CREDIT FRAUD DETECTION

Presented by : ECDS TEAM 4 (Chye Rui Teng Reina, Low Shi-Jie Arissa)

CONTENTS

Motivation

Which factors are more important in determining credit fraud?

Data Preparation

Cleaning of data

EDA

Analysing variables

Models

- Lasso Regression
- XGBoost

Secondary EDA

Analysing variables on our **balanced** dataset

Findings

Conclusions and Data-driven insights

PROBLEM FORMULATION

How can we detect fraudulent e-commerce transactions using machine learning models based on transaction data?

- Are we able to identify patterns in fraudulent transactions and which aspect of credit transactions do these patterns arise?
- Which model would be best to predict if a credit transaction is fraudulent based on the identified factors?

MOTIVATIONS

- Rise in Digital Transactions
- Technological Advancements
- Data Breaches and Identity Theft

By identifying fraud early, organizations can mitigate damage, safeguard sensitive information, and preserve the integrity of financial systems.

DATA PREPARATION

CLEANING OF DATA

```
creditdata.dropna(inplace = True) # removing null rows
print(creditdata.info())
```

1 Removing of null rows

```
credit_data = pd.DataFrame(creditdata)
credit_data = credit_data.drop(columns=['Transaction ID', 'Date', 'Day of Week', 'Gender'])
credit_data
```

2 Removal of negligible columns like Transaction ID, Day of Week and Gender

In [ ]:

```
# adding new columns, residence = transaction country and transaction country = shipping address
credit_data['Residence = Transaction Country'] = credit_data['Country of Residence'] == credit_data['Country of
credit_data['Transaction country = shipping address'] = credit_data['Country of Transaction'] == credit_data['Sh

# cleaning Amount column
credit_data['Amount'] = credit_data['Amount'].astype(str).str.replace('£', '', regex=True)
credit_data['Amount'] = pd.to_numeric(credit_data['Amount'], errors='coerce')

credit_data
```

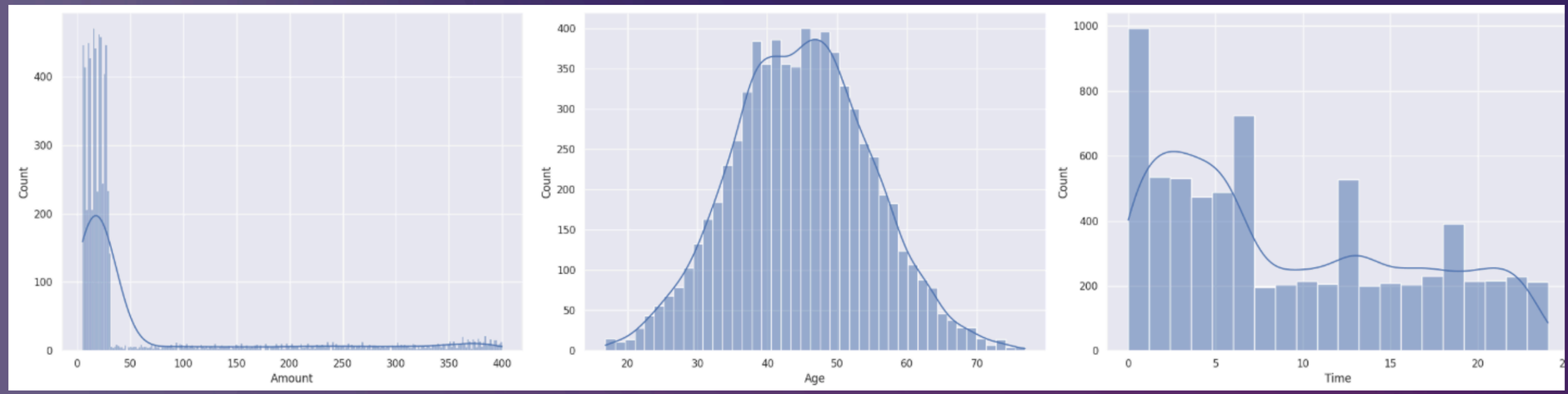
3 Addition of 'Residence = Transaction Country' & 'Transaction country = shipping address' columns

4 Cleaning of 'Amount' column

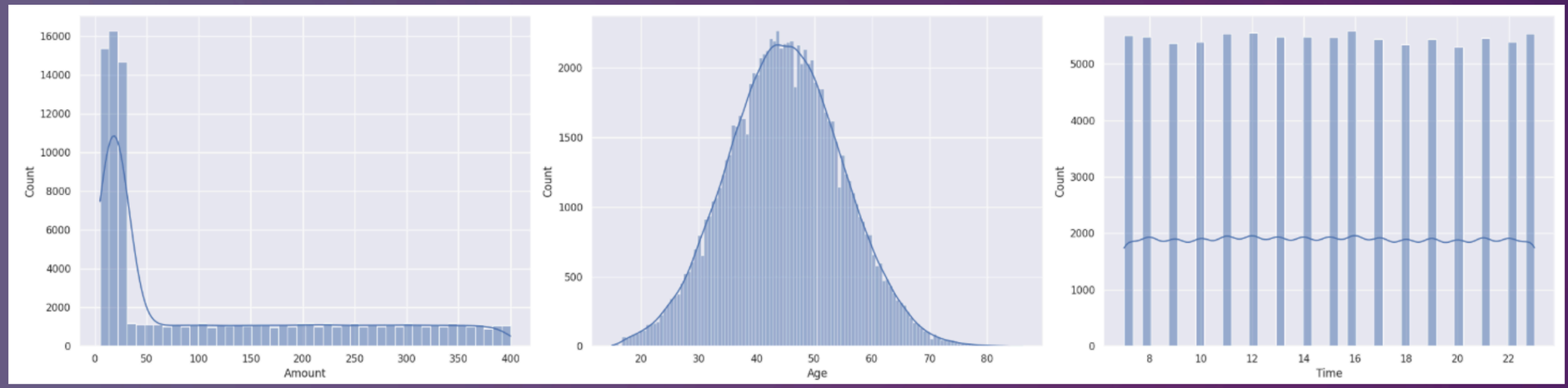
Out[ ]:

	Time	Type of Card	Entry Mode	Amount	Type of Transaction	Merchant Group	Country of Transaction	Shipping Address	Country of Residence	Age	Bank	Fraud	T
0	19	Visa	Tap	5	POS	Entertainment	United Kingdom	United Kingdom	United Kingdom	25.2	RBS	0	
1	17	MasterCard	PIN	288	POS	Services	USA	USA	USA	49.6	Lloyds	0	
2	14	Visa	Tap	5	POS	Restaurant	India	India	India	42.2	Barclays	0	
3	14	Visa	Tap	28	POS	Entertainment	United Kingdom	India	United Kingdom	51.0	Barclays	0	

