**HW5: Fraud Analytics**

**-Rijul Sherathia**

**Executive Summary**

Fraudulent credit card transactions can result in significant financial losses for businesses. Our project aimed to develop a model to improve fraud detection and reduce financial losses. By analyzing a dataset of credit card transactions, we were able to identify patterns and anomalies that are indicative of fraudulent activity. Our model was trained on historical data and achieved a False Discovery Rate (FDR) at 3% of 56.95% on out-of-time (OOT) data. Through our analysis, we estimated that our model could potentially save the business up to $21 million annually in fraudulent losses. This represents a significant return on investment and demonstrates the value of utilizing advanced analytics to improve fraud detection. Overall, our project provides a valuable tool for businesses to improve their fraud detection capabilities and protect their financial interests.

**Description of Data**

* Overview

The dataset contains credit card transactions obtained from **Mark Nigrini** website of a Tennessee located **US Government organization** for the business purposes in the **year 2010**, including information such as the card number, merchant number, transaction amount, and whether it was flagged as fraud. The data will likely be used to identify patterns and anomalies to improve fraud detection in credit card transactions. Dataset consists of **10 fields** populated with **96753 records**.

* Summary Tables

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Field names** | **# Populated Records** | **% populated** | **# zeros** | **Min** | **Max** | **Mean** | **Most Common** | **Std. dev** |
| Date | 96,753 | 100.00% | 0 | 1/1/2010 | 12/31/2010 | 6/25/2010 | 2/28/2010 | ~99 days |
| Amount | 96,753 | 100.00% | 0 | $0.01 | $3,102,045.53 | $427.88 | $3.62 | $10,006.14 |

**Table 1** : Numerical Fields Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **# Records With Values** | **% Populated** | **# Zeros** | **# Unique Values** | **Most Common** |
| Recnum | 96,753 | 100.00% | 0 | 96,753 | 1 |
| Cardnum | 96,753 | 100.00% | 0 | 1,645 | 5142148452 |
| Merchnum | 93,378 | 96.51% | 0 | 13,091 | 930090121224 |
| Merch description | 96,753 | 100.00% | 0 | 13,126 | GSA-FSS-ADV |
| Merch state | 95,558 | 98.76% | 0 | 227 | TN |
| Merch zip | 92,097 | 95.19% | 0 | 4,567 | 38,118 |
| Transtype | 96,753 | 100.00% | 0 | 4 | P |
| Fraud | 96,753 | 100.00% | 95,694 | 2 | 0 |

**Table 2** : Categorical Fields Table

**Data Exploration and Cleaning**

1. The dataset in CSV format contains several columns with missing values denoted as 'NaN'. We will exclude these columns from our analysis.
2. The data type of the date field in the dataset is currently set as 'object'. We will convert it to the 'datetime' format.
3. To comply with the given requirements, we will retain only the data where the transaction type is 'P’ and remove all other transactions from the dataset.
4. In addition, there is an outlier or spike in the data where the transaction amount exceeds $3000000. We will remove this outlier from the dataset.
5. After cleaning the dataset, we will check the number of rows that contain missing or 'NaN' values.

* **Clean and impute ‘merchnum’ field :**
  1. Initially, there were 3198 null rows in the 'Merchnum' field. We dropped all the columns with NaN values.
  2. Rows with 'Merchnum' as 0 were replaced with NaN values, increasing the count of NaN rows to 3251.
  3. We created a dictionary 'merchdes\_merchnum' to store unique non-null 'Merch description' with their corresponding 'Merchnum' values. We skipped rows where 'Merchnum' and 'Merch description' were both null.
  4. We replaced all Null or NaN values in the 'Merchnum' column using the 'merchdes\_merchnum' dictionary, reducing the null values from 3198 to 2094.
  5. We manually set 'Merchnum' as 'unknown' for all rows with 'Merch description' - 'RETAIL CREDIT ADJUSTMENT' or 'RETAIL DEBIT ADJUSTMENT', further reducing null rows to 1403.
  6. We created a new dictionary 'merchnum\_create' to map unique non-null 'Merch description' values to corresponding new unique 'Merchnum' values for rows with missing 'Merchnum' values. We iterated over the dataset to replace all NaN values in the 'Merchnum' column with the corresponding value in the 'merchnum\_create' dictionary.
  7. As a result, the number of null/NaN values in the 'Merchnum' column was reduced to 0. The imputation of the 'Merchnum' field is complete.

