Team 2 Report

Name	Sec	B.N	Code
Yousef Osama Mohamed	2	32	9211386
Nesma Abdelkader	2	28	9211292
Sara Gamal Gerges	1	20	9210455
Eman Ibraheem	1	14	9210265

Brief problem description

This project focuses on strengthening financial services in three key areas: fraud detection, customer analytics, and economic forecasting. We worked on building smart systems that help predict fraudulent transactions and deliver personalized customer insights. As part of our process, we thoroughly analyzed the data, printed and visualized it to understand the patterns better, and investigated which features strongly correlated with fraudulent activities. We aimed to create solutions that improve security, enhance customer engagement, and enable smarter, data-driven financial decisions.

Dataset

- **Main Dataset:** Transactions fraud dataset

User Table

- `id`
  - Unique identifier for the user
- `current_age`
  - Current age of the user
- `gender`
  - Gender of the user (Male, Female, Other)
- `birth_year`
  - Year of birth of the user
- `birth_month`
  - The month of birth of the user
- `retirement_age`
  - Expected retirement age of the user
- `address`
  - Residential address of the user
- `latitude`
  - Latitude coordinate of the user's address

- **longitude**
  - Longitude coordinate of the user's address
- **per_capita_income**
  - Average income per person in the user’s residential area
- **yearly_income**
  - The user’s self-reported or estimated annual income
- **total_debt**
  - The total outstanding debt the user has
- **credit_score**
  - Creditworthiness score of the user
- **num_credit_cards**
  - Total number of credit cards owned by the user

Cards Table

- **Id**
  - Unique identifier for the card record
- **Client_id**
  - An identifier linking the card to a specific user or client
- **Card_brand**
  - Brand of the card (e.g., Visa, MasterCard, American Express)
- **Card_type**
  - Type of the card (e.g., Credit, Debit, Prepaid)
- **Card_number**
  - Full card number (PAN) assigned to the user
- **Expires**
  - Expiration date of the card (MM/YY format)
- **CVV**
  - Card Verification Value (security code)
- **Has_chip**
  - Indicates if the card has an EMV chip (Yes/No)

- `num_cards_issued`
  - Number of cards issued under the same account
- `credit_limit`
  - The maximum credit amount available on the card
- `acct_open_date`
  - Date when the account associated with the card was opened
- `year_pin_last_changed`
  - The year when the PIN for the card was last updated
- `Card_on_dark_web`
  - indicates if the card information was found in dark web monitoring (Yes/No)

