Ritwik Kashyap
IIIT Delhi : B.Tech CS graduate 2025
Data Scientist at ImagingIQ, Gurugram

REAL TIME FRAUD DETECTION IN ONLINE TRANSACTIONS:-

DATASET DESCRIPTION:-

Each row has 18 descriptors for a transaction that has occurred in a bank or some place.

18 features:-

X: 18 features:-

1. **TransactionID** of the transaction
2. **UserID** of the person who has made the transaction
3. **Amount of the transaction**
4. **Type of transaction**: ATM, Bank, POS, Online etc
5. **Device Type** : Laptop, mobile, Tablet
6. **Account balance** of the person
7. **Location of the person**
8. **Merchant category** : electronics, clothing, restaurants (kaha pay kiya)
9. **IP_address_flag**: if the ip used is malicious
10. **Previous fraudulent activity** : if the person has done fraud before
11. **Daily transactions count** : how many does he/she do
12. **Avg_transaction amount past 7 days**
13. **Failed transactions count past 7 days**
14. **Card type** used
15. **Transaction distance:** how far in KMs is the transaction done
16. **Authentication method** : Password, OTP, Biometric, PIN etc
17. **Risk score**: a score giving how much probable is the transaction fraud
18. **Is_weekend**: done on weekend or not

Y:-
Fraud label

ML MODEL DEVELOPEMENT:-

0) Dataset EDA

No. of samples : 50,000
No. of fraudulent transactions :
No. of non-fradulent transactions :

Fraud transactions statistics :-
Non-Fraud transactions statistics :-

Daily transaction count?

1) Feature classification:-

- **Dropped features:** does not account to predictions, just leaks information.
  (2) TransactionID, UserID
- **High cardinality categorical variables :**
  Merchant category,
- **Low cardinality categorical variables :**
  Transaction type, Device Type,
- **Continuous variables :**
  Transaction amount, Account balance,
- **Binary features :**
  IP_address_Flag, Previous_Fraudulant_Activity,

2) Feature preprocessing:-

Imputation: fill the missing values in columns

- Numerical features: fill by **median** (robust to outliers) (or by mean imputation or KNN imputation) of the rest of the column.
  Other options: interpolation (for time series), regression to predict.
- Categorical features:
  Make a “Missing” or “Unknown” category
  Fill with Mode, most common category (when “Missing” may not have value)
- Binary features:
  Fill all with 0
  Fill all with the mode of column.

3) Feature encoding

Trees need encoding because even though they compute information gain, they still make splits based on numeric thresholds, and categorical strings have no numeric meaning.

- High cardinality features: features that are categorical, but have very high no. of categories, so we cannot convert them using OHE, as they will blow up the columns.
  Target encoding or frequency encoding is used here
  Target encoding: encode based on the target frequencies for a particular category, this is more better but has data leakage problems
  Frequency encoding: encodes based on the frequency of categories, as does not take into account the relationship between category and labels, but works well.

- Low categorical features: features that are categorical, but categories are limited or less.
  One Hot Encoding (OHE) is used for these.
- Continuous features: these are numeric features, natural numbers, they don't need encoding
  Caution: if in dataset integers are meant in terms of categories then they need to be changed to OHE or frequency or target encoding.

NOTE: after changing these features to encoded format, we DON'T need to do scaling or any similar operations in tree based models, as trees don't care about distances.
LoR, LR, SVM, KNN, Neural Networks need scaling of features.

Do we need a correlation heatmap matrix?
No, XGBoost models do not need such analysis for training, as they are robust to multicollinearity and automatically remove the redundant features from the feature set.
But it is useful for EDA.
As, if two features are 100% correlated we can easily remove one.
If a feature risk_score is 100% correlated to frau_label we can remove that feature.

Linear regression, Logistic regression do need a correlation analysis.

Do we need PCA (feature selection)?
No, PCA would in most cases reduce performance only as they combine the features, would disturb the encodings also, trees are not performing any matrix operations so dimensionality is not a problem.

Do we need SMOTE?

Real fintechs never use SMOTE.

SMOTE would generate imaginary transactions and would destroy relationships, and can even cause overfitting.
As frauds do not add linearly, the average resultant transaction may not be a fraud, also frauds do not have a clear boundary, and patterns, they are adversarial in nature, i.e. fraudsters try to trick the algorithm by sending fake examples.

So, we use class weighting, i.e. setting the scale_pos_ratio as the ratio of frauds and non-frauds to give more importance to frauds then non frauds.

How to do all this?
Sklearn preprocessing pipeline or pandas

For production a preprocessing pipeline is better as we can simply save the full model pipeline along with the weights in one joblib file and transform the incoming data in the same way as the training pipeline to ensure consistency.

