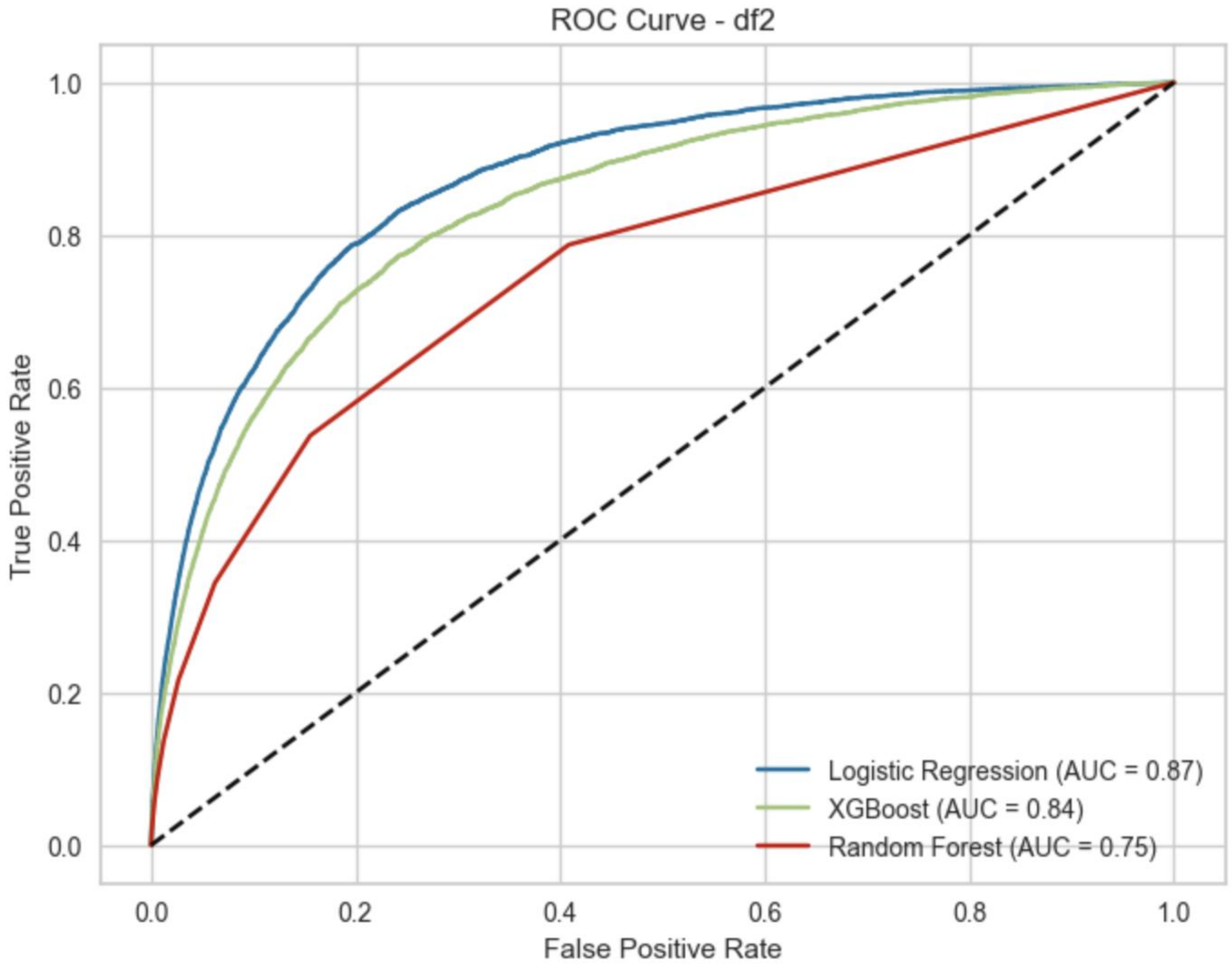
3. Random Forest:



- ❖ With an AUC of 1.00, the model correctly distinguishes between positive and negative instances approximately 100% of the time.

5.5 Testing the models

All models:



- With an AUC of 0.87, the LR model correctly distinguishes between positive and negative instances approximately 87% of the time.
- With an AUC of 0.84, the XGB model correctly distinguishes between positive and negative instances approximately 84% of the time.
- With an AUC of 0.75, the RF model correctly distinguishes between positive and negative instances approximately 75% of the time.

5.6 Testing the model

Method 1:

using average of all models and adjusting the threshold

- Threshold > 67.1%

Result:

- Fraudulent Accounts: 102
- Accuracy: 56.86% (58)

Method 2:

using each model separately and then intersecting

- Threshold > 50%

Result (LR):

- Fraudulent Accounts: 57,617

Result (XGB):

- Fraudulent Accounts: 22,448

Result (RF):

- Fraudulent Accounts: 4,814

Result (Intersected):

- Common account numbers: 213
- Accuracy: 9.86% (21)

5.7

Observations

- When the testing data is unseen (real scenario) the accuracy decreases.

6.1 Dataset (2)

<u>Each dataset is composed of:</u>

- 1 million (***10 lakh***) accounts.
- ***21 realistic features*** (***dropped 9 low priority features*** from the original dataset) used in the fraud detection use-case.
- Dropped features: housing_status, phone_home_valid, phone_mobile_valid, bank_months_count, has_other_cards, proposed_credit_limit, source, device_os, device_fraud_count
- ***Protected attributes***: age group, employment status and income.

7.1

Scenario - 1

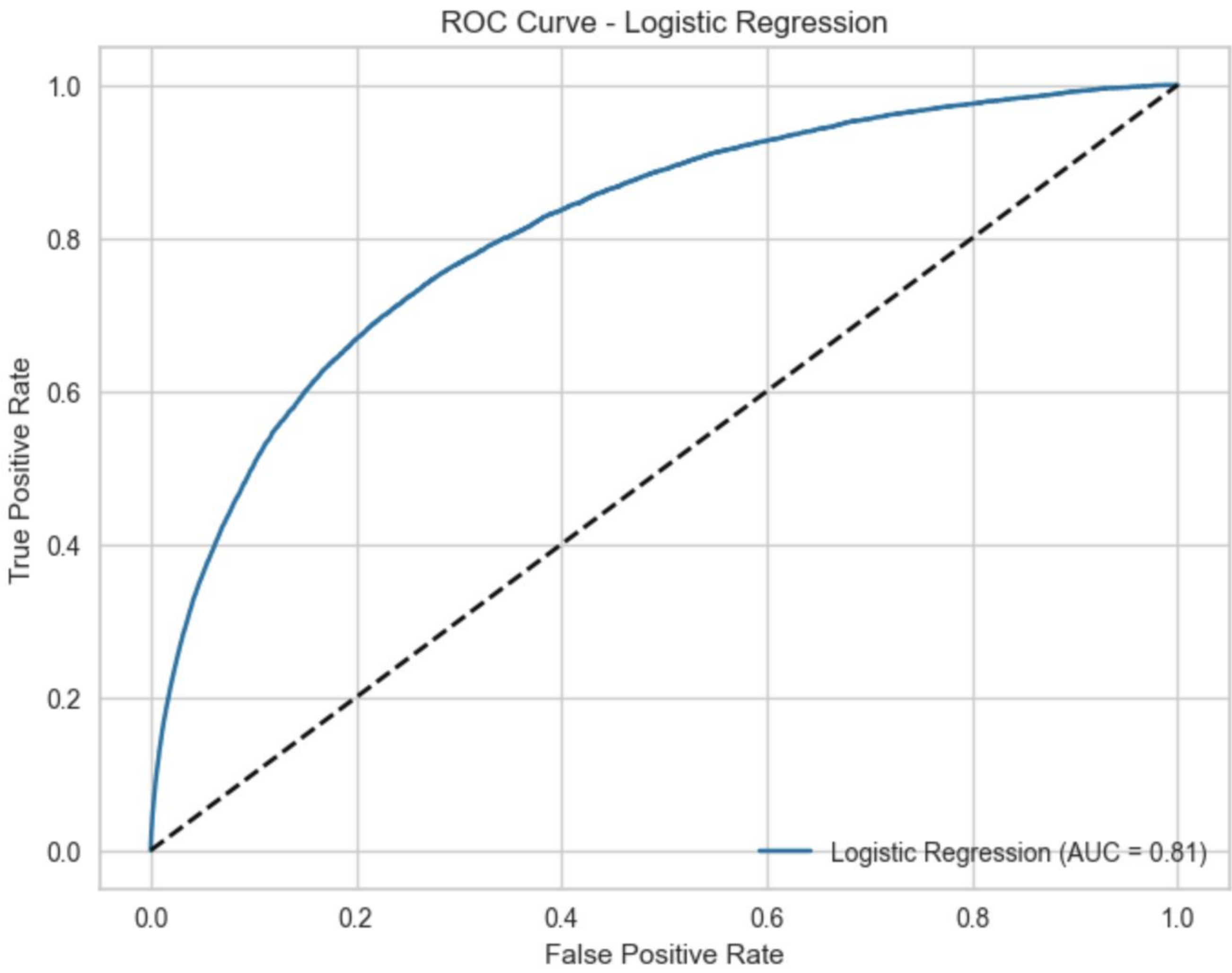
Training data: Account numbers 1-10,00,000 (complete dataset)

Testing data: Account numbers 1-10,00,000 (same complete dataset)

- Testing data is ***seen*** by the models.

