Darwhin_GomezPP

November 9, 2024

1 Project Proposal: Fraud Detection in Bank Transactions

1.1 2. Research Question

- **Null Hypothesis (H):**
  "Anomaly detection techniques, specifically an isolation-forrest model, does not significantly identify outliers in transaction data that correspond to fraudulent activities."
- **Alternative Hypothesis (H):**
  "Anomaly detection techniques, specifically an isolation-forrest model, can effectively identify outliers in transaction data that correspond to fraudulent activities."

1.2 2. Justification

Fraud detection remains a critical challenge in the financial industry. As fraudulent activities can involve abnormal transaction behaviors, detecting outliers or anomalies in large transaction datasets is key to identifying potential fraud. This project uses unsupervised learning techniques such as **anomaly detection** to flag suspicious transactions based on transaction data.

1.3 3. Data Source

https://www.kaggle.com/datasets/valakhorasani/bank-transaction-dataset-for-fraud-detection/data

The dataset, **bank_transaction_data_2.csv**, contains transaction records for multiple customer accounts. The dataset includes the following features:

- **TransactionID**: Unique alphanumeric identifier for each transaction.
- **AccountID**: Unique identifier for each account, with multiple transactions per account.
- **TransactionAmount**: Monetary value of each transaction.
- **TransactionDate**: Timestamp of each transaction.
- **TransactionType**: Categorical field indicating 'Credit' or 'Debit' transactions.
- **Location**: Geographic location of the transaction, represented by U.S. city names.
- **DeviceID**: Alphanumeric identifier for devices used to perform the transaction.
- **IP Address**: IPv4 address associated with the transaction.
- **MerchantID**: Unique identifier for merchants.
- **AccountBalance**: Balance in the account post-transaction.
- **PreviousTransactionDate**: Timestamp of the last transaction for the account.
- **Channel**: Channel through which the transaction was performed (e.g., Online, ATM, Branch).
- **CustomerAge**: Age of the account holder.

- **CustomerOccupation**: Occupation of the account holder (e.g., Doctor, Engineer, Student, Retired).
- **TransactionDuration**: Duration of the transaction in seconds.
- **LoginAttempts**: Number of login attempts before the transaction, with higher values indicating potential anomalies.

1.4 4. Tools and Libraries

- **Pandas**: For data manipulation and cleaning.
- **NumPy**: For numerical operations.
- **Matplotlib** and **Seaborn**: For data visualization (distributions, scatter plots, box plots, etc.).
- **Scikit-Learn**:
  - **Isolation Forest**: For unsupervised anomaly detection.
  - **StandardScaler**: For feature scaling and normalization.
- **Datetime**: For handling and extracting features from the `TransactionDate` and `PreviousTransactionDate` columns.

1.5 5. Proposed Methodology

1.5.1 Data Preprocessing

- **Handle missing values**: Ensure no missing values are present (if any, apply appropriate imputation or removal).
- **Categorical feature encoding**: Convert categorical variables (e.g., `TransactionType`, `Location`, `Channel`) into numerical values via **One-Hot Encoding** or **Label Encoding**.
- **Datetime feature extraction**: Convert `TransactionDate` and `PreviousTransactionDate` into `datetime` format, and extract useful features such as the transaction hour, day, and time difference between consecutive transactions.
- **Standardize numerical features**: Normalize or standardize continuous variables like `TransactionAmount`, `AccountBalance`, and `TransactionDuration` to ensure consistency across the models.

1.5.2 Modeling

- **Isolation Forest**:
  - Train an **Isolation Forest** model to detect outliers in the transaction data, flagging unusual transactions based on their deviation from normal patterns.
  - Fine-tune model parameters (e.g., contamination level) to improve anomaly detection based on observed patterns.

1.5.3 Evaluation

- **Fraud Flagging**: Transactions identified as outliers by the Isolation Forest model will be flagged as potential fraud.

- **Visual Inspection**: Use scatter plots and boxplots to visualize flagged anomalies, validating if they align with typical fraud indicators (e.g., high transaction amounts, multiple login attempts).

- **Transaction Patterns**: Analyze patterns of flagged transactions, such as unusual transaction amounts or rapid transaction frequency, to assess the model's ability to detect suspicious behaviors.