However, at this point, we observe that except the columns ‘Merch description’, ‘Merch state’ and ‘Merch zip’ all others have 0 null counts and hence we now proceed towards cleaning and imputing these fields.

* **Clean and Impute ‘Merch state’ Field :**
  1. Zipcodes and states are related, so we'll start by analyzing them.
  2. We find 'Merch zip' with non-missing values and 'Merch state' with missing values, and create a dictionary called 'zip\_state' to store unique zipcodes as keys and their corresponding state values.
  3. We populate 'zip\_state' with unique values of 'Merch zip' and their corresponding 'Merch state' values where 'Merch zip' has non-missing values. We manually add some known key-value pairs to the dictionary.
  4. We create another dictionary called 'merchnum\_state' to store unique 'Merchnum' values and their corresponding 'Merch state' values where 'Merchnum' has non-missing values.
  5. We create a third dictionary called 'merchdes\_state' to store unique 'Merch description' values and their corresponding 'Merch state' values where 'Merch description' has non-missing values. We fill the missing 'Merch state' values using these three dictionaries, reducing the number of missing values.
  6. We replace missing 'Merch state' values with 'unknown' where the 'Merch description' is 'RETAIL CREDIT ADJUSTMENT' or 'RETAIL DEBIT ADJUSTMENT', further reducing the number of missing values.
  7. We store all possible state values in an array and set the value of all foreign states to 'foreign' if they are not in the array.
  8. Finally, we replace all remaining null values in the 'Merch state' column with 'unknown', reducing the number of null or missing data in the 'Merch state' column to 0.
* **Clean and Impute Merch zip Column:**

To impute missing values in the 'Merch zip' column, the following steps were taken:

* 1. Created dictionaries 'merchnum\_zip' and 'merchdes\_zip' to store non-null values of 'Merchnum' and 'Merch description' columns as keys and their corresponding 'Merch zip' column values as values.
  2. Used the above dictionaries to replace null values in the 'Merch zip' column in accordance with the respective 'Merchnum' and 'Merch description' column values.
  3. Replaced missing values in the 'Merch zip' column with 'unknown' where the 'Merch description' column was 'RETAIL CREDIT ADJUSTMENT' or 'RETAIL DEBIT ADJUSTMENT'.
  4. Created a dictionary 'state\_zip' to store non-null values of the 'Merch state' column as keys and corresponding non-null value of one of the zip codes belonging to this state.
  5. Used the above dictionary to replace null values in the 'Merch zip' column with the zip code from the dictionary according to the 'Merch state' column value.
  6. Replaced remaining null values in the 'Merch zip' column with 'unknown'. This resulted in the missing data count for 'Merch zip' being reduced to 0.

**Variable Creation**

Credit card fraud occurs when someone uses another person's credit card information without their consent to make unauthorized purchases or access funds. Fraudsters can obtain credit card information in various ways, including stealing physical cards, phishing, skimming, or hacking into databases.

To detect fraudulent transactions, we can create variables based on the transaction data, such as:

* Frequency of transactions: This variable counts the number of transactions made by a particular credit card in a given time period. Fraudsters may use a stolen credit card for a high volume of transactions in a short period.
* Time between transactions: This variable calculates the time interval between consecutive transactions made by a credit card. Unusual time gaps between transactions could indicate that a card has been stolen.
* Transaction amount: This variable measures the amount of money spent in each transaction. Fraudsters may make multiple small transactions instead of one large transaction to avoid triggering fraud detection systems.
* Merchant category code (MCC): This variable classifies merchants into different categories based on their business type. Certain MCCs may be more prone to fraud than others.
* Geographic location of transactions: This variable captures the location of transactions. Unusual geographic patterns, such as transactions made in multiple countries within a short time, could indicate fraudulent activity.
* Time of day of transactions: This variable captures the time of day of transactions. Unusual transaction times, such as transactions made in the middle of the night, could also indicate fraudulent activity.