INITIAL EXPLORATORY DATA ANALYSIS



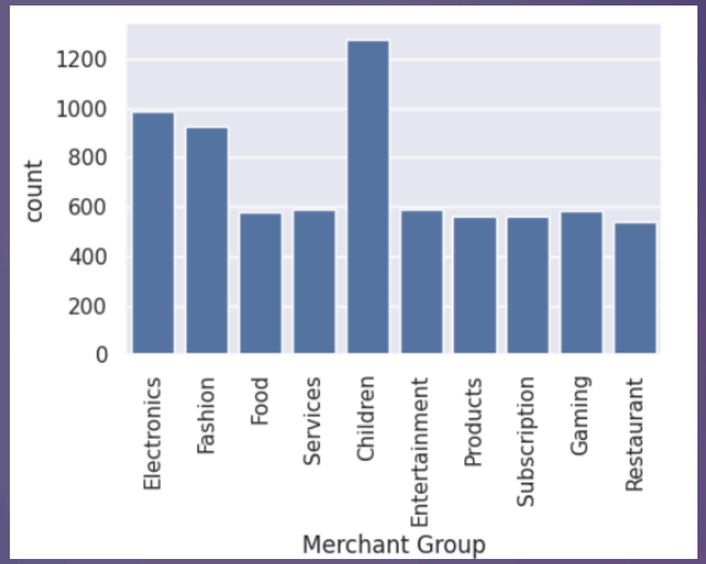
Non-Fraud

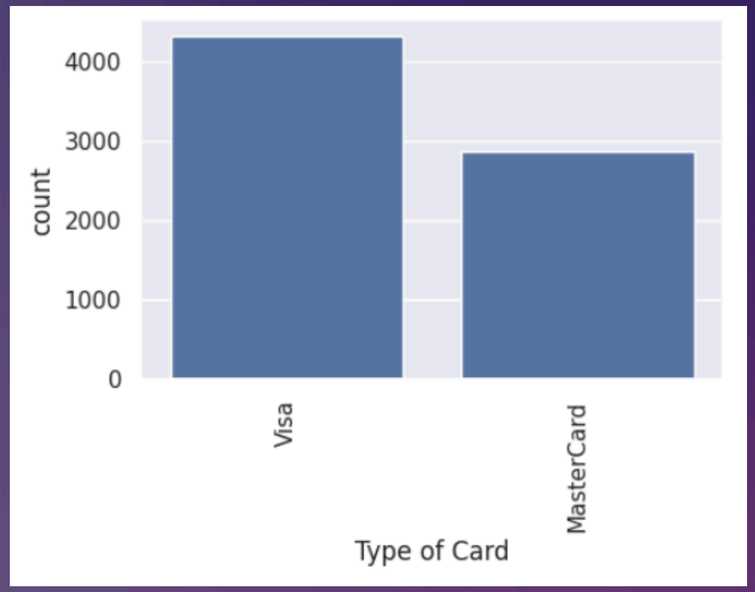


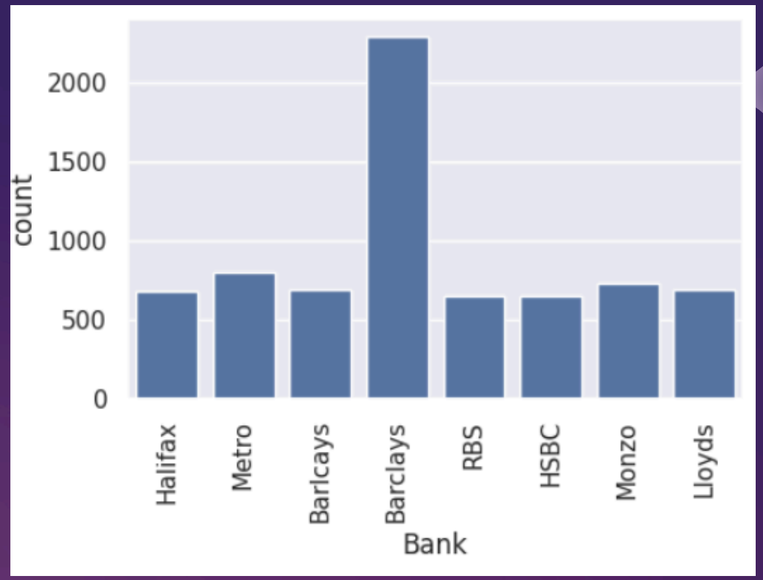
Fraud

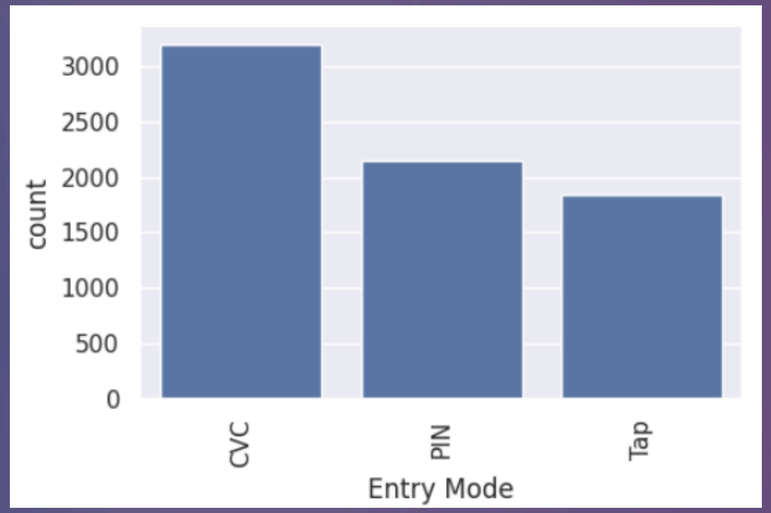
NUMERIC

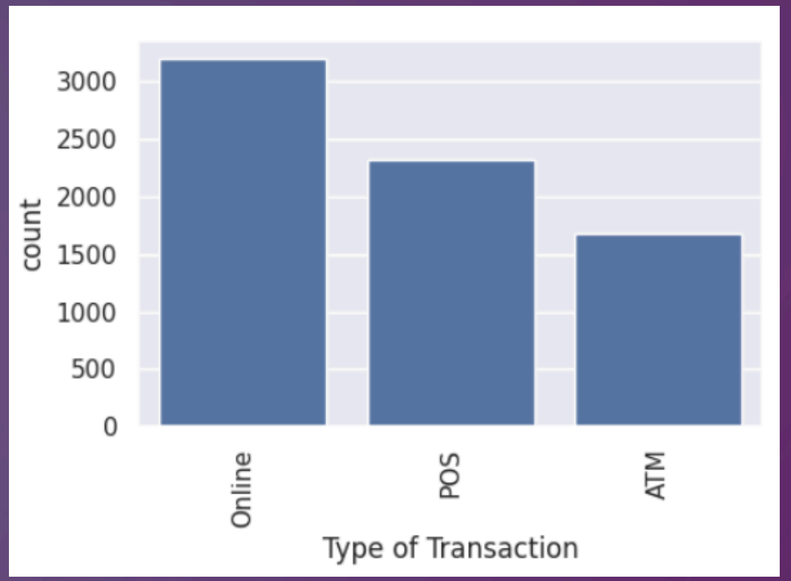
INITIAL EXPLORATORY DATA ANALYSIS





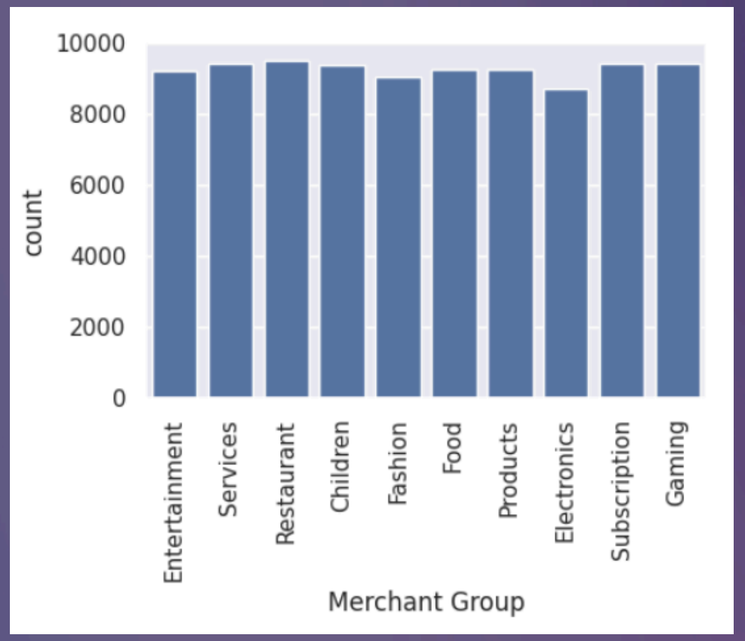


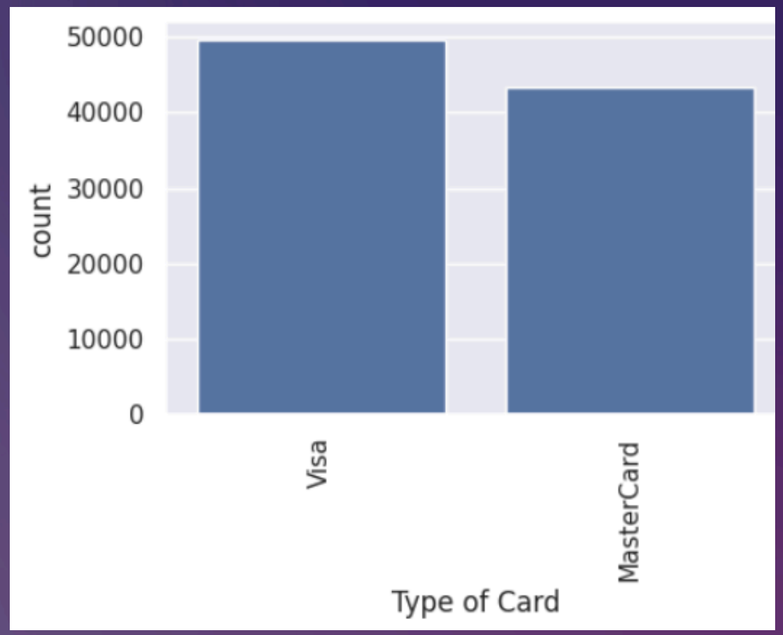


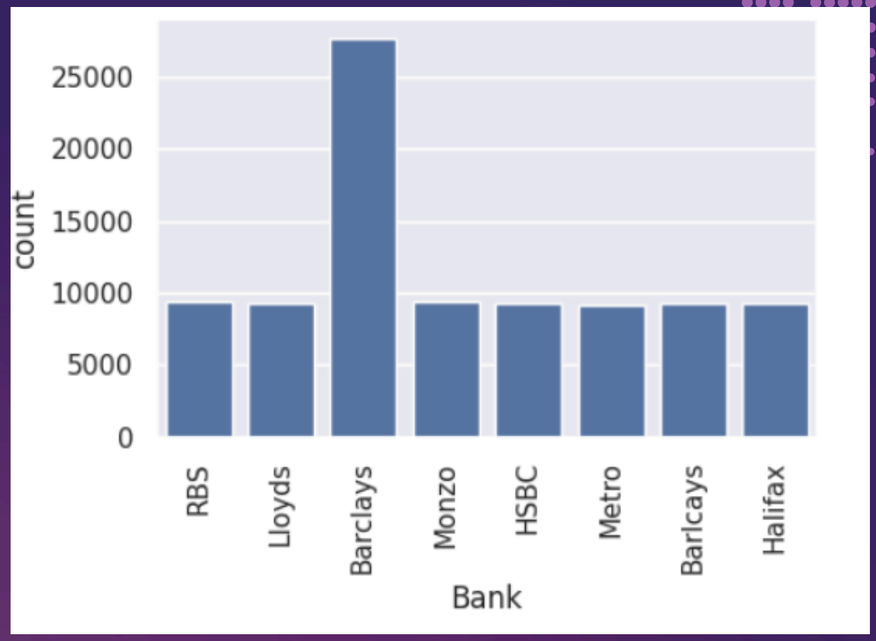


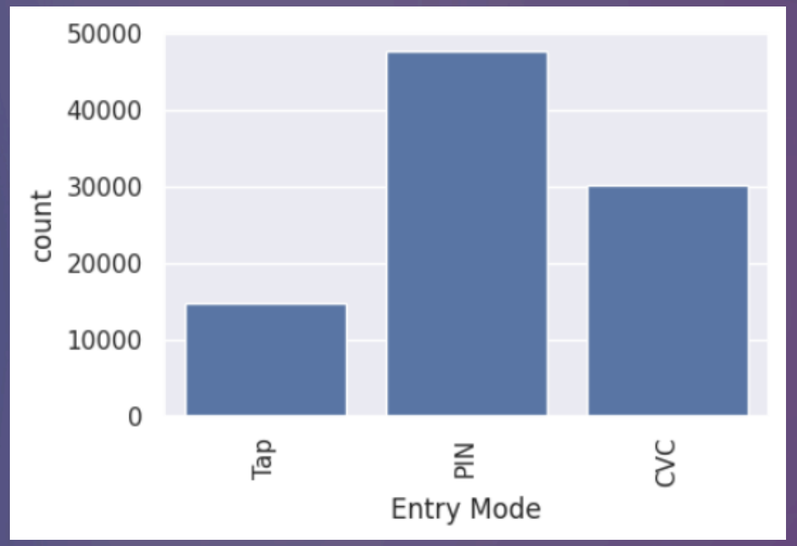
CATEGORICAL
(**Fraud**)

