"This project aims to develop a machine learning model for detecting online fraud transactions. By analyzing patterns in transaction data, we will identify fraudulent activities and predict potential fraud. The goal is to build a model with high accuracy and recall to minimize both false positives and false negatives. This solution will help in improving security and reducing financial losses for online platforms."

```
# import libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# load data
df = pd.read_excel(r"E:\data science with python & ai\all projects\
machine learning projects\weather data\
Updated_Inclusive_Indian_Online_Scam_Dataset (1).xlsx")
df.head()
```

```
   transaction_id  customer_id  merchant_id    amount
transaction_time  \
0             1.0     684415.0       2028.0  1262.770 2023-11-24
22:39:00
1             2.0     447448.0       2046.0  2222.928 2024-03-30
16:18:00
2             3.0     975001.0       2067.0  7509.832 2024-03-07
18:27:00
3             4.0     976547.0          NaN  2782.965 2024-02-01
00:58:00
4             5.0     935741.0       2044.0       NaN 2023-12-22
18:42:00

   is_fraudulent   card_type   location purchase_category
customer_age  \
0            0.0       Rupay  Bangalore               NaN
28.0
1            0.0  MasterCard      Surat               POS
62.0
2            0.0  MasterCard  Hyderabad               POS
24.0
3            0.0       Rupay  Hyderabad           Digital
62.0
4            0.0         NaN  Bangalore           Digital
19.0

           fraud_type
```

```
0      Identity theft
1             Malware
2             Malware
3  Payment card fraud
4                scam

# delete column which is need
update_df =
df.drop(columns=["transaction_id","customer_id","merchant_id","transac
tion_time","location","purchase_category"],inplace=True)

df.head()

     amount  is_fraudulent   card_type  customer_age
fraud_type
0  1262.770            0.0       Rupay          28.0      Identity
theft
1  2222.928            0.0  MasterCard          62.0
Malware
2  7509.832            0.0  MasterCard          24.0
Malware
3  2782.965            0.0       Rupay          62.0  Payment card
fraud
4       NaN            0.0         NaN          19.0
scam

# check missing value
df.isnull().sum()

amount           692
is_fraudulent    725
card_type        567
customer_age     668
fraud_type       498
dtype: int64

# remove missing value
df.fillna(df["amount"].mean(),inplace=True)
df.fillna(df["is_fraudulent"].mean(),inplace=True)
df.fillna(df["customer_age"].mean(),inplace=True)
df.fillna(df["card_type"].mode(),inplace=True)
df.fillna(df["fraud_type"].mode(),inplace=True)

df.dtypes

amount           float64
is_fraudulent    float64
card_type         object
customer_age     float64
fraud_type        object
dtype: object
```