Transactions Table

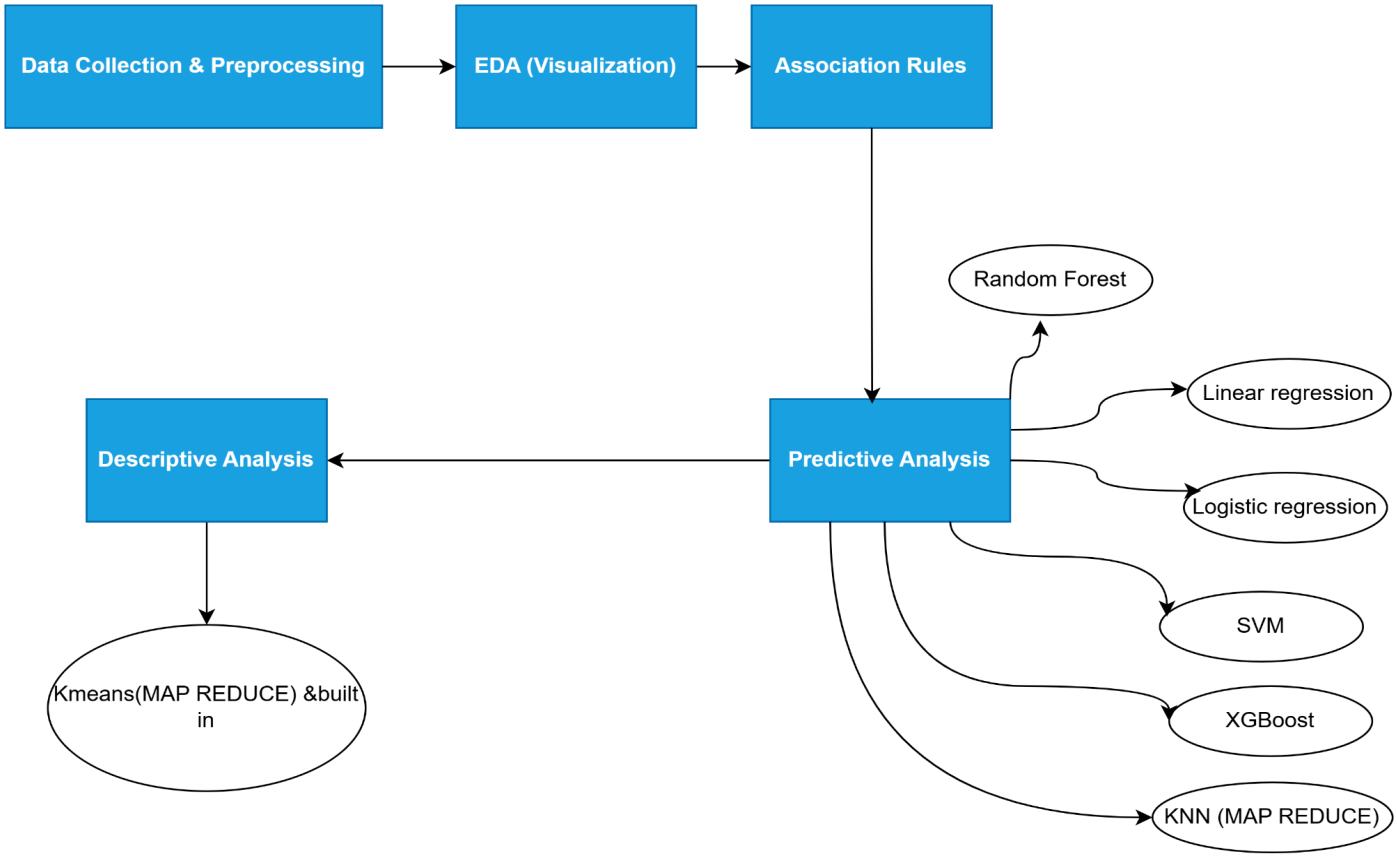
- `id`
  - Unique identifier for the transaction
- `date`
  - Date when the transaction occurred
- `client_id`
  - An identifier linking the transaction to the user or client
- `card_id`
  - Identifier of the card used in the transaction
- `amount`
  - Amount spent in the transaction
- `use_chip`
  - Indicates whether the card's chip was used (Yes/No)
- `merchant_id`
  - Unique identifier for the merchant
- `merchant_city`
  - City where the merchant is located
- `merchant_state`
  - State where the merchant is located

- `zip`
  - ZIP code of the merchant's location

- `mcc`
  - Merchant Category Code indicating the merchant’s business type

- `errors`
  - Errors, if any, that occurred during the transaction

Pipeline


Data Collection & Preprocessing
EDA (Visualization)
Association Rules
Random Forest
Linear regression
Descriptive Analysis
Predictive Analysis
Logistic regression
SVM
Kmeans(MAP REDUCE) &built in
XGBoost
KNN (MAP REDUCE)

Data Preprocessing

Transactions.csv

1. Load `mcc_codes.json` to retrieve error codes and convert it into a Spark DataFrame.
2. Load `fraud_labels.json` to retrieve fraud targets (Yes/No) and convert it into a Spark DataFrame.
3. Drop unnecessary columns in `transaction_df,` such as `merchant_state` and `zip`.
4. Fill null values in the `error` column with "No Error".
5. Join both the `mcc_codes` and `fraud_labels` DataFrames with `transaction_df`.
6. Change the data types of the `date` and `amount` columns in `transaction_df` to **double**.