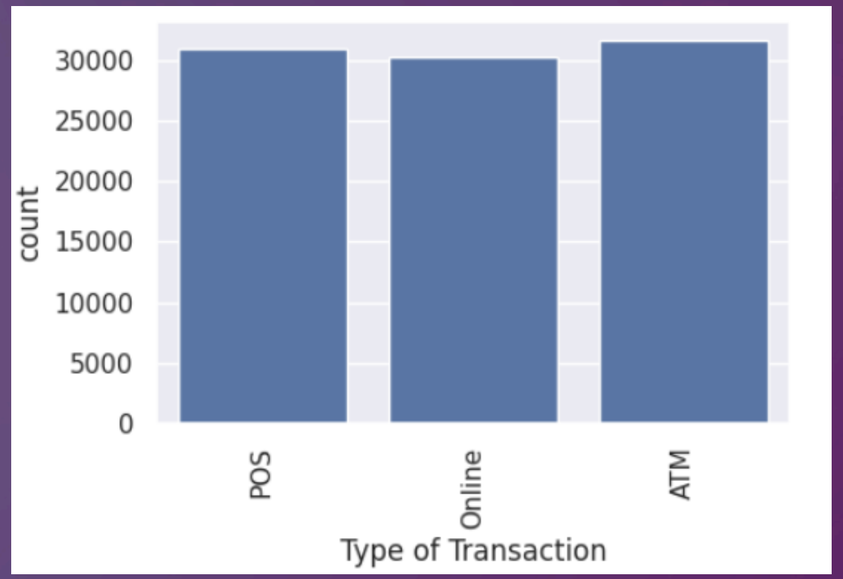
INITIAL EXPLORATORY DATA ANALYSIS





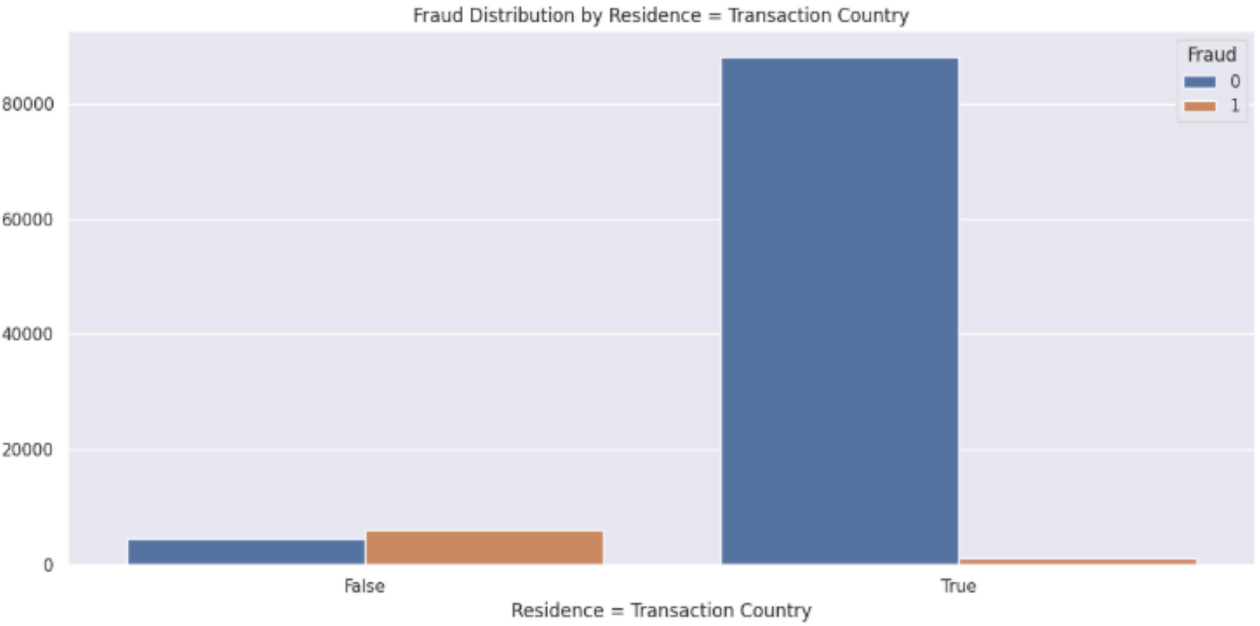


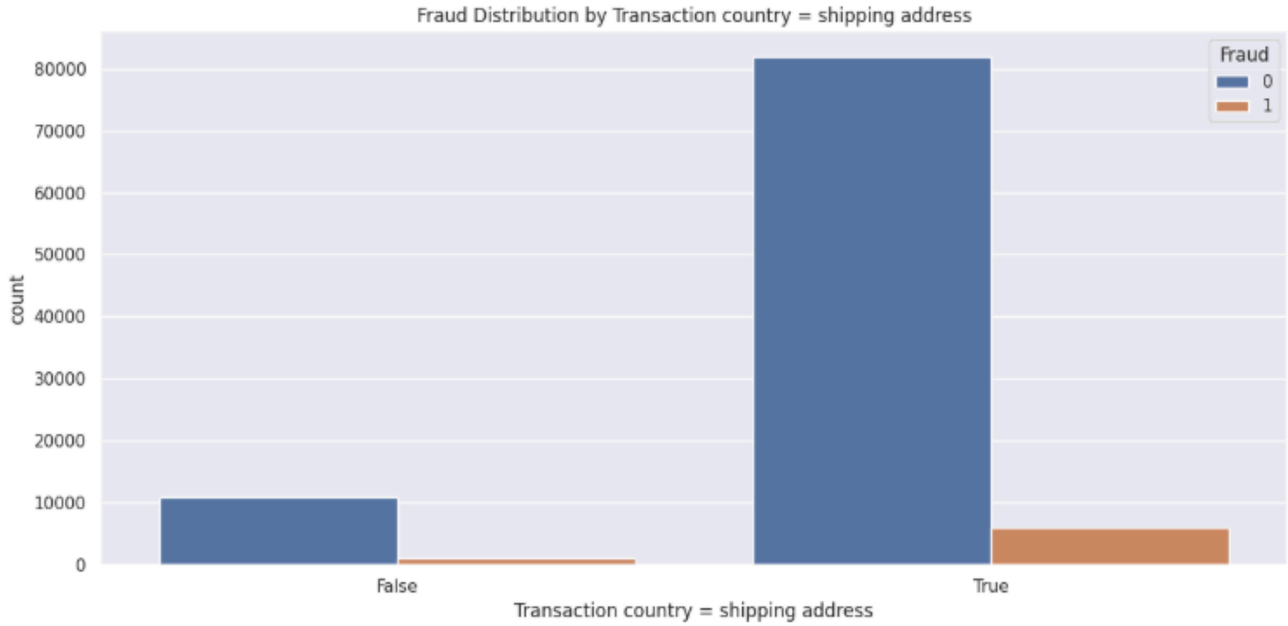




CATEGORICAL
(**Non-fraud**)

INITIAL EXPLORATORY DATA ANALYSIS





BOOLEAN

TRAINING OUR MACHINE LEARNING MODEL

XGBoost

Extreme Gradient Boosting

A scalable, distributed gradient boosted decision tree. A gradient boosting decision tree is a decision tree ensemble learning algorithm which is similar to random forest for classification and regression.

Our model parameters:

1. Binary classification (fraud or not fraud)
2. Each decision tree goes 3 levels deep
3. Learning rate = 0.1 (Slower, but safer)

```
# define parameters
params = {
    'objective': 'binary:logistic',
    'max_depth': 3,
    'learning_rate': 0.1,
}
```

Steps in training our model:

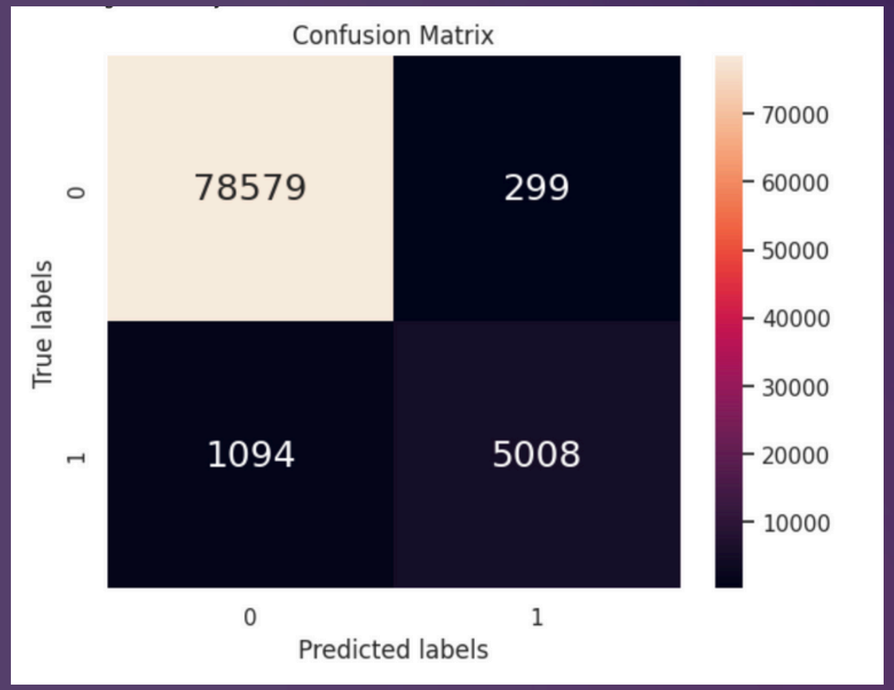
1. Splitting dataset 85-15 into train and test
2. Converting dataset into DMatrix
3. Train model on train data
4. Make predictions based on test data

TRAINING OUR MACHINE LEARNING MODEL

XGBoost

Train data

Accuracy: 0.9836 (Rather high!)