7.2 Training using ML models

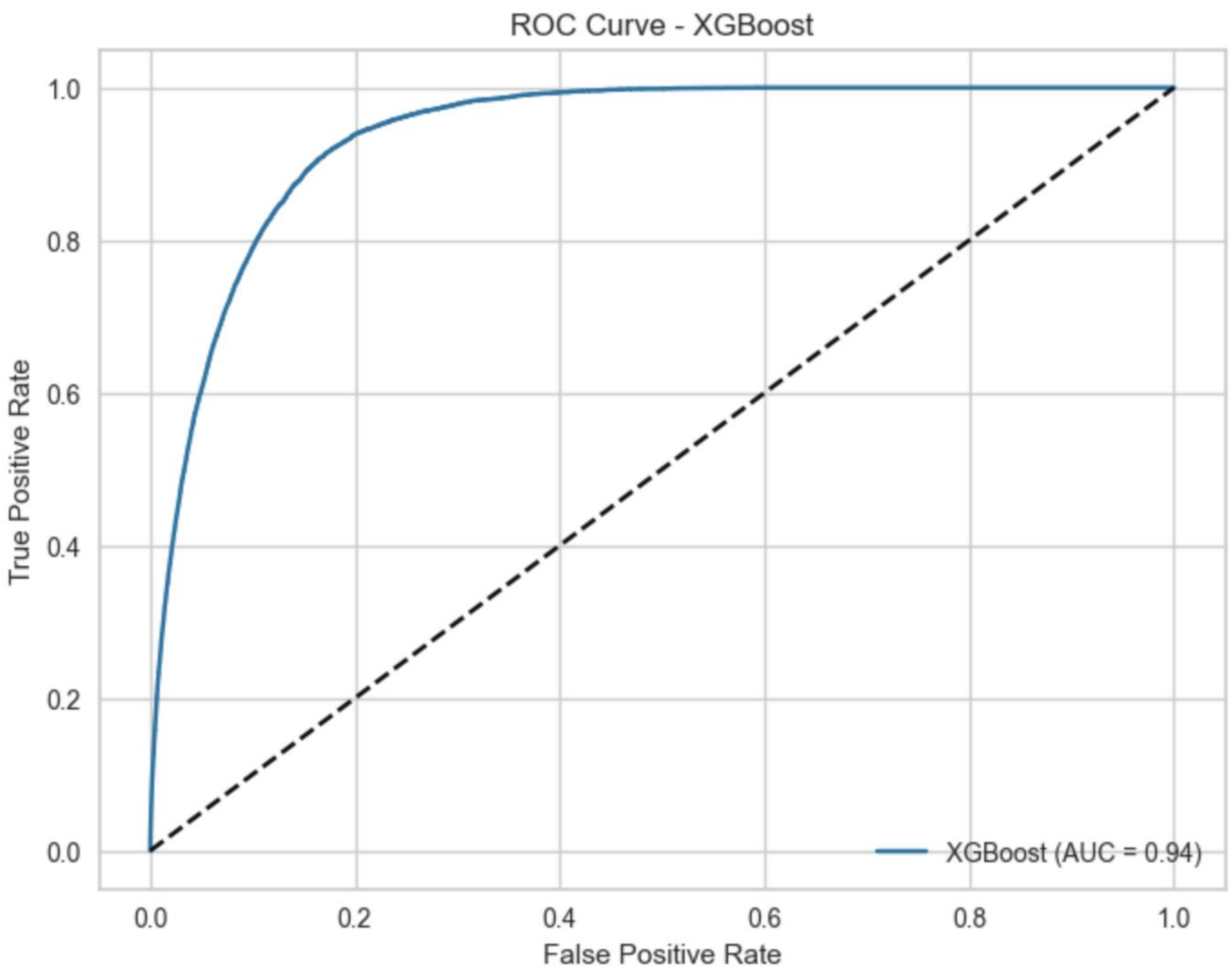
1. Logistic Regression:



- With an AUC of 0.81, the model correctly distinguishes between positive and negative instances approximately 81% of the time.

7.3 Training using ML models

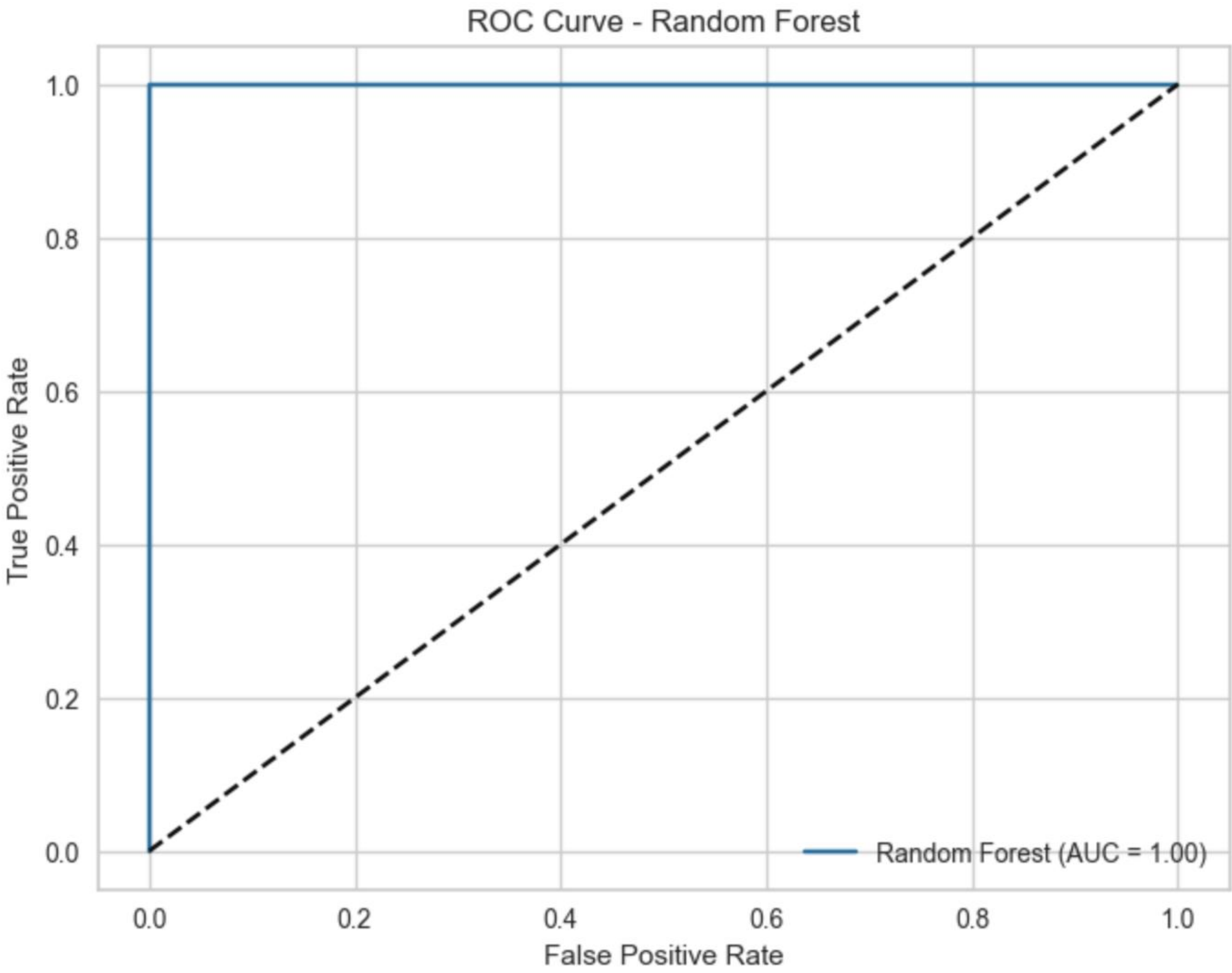
2. XGBoost:



- With an AUC of 0.94, the model correctly distinguishes between positive and negative instances approximately 94% of the time.

