Fraud Detection Model Documentation

Overview

This project addresses the binary classification problem of detecting fraudulent transactions using two modeling approaches:

1. Traditional Machine Learning: Random Forest
2. Deep Learning: MLP with categorical embeddings

The dataset includes transaction-related features (e.g., amount, time step, customer, merchant, category), as well as behavioral and statistical aggregates derived through custom feature engineering.

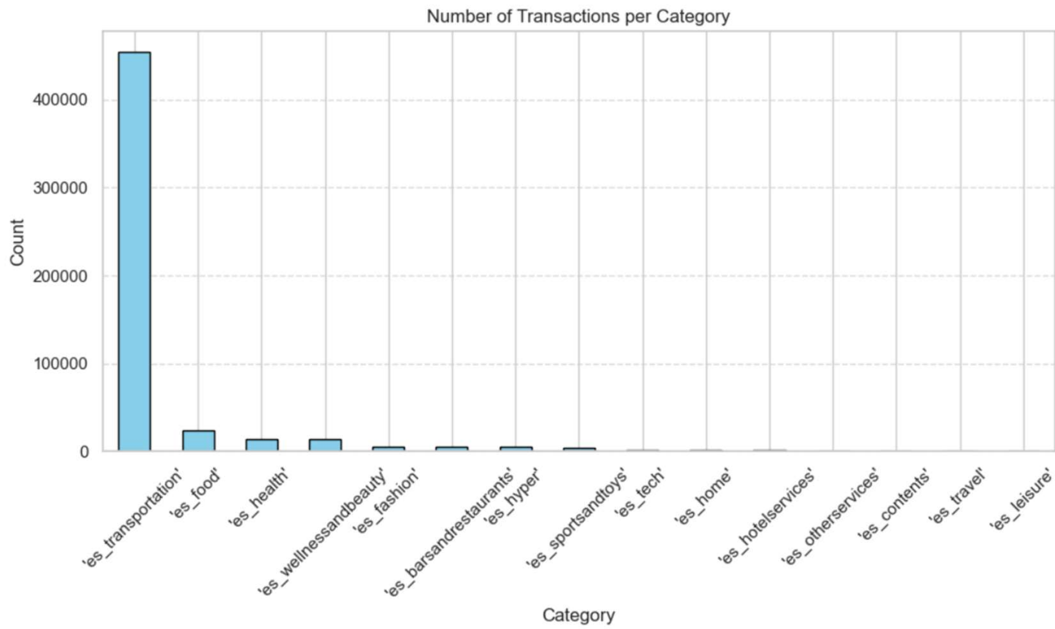
Feature Engineering

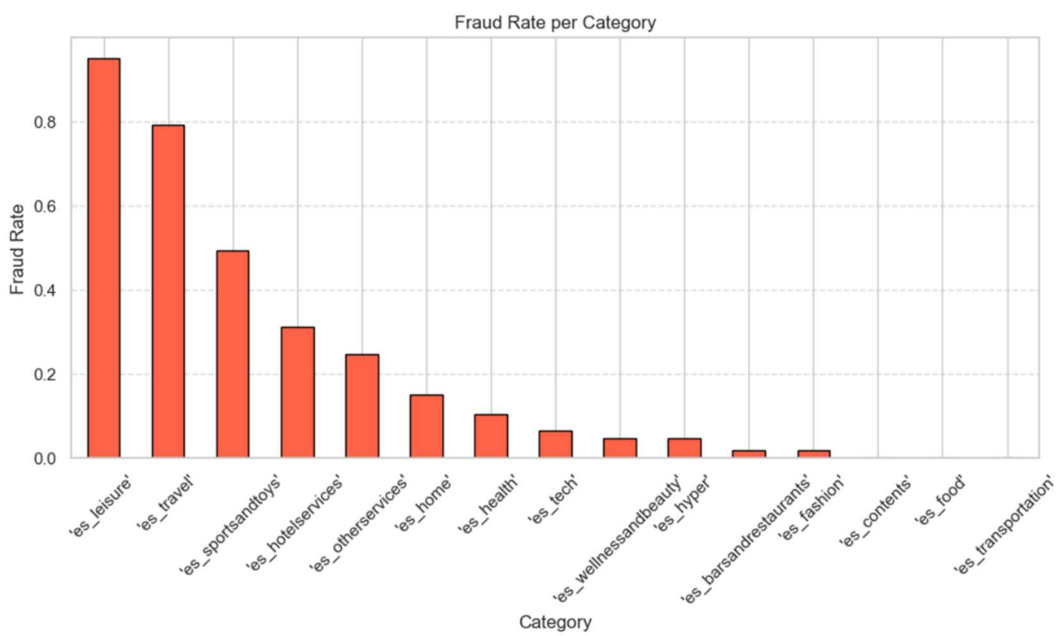
Firstly, I tried encoding all 50 merchants to OHE and used random forest on it, but it becomes too much overhead and memory consuming.