**Summary Table:**

|  |  |
| --- | --- |
| **Description** | **# Variables Created** |
| **Date of week target encoded**:  average fraud percentage of that day | 1 |
| **Month target encoded**:  average fraud percentage of that month | 1 |
| **State Risk target encoded**:  Top 15 Fraud Merchant State with highest fraud percentage | 1 |
| **Benford’s Law Card number Variable**:  count the # of first digits in card number | 1 |
| **Benford’s Law Merchant number Variable**:  count the # of first digits in Merchant number | 1 |
| **Benford’s Law Merchant Zip Variable**:  count the # of first digits in Merchant Zip | 1 |
| **Day Since Variable**:  Number of days since application with that entity was seen | 18 |
| **Velocity Variable**:  Number of records with the same entity over the last {0,1,3,7,14,30,60} days | 1,134 |
| **Relative Velocity Variable**:  Count of records and total amount with same entities seen in the past {0,1} day divided by the number of applications with those same entities seen in the last {7,14,30,60} days | 288 |
| **Velocity Density Variable**:  Count of records with same entities seen in the past {0,1} day divided by day since those same entities seen in the last {7,14,30,60} days | 144 |
| **Counts by Entity:**  Number of unique entities for a particular field over the last {0,1,3,7,14,30,60} days | 1,836 |
| **Cross Entity Uniqueness Variable**:  Number of records with unique combinations of all the entities | 306 |
| **Entity Amount Variability**:  Amount variability with the amount of same entity seen over the last {0,1,3,7,14,30,60} days | 324 |
| **Relative Velocity Variable (squared)**:  Count of records and total amount with same entities seen in the past {0,1} day divided by square of the number of applications with those same entities seen in the last {3,7,14,30} days | 144 |
| **Binning Amount:**  Creates 5 equal sized bins to divide amount column and assigns labels (1,5) to each bin | 1 |

**Feature selection**

What: Feature selection involves selecting the most relevant variables or features from a dataset that will be used in building a predictive model. This process involves analyzing the correlation between variables, identifying redundant variables, and removing variables that are not relevant to the outcome.

Why: The goal of feature selection is to improve the predictive accuracy of a model by reducing the complexity of the model and eliminating noise and irrelevant variables. By selecting only the most important variables, feature selection can help reduce overfitting, decrease the computational cost of building the model, and improve the interpretability of the model.

How: Feature selection was performed using a combination of techniques, including LGBM and Random Forest models with forward or backward selection, as well as variable wrappers and filters. The aim was to identify the most important features that contribute significantly to fraud detection while eliminating redundant or irrelevant features. Variable wrappers use a search algorithm to find the optimal subset of features that maximizes the model's performance, while filters use statistical tests to identify the most relevant features based on their correlation with the target variable.

Methods:

1. LGBM Classifier with Forward Selection
   1. Number of wrappers = 20 ; Number of filters = 200
   2. Number of wrappers = 30 ; Number of filters = 200
   3. Number of wrappers = 20 ; Number of filters = 300
   4. Number of wrappers = 20 ; Number of filters = 400
2. LGBM Classifier with Backward Selection
   1. Number of wrappers = 20 ; Number of filters = 50
3. Random Forest Classifier with Forward Selection
   1. Number of wrappers = 20 ; Number of filters = 200
   2. Number of wrappers = 30 ; Number of filters = 200
   3. Number of wrappers = 20 ; Number of filters = 300

Chart, line chart

Description automatically generated

|  |  |  |
| --- | --- | --- |
| **wrapper order** | **variable** | **filter score** |
| 1 | card\_merch\_total\_14 | 0.630048056 |
| 2 | card\_zip3\_max\_14 | 0.629514577 |
| 3 | card\_zip3\_count\_7 | 0.387860248 |
| 4 | Merchnum\_desc\_total\_1 | 0.528444884 |
| 5 | Merchnum\_desc\_max\_1 | 0.523694225 |
| 6 | Merchnum\_desc\_med\_3 | 0.429393431 |
| 7 | card\_zip3\_variability\_max\_3 | 0.385868376 |
| 8 | zip3\_variability\_avg\_3 | 0.405014398 |
| 9 | merch\_zip\_total\_14 | 0.44001854 |
| 10 | merch\_zip\_max\_3 | 0.514480955 |
| 11 | Card\_Merchnum\_desc\_total\_60 | 0.595018827 |
| 12 | state\_des\_med\_3 | 0.425544947 |
| 13 | Merchnum\_desc\_total\_7 | 0.517123406 |
| 14 | merch\_zip\_max\_1 | 0.522152981 |
| 15 | card\_merch\_total\_30 | 0.615461086 |
| 16 | Card\_Merchnum\_desc\_total\_30 | 0.606279583 |
| 17 | Card\_Merchnum\_Zip\_total\_30 | 0.612931392 |
| 18 | Card\_Merchnum\_Zip\_total\_14 | 0.62742092 |
| 19 | state\_des\_total\_14 | 0.490871546 |
| 20 | Merchnum\_desc\_max\_3 | 0.516807894 |