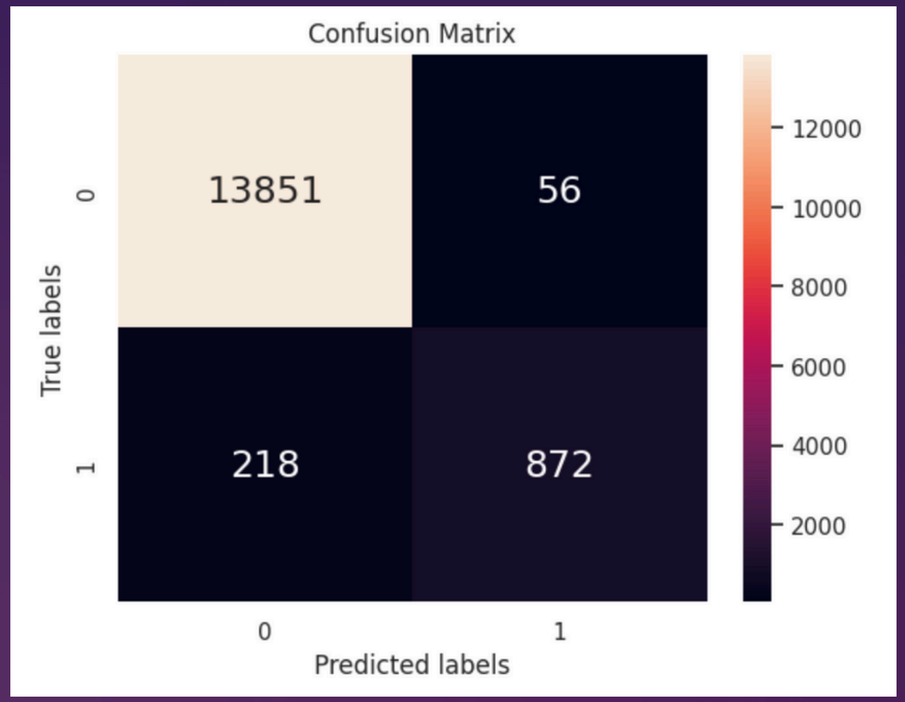


TPR: 0.821, TNR: 0.996

FPR: 0.00379, **FNR: 0.179** (**High!**)

Test data

Accuracy: 0.9817 (Rather high!)

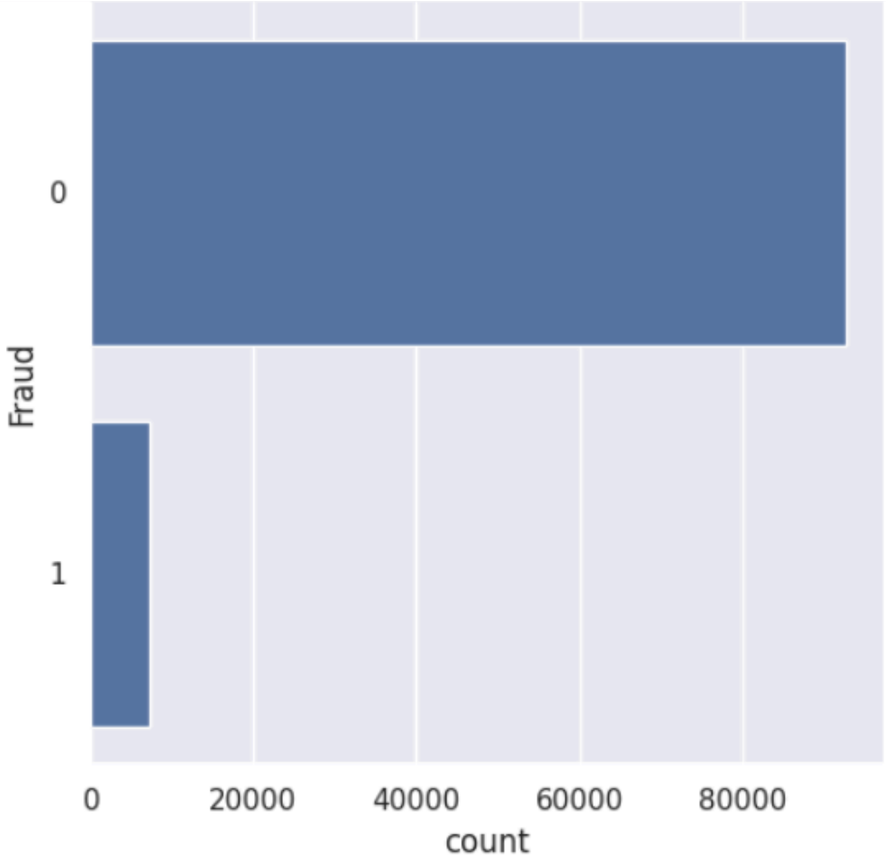


TPR: 0.8, TNR: 0.996

FPR: 0.00403, **FNR: 0.2** (**High!**)

From the high false negative rates above, the model missed detecting many frauds. Hence, we decided to **balance the data set** so the model does not become biased to the non-fraud majority. This way, the model is able to better predict fraudulent transactions

BALANCING OUR DATA



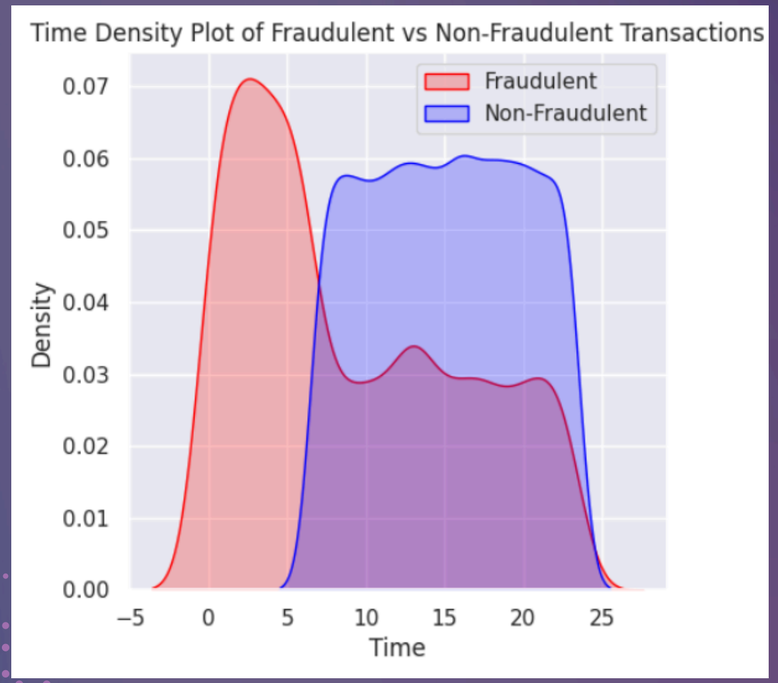

```python
from sklearn.utils import resample

# Separate fraud and non-fraud cases
credit_fraud = credit_data[credit_data['Fraud'] == 1]
credit_non_fraud = credit_data[credit_data['Fraud'] == 0]

# Count the number of fraud cases
fraud_count = len(credit_fraud)

# Undersample non-fraud cases to match the fraud count
non_fraud_sampled = resample(credit_non_fraud,
                             replace=False,  # No replacement
                             n_samples=fraud_count,  # Match fraud cases
                             random_state=42)  # For reproducibility

# Combine the undersampled non-fraud cases with fraud cases
credit_balanced = pd.concat([credit_fraud, non_fraud_sampled])

# Shuffle the dataset to mix fraud and non-fraud cases
credit_balanced = credit_balanced.sample(frac=1, random_state=42).reset_index(drop=True)

# Verify new class distribution
print(credit_balanced['Fraud'].value_counts())
```

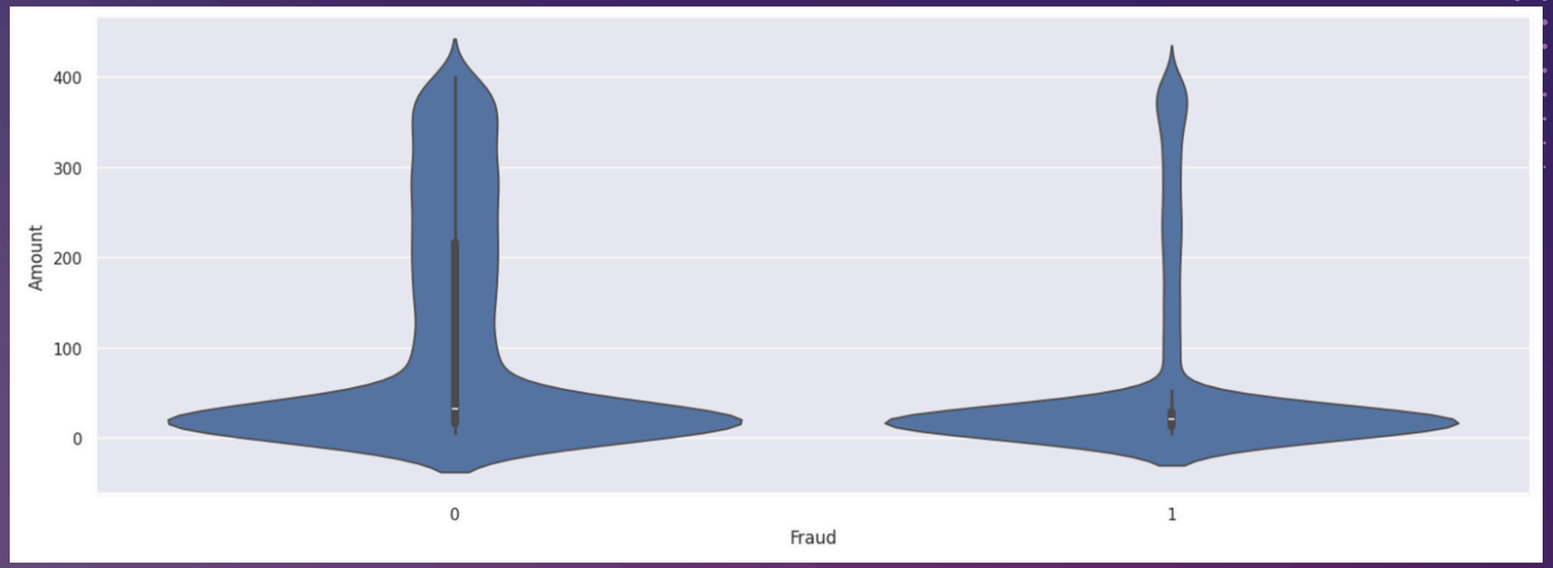
```
Fraud
0    7192
1    7192
Name: count, dtype: int64
```