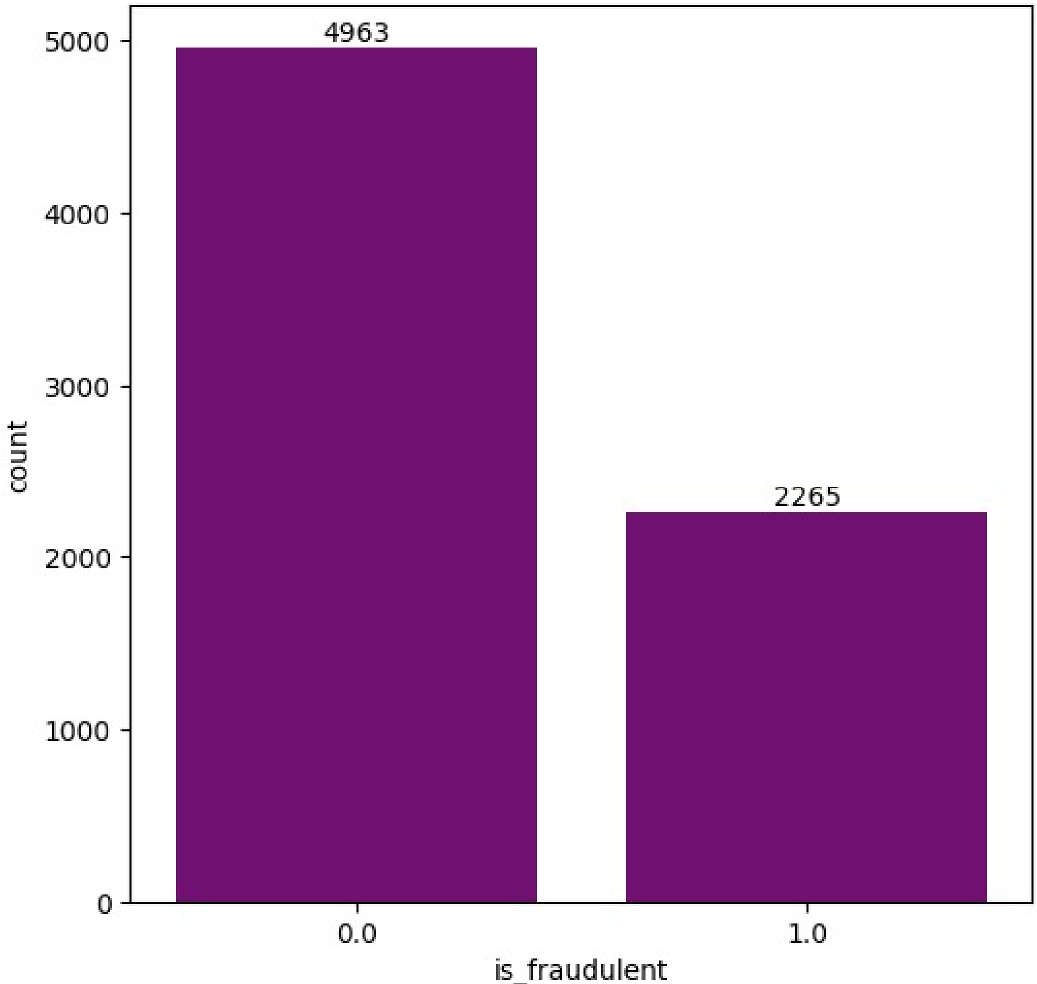
```python
df.describe()
```

```
             amount  is_fraudulent  customer_age
count   7953.000000    7953.000000   7953.000000
mean    6149.985080     560.920933    556.277635
std     3635.761569    1770.216337   1694.009825
min       84.711000       0.000000     18.000000
25%     3385.320000       0.000000     32.000000
50%     6149.985080       0.000000     46.000000
75%     8428.230000       1.000000     59.000000
max    17960.976000    6149.985080   6149.985080
```

```python
df.info()
```