**Model Exploration**

Model exploration with hyperparameter tuning involves trying out different machine learning models and adjusting their hyperparameters to achieve the best performance on the given dataset. In this project, several models were explored like Logistic Regression, Decision Tree, Random Forest, LightGBM, Neural Networks, XGBoost and Gradient Boosting Classifier. The hyperparameters that were tuned for each model depended on the specific algorithm, but generally included parameters like learning rate, number of estimators, maximum depth, minimum samples per leaf, regularization strength, and activation function. After exploring and tuning each model, the best performing model was selected based on its performance metrics and used to make predictions on the test data.

Calendar

Description automatically generated

Calendar

Description automatically generated

Chart

Description automatically generated

**Final Model Performance**

Description:

The final best model selected is the RandomForestClassifier, which is an ensemble model that uses a collection of decision trees to make predictions.

The hyperparameters for the model are as follows:

* criterion: This parameter is used to measure the quality of the split. 'entropy' is the criterion used in this case, which measures the amount of information gained in each split.
* n\_estimators: This parameter specifies the number of trees in the forest. In this case, 80 trees are used.
* max\_depth: This parameter specifies the maximum depth of each decision tree. In this case, the maximum depth is set to 7, which means that each decision tree can have at most 7 levels.
* min\_samples\_split: This parameter specifies the minimum number of samples required to split an internal node. In this case, the minimum number of samples required is 20.
* min\_samples\_leaf: This parameter specifies the minimum number of samples required to be at a leaf node. In this case, the minimum number of samples required is 60.

By tuning these hyperparameters, the model is able to achieve the best possible performance on the dataset. The combination of these hyperparameters has been found to produce the highest accuracy while avoiding overfitting.

Summary Tables:

Table

Description automatically generated

Table, Excel

Description automatically generated

Graphical user interface, application, table, Excel

Description automatically generated

**Financial Curve and Recommended Cutoff:**

Chart, line chart

Description automatically generated

The cutoff percentage for fraud detection can be decided by analyzing the financial curve representing the fraud amount caught, overall savings, and lost revenue. The financial curves show that at a **cutoff of 3%,** the model is able to catch a significant amount of fraud while also minimizing the amount of lost revenue. This cutoff point provides a good balance between fraud detection and minimizing false positives.

To decide the cutoff percentage, one needs to look at the point where the fraud amount caught starts to level off and the overall savings start to decrease. This is because setting the cutoff too high can result in missing some fraudulent transactions, while setting it too low can result in too many false positives, which can be costly for the business. One approach is to set the cutoff percentage at the point where the slope of the curve starts to flatten out, which is typically the point of diminishing returns. Another approach is to set the cutoff percentage based on the organization's risk appetite and cost-benefit analysis. It is important to note that the cutoff percentage should be regularly reviewed and adjusted based on the changing fraud patterns and business needs.

**Summary:**

The data exploration and cleaning process for a dataset in CSV format involved various steps taken include removing columns with missing values, converting the data type of the date field to 'datetime', retaining only transactions with a specific transaction type, removing an outlier, and cleaning and imputing the 'Merchnum', 'Merch state', and 'Merch zip' fields. The 'Merchnum' field is cleaned and imputed using dictionaries and manual methods. 'Merch state' and 'Merch zip' fields are cleaned and imputed using dictionaries based on related fields. Missing values are replaced with 'unknown', 'foreign', or corresponding values from dictionaries. The process results in the reduction of the missing data count for each field to 0.