Then, I used-

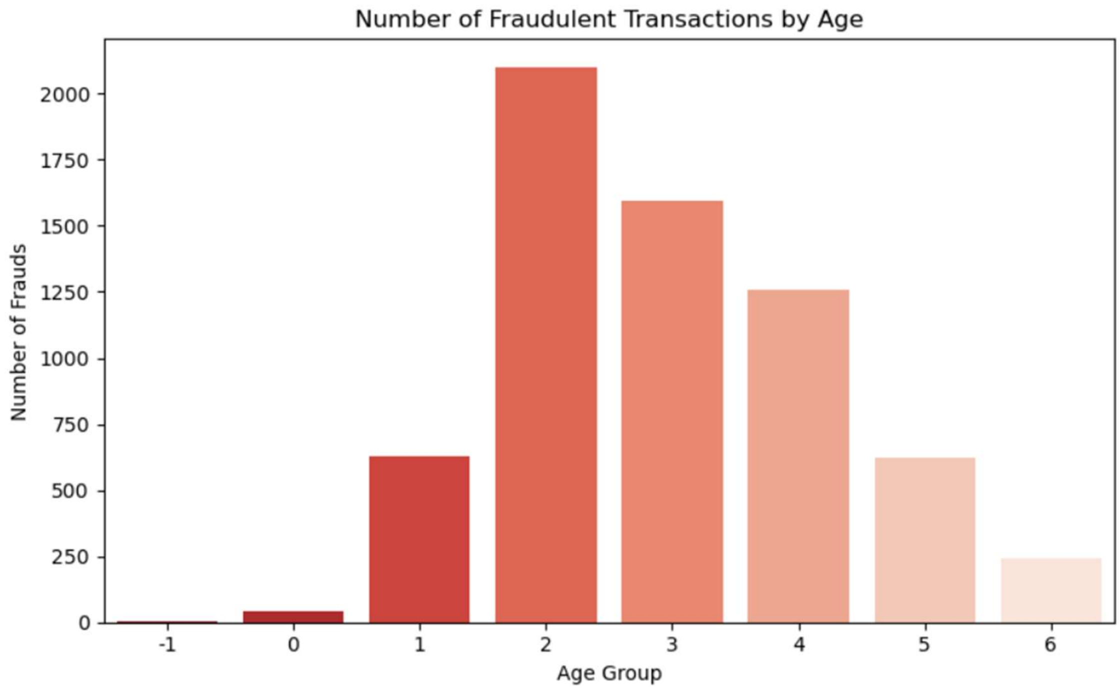
Implemented via a custom FraudFeatureEngineerDL transformer:

- Encodes and cleans data ('U' age values, quote removal)
- Derives features:
  - Time: hour_of_day, day, is_night
  - User behavior: transaction count, average/std of amount, z-score
  - Risk: is_high_amt, merchant_fraud_rate, category_fraud_rate
- Encodes category using LabelEncoder (used for embeddings in DL model)
- Drops ID/low-correlation features post-processing