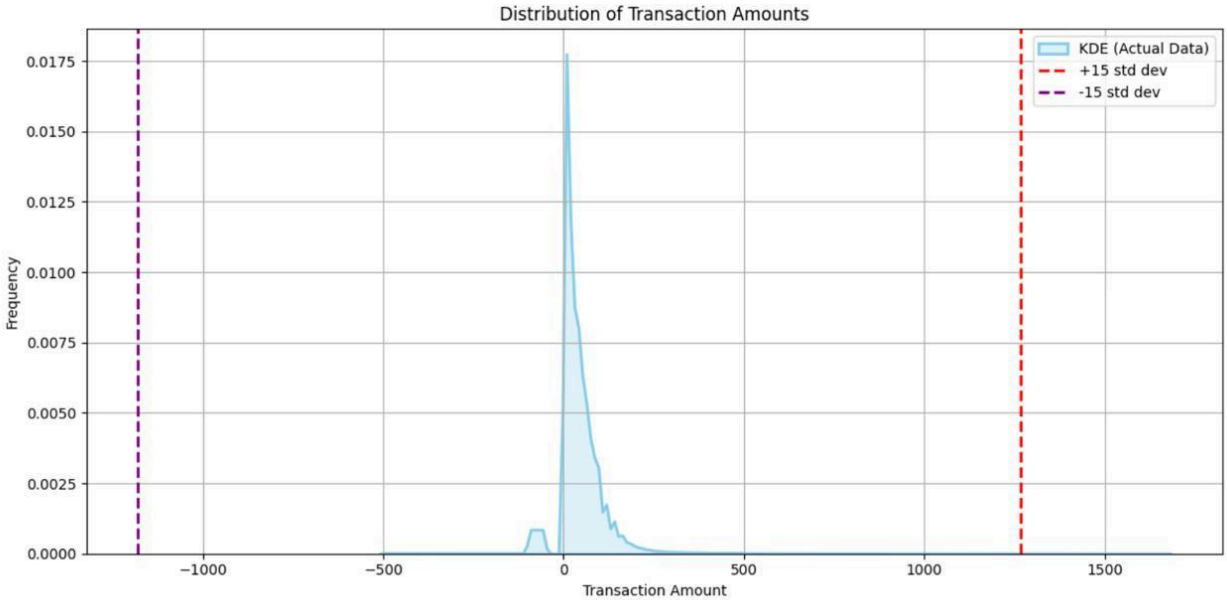
Users.csv

1. Change the data types of `Per_capita_income`, `total_debt`, and `yearly_income` columns in `users_df` to **double**.
2. Filter `users_df` to retain only users who have transactions.