1.6 6. Exploratory Data Analysis

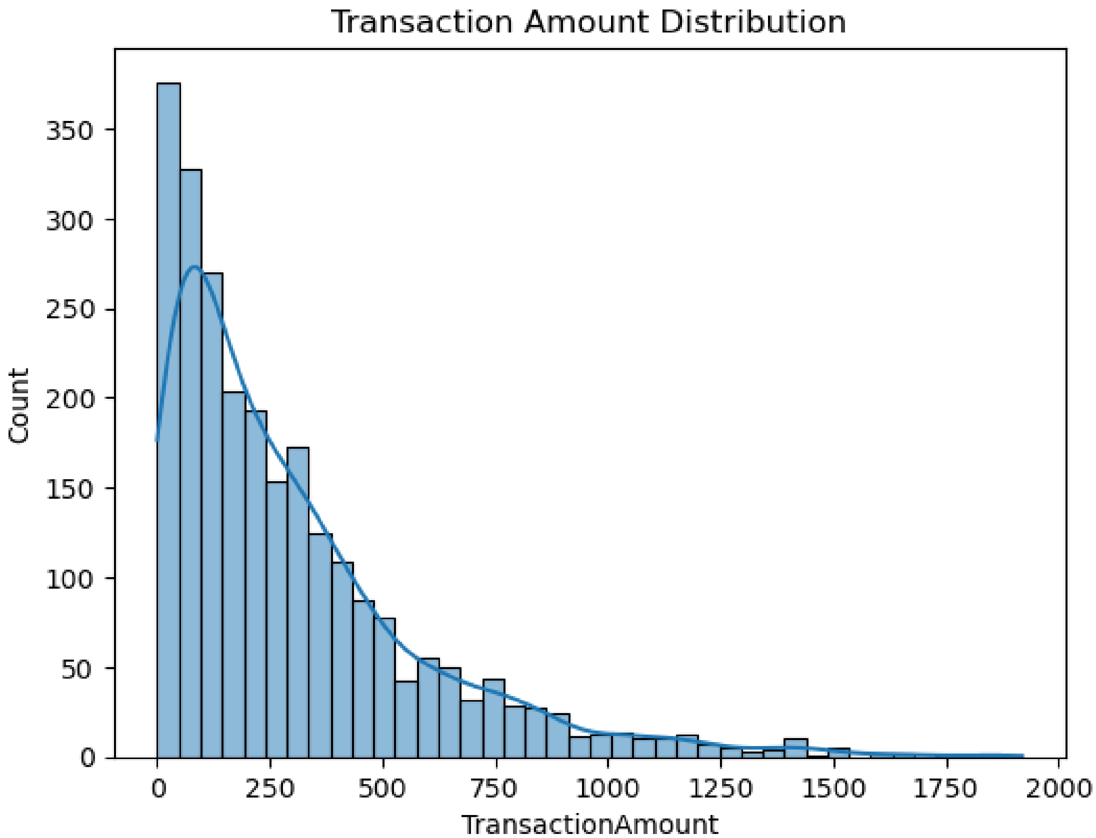
The Following are summary statistics and some exploratory plots of some variables of interest just to visulaize some of the data, eg. distrubutions and box plots to recognize outliers. The code will be hidden but avaiable through this link on GitHub