7.4 Training using ML models

3. Random Forest:



- With an AUC of 1.00, the model correctly distinguishes between positive and negative instances approximately 100% of the time.

7.5 Testing the models

Method 1:

using average of all models and adjusting the threshold

- Threshold > 89.2%

Result:

- Fraudulent Accounts: 101
- Accuracy: 100%

Method 2:

using each model separately and then intersecting

- Threshold > 50%

Result (LR):

- Fraudulent Accounts: 2,69,589

Result (XGB):

- Fraudulent Accounts: 1,69,833

Result (RF):

- Fraudulent Accounts: 11,004

Result (Intersected):

- Common account numbers: 7,894
- Accuracy: 100%

7.6

Observations

- When the testing data is seen all fraud accounts are again successfully detected.

8.1

Scenario - 2

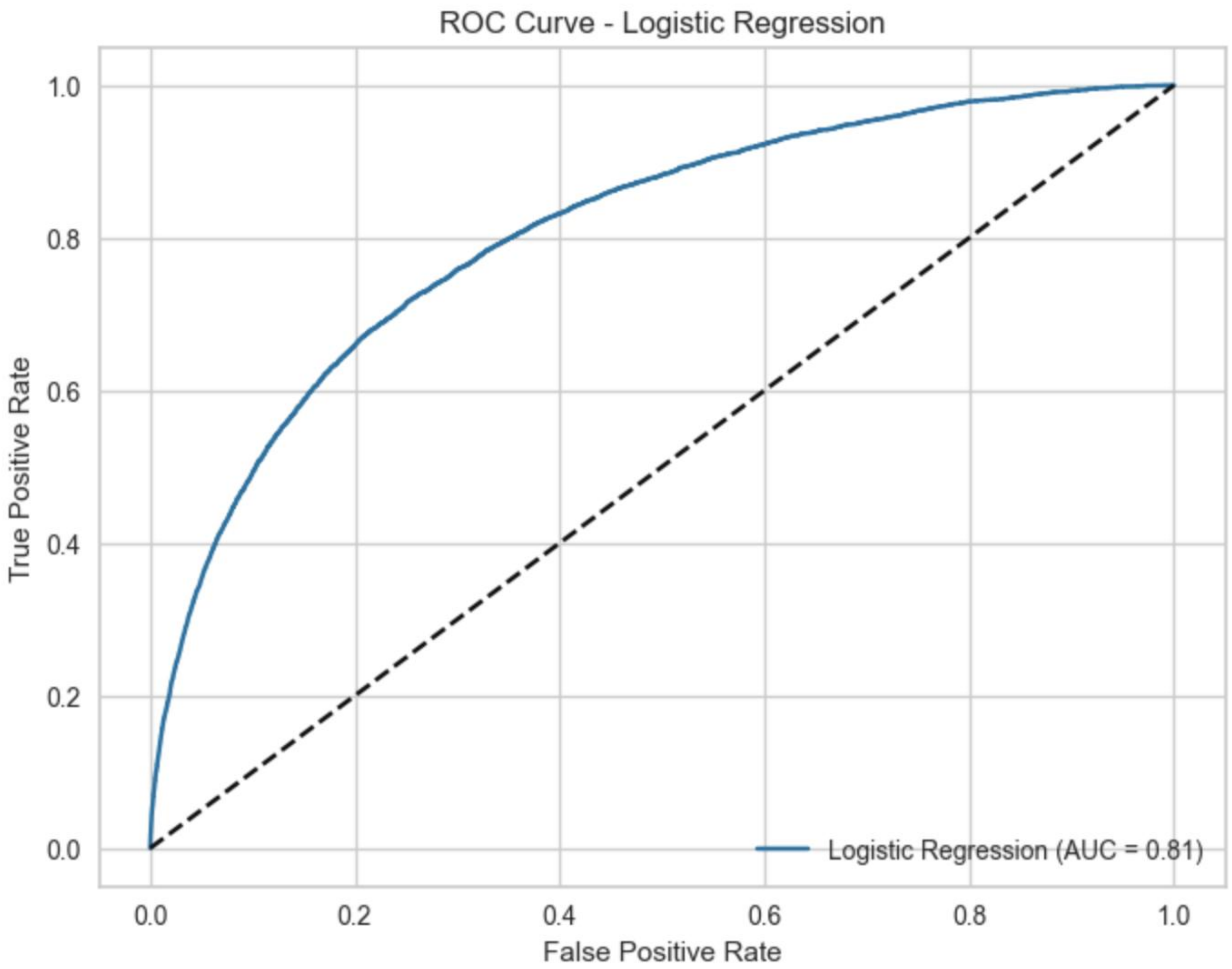
Training data: Account numbers 1-5,00,000 (first 5 lakh a/c)

Testing data: Account numbers 5,00,001-10,00,000 (next 5 lakh a/c)

❖ Testing data is ***unseen*** by the models.

8.2 Training using ML models

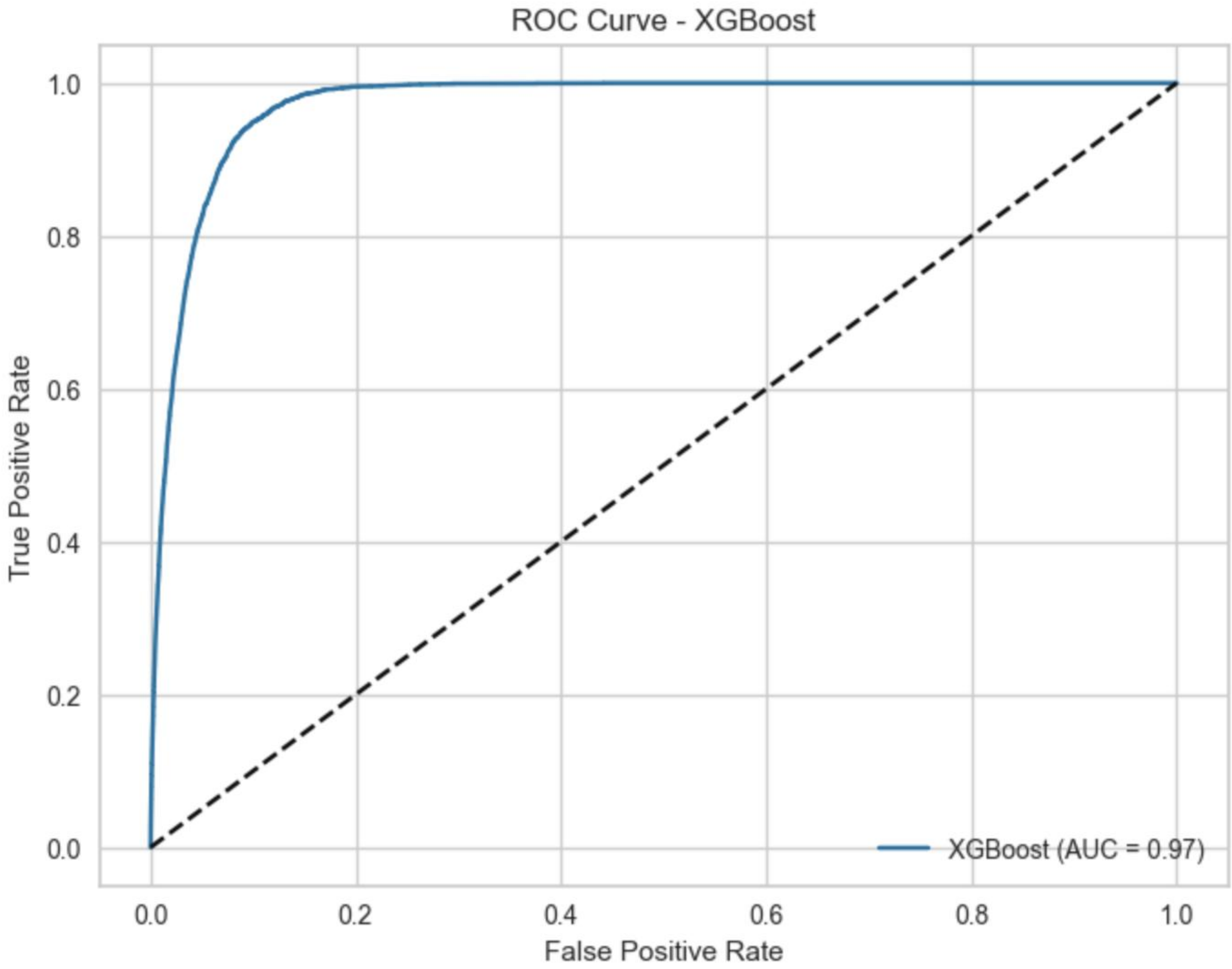
1. Logistic Regression:



- With an AUC of 0.81, the model correctly distinguishes between positive and negative instances approximately 81% of the time.