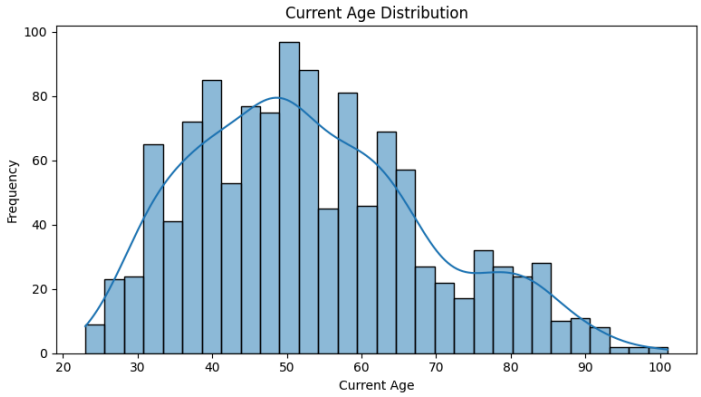
Cards.csv

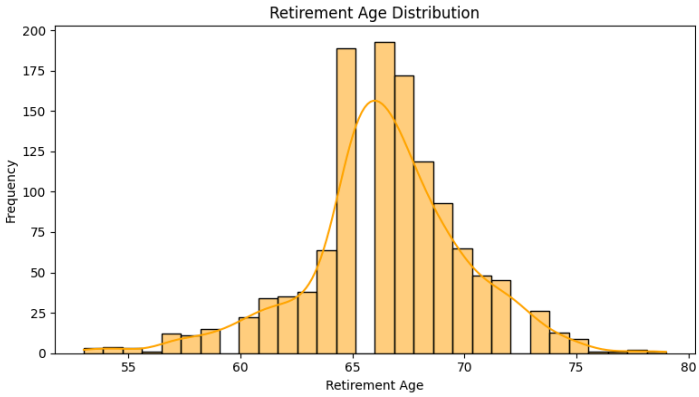
1. Change the data type of `credit_limit` in `cards_df` to **double**.
2. Filter `cards_df` to retain only cards associated with users in `users_df`.