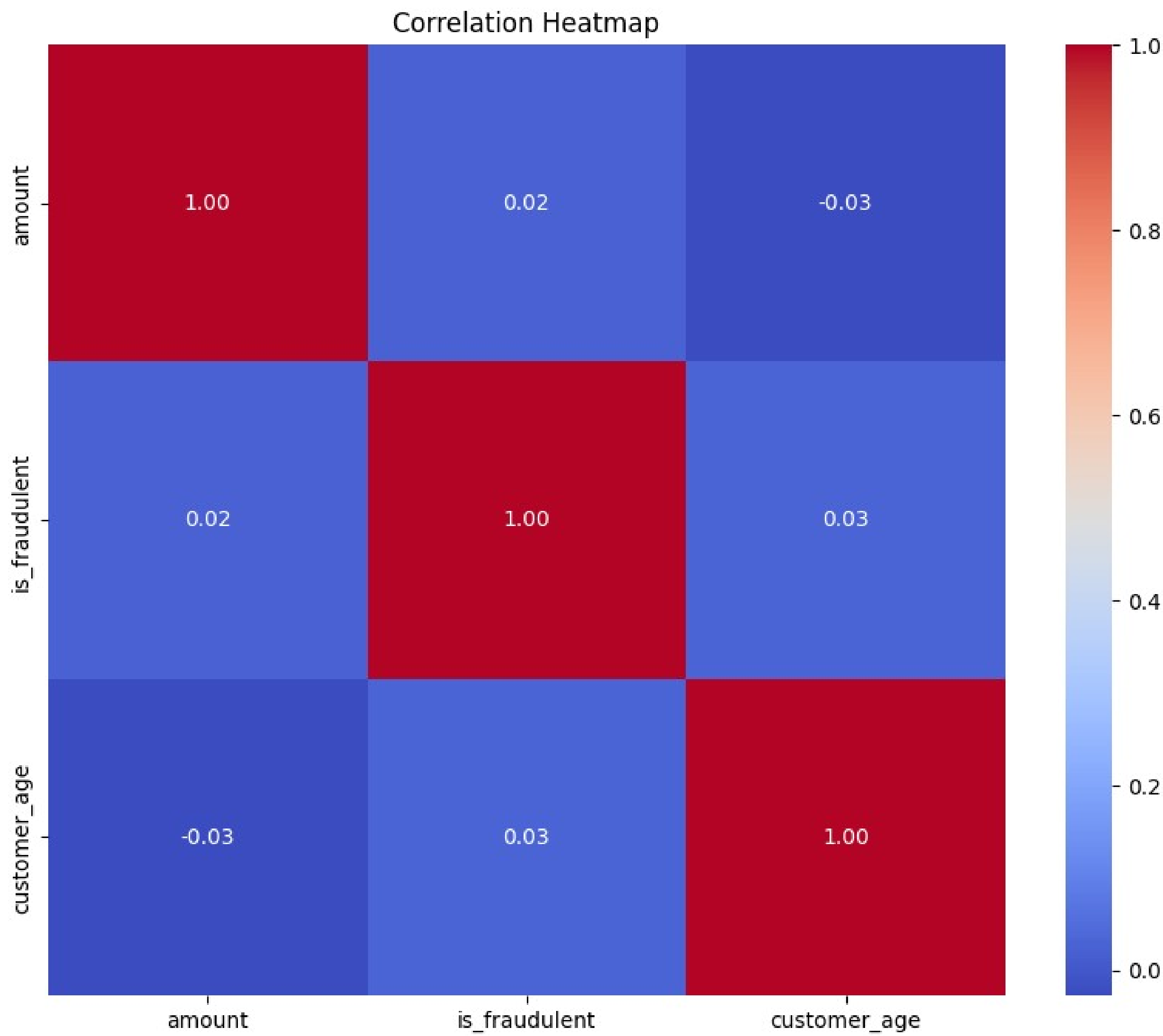
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7953 entries, 0 to 7952
Data columns (total 5 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   amount         7953 non-null   float64
 1   is_fraudulent  7953 non-null   float64
 2   card_type      7953 non-null   object 
 3   customer_age   7953 non-null   float64
 4   fraud_type     7953 non-null   object 
dtypes: float64(3), object(2)
memory usage: 310.8+ KB
```

```python
# isin use for incoreect fixed value like 0,0,1,0,1,0,45487
df = df[df['is_fraudulent'].isin([0, 1])]