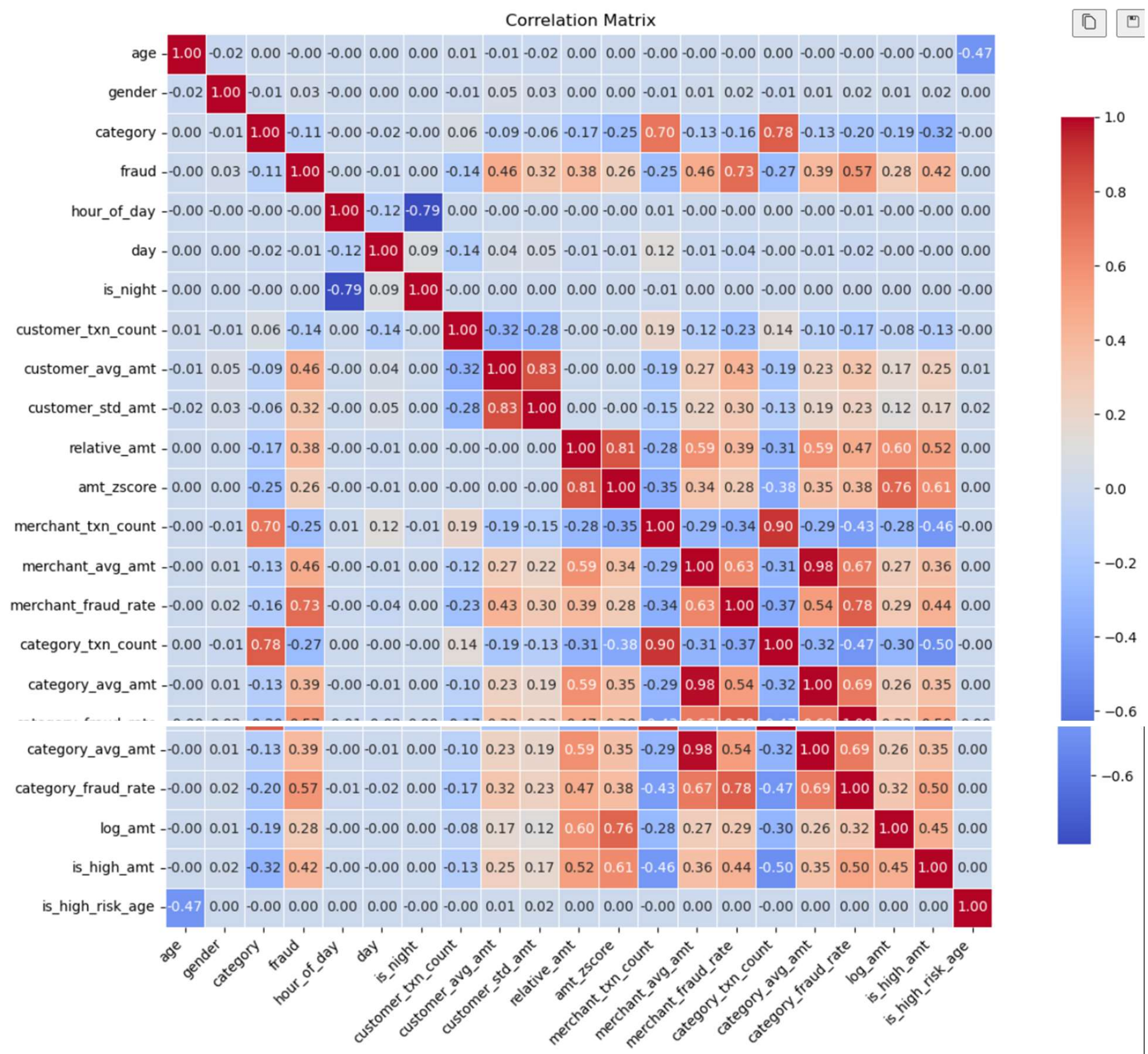




- We can observe from above 2 graphs that es_transport has more transactions but less fraud rate and similarly for es_leisure. So, from this we can take category_fraud_rate as extra feature to add more weight to categories doing more fraud.



- Age group from 2 to 4 are doing more frauds, so I have added is_high_risk_age as another feature
- Can also use, merchant_customer_combo as another feature- it gives unque pairs of both doing frauds, I got ~4k unique pairs of them out of 4000 customers and 50 merchants.
- Corresponding correlation matrix-


Correlation Matrix
age
gender
category
fraud
hour_of_day
day
is_night
customer_txn_count
customer_avg_amt
customer_std_amt
relative_amt
amt_zscore
merchant_txn_count
merchant_avg_amt
merchant_fraud_rate
category_txn_count
category_avg_amt
category_fraud_rate
log_amt
is_high_amt
is_high_risk_age
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

```
fraud                  1.000000
merchant_fraud_rate    0.731215
category_fraud_rate    0.571078
customer_avg_amt       0.464966
merchant_avg_amt       0.463433
is_high_amt            0.415371
category_avg_amt       0.391471
relative_amt           0.384656
customer_std_amt       0.324238
log_amt                0.284242
category_txn_count    -0.267877
amt_zscore             0.259546
merchant_txn_count    -0.246864
customer_txn_count    -0.143394
category              -0.113529
gender                 0.025252
day                   -0.011875
age                   -0.003412
hour_of_day           -0.001914
is_night               0.001823
is_high_risk_age       0.001499
Name: fraud, dtype: float64
```

- low_corr_features = [
    'gender',
    'day',
    'age',
    'hour_of_day',
    'is_night',
    'is_high_risk_age'
  ] are dropped due to cor val < 0.05

Engineered Features (Used in Model)

1. **hour_of_day**
   Calculated as step % 24. Captures the hour of the day when the transaction occurred.
2. **day**
   Calculated as step // 24. Represents the day number in the dataset timeline.
3. **is_night**
   A binary feature:

1 if the transaction occurred between 12:00 AM and 6:00 AM (potentially high-risk hours), else 0.