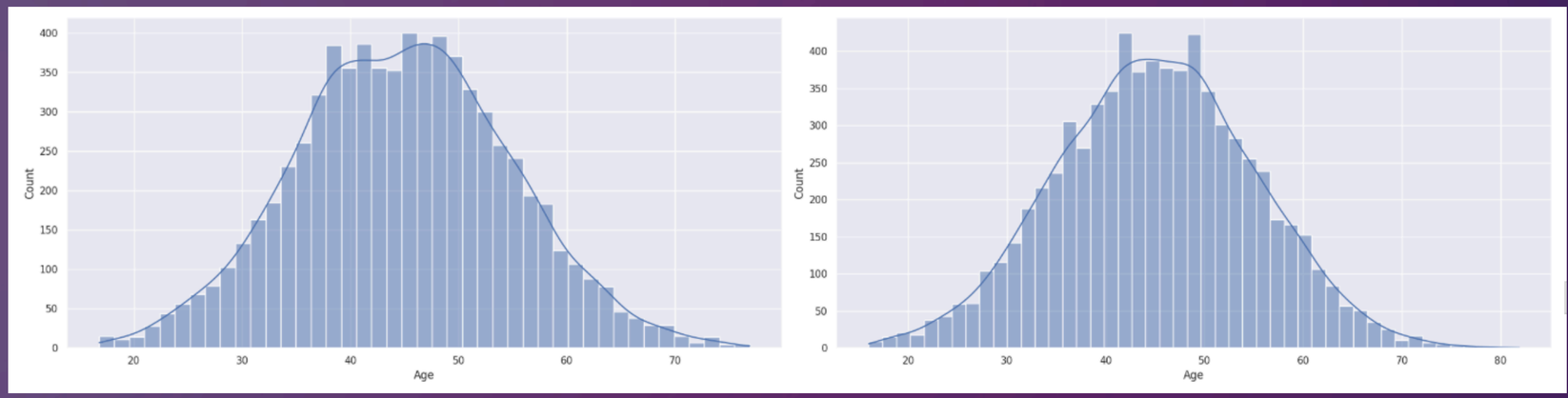
INITIAL UNBALANCED

BALANCED

SECONDARY EXPLORATORY DATA ANALYSIS

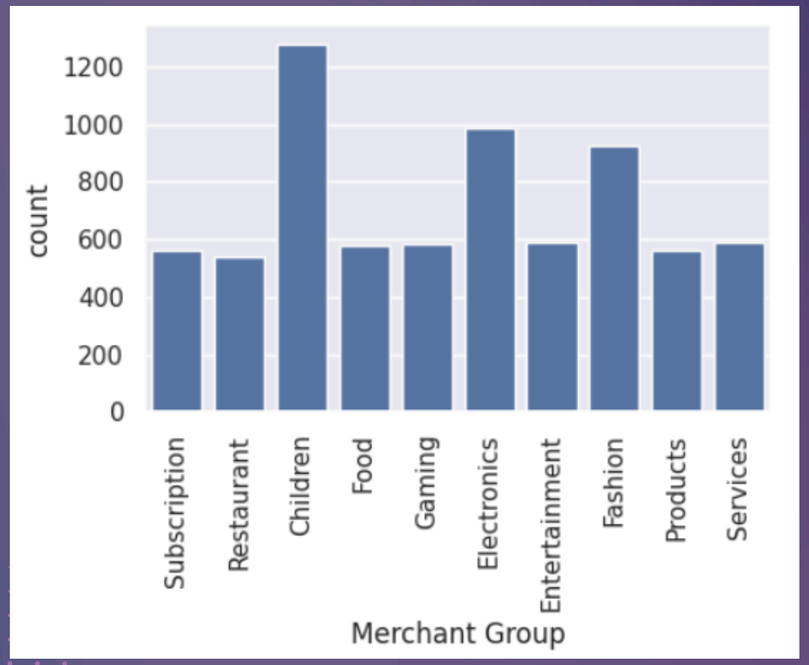


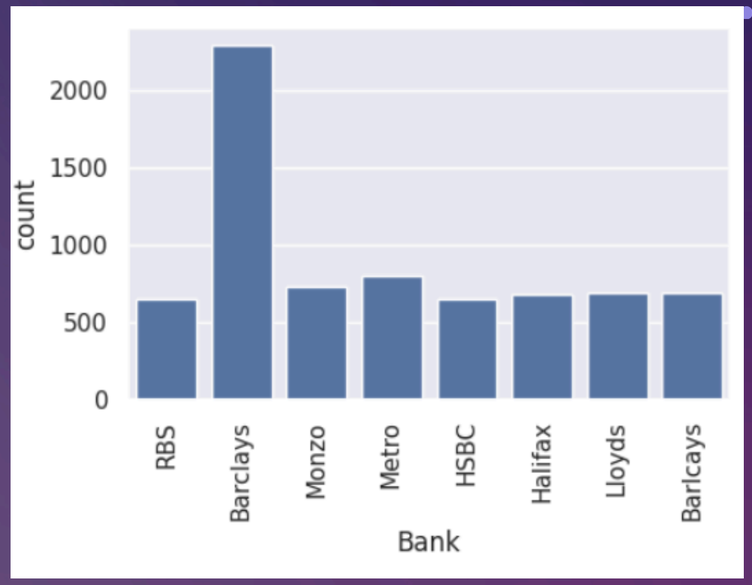


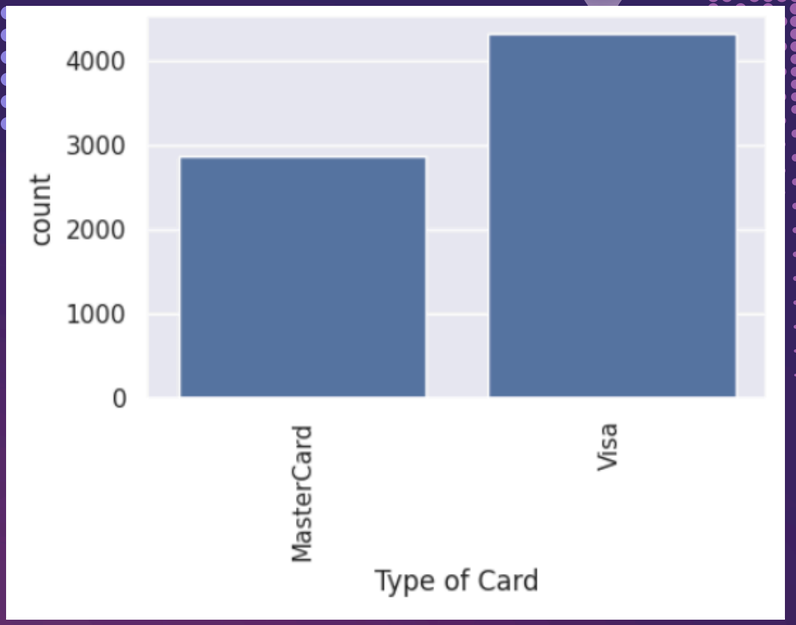


NUMERIC

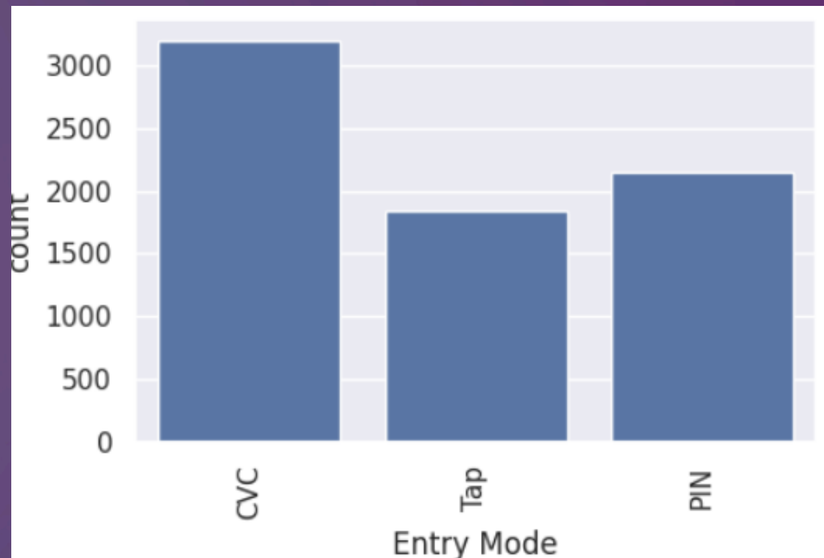
SECONDARY EXPLORATORY DATA ANALYSIS

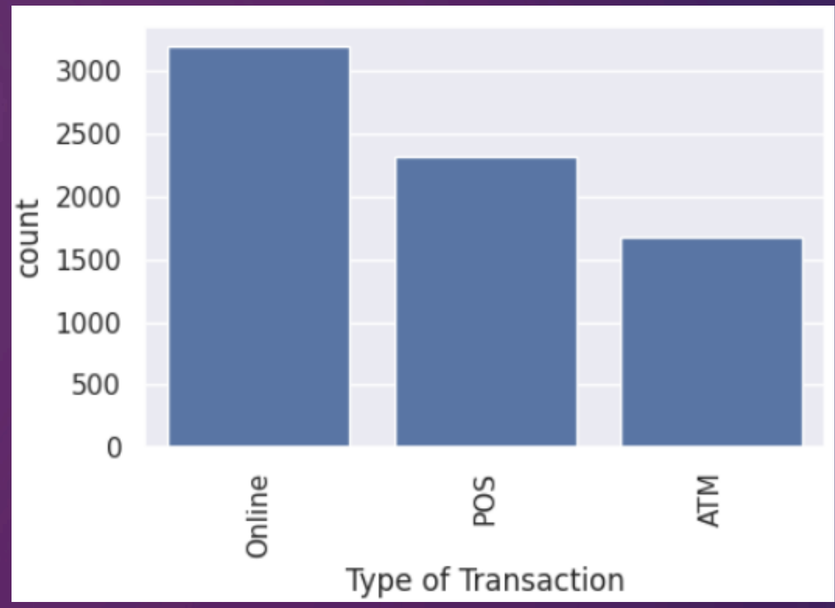




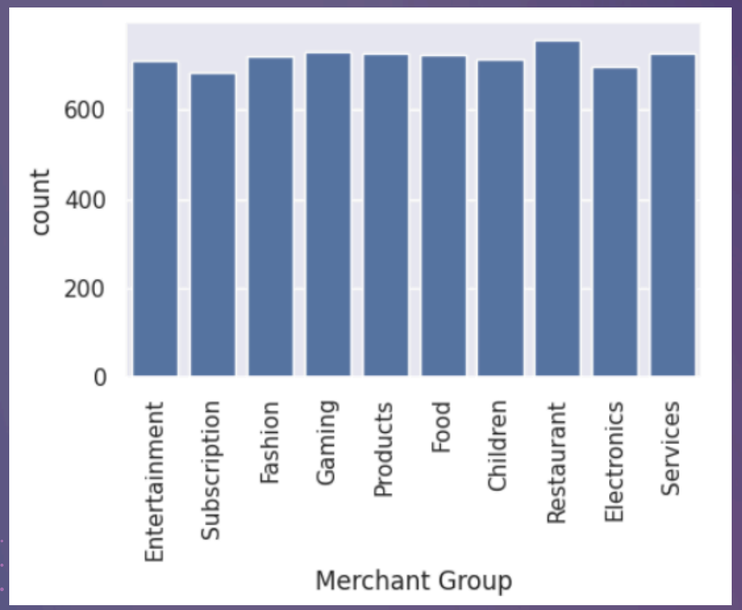


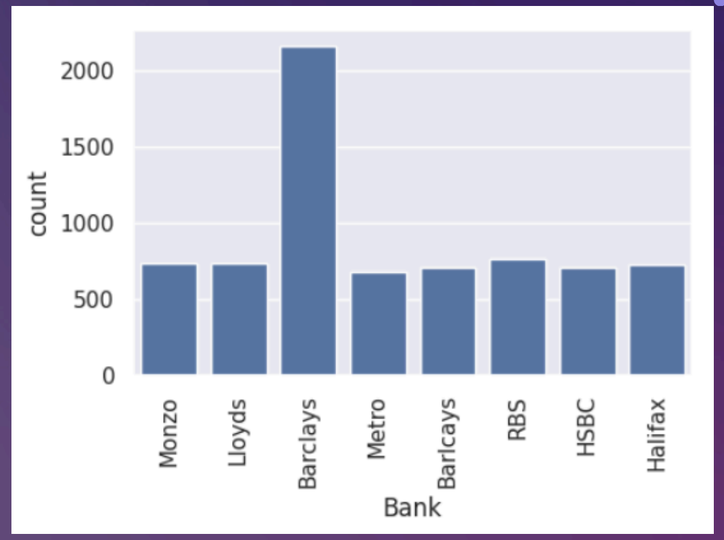