The variables created for detecting credit card fraud include frequency of transactions, time between transactions, transaction amount, merchant category code, geographic location of transactions, and time of day of transactions. Additionally, several variables are created based on the application data, such as the number of days since an application with that entity was seen, the number of records with the same entity over a given time period, and the amount variability with the same entity seen over a given time period. Other variables are created based on Benford's Law, including counting the number of first digits in the card number, merchant number, and merchant zip. Finally, variables are created based on the unique entities and their combinations, including counts by entity, cross-entity uniqueness, and entity amount variability. These variables aim to capture unusual patterns or behaviors that may indicate fraudulent activity.

The feature selection process involved using LGBM and Random Forest classifiers with forward or backward selection, as well as variable wrappers and filters. The goal was to identify the most important features that contribute significantly to fraud detection while eliminating redundant or irrelevant features. Variable wrappers were used to find the optimal subset of features that maximizes the model's performance. The search algorithm was used to identify the best combination of features by evaluating the model's performance on different subsets of features. The algorithm added or removed features iteratively until it found the best subset. Filters, on the other hand, used statistical tests to identify the most relevant features based on their correlation with the target variable. The filter score was computed for each feature, and the features were ranked based on their filter scores. Finally, the most important features for fraud detection were identified based on the results of the different techniques used. The top 20 features, ranked by their filter scores, were identified, and used in building the predictive model for fraud detection.

In this step, several models were explored, including Logistic Regression, Decision Tree, Random Forest, LightGBM, Neural Networks, XGBoost and Gradient Boosting Classifier. After training and evaluating each model, the next step is to tune the hyperparameters. Hyperparameters were tuned using grid search, which involves defining a grid of possible hyperparameter values and evaluating the performance of the model for each combination of hyperparameters in the grid. Once the hyperparameters are tuned, the best performing model is selected based on its performance metrics. The selection process involves comparing the performance of each model using a performance vs complexity plot and a box plot.

Statement of Model Performance:

Model performance can be determined using train-test split and out-of-time (OOT) accuracy by evaluating the performance of the model on different subsets of the data. The financial curve analysis involved calculating the fraud amount caught, overall savings, and lost revenue at different cutoff percentages. The financial curves were used to determine the optimal cutoff percentage that balances the trade-off between fraud detection and minimizing false positives. The cutoff percentage selected was 3%, which resulted in significant fraud detection while minimizing lost revenue.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | **# Records** | **# Goods** | **# Bads** | **Fraud Rate** |
| **Training** | 58,779 | 58,182 | 597 | 1.0157% |
| **Testing** | 25,191 | 24,908 | 283 | 1.1234% |
| **OOT** | 12,427 | 12,248 | 179 | 1.4404% |

Train Accuracy: 78.72%

Train Accuracy: 78.09%

OOT Accuracy: 56.98%

Estimated $ Fraud Savings: $ 21,228,000

FDR 3% for OOT: 56.98%

**Appendix:**

Data Quality Report

1. Data Description

The dataset contains credit card transactions obtained from **Mark Nigrini** website of a Tennessee located **US Government organization** for the business purposes in the **year 2010**, including information such as the card number, merchant number, transaction amount, and whether it was flagged as fraud. The data will likely be used to identify patterns and anomalies to improve fraud detection in credit card transactions. Dataset consists of **10 fields** populated with **96753 records**.

1. Summary Tables

**Table 1** : Numerical Fields Table

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Field names** | **# Populated Records** | **% populated** | **# zeros** | **Min** | **Max** | **Mean** | **Most Common** | **Std. dev** |
| Date | 96,753 | 100.00% | 0 | 1/1/2010 | 12/31/2010 | 6/25/2010 | 2/28/2010 | ~99 days |
| Amount | 96,753 | 100.00% | 0 | $0.01 | $3,102,045.53 | $427.88 | $3.62 | $10,006.14 |

**Table 2** : Categorical Fields Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field Name** | **# Records With Values** | **% Populated** | **# Zeros** | **# Unique Values** | **Most Common** |
| Recnum | 96,753 | 100.00% | 0 | 96,753 | 1 |
| Cardnum | 96,753 | 100.00% | 0 | 1,645 | 5142148452 |
| Merchnum | 93,378 | 96.51% | 0 | 13,091 | 930090121224 |
| Merch description | 96,753 | 100.00% | 0 | 13,126 | GSA-FSS-ADV |
| Merch state | 95,558 | 98.76% | 0 | 227 | TN |
| Merch zip | 92,097 | 95.19% | 0 | 4,567 | 38,118 |
| Transtype | 96,753 | 100.00% | 0 | 4 | P |
| Fraud | 96,753 | 100.00% | 95,694 | 2 | 0 |

1. Visualization of each Field
2. Field Name: *Record Number*

Description: Recnum is a field that represents a **unique identifier for each transaction record**. It is an incremental integer that starts from 1 and increments by 1 for each new transaction based on the date field. This field can be used as an alternative to identify the sequence of transactions performed for the same date.