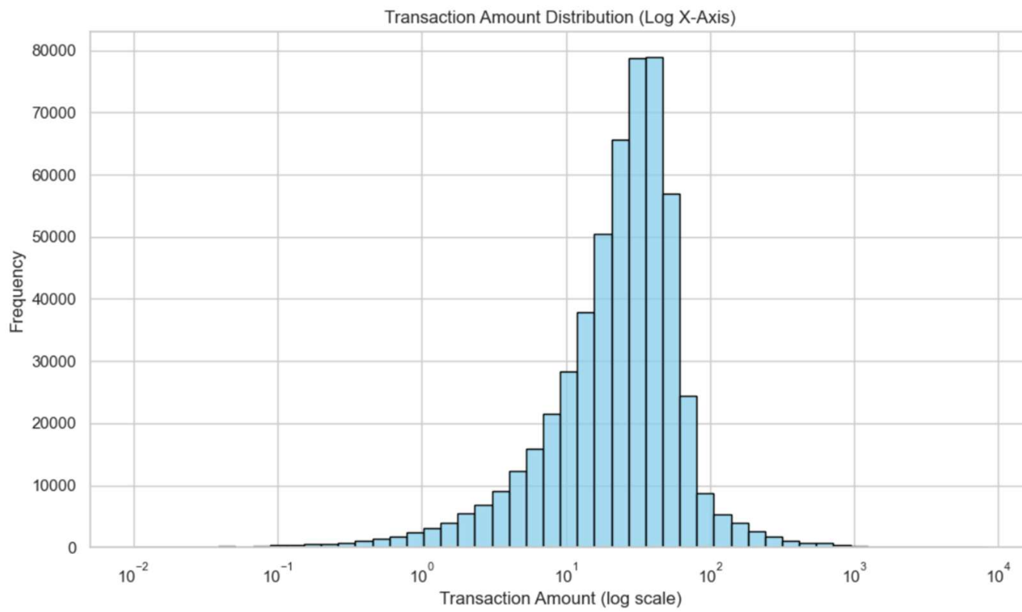
4. **gender**
   Mapped to numeric values:
   'M' → 0, 'F' → 1. Unknown → -1.
5. **customer_txn_count**
   Total number of transactions made by a customer. Indicates transaction volume history.
6. **customer_avg_amt**
   Average transaction amount for the customer across all transactions.

```
: df["amount"].describe().astype(int)

: count    535178
  mean         37
  std         112
  min           0
  25%          13
  50%          26
  75%          42
  max        8329
  Name: amount, dtype: int64
```

7. **customer_std_amt**
   Standard deviation of transaction amounts for the customer. If undefined, set to 0.
8. **relative_amt**
   Ratio of current transaction amount to the customer's average amount.
9. **amt_zscore**
   Z-score of the transaction amount relative to customer's average and standard deviation. Helps in detecting anomalies.
10. **merchant_txn_count**
    Total number of transactions conducted with this merchant.
11. **merchant_avg_amt**
    Average amount per transaction for the merchant.
12. **merchant_fraud_rate**
    Historical fraud rate for this merchant (based on training data).
13. **category_txn_count**
    Number of transactions belonging to the same category.
14. **category_avg_amt**
    Average transaction amount for the category.