8.3 Training using ML models

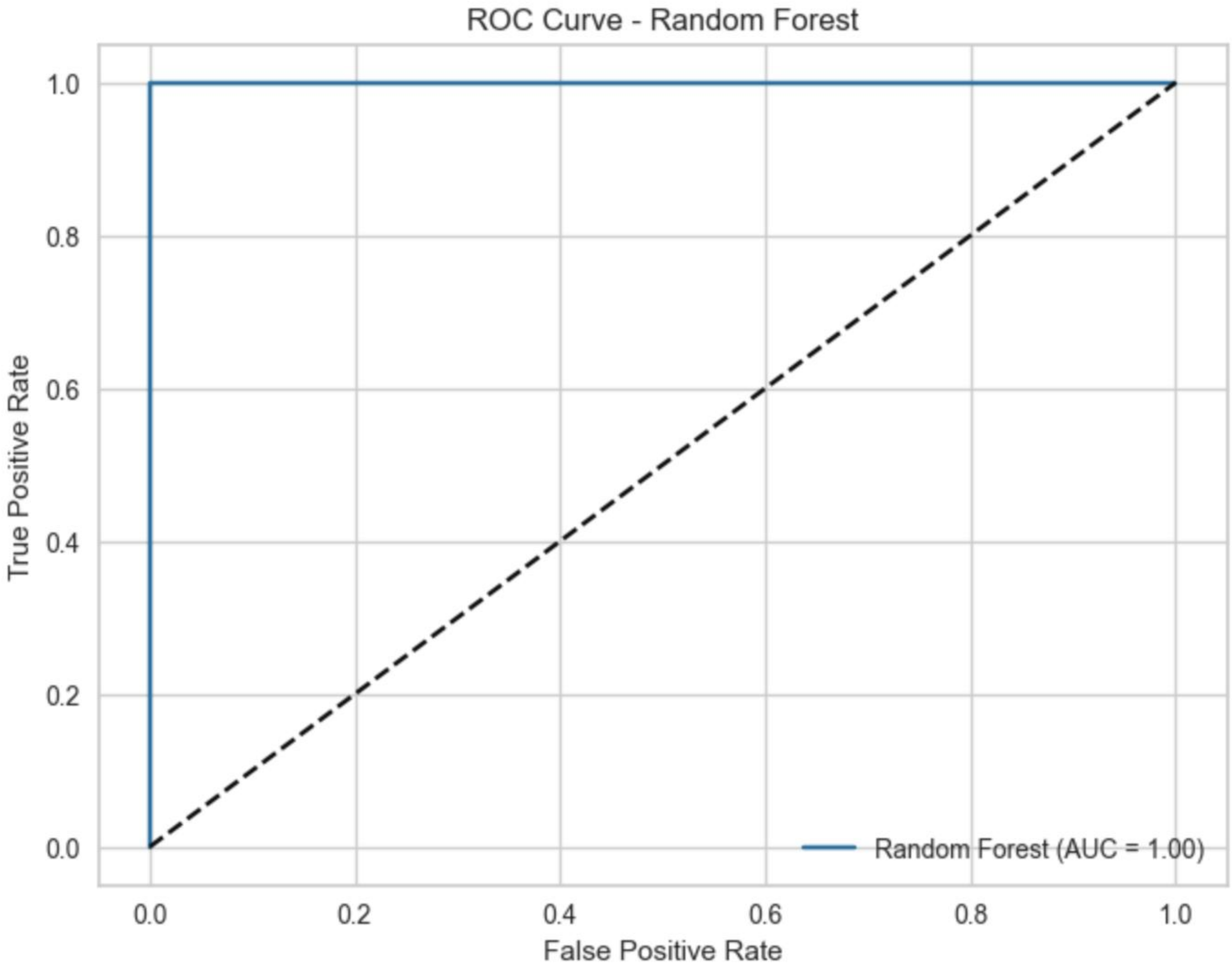
2. XGBoost:



- With an AUC of 0.97 , the model correctly distinguishes between positive and negative instances approximately 97% of the time.

8.4 Training using ML models

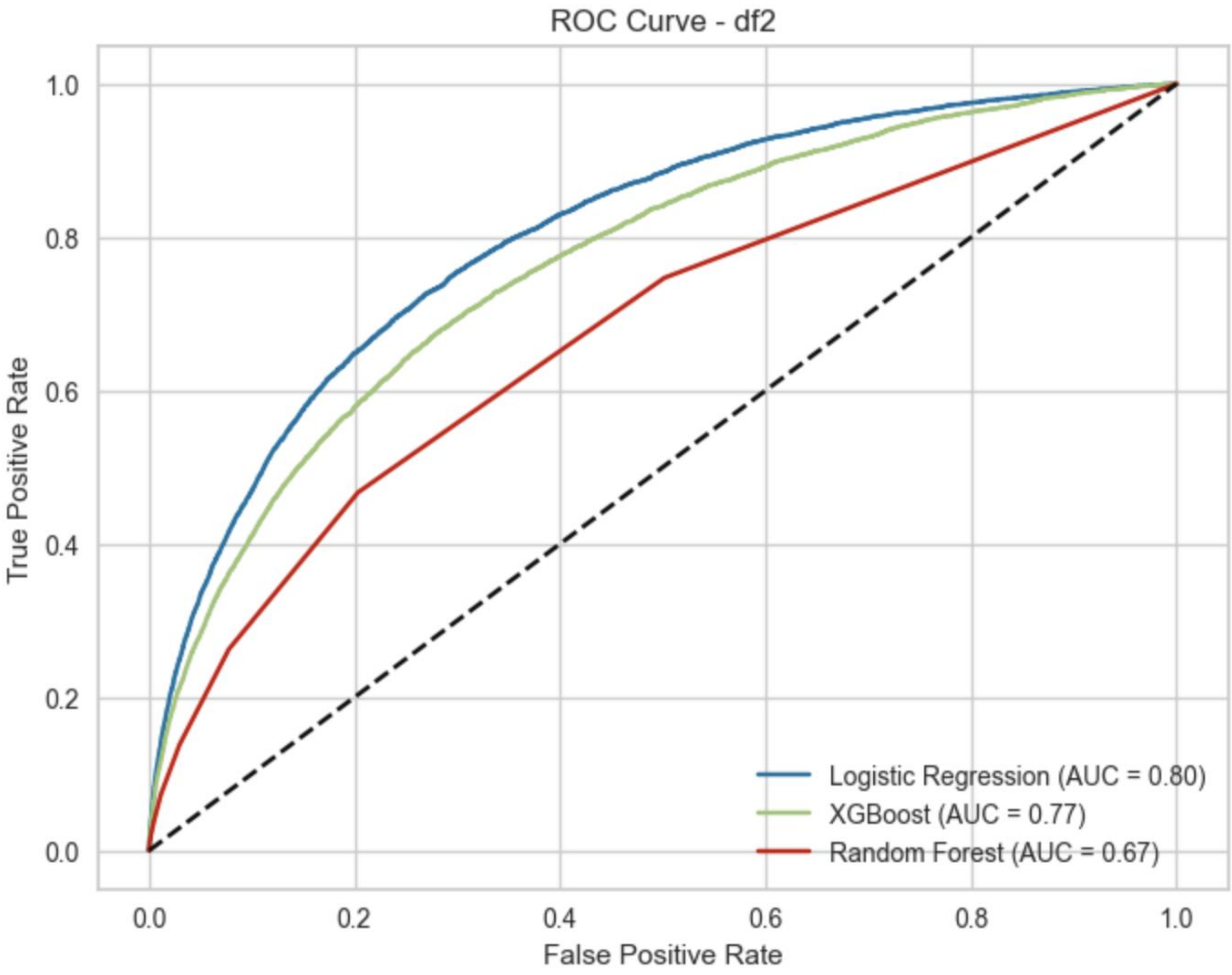
3. Random Forest:



- ❖ With an AUC of 1.00, the model correctly distinguishes between positive and negative instances approximately 100% of the time.

8.5 Testing the models

All models:



- With an AUC of 0.80, the LR model correctly distinguishes between positive and negative instances approximately 80% of the time.
- With an AUC of 0.77, the XGB model correctly distinguishes between positive and negative instances approximately 77% of the time.
- With an AUC of 0.67, the RF model correctly distinguishes between positive and negative instances approximately 67% of the time.