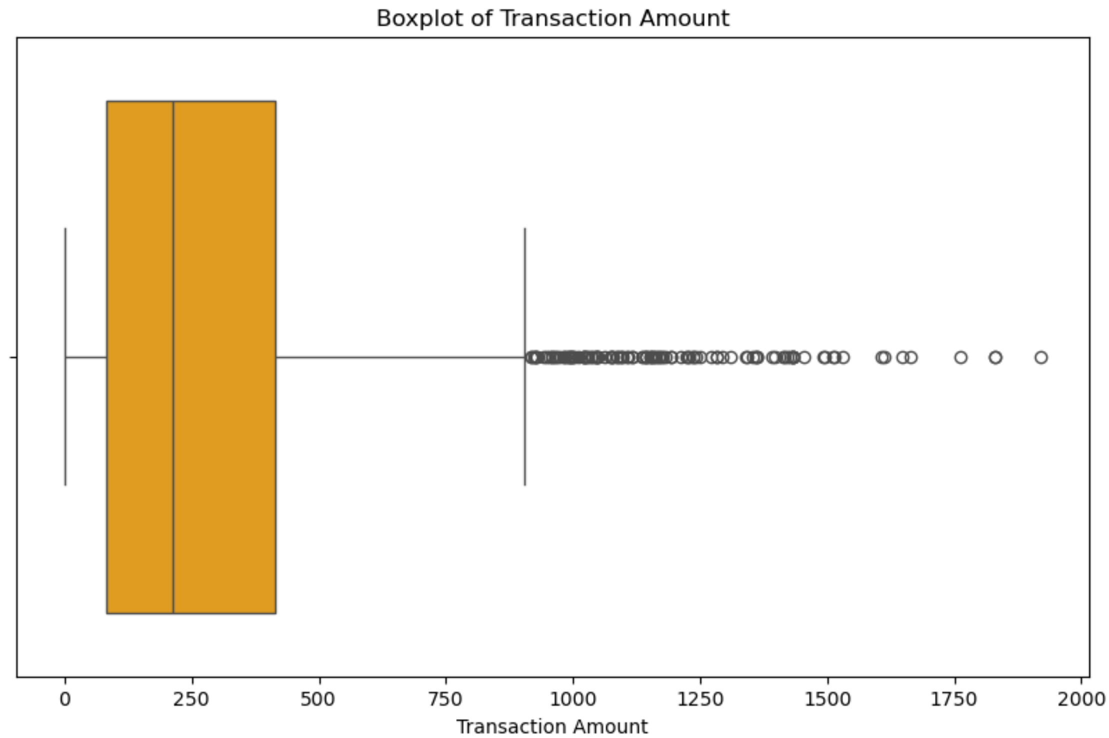
```
Dataset Preview:
  TransactionID AccountID  TransactionAmount      TransactionDate  \
0      TX000001   AC00128              14.09  2023-04-11 16:29:14
1      TX000002   AC00455             376.24  2023-06-27 16:44:19
2      TX000003   AC00019             126.29  2023-07-10 18:16:08
3      TX000004   AC00070             184.50  2023-05-05 16:32:11
4      TX000005   AC00411              13.45  2023-10-16 17:51:24

  TransactionType   Location DeviceID      IP Address MerchantID Channel  \
0           Debit  San Diego  D000380  162.198.218.92       M015     ATM
1           Debit    Houston  D000051     13.149.61.4       M052     ATM
2           Debit       Mesa  D000235  215.97.143.157       M009  Online
3           Debit    Raleigh  D000187  200.13.225.150       M002  Online
4          Credit    Atlanta  D000308    65.164.3.100       M091  Online

   CustomerAge CustomerOccupation  TransactionDuration  LoginAttempts  \
0           70             Doctor                   81              1
1           68             Doctor                  141              1
2           19            Student                   56              1
3           26            Student                   25              1
4           26            Student                  198              1

   AccountBalance PreviousTransactionDate
0         5112.21     2024-11-04 08:08:08
1        13758.91     2024-11-04 08:09:35
2         1122.35     2024-11-04 08:07:04
3         8569.06     2024-11-04 08:09:06
4         7429.40     2024-11-04 08:06:39

Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2512 entries, 0 to 2511
Data columns (total 16 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   TransactionID            2512 non-null   object
 1   AccountID                2512 non-null   object
 2   TransactionAmount        2512 non-null   float64
```

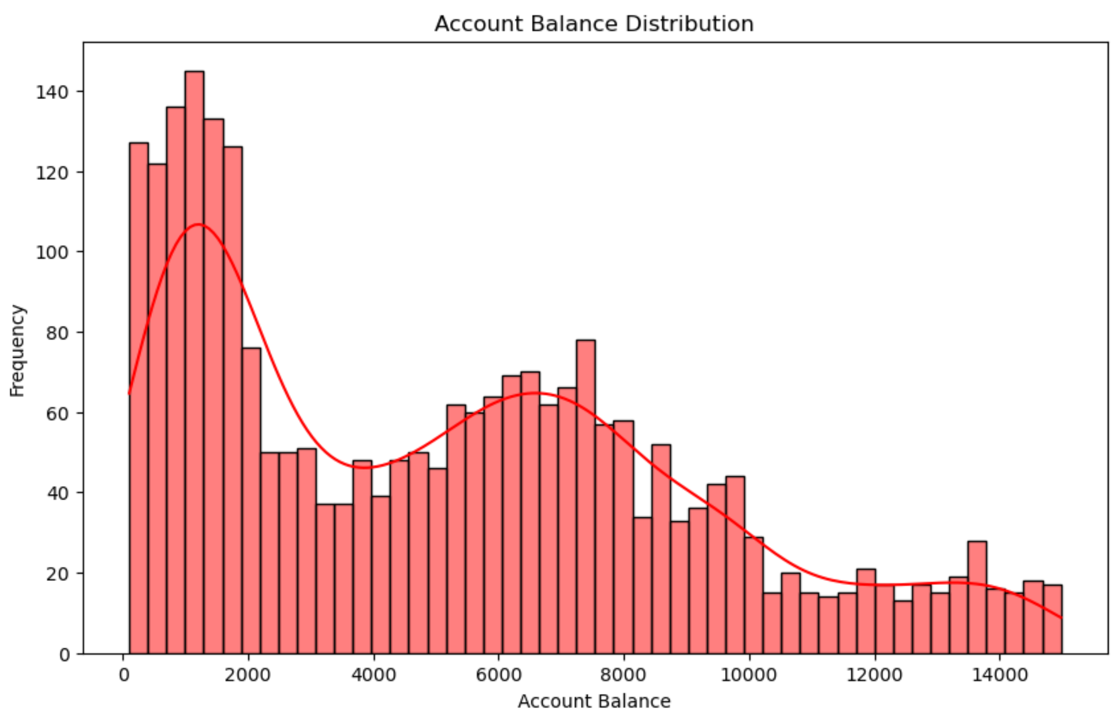
```
 3   TransactionDate          2512 non-null   object
 4   TransactionType          2512 non-null   object
 5   Location                 2512 non-null   object
 6   DeviceID                 2512 non-null   object
 7   IP Address               2512 non-null   object
 8   MerchantID               2512 non-null   object
 9   Channel                  2512 non-null   object
 10  CustomerAge              2512 non-null   int64
 11  CustomerOccupation       2512 non-null   object
 12  TransactionDuration      2512 non-null   int64
 13  LoginAttempts            2512 non-null   int64
 14  AccountBalance           2512 non-null   float64
 15  PreviousTransactionDate  2512 non-null   object
dtypes: float64(2), int64(3), object(11)
memory usage: 314.1+ KB
None

Summary Statistics:
       TransactionAmount  CustomerAge  TransactionDuration  LoginAttempts  \
count        2512.000000  2512.000000          2512.000000    2512.000000
mean          297.593778    44.673965           119.643312       1.124602
std           291.946243    17.792198            69.963757       0.602662
min             0.260000    18.000000            10.000000       1.000000
25%            81.885000    27.000000            63.000000       1.000000
50%           211.140000    45.000000           112.500000       1.000000
75%           414.527500    59.000000           161.000000       1.000000
max          1919.110000    80.000000           300.000000       5.000000

       AccountBalance
count     2512.000000
mean      5114.302966
std       3900.942499
min        101.250000
25%       1504.370000
50%       4735.510000
75%       7678.820000
max      14977.990000

Missing Values in Each Column:
Series([], dtype: int64)
Unique Counts for Each Column:
TransactionID              2512
AccountID                   495
TransactionAmount          2455
TransactionDate            2512
TransactionType               2
Location                     43
DeviceID                    681
```