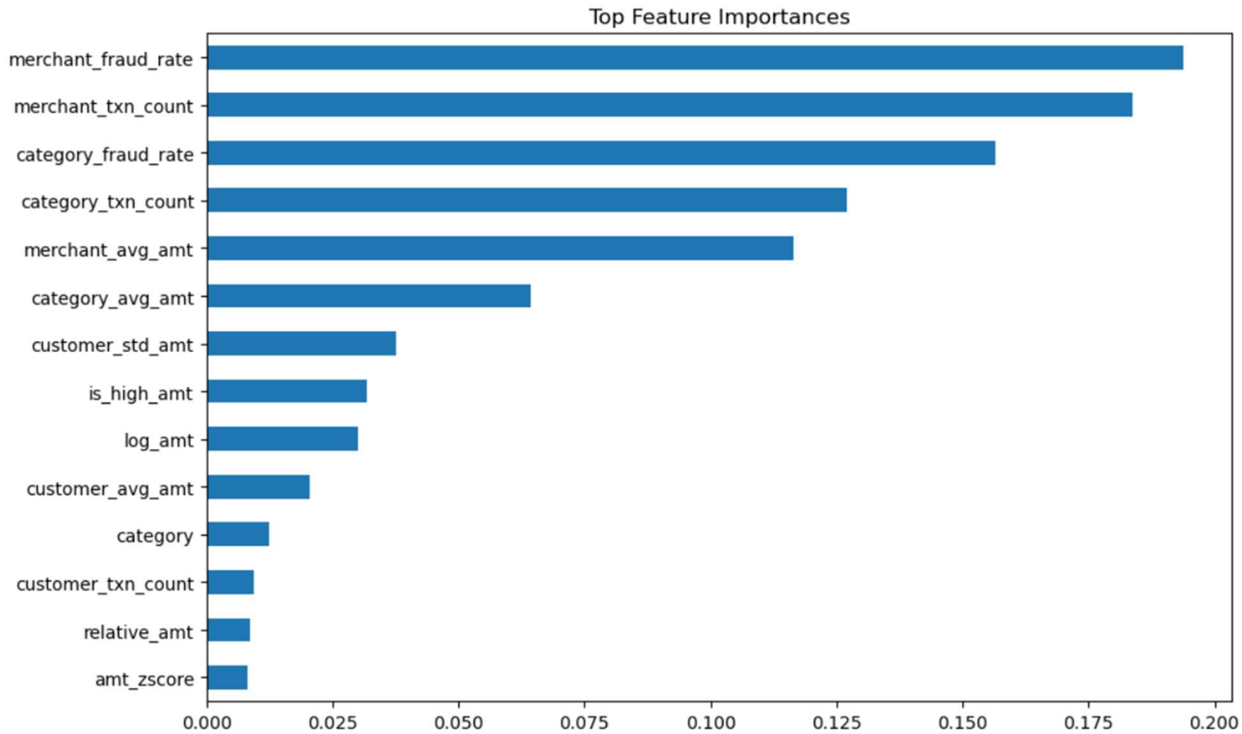
15. **category_fraud_rate**
    Historical fraud rate for the transaction category.
16. **log_amt**
    Log-transformed transaction amount: log(1 + amount). Helps normalize skewed distributions.



17. **is_high_amt**
    Binary flag indicating whether the amount is above the 95th percentile (i.e., unusually high).
18. **is_high_risk_age**
    Binary feature: 1 if age group is '2' or '3' — empirically observed to have higher fraud risk.
19. **category (encoded)**
    Label-encoded version of the merchant category. This is passed through an embedding layer in the deep learning model.

DROPPED- zipcodes, step, customer, merchant, amount

Feature Importance



Model Architectures

1. Random Forest

- Model: RandomForestClassifier(n_estimators=100, class_weight='balanced', n_jobs=-1)
- Pipeline: FraudFeatureEngineer → RandomForest
- Serialization: joblib

2. Deep Learning (MLP + Embeddings)

- Model: MLPWithEmbeddings
- Framework: PyTorch
- Architecture:
  - Embedding layer for category (learned embedding)
  - Fully connected layers: [64 → 32 → 1]
  - Activations: ReLU, Sigmoid

- BatchNorm, dropout can be added

- Loss: BCELoss
- Optimizer: Adam
- Scaler: StandardScaler applied to numerical features
- Serialization: torch.save for model, scaler, feature engineer

Performance

1. **Deep Learning (MLP + Embedding)**

```
📊 Training Metrics:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00    422963
         1.0       0.92      0.84      0.88      5179

    accuracy                           1.00    428142
   macro avg       0.96      0.92      0.94    428142
weighted avg       1.00      1.00      1.00    428142

Confusion Matrix:
 [[422605    358]
 [   854   4325]]
ROC-AUC Score: 0.999111838182553
```

- Strengths:
  - High precision and recall on minority class (fraud)
  - Learns complex feature interactions (e.g. categorical embeddings + behavior features)
  - Model Size is just 28kb.

2. **Random Forest**

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    105738
           1       0.84      0.80      0.82      1298

    accuracy                           1.00    107036
   macro avg       0.92      0.90      0.91    107036
weighted avg       1.00      1.00      1.00    107036
```

```
ROC-AUC Score: 0.9886008184721249
```

- Strengths:
  - Robust performance out-of-the-box
  - Faster to train, no GPU required
- Weakness:
  - Slightly lower recall and F1 for fraud class
  - Struggles with interactions between high-cardinality categories
  - Model Size is 42mb