8.6 Testing the model

Method 1:

using average of all models and adjusting the threshold

- Threshold > 64%

Result:

- Fraudulent Accounts: 99
- Accuracy: 35.35% (35)

Method 2:

using each model separately and then intersecting

- Threshold > 50%

Result (LR):

- Fraudulent Accounts: 72,558

Result (XGB):

- Fraudulent Accounts: 20,062

Result (RF):

- Fraudulent Accounts: 4,813

Result (Intersected):

- Common account numbers: 145
- Accuracy: 9.66% (14)

8.7

Observations

- When the testing data is unseen (real scenario) the accuracy decreases.
- The accuracy is lesser than the dataset (1) i.e. 30 features: decreased from 56.86% to 35.35%

9.1

Conclusion

- We conclude that an increase in the amount of training data leads to improved performance during testing.

10.1 Dataset (3)

Composed of:

- 5 million (***50 lakh***) accounts.
- ***30 realistic features*** used in the fraud detection use-case.
- ***Protected attributes***: age group, employment status and income.

11.1

Scenario

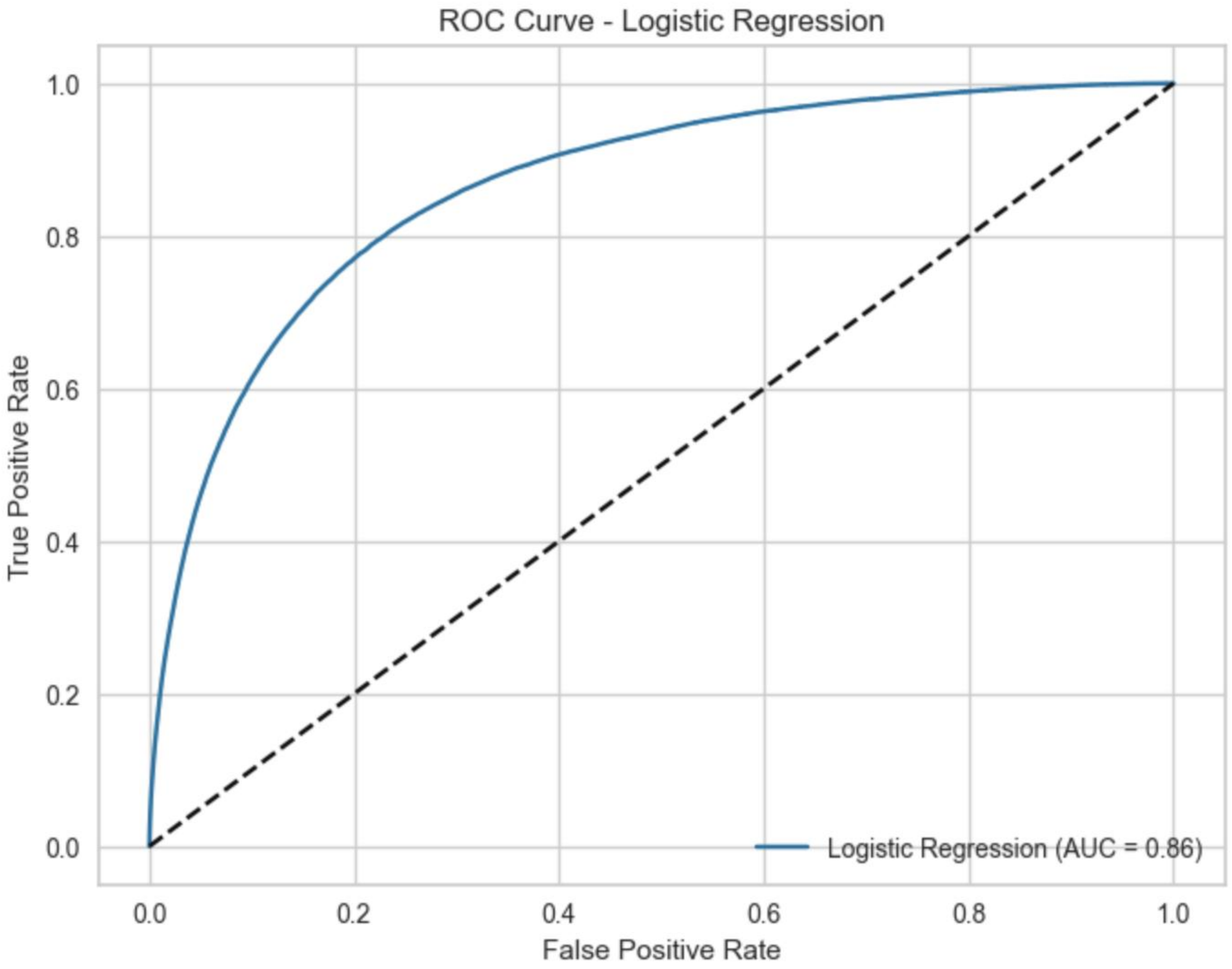
Training data: 50 lakh accounts from all variants

Testing data: 10 lakh accounts from base

❖ Testing data is ***unseen*** by the models.

11.2 Training using ML models

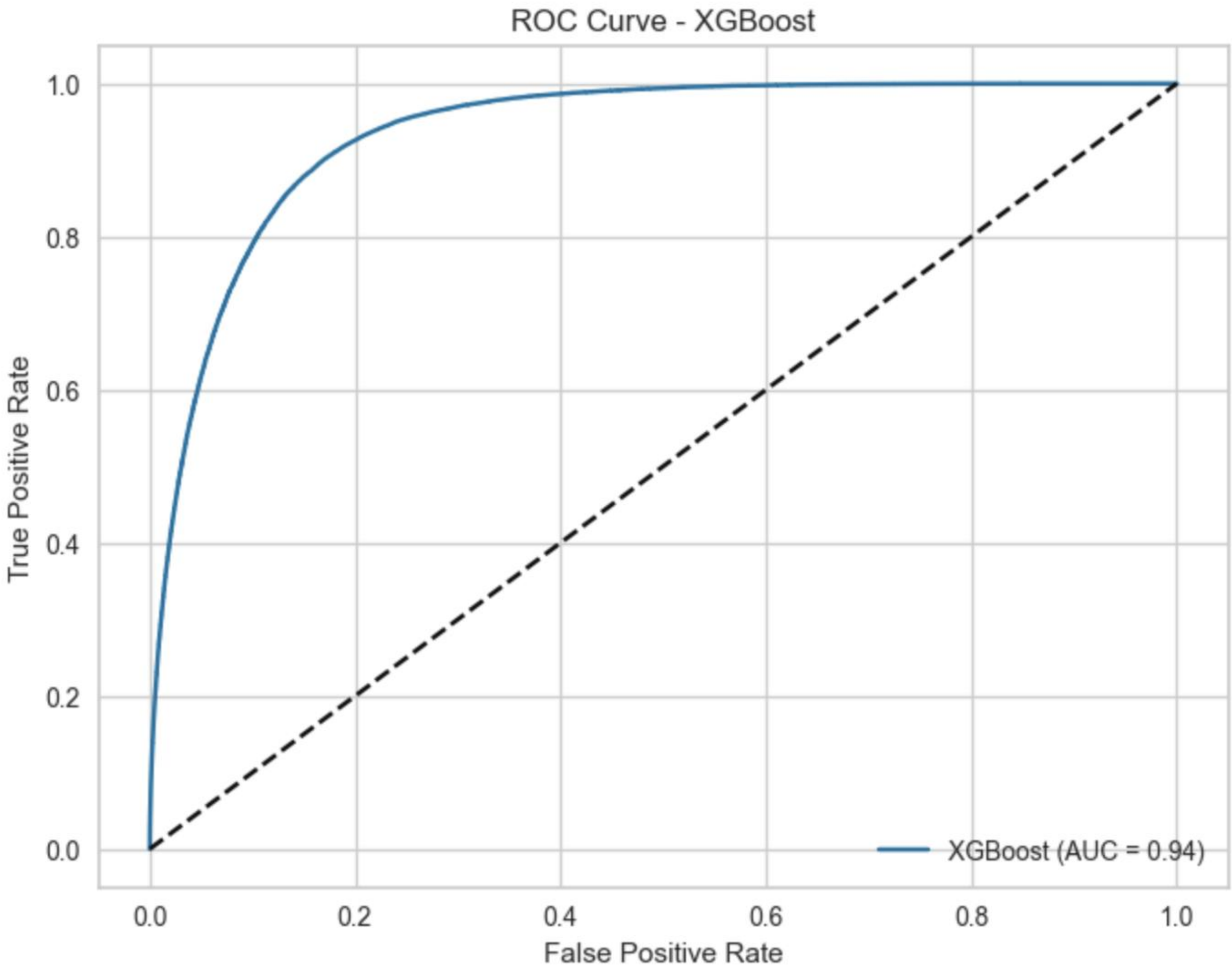
1. Logistic Regression:



- With an AUC of 0.86, the model correctly distinguishes between positive and negative instances approximately 86% of the time.