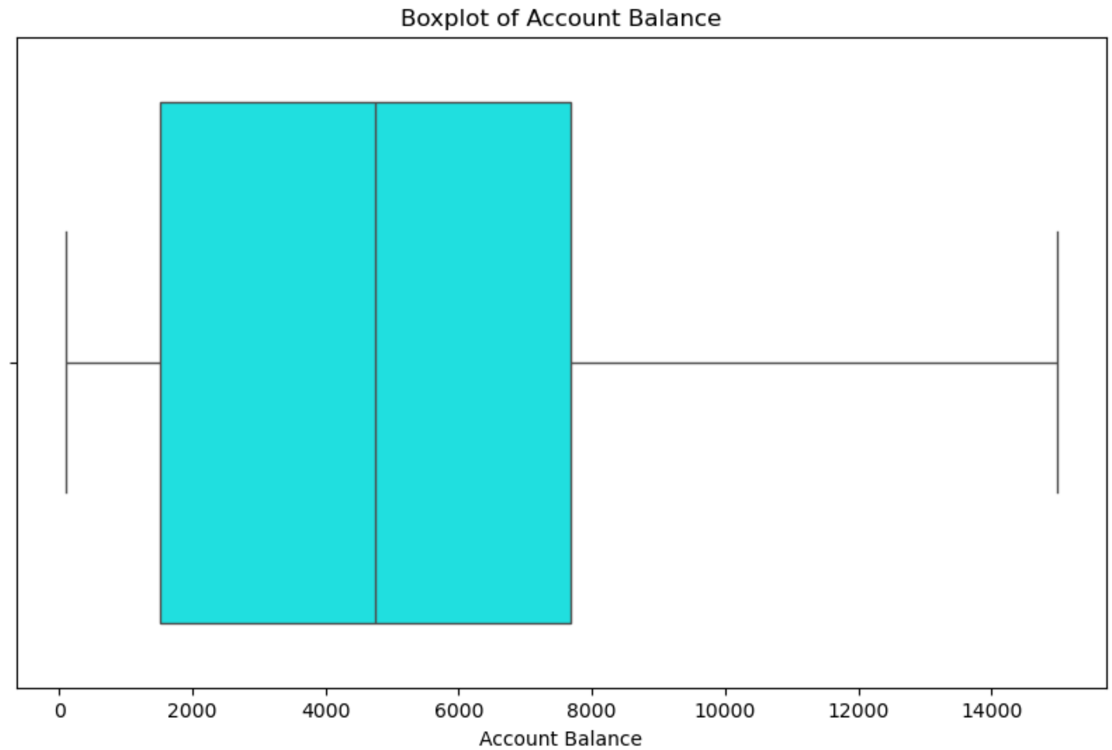
```
IP Address                 592
MerchantID                 100
Channel                      3
CustomerAge                 63
CustomerOccupation           4
TransactionDuration        288
LoginAttempts                5
AccountBalance            2510
PreviousTransactionDate    360
dtype: int64
TransactionID               object
AccountID                   object
TransactionAmount          float64
TransactionDate             object
TransactionType             object
Location                    object
DeviceID                    object
IP Address                  object
MerchantID                  object
Channel                     object
CustomerAge                  int64
CustomerOccupation          object
TransactionDuration          int64
LoginAttempts                int64
AccountBalance             float64
PreviousTransactionDate     object
dtype: object

Value Counts for Channel:
Channel
Branch    868
ATM       833
Online    811
Name: count, dtype: int64

Value Counts for CustomerOccupation:
CustomerOccupation
Student     657
Doctor      631
Engineer    625
Retired     599
Name: count, dtype: int64

Value Counts for Location:
Location
Fort Worth          70
Los Angeles         69
Oklahoma City       68
```

