I have answered the following questions mentioned in the Internship Guideline PDF.

1. Data cleaning including missing values, outliers and multi-collinearity.
2. Describe your fraud detection model in elaboration.
3. How did you select variables to be included in the model?
4. Demonstrate the performance of the model by using best set of tools.
5. What are the key factors that predict fraudulent customer?
6. Do these factors make sense? If yes, How? If not, How not?
7. What kind of prevention should be adopted while company update its infrastructure?
8. Assuming these actions have been implemented, how would you determine if they work?

1. Data cleaning including missing values, outliers and multi-collinearity.

The dataset (~6.3M rows) had almost no missing values but showed some outliers.
I removed them using conditions like:

- Old and new balances both zero after a transaction,
- Sending money with zero old balance,
- Old balance same as new balance,
- Destination balance jumping unusually high.

For multicollinearity, I created features like balanceDiffOrig, balanceDiffDest, and errorOrig, checked correlations, and dropped redundant ones to keep the data simple and meaningful.

2. Describe your fraud detection model in elaboration.

I tried out different models but mainly focused on Decision Tree, XGBoost, and Random Forest because they work really well with tabular data and are easy to understand.
I trained models under different fraud-to-non-fraud ratios (1:1, 1:2, 1:3, 1:10) using SMOTE and resampling methods.
For each setup, I tracked key metrics like Recall, Precision, F1-Score, and AUC-ROC.
After comparing everything, **XGB Classifier** gave the most stable and high-performing results.

XGBoost Model - Performance Comparison

For example, here (1:1) means (fraud:non_fraud)

Metric	SMOTE (1:1) No Resampling	SMOTE (1:1) With Resampling	SMOTE (1:2) No Resampling	SMOTE (1:2) With Resampling	SMOTE (1:3) No Resampling	SMOTE (1:3) With Resampling	SMOTE (1:10) No Resampling	SMOTE (1:10) With Resampling
Recall (Fraud)	97.21%	97.36%	95.1%	96.4%	94.69%	97.06%	90.64%	95.88%
Precision (Fraud)	96.46%	96.54%	96.49%	94.33%	95.97%	92.07%	96.71%	81.21%
F1 Score	96.84%	96.95%	95.79%	95.35%	95.32%	94.5%	93.58%	87.94%
AUC	99.7%	99.7%	99.75%	99.73%	99.74%	99.74%	99.77%	99.76%

3. How did you select variables to be included in the model?

I created new features to pull out more useful information from the data, like:

- **balanceDiffOrig** = oldbalanceOrig – newbalanceOrig
- **balanceDiffDest** = oldbalanceDest – newbalanceDest
- **errorOrig** = oldbalanceOrig - amount – newbalanceOrig

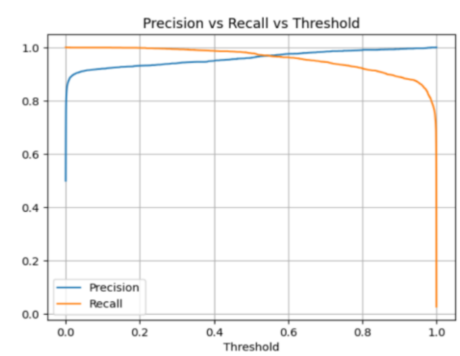
By keeping **isFlaggedFraud** didn't have much impact.

To pick the best features, I checked their importance using Random Forest's feature_importances_ attribute. I focused on features that made sense both statistically and from a business point of view.
Later, I dropped **errorOrig** during the final model run because it was leaking information — basically making the prediction unfairly easy.

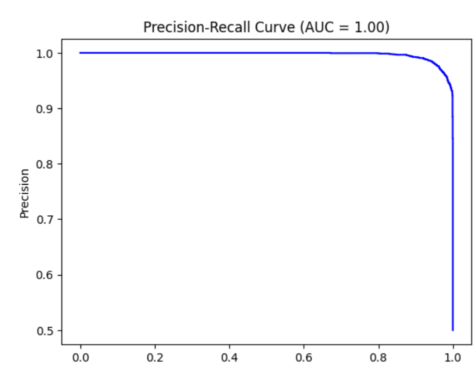
4. Demonstrate the performance of the model by using best set of tools.