11.3 Training using ML models

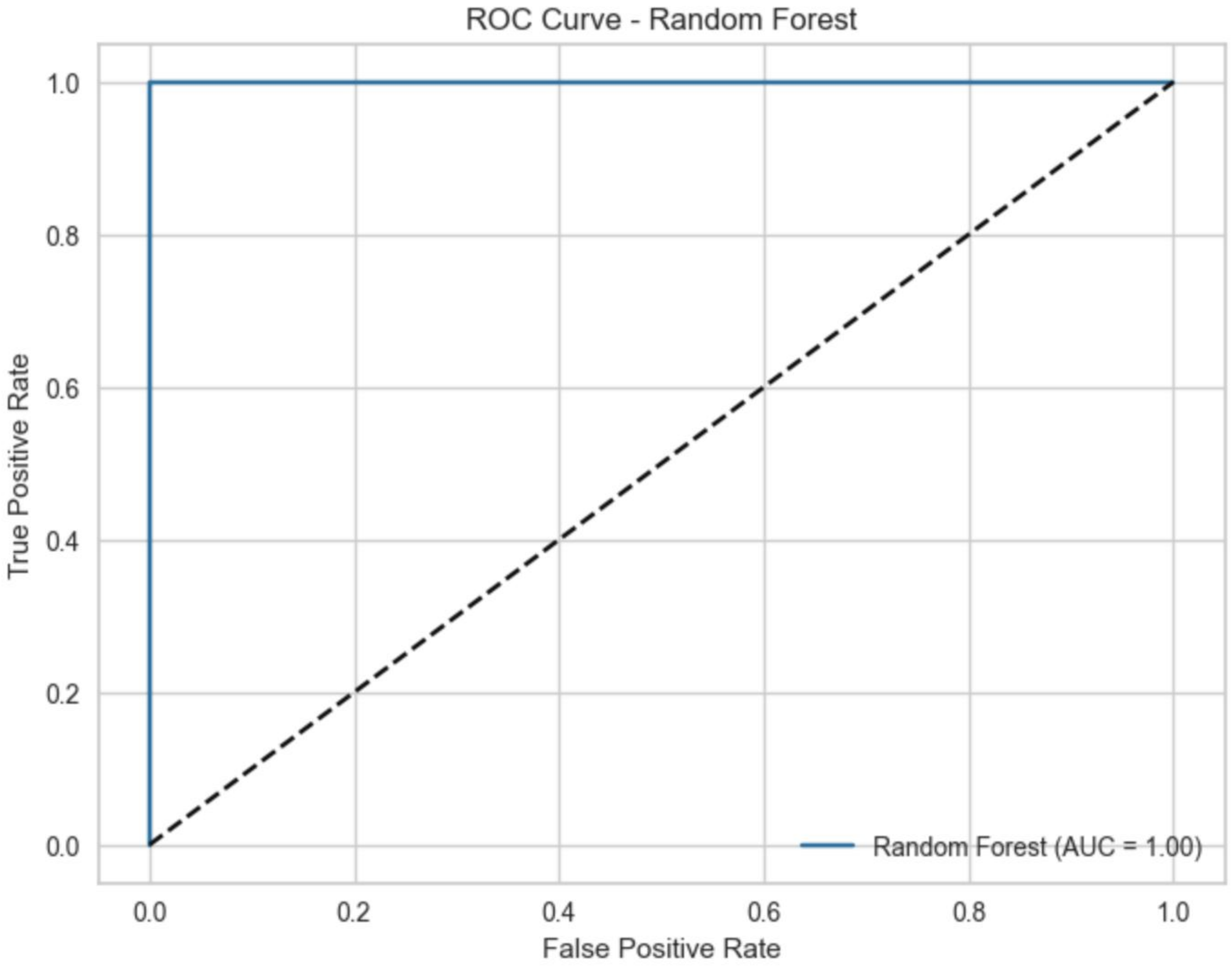
2. XGBoost:



❖ With an AUC of 0.94, the model correctly distinguishes between positive and negative instances approximately 94% of the time.

11.4 Training using ML models

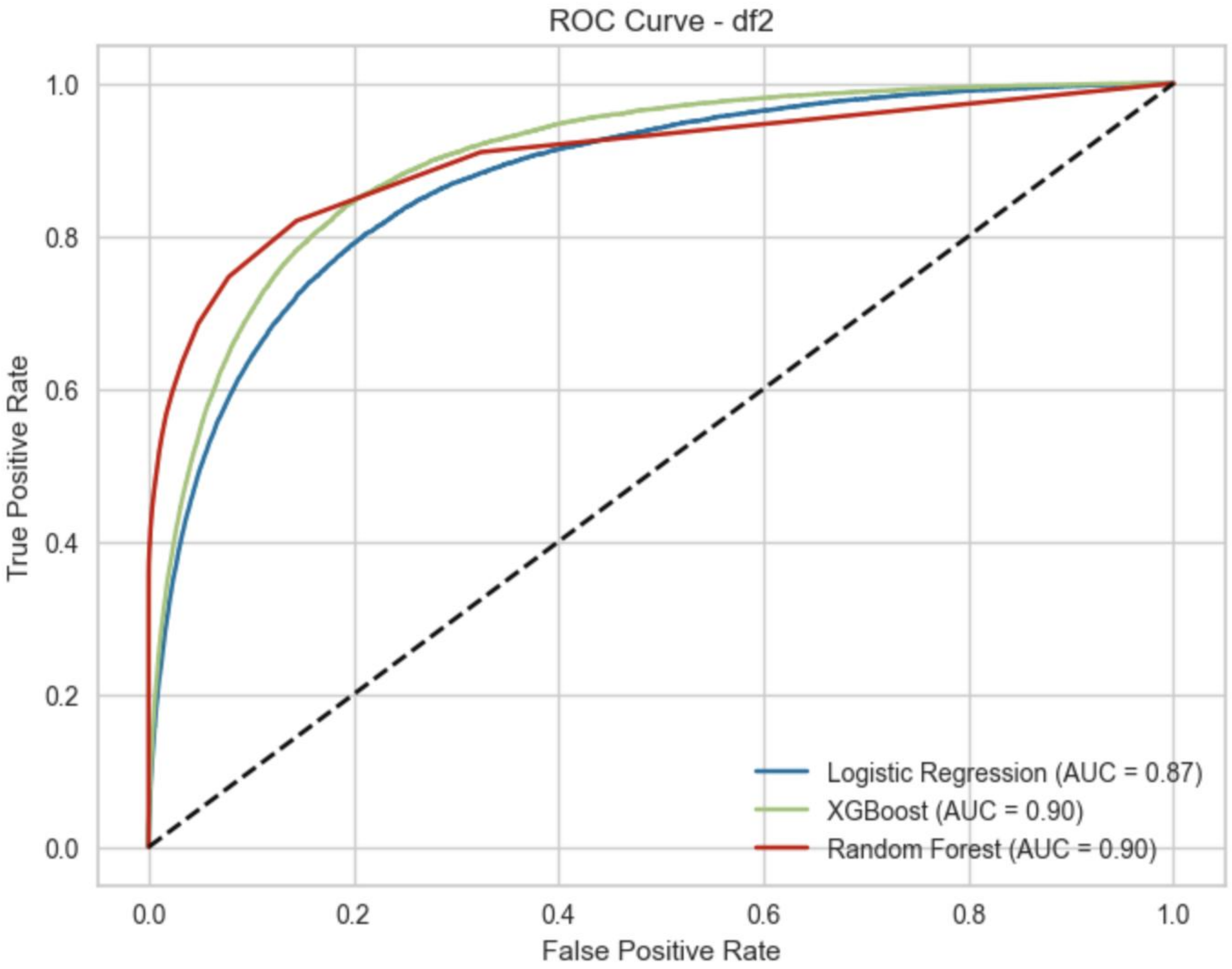
3. Random Forest:



- With an AUC of 1.00, the model correctly distinguishes between positive and negative instances approximately 100% of the time.

11.5 Testing the models

All models:



- ❖ With an AUC of 0.87, the LR model correctly distinguishes between positive and negative instances approximately 87% of the time.
- ❖ With an AUC of 0.90, the XGB model correctly distinguishes between positive and negative instances approximately 90% of the time.
- ❖ With an AUC of 0.90, the RF model correctly distinguishes between positive and negative instances approximately 90% of the time.