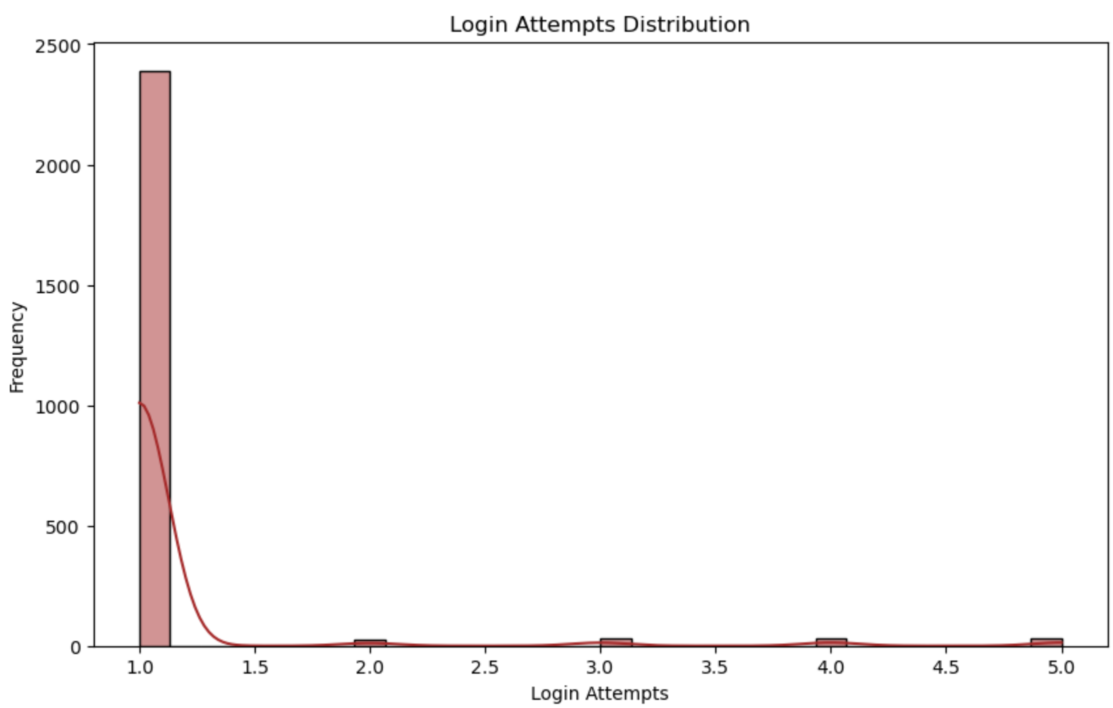
```
Charlotte           68
Tucson              67
Philadelphia        67
Omaha               65
Miami               64
Detroit             63
Houston             63
Memphis             63
Denver              62
Kansas City         61
Boston              61
Mesa                61
Atlanta             61
Seattle             61
Colorado Springs    60
Jacksonville        60
Fresno              60
Chicago             60
Austin              59
San Jose            59
Raleigh             59
San Antonio         59
San Diego           59
Indianapolis        58
New York            58
San Francisco       57
Nashville           55
Milwaukee           55
Las Vegas           55
Virginia Beach      55
Phoenix             55
Columbus            54
Sacramento          53
Baltimore           51
Louisville          51
Dallas              49
Washington          48
El Paso             46
Portland            42
Albuquerque         41
Name: count, dtype: int64

Value Counts for TransactionType:
TransactionType
Debit     1944
Credit     568
Name: count, dtype: int64
```