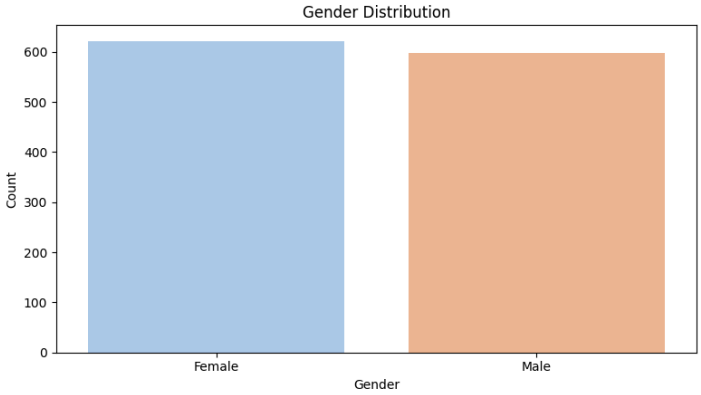
Data visualization

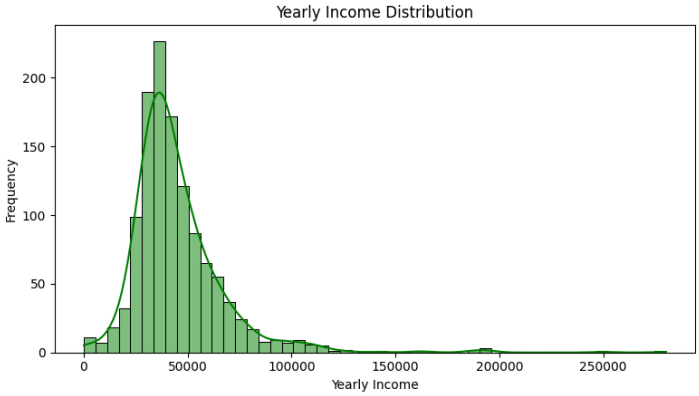


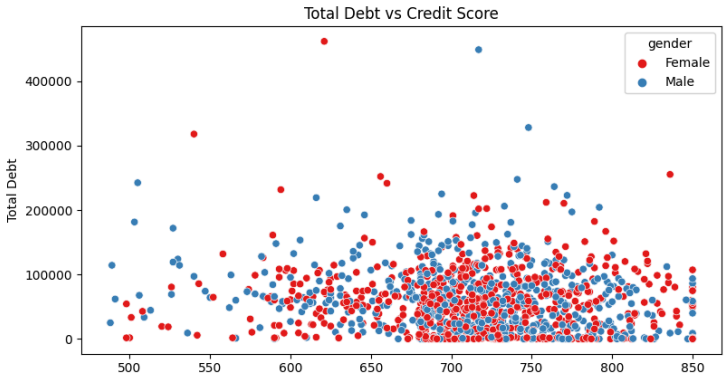
User Data Distributions

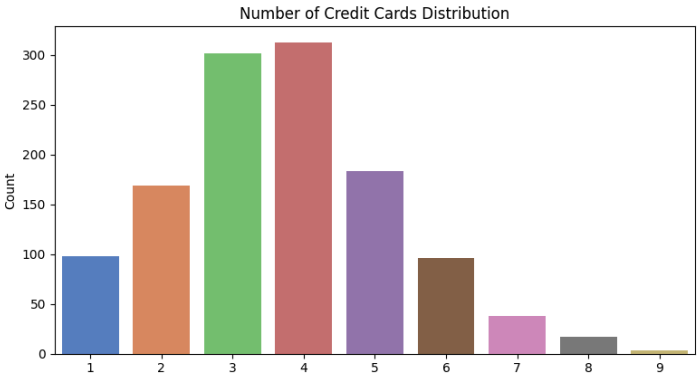


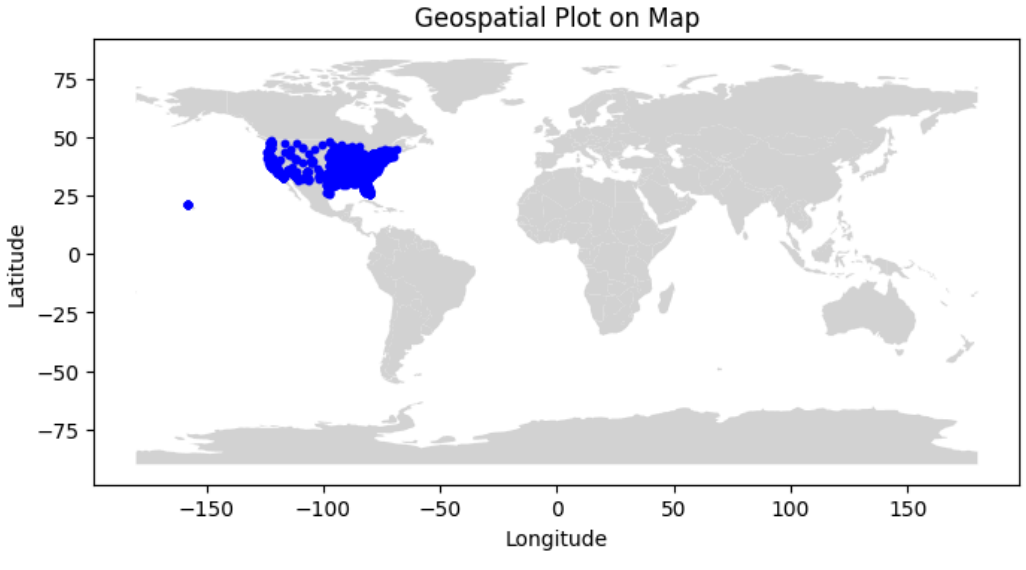








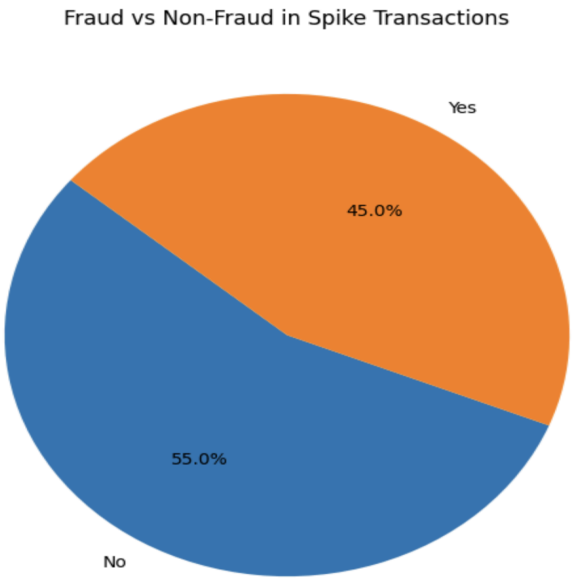




Users Insights

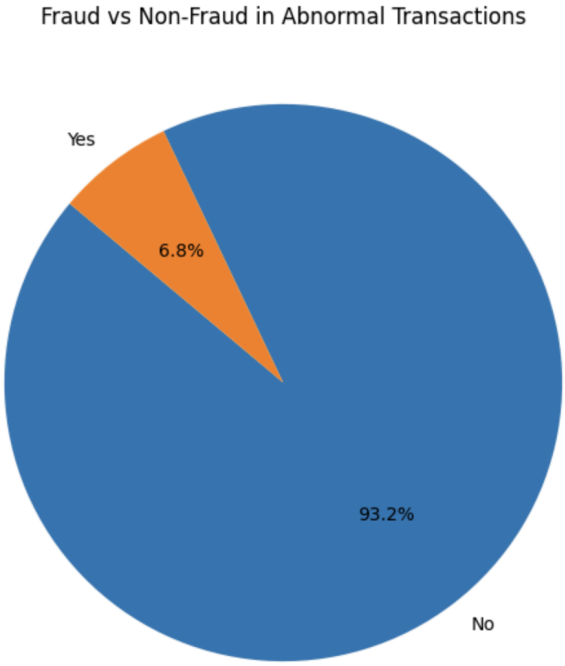
- **Spikes in Transaction Frequency per User (daily)**

  We identify days when a user's transaction count is unusually high compared to normal behavior. A spike day is flagged when the daily transactions amount exceeds the user's mean by over 10 standard deviations. We then analyze transactions on these spike days to assess the link between sudden activity surges and fraud.