1. Field Name: *Card Number*

Description: Cardnum is a field that represents the **credit card number used for the transaction**. The Cardnum field can be useful for identifying patterns of fraud or suspicious activity associated with a particular credit card, such as multiple transactions occurring in a short time-period or transactions occurring in geographically distant locations. This field contains sensitive information and proper measures should be taken to protect this data from unauthorized access.

Visualization:

Chart, bar chart

Description automatically generated

1. Field Name: *Date*

Description: Date is a field that represents the **date on which the transaction occurred** and is a critical piece of information for analyzing transaction trends and patterns over time. The date field can be useful for identifying seasonality or other trends in transaction volume, as well as identifying periods of increased or decreased fraud activity. Additionally, the date field can be used to filter transactions based on a specific date range or to aggregate transactions by week, month, or year.

Visualization:

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

1. Field Name: *Merchant Number*

Description: Merchnum is a field that represents a **unique identifier for the merchant** where the transaction occurred. The merchnum field can be useful for identifying patterns of fraud or suspicious activity associated with specific merchants or groups of merchants, such as unusually high transaction volumes. Additionally, the merchnum field can be used to group transactions by merchants for reporting and analysis purposes.

Visualization:

Chart, waterfall chart

Description automatically generated

Chart, bar chart

Description automatically generated

1. Field Name: *Merchant Description*

Description: Merch description is a field that provides a **brief description of the goods or services purchased from the merchant**. This field may contain text descriptions or codes that provide additional information about the transaction. The field can be useful for identifying patterns in transaction activity, such as the types of products or services that are frequently purchased, or for identifying potentially fraudulent transactions based on unusual or unexpected merchandise descriptions. This field can also be used for reporting and analysis purposes, such as grouping transactions by product category or identifying trends in purchasing behavior over time.

Visualization:

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

1. Field Name: *Merchant state*

Description: Merch state is a field representing the **state where the merchant is located**. The merch state field can be useful for identifying patterns in transaction activity by geographic region, such as identifying areas where a particular merchant is more popular or where fraud is more prevalent.

Visualization:

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

1. Field Name: *Merchant Zip*

Description: Merchant Zip is a field that represents the **zip code of the merchant's location** where the credit card transaction was made. The zip code can help identify where most of the transactions are taking place and provide insights into the merchant's location. This information can be used to identify unusual or suspicious transactions that occur outside the expected geographic area.

Visualization:

Chart, bar chart

Description automatically generated

Chart

Description automatically generated

1. Field Name: *Transaction Type*

Description: Transtype is a field that represents the **type of transaction made with the credit card**. It provides information on whether the transaction was a Purchase(P), Authorization(A), or some other type of transaction. Transtype is a useful field for detecting fraud, as certain transaction types may be more likely to be fraudulent than others.

Visualization:

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

1. Field Name: *Amount*

Description: Amount is a field that represents the **dollar amount of the transaction** made with the credit card. Analyzing the Amount field can provide insights into customer behavior, such as what types of products or services are being purchased and at what price, but it is important to consider the context of the transaction. For Ex - A $500 transaction at a high-end department store may be perfectly legitimate, while a $500 transaction at a discount store may be more suspicious.

Visualization:

Chart, histogram

Description automatically generated

Chart, line chart

Description automatically generated

1. Field Name: *Fraud*

Description: Fraud is a field that **indicates whether a transaction is fraudulent or not**. A value of 0 typically indicates that the transaction is legitimate, while a value of 1 indicates that the transaction is fraudulent. Count of Fraud Transactions – 1,059.

Visualization:

Chart, waterfall chart

Description automatically generated

Chart, bar chart

Description automatically generated