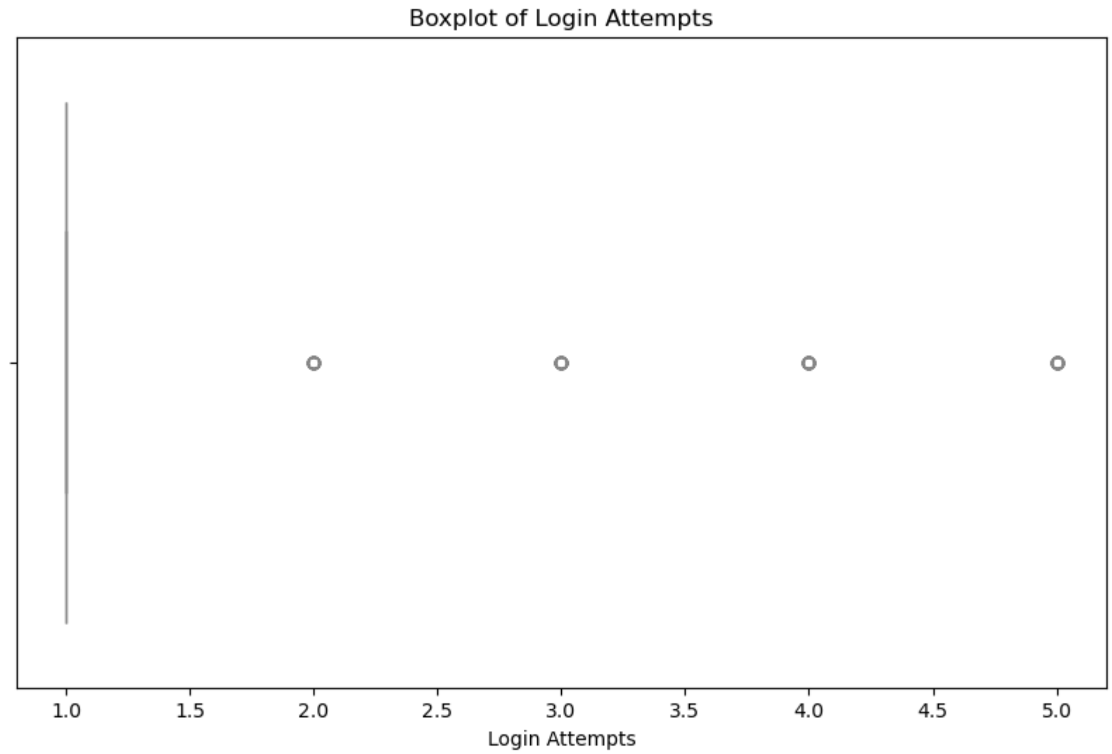

Transaction Amount Distribution
Count
0
50
100
150
200
250
300
350
0
250
500
750
1000
1250
1500
1750
2000
TransactionAmount

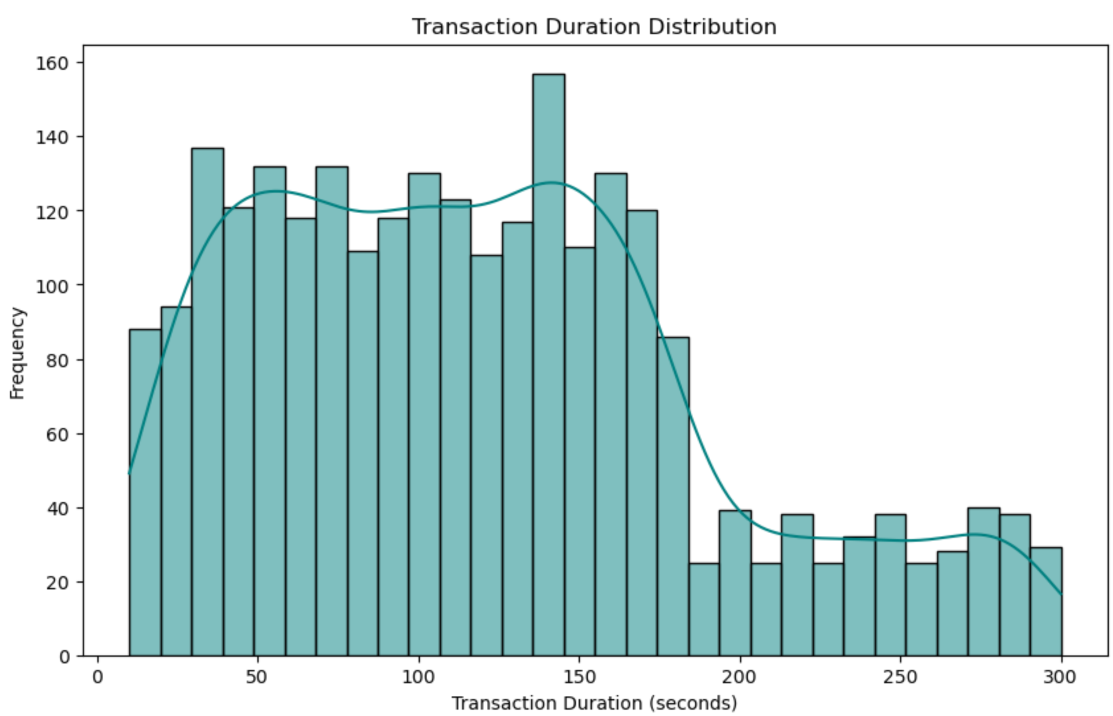

Boxplot of Transaction Amount
0
250
500
750
1000
1250
1500
1750
2000
Transaction Amount