# Fraud vs Non-Fraud
plt.figure(figsize=(6,6))
ax = sns.countplot(x="is_fraudulent",data=df,color="purple")
for bars in ax.containers:
    ax.bar_label(bars)
```

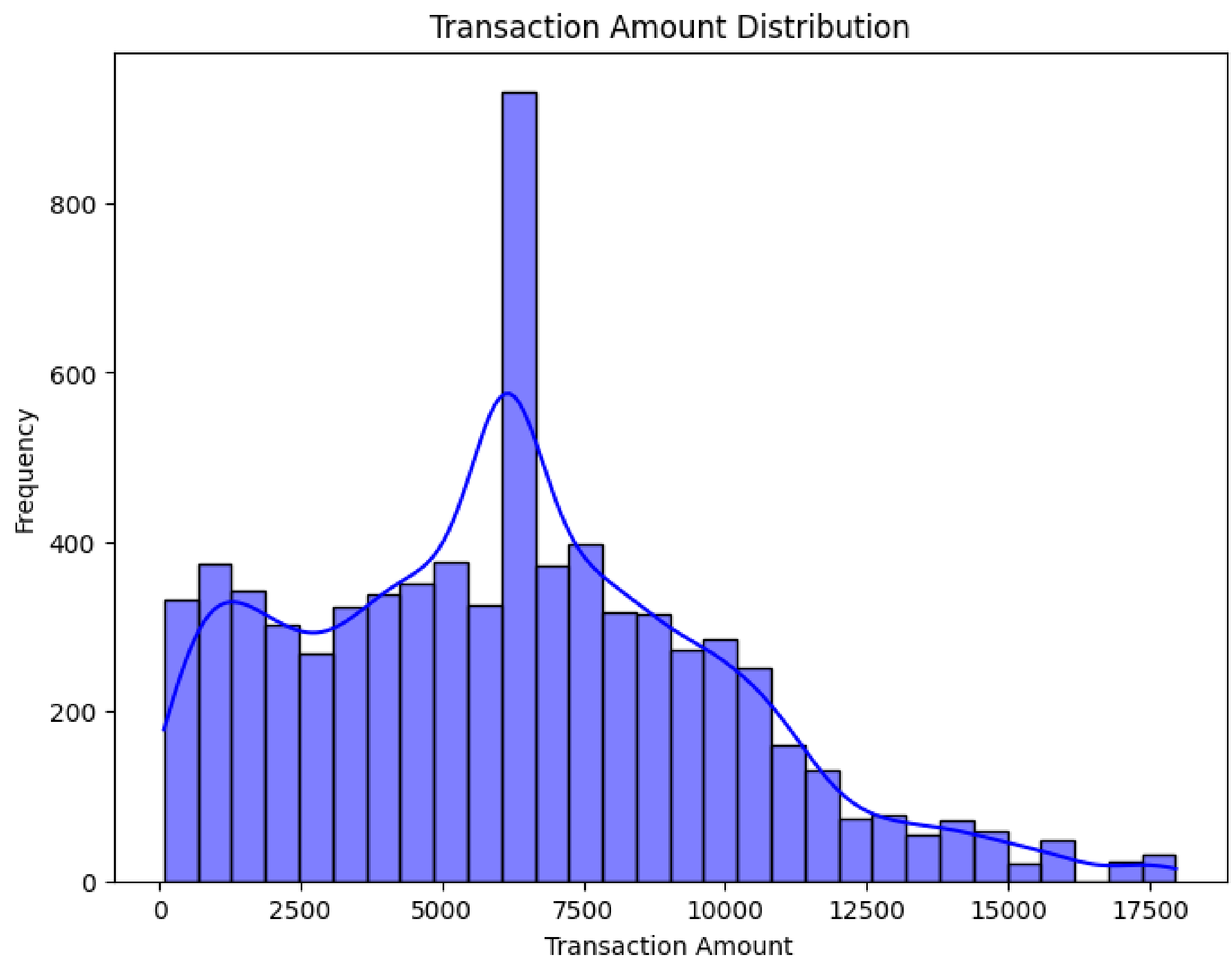


In the dataset, there are 4963 non-fraudulent transactions (0) and 2265 fraudulent transactions (1). This indicates that the majority of transactions are non-fraudulent, while a smaller proportion are flagged as fraudulent.

```
# Correlation Heatmap
numeric_df = df.select_dtypes(include=[float, int])

# Plot the heatmap with only numeric data
plt.figure(figsize=(10,8))
sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```