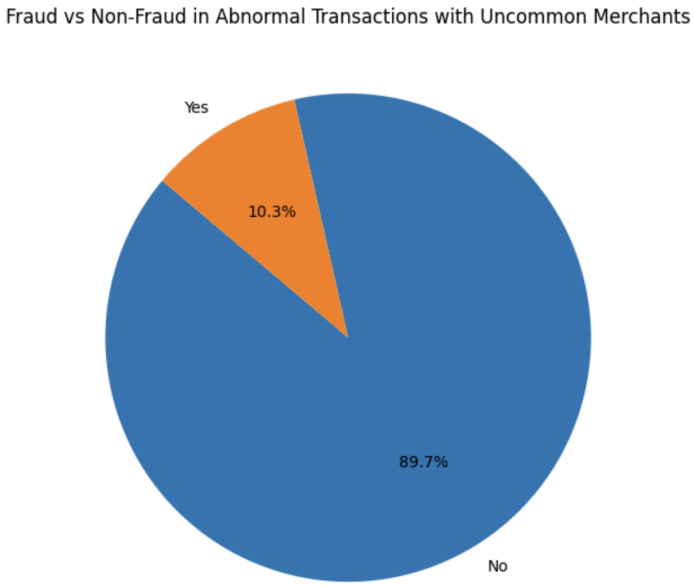
- **Unusual Transaction Amounts Per User**

  We detect transactions where the amount significantly exceeds a user's normal spending pattern. A transaction is flagged if its amount exceeds the user's average plus 15 standard deviations, helping highlight potential fraud linked to abnormal spending.



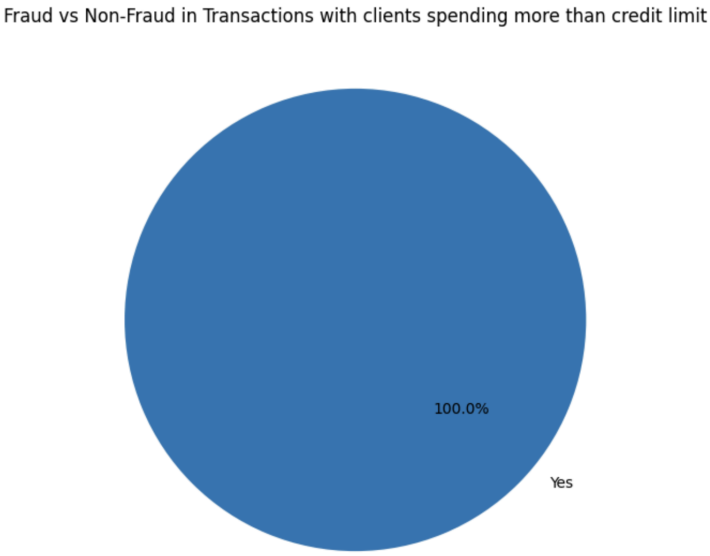
- **Uncommon Merchant Interaction Patterns**

  We also added a constraint to identify transactions with merchants that a user rarely interacts with. A merchant is considered "common" if a user has transacted with them more than twice.