4) Model hyperparameters:-

Model: XGBoost hyperparameters:

Tunable

- **n_estimators**: no. of trees
- **max_depth**: max no. of levels in a single tree
- **learning rate**
- **Model_subsample**: no. of rows in sample tree is trained
- **Model_colsample_bytree**: no. of columns in sample tree is trained

Non Tunable

- **use_label_encoder**: removed in versions now, just encodes the labels
- **eval_metric**: “logloss”
- **scale_pos_weight**

eval_metric : logloss **:** as binary cross entropy loss is the fundamental choice for binary classification problems, and it is differentiable, and works with imbalance dataset also
We can also use hinge loss also (like in SVM)
It is the internal
scale_pos_weight : class weighting, gives more weight to the minority class, so that model pays more attention to it. It biases the tree splits toward detecting fraud

model_subsample when low gives more generalization, and reduces overfitting

model_colsample_bytree : forces model to use different subsets of features, reduces overfitting, correlation between trees.

5) Hyperparameter selection :-

1) GridSearchCV: we will select the best model using GridSearchCV, that takes in a grid of parameters like n_estimators, max_depth, learning rate and then gives you the final model based on some evaluation criteria.

Here scoring is based on : **“average_precision” or “PR-AUC”**

NOTE 1:-

PR-AUC measures the area under P, R curve,
It says "How good is the model at detecting fraud without raising too many false alarms?"

ROC-AUC whereas measures AUC under TPR-FPR
It says "How good is my model at separating fraud from non-fraud overall?"
(This is better)

Basically, ROC-AUC is is biased by TN cases, as FPR has TN in denominator, and Fraud datasets have too many TN, so ROC-AUC would be high for a bad model also, so we need to have PR-AUC, as it will account for FP and FN and generally give more accurate model

NOTE 2:-

precision and recall, F1-score, accuracy is useless mostly

Most important is recall : TP/ TP + FN , precision : TP / TP + FP

Recall pays attention to False negatives (fraud tha par, negative bataya)
Precision pays attention to False positives (fraud nhi tha par, positive bataya)

Ek bar ko sahi transaction fraud mai aajaye, koini, admin ka workload hi badhega, but fraud wali nahi aayi to dikkat hojayegi

So, we need a high recall

ANALYSIS:-
This is giving hyperparameters as:-
{'model__colsample_bytree': 1.0, 'model__learning_rate': 0.1, 'model__max_depth': 7, 'model__n_estimators': 300, 'model__subsample': 1.0}

This is wrong as rows and columns samples must not be 1, that is each tree should not see full dataset, there should be some randomness, so we cannot accept these hyperparameters.

2) Manual setting:-

```
# subsample = 0.8
# colsample_bytree = 0.7-0.9
# max_depth = 4-6
# learning_rate = 0.05
# n_estimators = 300
# scale_pos_weight = ratio
```

6) Training and validation:-

Results set 1 on manual parameter setting:-

```
Accuracy: 0.8771
Confusion Matrix:
 [[6785    2]
 [1227 1986]]
Classification Report:
              precision    recall  f1-score   support

           0       0.85      1.00      0.92      6787
           1       1.00      0.62      0.76      3213

    accuracy                           0.88     10000
   macro avg       0.92      0.81      0.84     10000
weighted avg       0.90      0.88      0.87     10000
```

7) Output probabilities calibration:-

Done using sigmoid function on the output probabilities

8) Manual testing of the model/ self examination:-

I have observed that

On changing the transaction amount to very high, there is little change in the predictions

On changing the integer columns like “frauds in last 7 days by that person”, changing from 3 to 5 changes output probability from 0.13 to 0.24 to 0.98

Recall of the model is poor at 0.62, which says that it cannot detect all the frauds correctly, it is marking frauds as non-frauds also
I.e. it is missing 38% of the frauds => NOT ACCEPTABLE

HOW TO SOLVE THIS, IMPROVE MODEL

SOFTWARE ENGINEERING PART:-

MODEL INFERENCE API DEVELOPEMENT:-

Way 1: Either I can create dummy transactions in a DB and fetch in every 10 seconds for new transactions , **(DB Polling)**

Way 2 (will do this): I can introduce real time streaming, using **Kafka (a streaming pipe)** and create a generator which generates transactions real time and they go to the FastAPI service and get processed and come back to Kafka, and get streamed back to the frontend dashboard.

To figure out how to add LLMs for explainability in this architecture.

Also add CI/CD and MLFlow for automated retraining of XGBoost model and versioning of the model periodically

EXPLAINABILITY USING LLMs OR SOMETHING ELSE:-

FRONTEND DEVELOPEMENT:-

SOFTWARE TESTING:-