11.6 Testing the model

Method 1:

using average of all models and adjusting the threshold

- Threshold > 91%

Result:

- Fraudulent Accounts: 282
- Accuracy: 100%

- Threshold > 75%

Result:

- Fraudulent Accounts: 2,404
- Accuracy: 90.35%

Method 2:

using each model separately and then intersecting

- Threshold > 50%

Result (LR):

- Fraudulent Accounts: 2,02,696

Result (XGB):

- Fraudulent Accounts: 1,62,877

Result (RF):

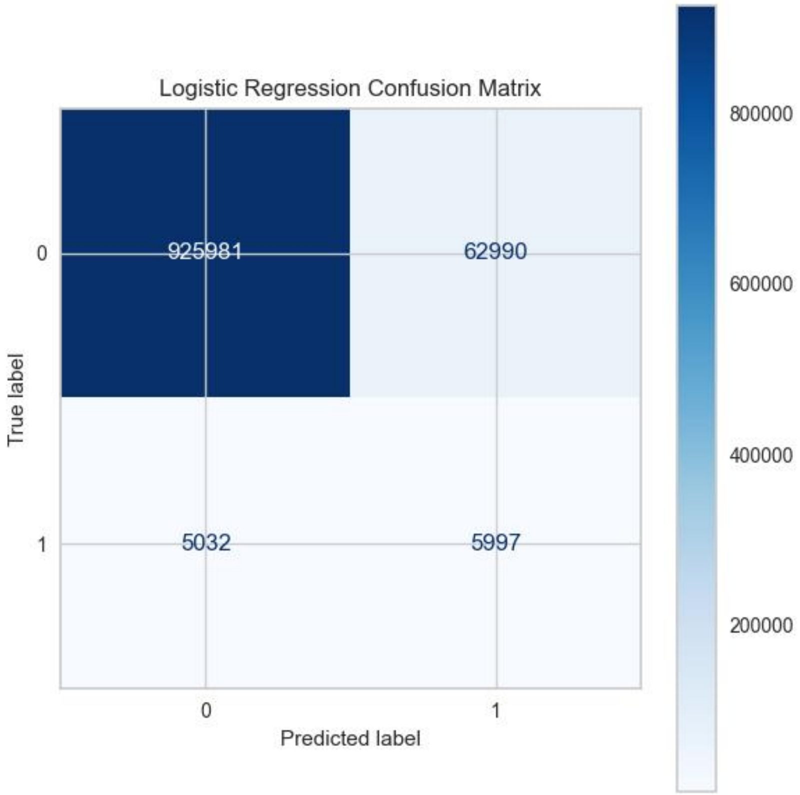
- Fraudulent Accounts: 3,149

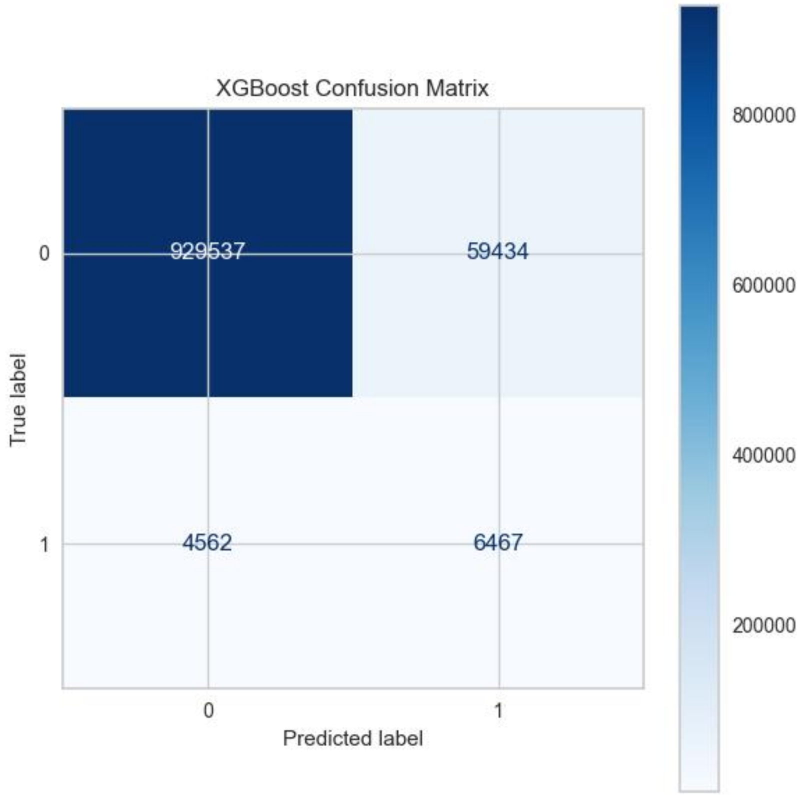
Result (Intersected):

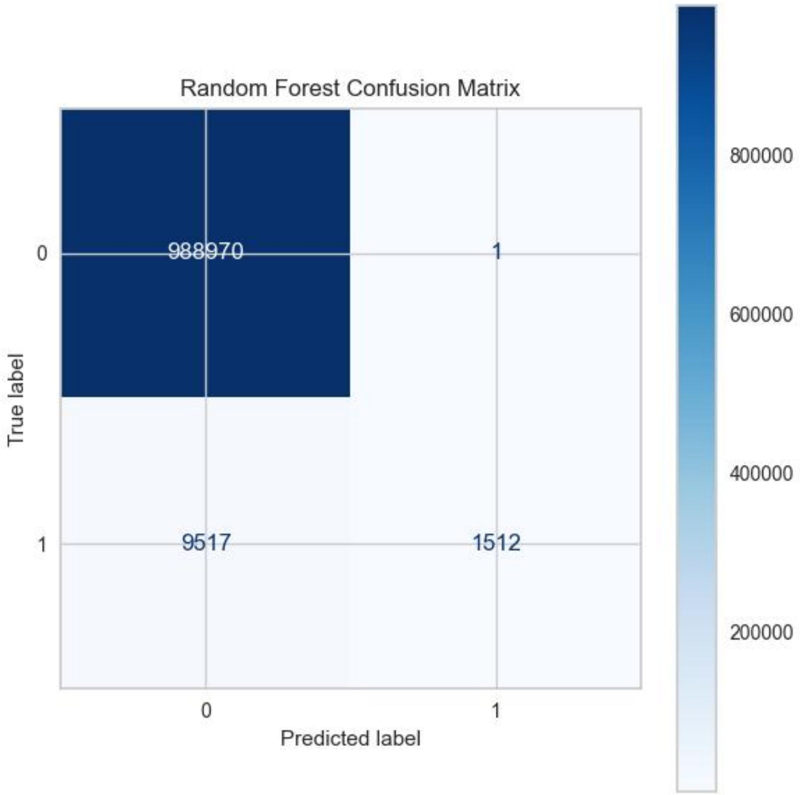
- Common account numbers: 2,417
- Accuracy: 99.01%

11.7 Validation

For each model :