- **Fraud Analysis for Overspending Clients**

  We identified users whose average transaction amount exceeds their average credit limit, and analyzed the fraud distribution within these transactions.

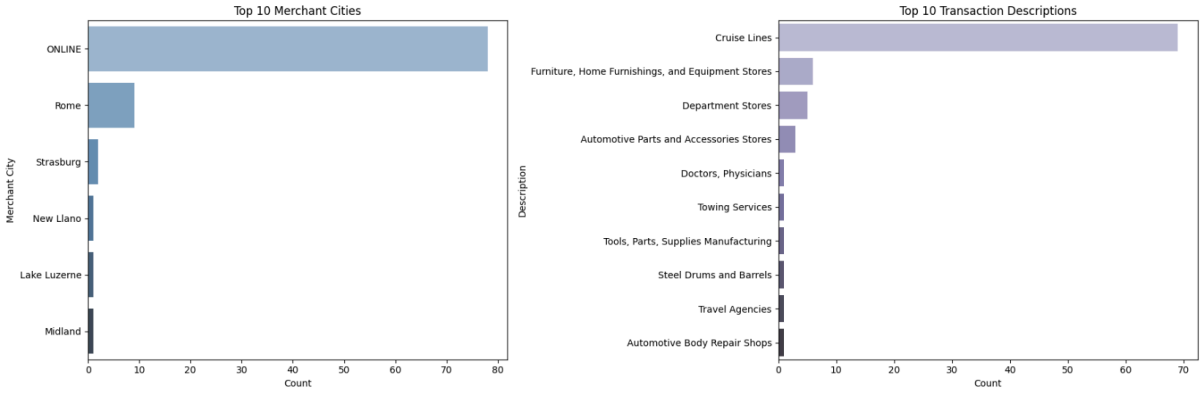


Observation:

We observed that the percentage of fraudulent transactions increases as more constraints are applied to filter specific abnormal behavior patterns, indicating that tighter behavioral filters help better isolate high-risk activities.

- **Fraud Distribution by Merchant City and Transaction Description**

  We analyzed fraudulent transactions from overspending users and identified the top 10 merchant cities and transaction descriptions associated with fraud. Bar charts visualize the most frequent locations and transaction types linked to fraudulent activities.



Does it make any sense for “Cruise Lines” to be the most frequent fraud description of unusual users' transactions?

Actually, yes, if someone steals a credit card, they might use it to buy non-essential things like travel, furniture, etc.