I showed model performance through:

- **Confusion Matrix**
- **ROC-AUC Curve**
- **Metrics** like Precision, Recall, F1 Score, and Accuracy
- **Tabular comparisons** of results (like below examples)



I used sklearn to track everything, and organized the results into clean markdown tables to easily compare different SMOTE along with ratios, with and without resampling.

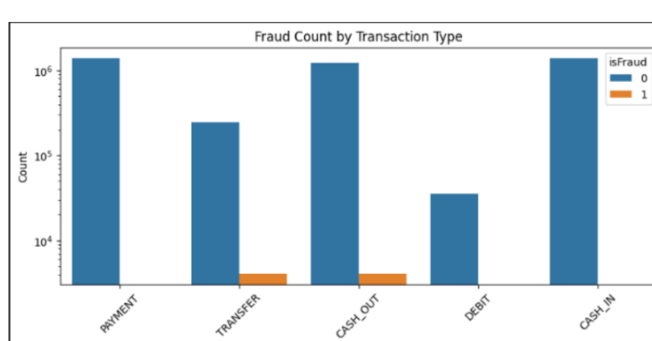


Random Forest Model - Performance Comparison

For example, here (1:1) means (fraud:non_fraud) ratio

Metric	SMOTE (1:1) No Resample	SMOTE (1:1) Resample	SMOTE (1:2) No Resample	SMOTE (1:2) Resample	SMOTE (1:3) No Resample	SMOTE (1:3) Resample
Recall (Fraud)	98.25%	98.18%	97.33%	97.55%	97.06%	97.32%
Precision (Fraud)	95.45%	95.38%	92.09%	91.02%	87.98%	87.24%
F1 Score	96.83%	96.76%	94.64%	94.17%	92.3%	92.01%
AUC	99.64%	99.64%	99.62%	99.59%	99.59%	99.58%

5. **What are the key factors that predict fraudulent customer?**



The top predictors were:

- **Transaction type** (especially CASH_OUT and TRANSFER)
- **Error in balance calculation** (errorOrig) — though it leaked information, so I avoided using it in the final model
- **Transaction amount**

- **Balance differences** (balanceDiffOrig and balanceDiffDest)

Fraudulent transactions often had unusual differences in the sender's balance after cash outs and transfers — a clear sign of possible fraud or tampering.

6. Do these factors make sense? If yes, How? If not, How not?

Absolutely, yes! These features make perfect sense:

- Fraudsters mainly target **CASH_OUT** and **TRANSFER** transactions.
- **Balance mismatches** (like errorOrig, balanceDiffOrig and balanceDiffDest) clearly hint at fraud — when the money deducted doesn't match the actual balance change.
- A **zero new balance** often means the account was completely drained — majority fraud behavior.
- And **large amounts** with strange balance shifts? Huge red flags.

7. **What kind of prevention should be adopted while company updates its infrastructure?**

To detect fraud in real-time, I'd:

1. **Use errorOrig** to flag balance inconsistencies, a strong fraud indicator.
2. **Track user transaction patterns** to spot unusual behavior like large transfers.
3. **Set limits or add verification** for high-risk transactions like **CASH_OUT** and **TRANSFER**.
4. **Log flagged transactions** for auditing and continuous improvement.

8. Assuming these actions have been implemented, how would you determine if they work?

- Conduct **A/B testing** between old and updated systems.
- Monitor the **fraud rate over time**: if it's dropping, the system is working.
- Evaluate **precision and recall** of live fraud alerts versus actual fraud cases.

Check for **customer complaints**, false positives, or financial losses—those should decline

Thank you

Regards,

Jayanta Nath

Mail: nathjayanta772@gmail.com