CATEGORICAL
(**Fraud**)

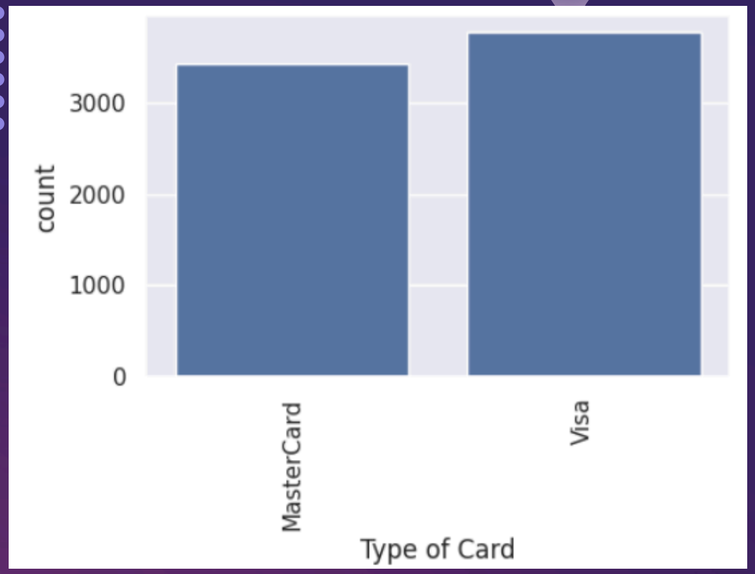


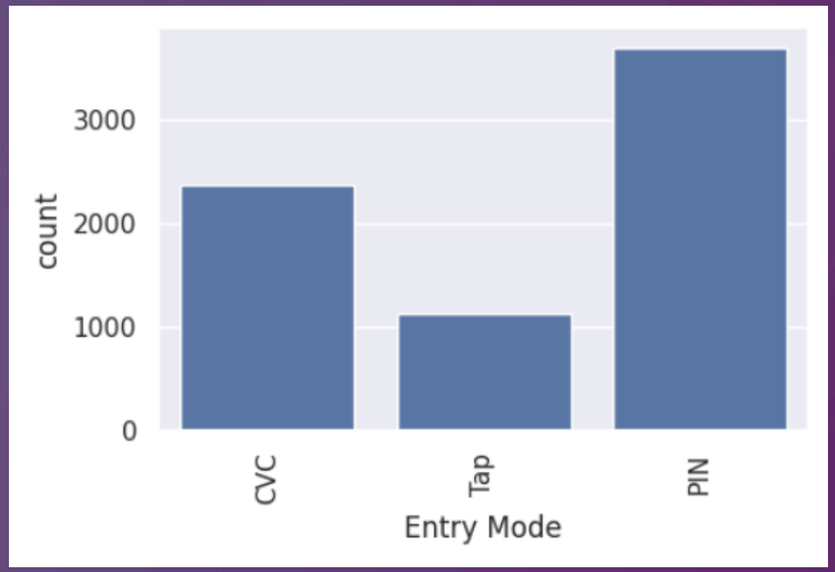


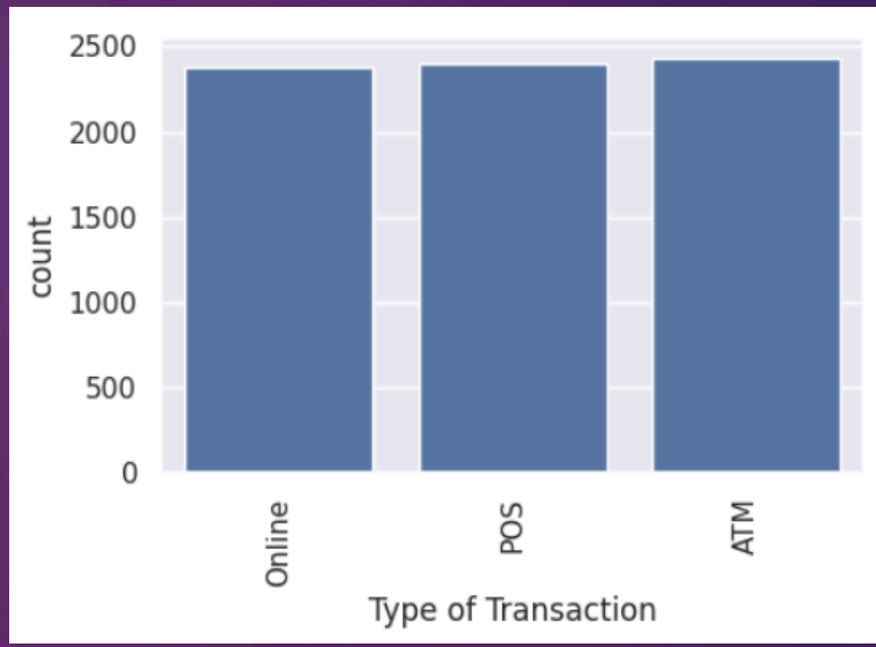
SECONDARY EXPLORATORY DATA ANALYSIS





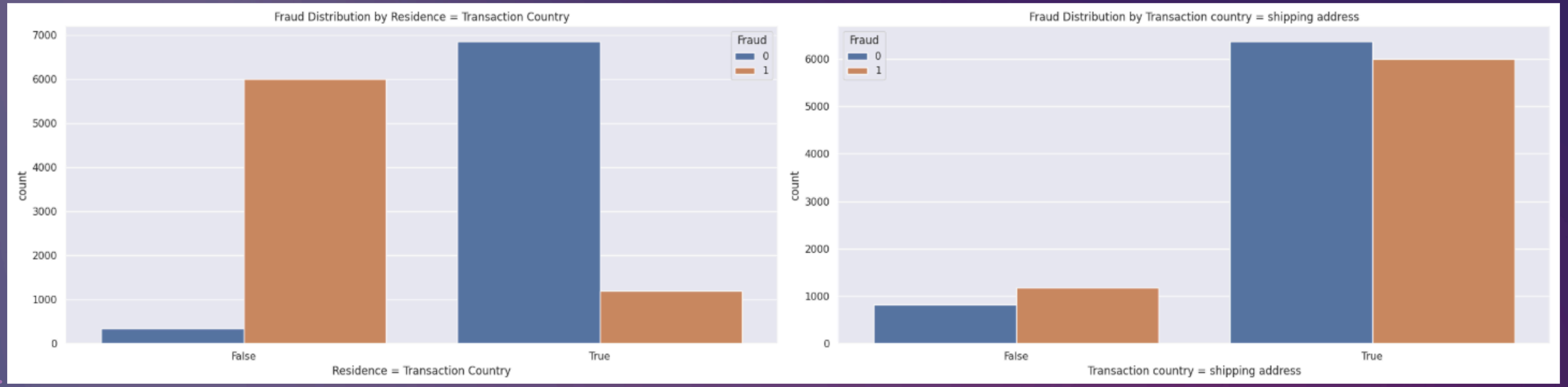






CATEGORICAL (non-fraud)

SECONDARY EXPLORATORY DATA ANALYSIS



BOOLEAN

FURTHER DETERMINATION OF IMPORTANT NUMERIC VARIABLES

Lasso Regression

Lasso regression is useful for feature selection and preventing overfitting.