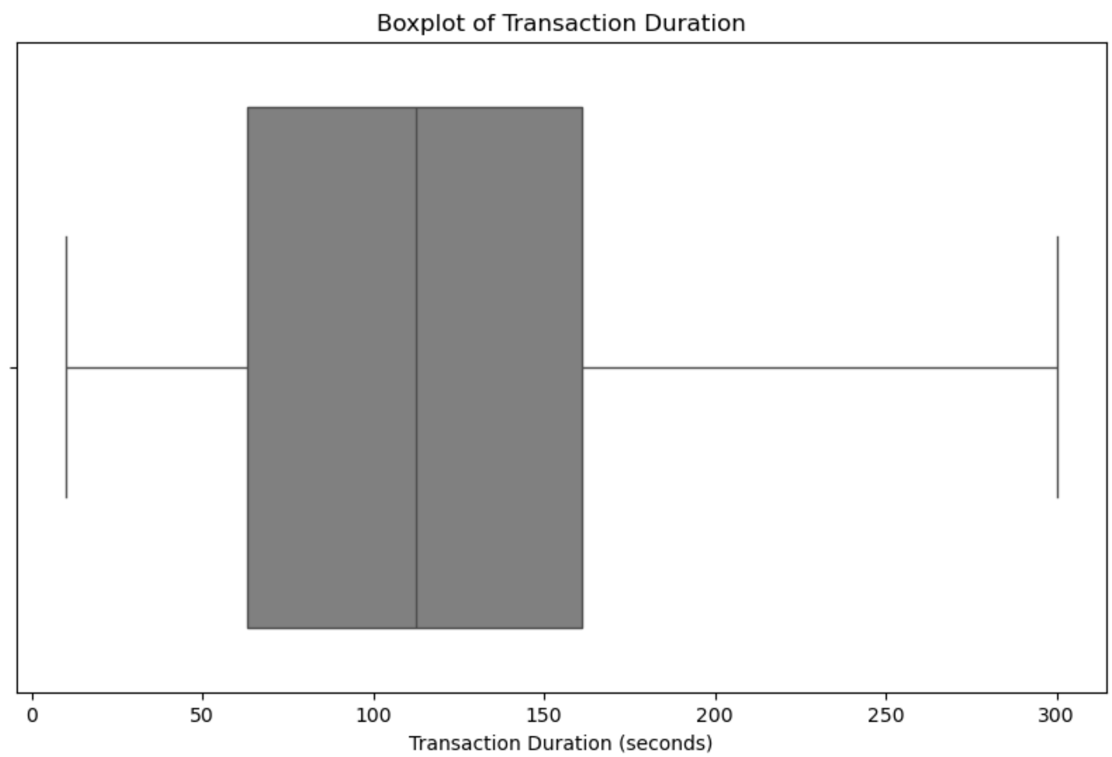

Account Balance Distribution
0
20
40
60
80
100
120
140
Frequency
0
2000
4000
6000
8000
10000
12000
14000
Account Balance


Boxplot of Account Balance
0
2000
4000
6000
8000
10000
12000
14000
Account Balance