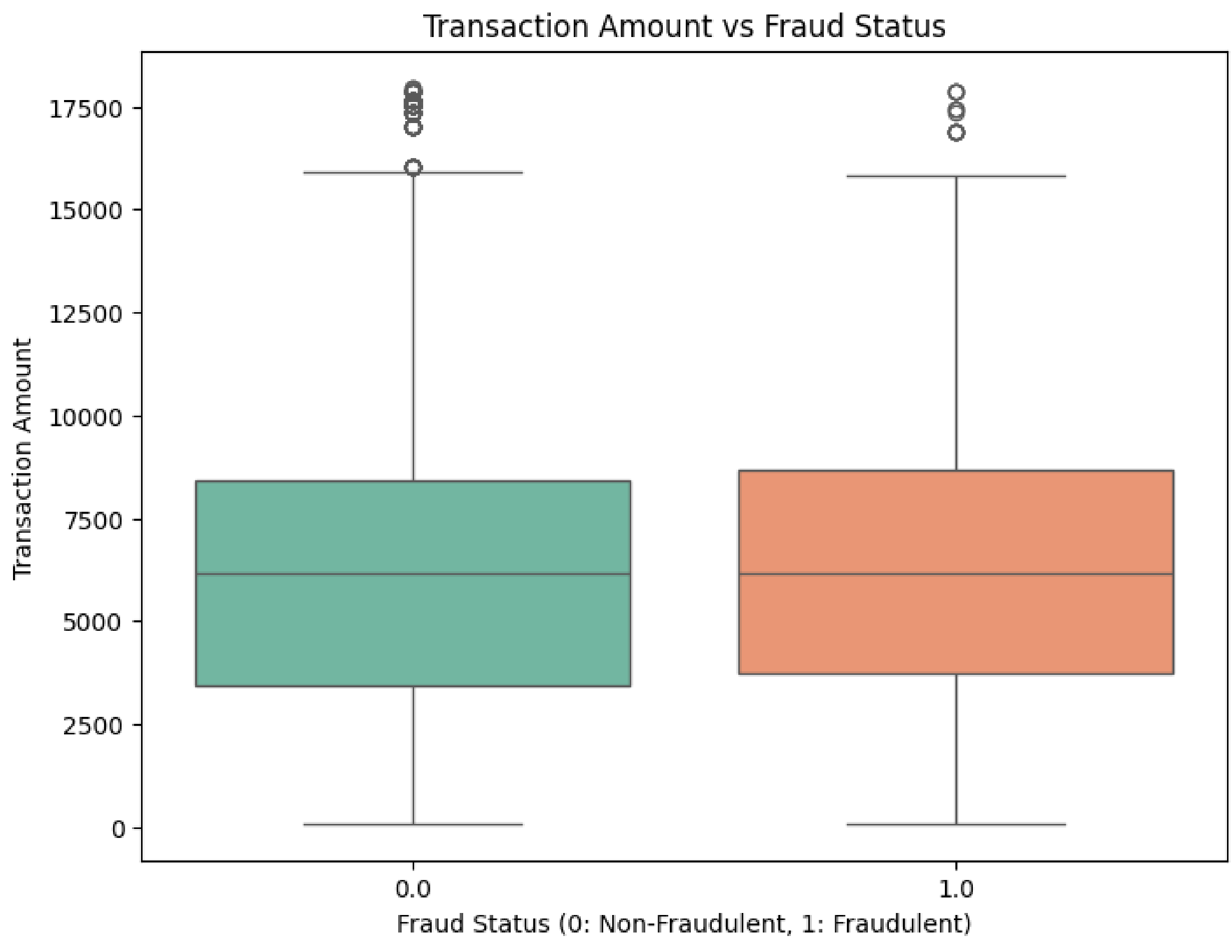



```python
# Distribution of Transaction Amounts
plt.figure(figsize=(8,6))
sns.histplot(df['amount'], kde=True, color='blue', bins=30)
plt.title('Transaction Amount Distribution')
plt.xlabel('Transaction Amount')
plt.ylabel('Frequency')
plt.show()
```




```
# Boxplot to see the relationship between Transaction Amount and Fraud
Status
plt.figure(figsize=(8,6))
sns.boxplot(x='is_fraudulent', y='amount', data=df, palette='Set2')
plt.title('Transaction Amount vs Fraud Status')
plt.xlabel('Fraud Status (0: Non-Fraudulent, 1: Fraudulent)')
plt.ylabel('Transaction Amount')
plt.show()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_12036\4080239114.py:3:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x='is_fraudulent', y='amount', data=df, palette='Set2')
```




```
# we need to change catagorical data to numerical data

# Replace non-categorical values (like 6149.98508) with NaN in
card_type and fraud_type
df['card_type'] = df['card_type'].apply(lambda x: np.nan if
isinstance(x, (int, float)) else x)
df['fraud_type'] = df['fraud_type'].apply(lambda x: np.nan if
isinstance(x, (int, float)) else x)

# Optionally, fill missing values (if you know the correct values to
fill, you can do so, or use 'mode' to fill with the most frequent)
df['card_type'] = df['card_type'].fillna(df['card_type'].mode()[0])  #
Fill with most frequent value
df['fraud_type'] = df['fraud_type'].fillna(df['fraud_type'].mode()[0])
# Fill with most frequent value

# Now both card_type and fraud_type should only contain valid string
values

from sklearn.preprocessing import LabelEncoder
```

```
# Create a label encoder for card_type
card_type_encoder = LabelEncoder()

# Convert card_type to numerical values
df['card_type'] = card_type_encoder.fit_transform(df['card_type'])

# Create a label encoder for fraud_type
fraud_type_encoder = LabelEncoder()

# Convert fraud_type to numerical values
df['fraud_type'] = fraud_type_encoder.fit_transform(df['fraud_type'])

df.head()
```