- Lasso helps in reducing overfitting by penalizing less useful features.
- It forces some coefficients of less important variables to become zero, effectively removing them
- Works primarily on numeric variables

After adjusting the penalty strength a few times, we chose the regularisation strength to be 1. The model selected both **‘time’** and **‘amount’** as important numeric variables

```
# Apply Lasso regression
lasso = Lasso(alpha=1)  # Regularization strength
lasso.fit(X_train, y_train)

# Predict and evaluate the model
y_pred = lasso.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
```

```
Coefficients: [-0.00887115 -0.00084529]
Intercept: 0.684921983380109
Mean Squared Error: 0.21598898111437775
Selected Features: ['time', 'amount']
```

RETRAINING OUR MACHINE LEARNING MODEL

XGBoost

Defined the same parameters

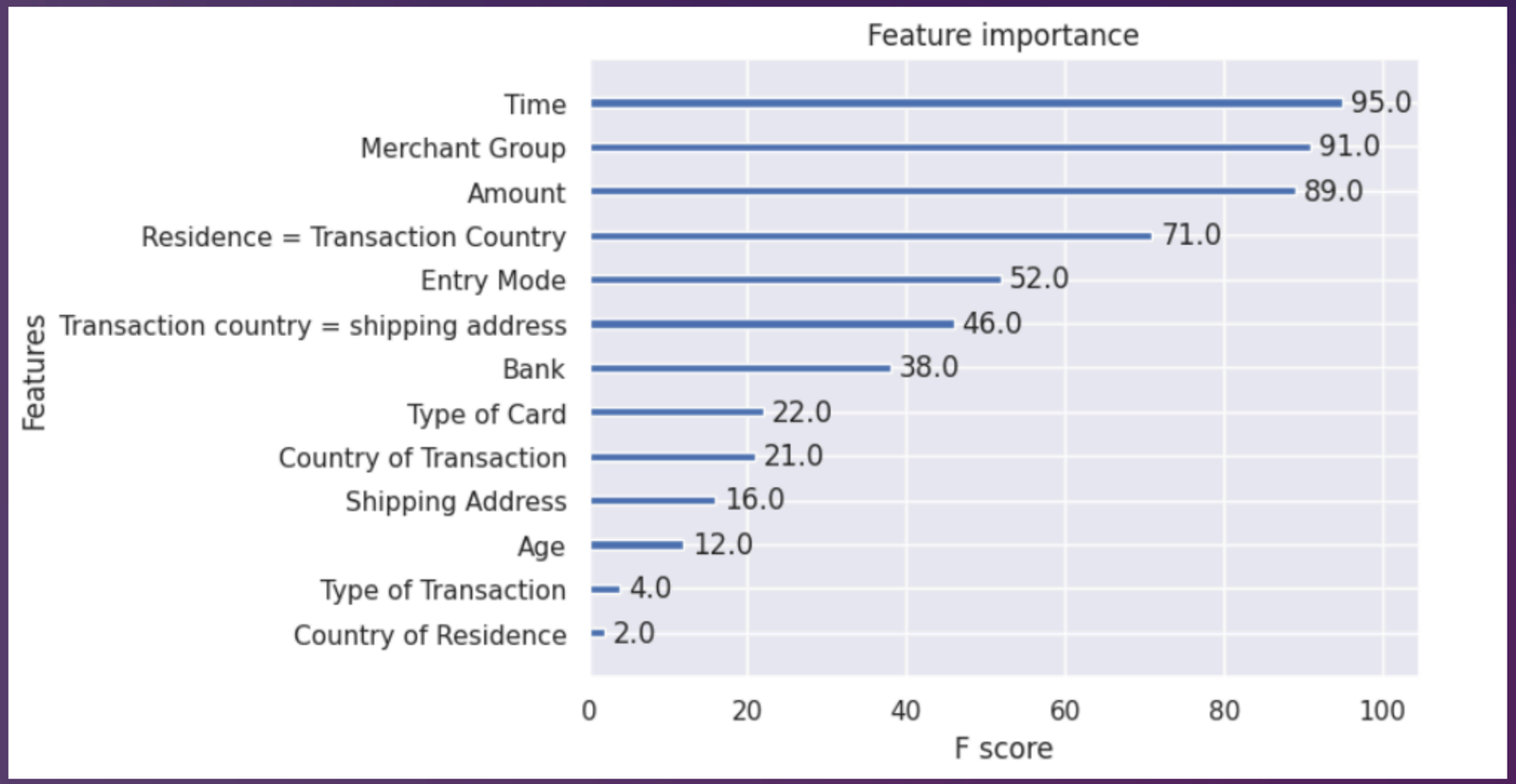
```
# define parameters
params = {
    'objective': 'binary:logistic',
    'max_depth': 3,
    'learning_rate': 0.1,
}
```

Conducting the same steps:

1. Splitting dataset 85-15 into train and test
2. Converting dataset into DMatrix
3. Train model on train data
4. Make predictions based on test data

RETRAINING OUR MACHINE LEARNING MODEL

XGBoost

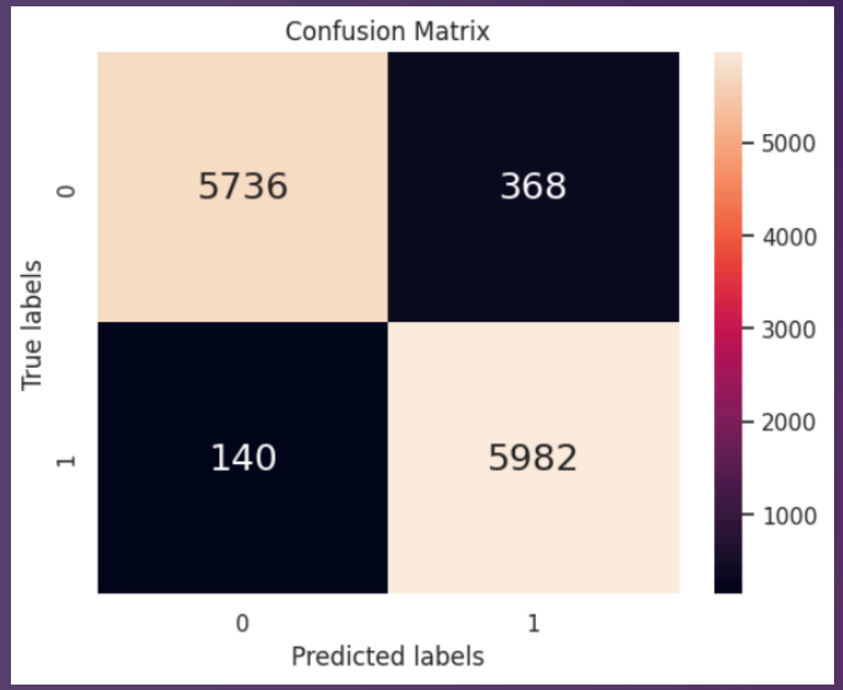


RETRAINING OUR MACHINE LEARNING MODEL

XGBoost

Train data

Accuracy: 0.9584 (Rather high!)

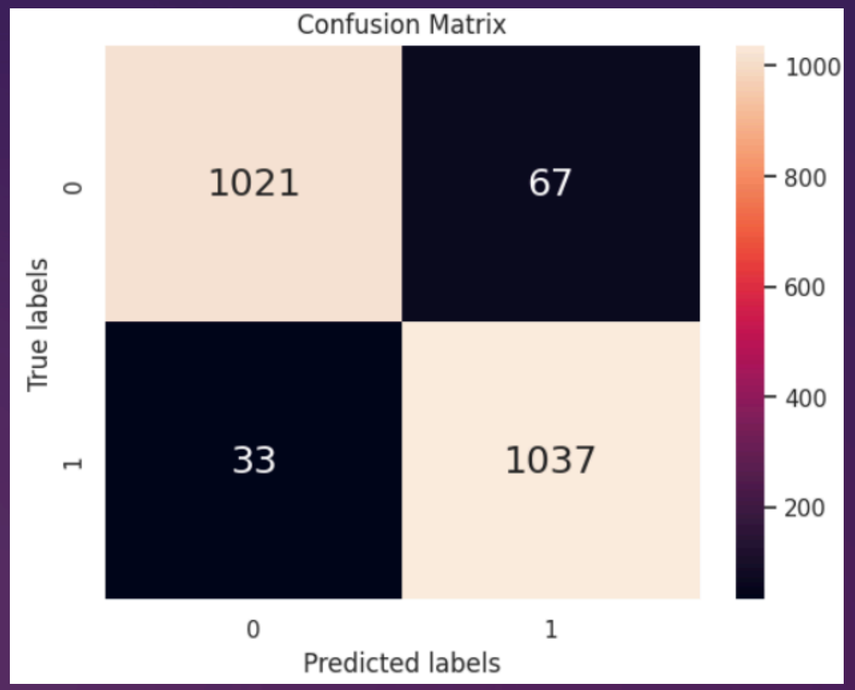


TPR: 0.977, TNR: 0.940

FPR: 0.060, **FNR: 0.0229**(**lower!**)

Test data

Accuracy: 0.9537 (Rather high!)



TPR: 0.821, TNR: 0.996

FPR: 0.00379, **FNR: 0.179** (**lower!**)

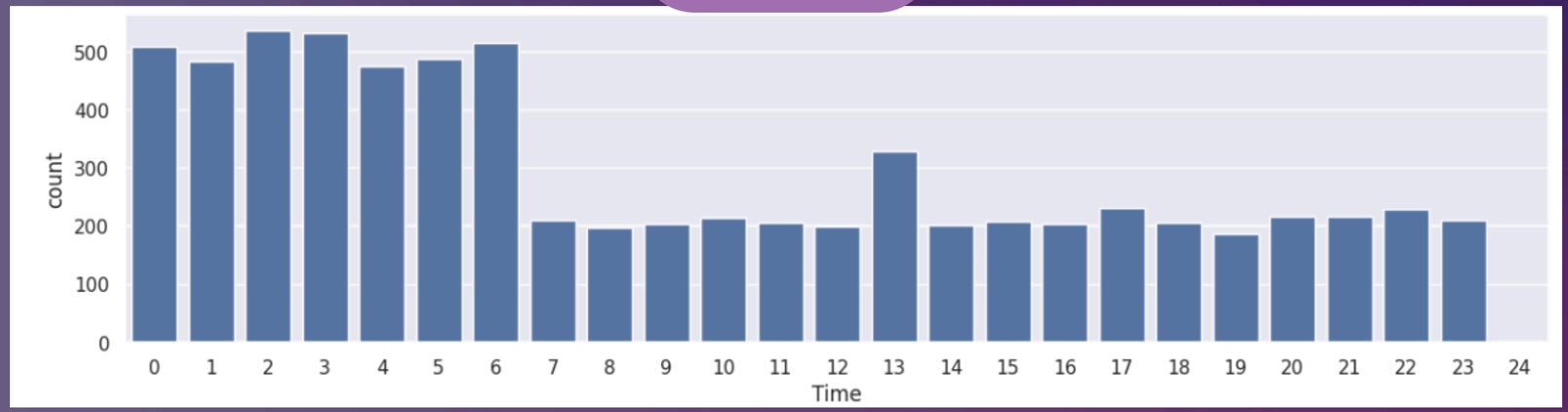
The FNR decreased and the TPR increased. Hence, balancing the dataset allowed our model to classify the fraud and non-fraud cases more accurately and with less bias.