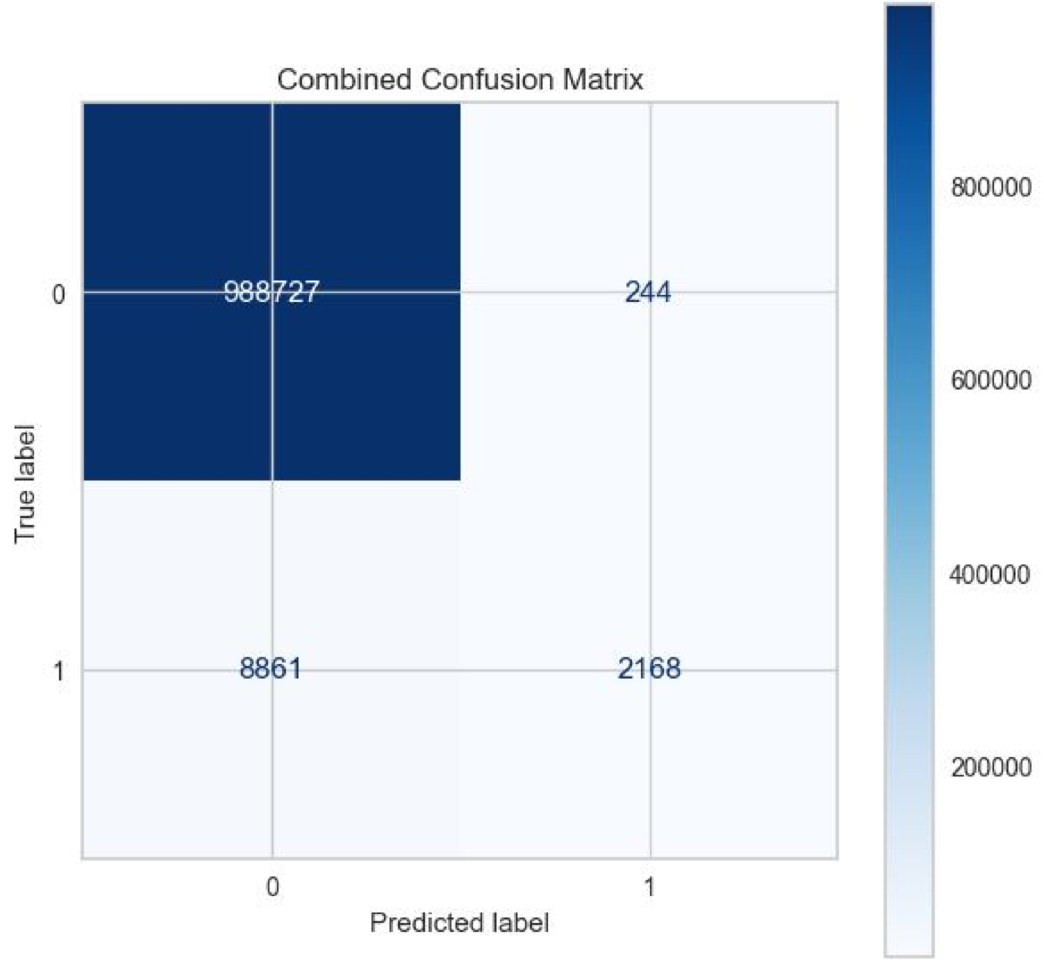
11.8 Validation

```
Logistic Regression Metrics:
Accuracy: 0.9320
Precision: 0.0869
Recall: 0.5437
F1-Score: 0.1499

XGBoost Metrics:
Accuracy: 0.9360
Precision: 0.0981
Recall: 0.5864
F1-Score: 0.1681

Random Forest Metrics:
Accuracy: 0.9905
Precision: 0.9993
Recall: 0.1371
F1-Score: 0.2411
```

```
Logistic Regression - TPR: 0.5437, FPR: 0.0637
XGBoost - TPR: 0.5864, FPR: 0.0601
Random Forest - TPR: 0.1371, FPR: 0.0000
```

11.9 Validation

For combined model :



11.10 Validation

```
Combined Metrics:
Accuracy: 0.9909
Precision: 0.8988
Recall: 0.1966
F1-Score: 0.3226
```

```
Combined - TPR: 0.1966, FPR: 0.0002
```

11.11

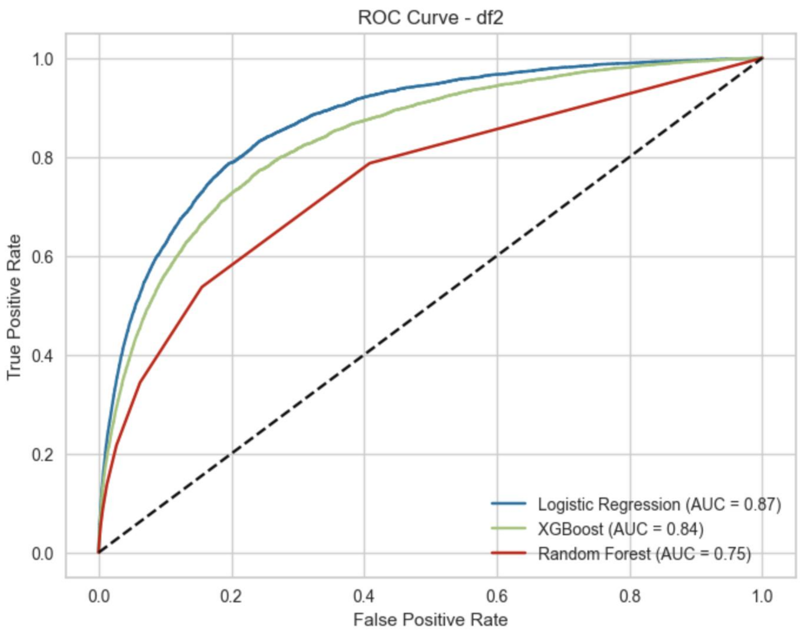
Observations

❖ Increasing the amount of training data results in highest accuracy.

12.1 Comparison

Trained: 5 lakh accounts
Tested : 5 lakh accounts

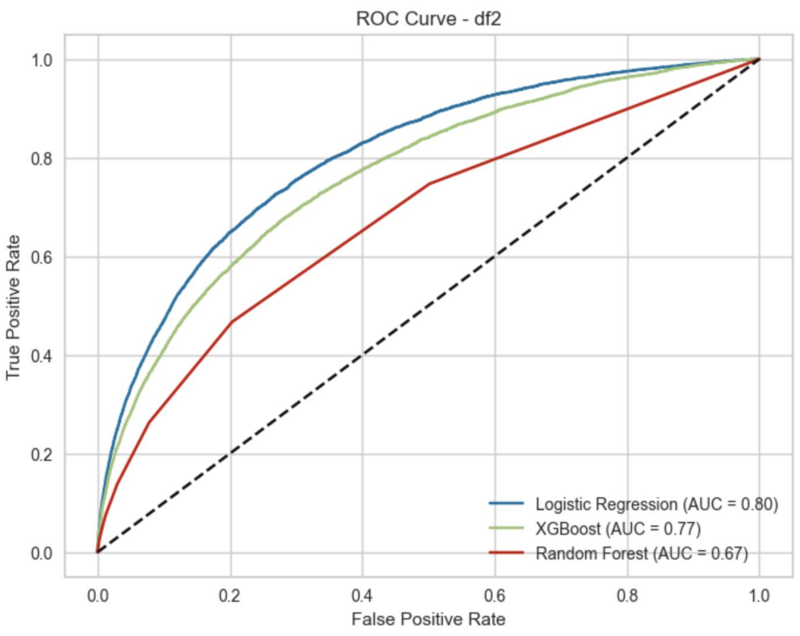
Features : 30



Testing AUCs

Trained: 5 lakh accounts
Tested : 5 lakh accounts

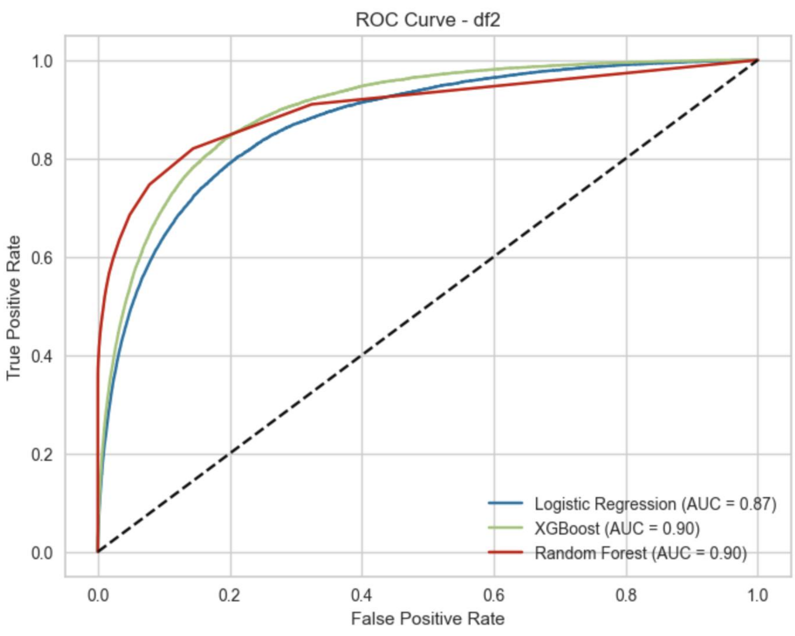
Features : 21



Testing AUCs

Trained: 50 lakh accounts
Tested : 10 lakh accounts

Features : 30



Testing AUCs

13.1 Conclusion

- We utilized a dataset that is the ***first publicly available, privacy-preserving, large-scale, realistic*** suite of tabular datasets. This dataset was ***generated using advanced tabular data generation techniques applied to an anonymized, real-world bank account opening fraud detection dataset***.
- Our model was ***trained on 50 lakhs accounts in the dataset, learning the relationships between each column and the fraudulent nature of the accounts.***
- We then ***applied this trained model to calculate the risk percentage of all accounts in the base dataset containing 10 lakh accounts.***
- As a result, ***our model successfully predicts the risk percentage*** (i.e., the likelihood of an account being fraudulent) ***for all accounts.***
- Furthermore, we concluded that, ***the accuracy rate is as high as 99%.***
- Our ***final model***, will be ***trained on 60 lakh accounts across 30 realistic features,*** leading to even ***higher accuracy.***