	amount	is_fraudulent	card_type	customer_age	fraud_type
0	1262.77000	0.0	0	28.0	0
1	2222.92800	0.0	0	62.0	1
2	7509.83200	0.0	0	24.0	1
3	2782.96500	0.0	0	62.0	2
4	6149.98508	0.0	0	19.0	4

```
# Split the data into features and target
X = df.drop('is_fraudulent', axis=1)
y = df['is_fraudulent']

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# scale the data
# Apply scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# model selection
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train_scaled, y_train)
```

```
RandomForestClassifier(random_state=42)
```

```
y_pred = model.predict(X_test_scaled)
y_pred
```

```
array([0., 0., 1., ..., 0., 1., 1.])
```

```
# evaluation the model
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
[[953  17]
 [  5 471]]
              precision    recall  f1-score   support

         0.0       0.99      0.98      0.99       970
         1.0       0.97      0.99      0.98       476

    accuracy                           0.98      1446
   macro avg       0.98      0.99      0.98      1446
weighted avg       0.99      0.98      0.98      1446
```

```python
import joblib
joblib.dump(model, 'fraud_detection_model.pkl')
```

```
['fraud_detection_model.pkl']
```

Key Metrics Breakdown: Accuracy: 98% This means the model correctly classifies transactions (both fraudulent and non-fraudulent) 98% of the time. A high accuracy indicates that the model is performing well overall.

Precision (Fraudulent Transactions): 97% Out of all the transactions predicted as fraudulent, 97% were actually fraudulent. High precision ensures that the model is not wrongly flagging legitimate transactions as fraudulent.

Recall (Fraudulent Transactions): 99% Out of all the actual fraudulent transactions, the model correctly identified 99%. A high recall indicates that the model is good at detecting fraud and minimizing missed fraudulent transactions.

F1-Score (Fraudulent Transactions): 98% This combines precision and recall, providing a balanced score for fraud detection. It suggests that the model performs well in both detecting fraud and avoiding false positives.

False Negatives (FN): Only 5 fraudulent transactions were misclassified as legitimate. This is excellent because it means the model rarely misses a fraud case.

False Positives (FP): 17 legitimate transactions were wrongly classified as fraudulent. Although the number is low, reducing false positives would further improve the user experience (less inconvenience to customers and merchants).

How to Reduce Fraud: Improve Model Training: Enhance the feature engineering process by including more relevant features (e.g., time of transaction, device information, merchant type) that could help better distinguish between fraudulent and non-fraudulent transactions.

Resampling Techniques: If fraud cases are rare in your dataset, consider using oversampling (SMOTE) or undersampling techniques to balance the class distribution. This could improve the model’s ability to detect fraud without increasing false positives.

Ensemble Methods: Combine multiple models like Decision Trees, Random Forests, or XGBoost to further improve detection and reduce the chances of both false positives and false negatives.

Threshold Adjustment: You can adjust the decision threshold for classification, aiming to minimize false negatives (missed fraud cases), which could slightly increase false positives but make the model more sensitive to fraud detection.

Conclusion: Your model is performing well with an overall accuracy of 98% and excellent detection of fraud (99% recall). By fine-tuning it further using the suggestions above, you can continue to reduce fraud detection errors and improve user experience.