ANALYSING PATTERNS IN FRAUDULENT TRANSACTIONS - CROSSCHECKING

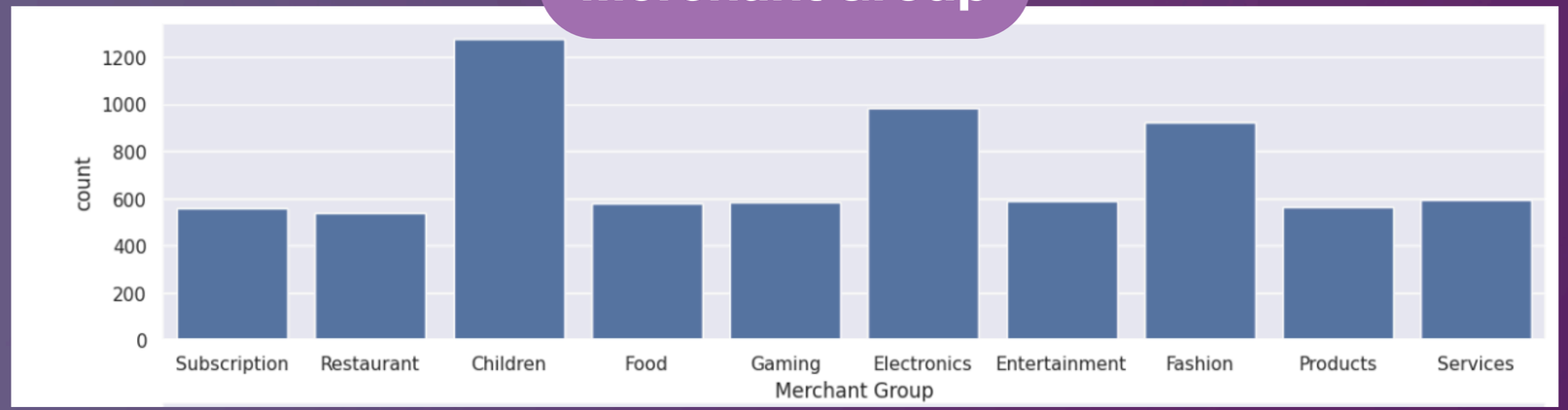
Selected the top 5 key features contributing to fraud detection and examined their patterns specifically within fraudulent cases

Time

Most fraudulent transactions occured from 0000 - 0600

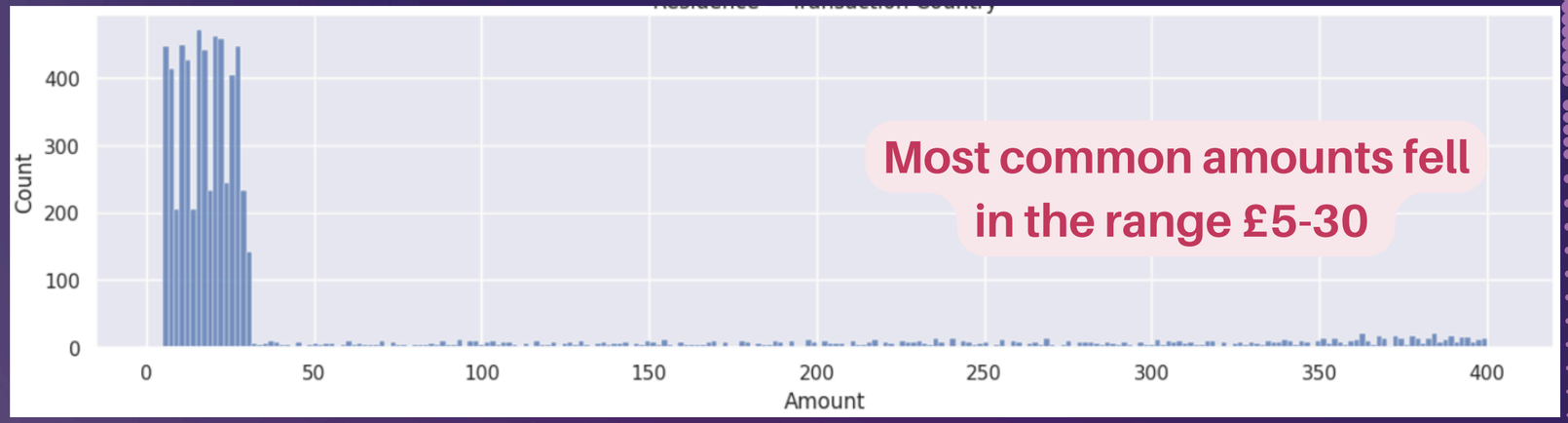


Merchant Group

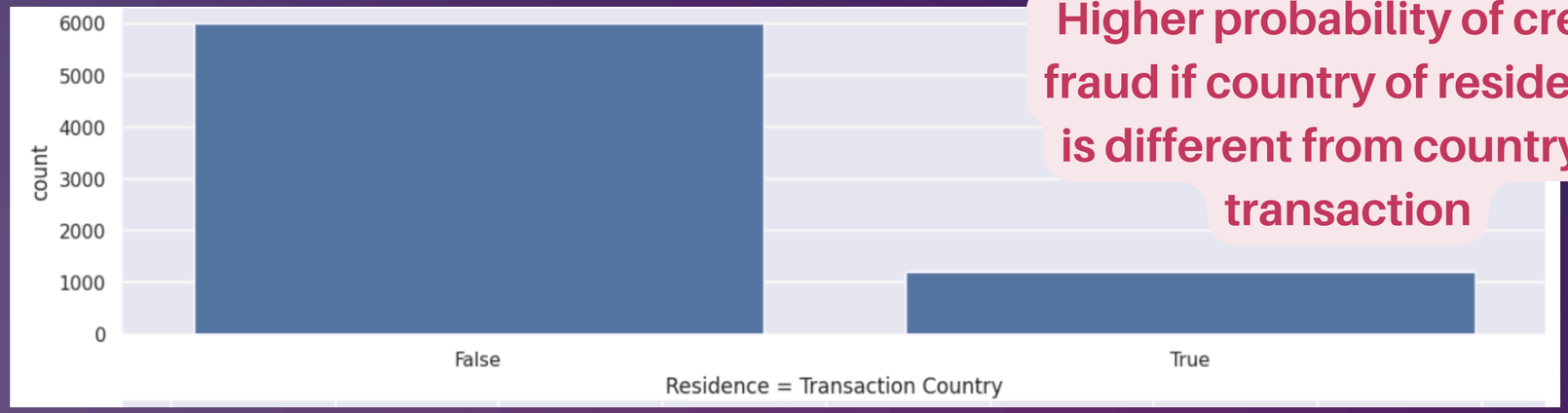


More fraudulent transactions happened in Children, Electronics and Fashion categories

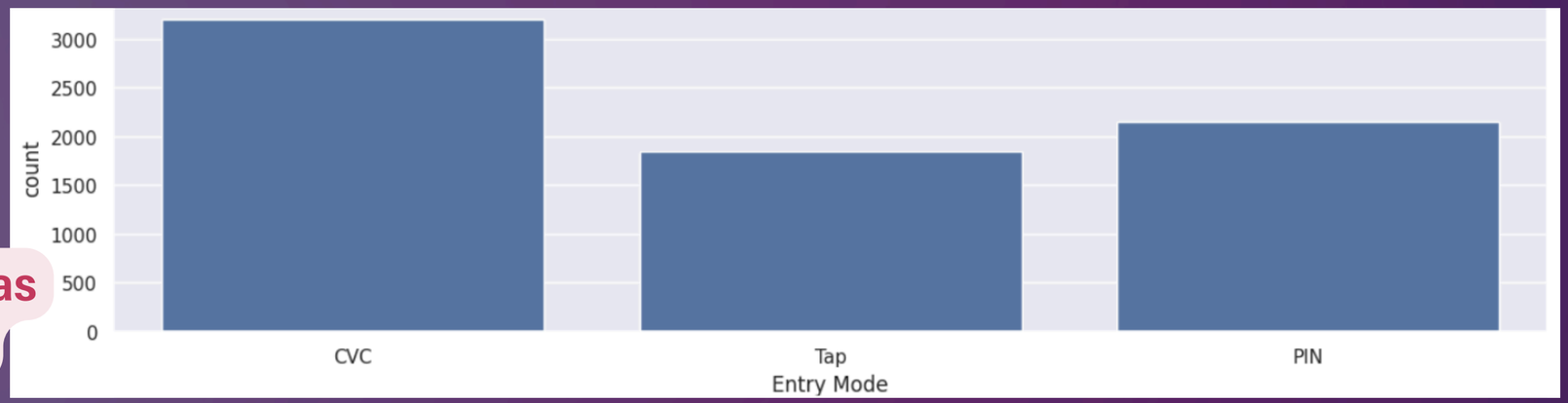
ANALYSING THE IMPORTANT VARIABLES



Most common amounts fell in the range £5-30



Higher probability of credit fraud if country of residence is different from country of transaction



Credit Verification Code was the most common entry mode for frauds

CONCLUSION

We can effectively detect fraudulent credit card transactions using machine learning models, which analyse patterns in data. XGBoost not only allowed us to identify important variables in detecting frauds but also used these important variables to detect the fraud cases in the test set. This machine learning model is very useful in real world applications.

CONCLUSION

Top 5 variables to be used in detecting fraudulent credit card transactions

1. Time of transaction
2. Merchant Group
3. Transaction Entry
4. Country of Residence = Country of transaction
5. Amount transacted

CONCLUSION

Possible implications

Time of transaction:

- Implement heightened fraud detection thresholds or automated flags from 0000 to 0600
- Retailers and financial institutions might consider limiting transaction size or frequency during these hours for high-risk profiles.

Merchant group:

- Merchants in Children, Electronics and Fashion sectors could benefit from insurance or chargeback protection partnerships.

Entry mode:

- Implement a CVC refresh feature allows the Card Verification Code (CVC) on a credit/debit card to change periodically, making it harder for fraudsters to use stolen card info.

CONCLUSION

Possible implications

Country of Residence = Country of transaction:

- Trigger extra authentication checks when location mismatch occurs

Amount:

- Train models to flag clusters of small transactions across short time windows from the same account/device/IP.

THANK YOU!