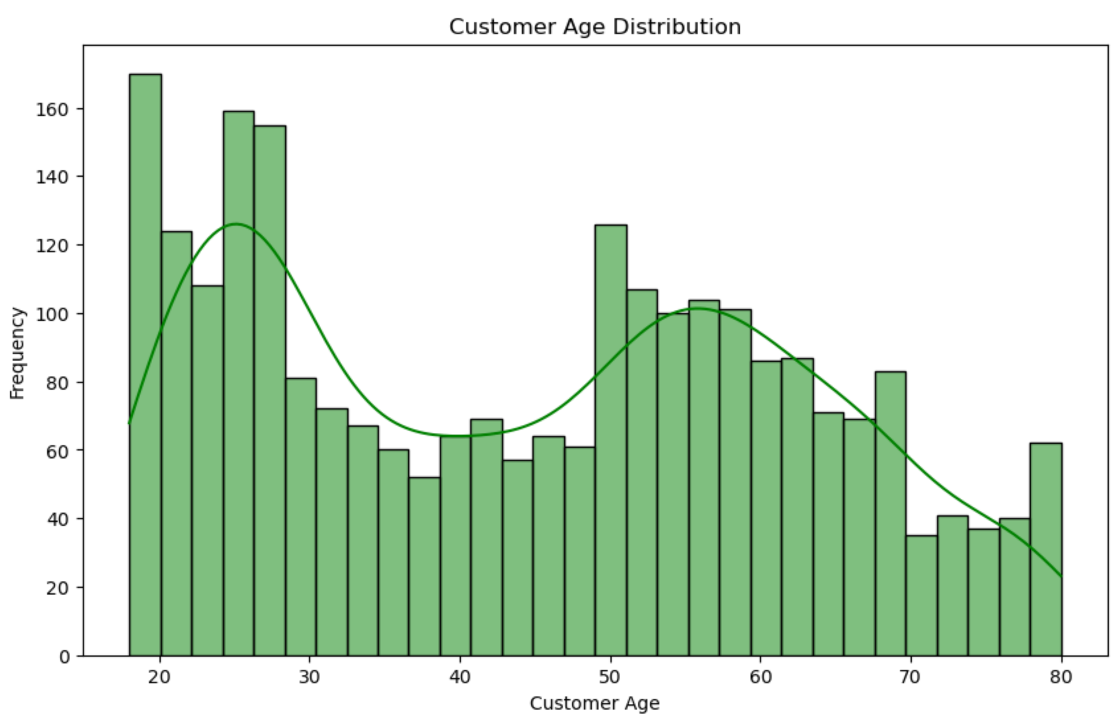

Login Attempts Distribution
2500
2000
1500
1000
500
0
Frequency
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Login Attempts


Boxplot of Login Attempts
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Login Attempts

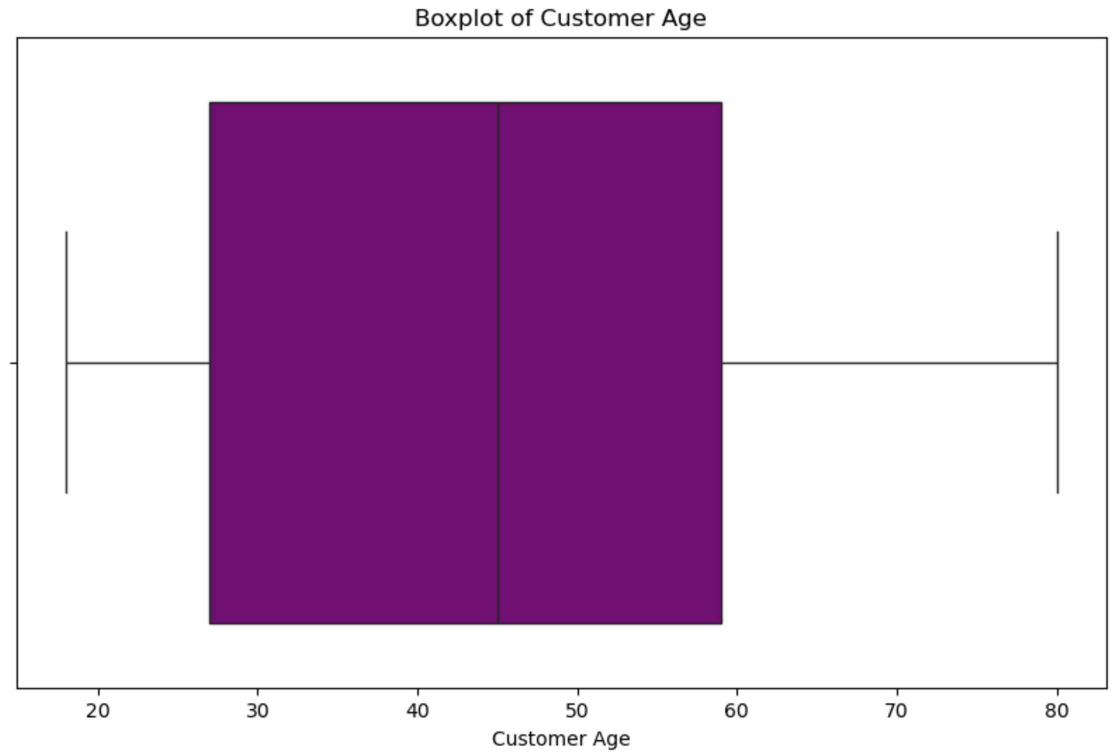

Transaction Duration Distribution
Frequency
0
20
40
60
80
100
120
140
160
0
50
100
150
200
250
300
Transaction Duration (seconds)



Boxplot of Transaction Duration
0
50
100
150
200
250
300
Transaction Duration (seconds)



Customer Age Distribution
Frequency
0
20
40
60
80
100
120
140
160
20
30
40
50
60
70
80
Customer Age


Boxplot of Customer Age
20
30
40
50
60
70
80
Customer Age