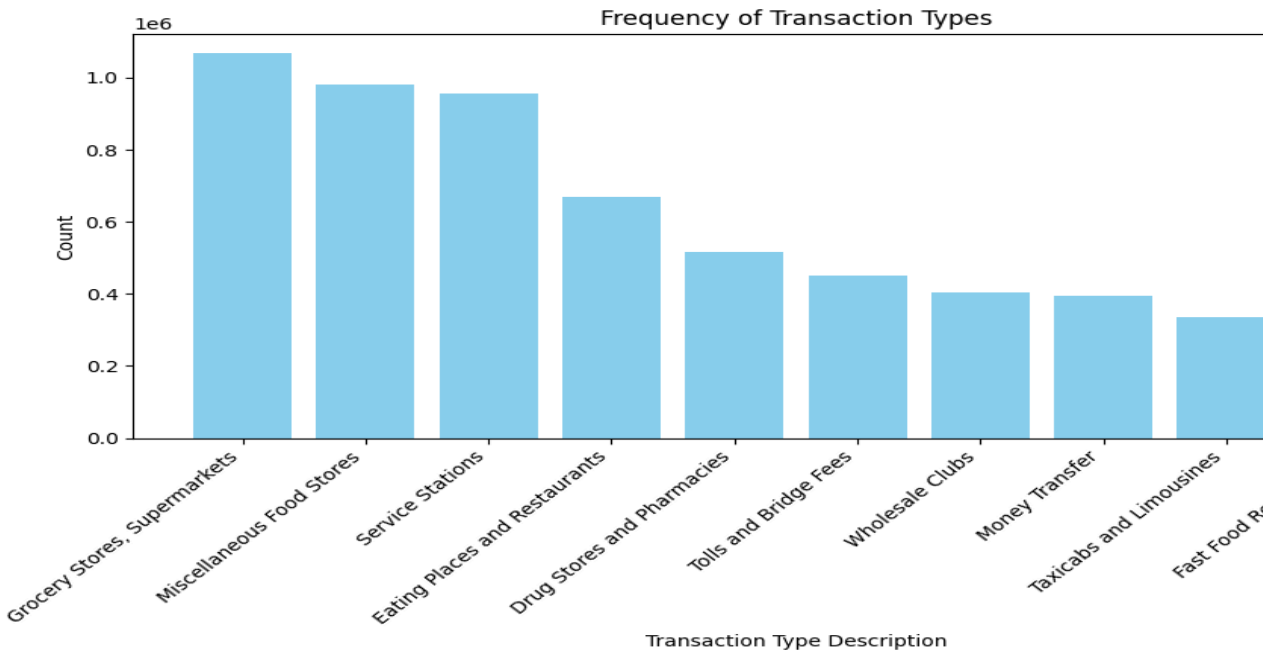
Cruise bookings may show up because:

- They’re hard to trace (online purchases, international companies)
- There's potential resale value (people reselling cruise tickets at discounts)

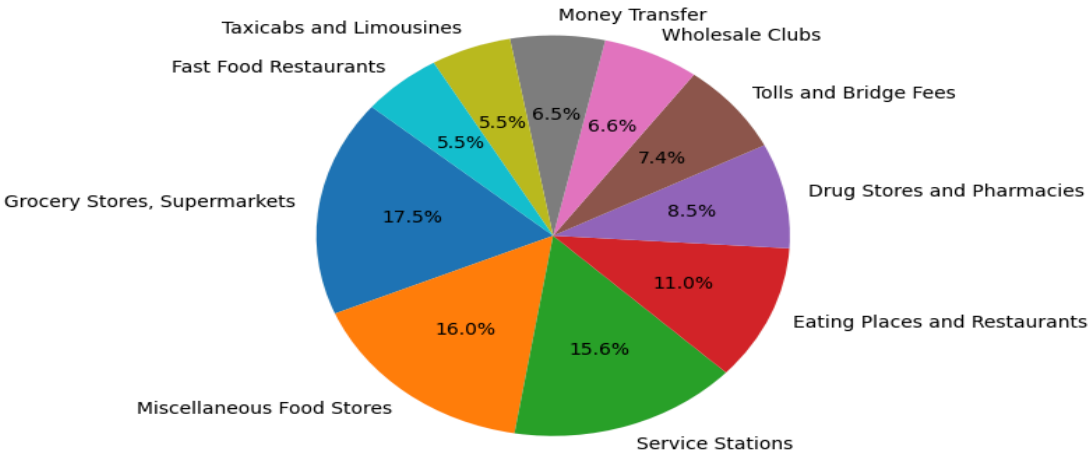
Transaction insights

- ## Analysis of Common Transaction Types

  This chart displays the frequency of transaction types on a logarithmic scale. Grocery stores and supermarkets lead with the highest frequency, followed by miscellaneous food stores, eating places/restaurants, and drug stores/pharmacies.



Top 10 Transaction Types (Percentage)

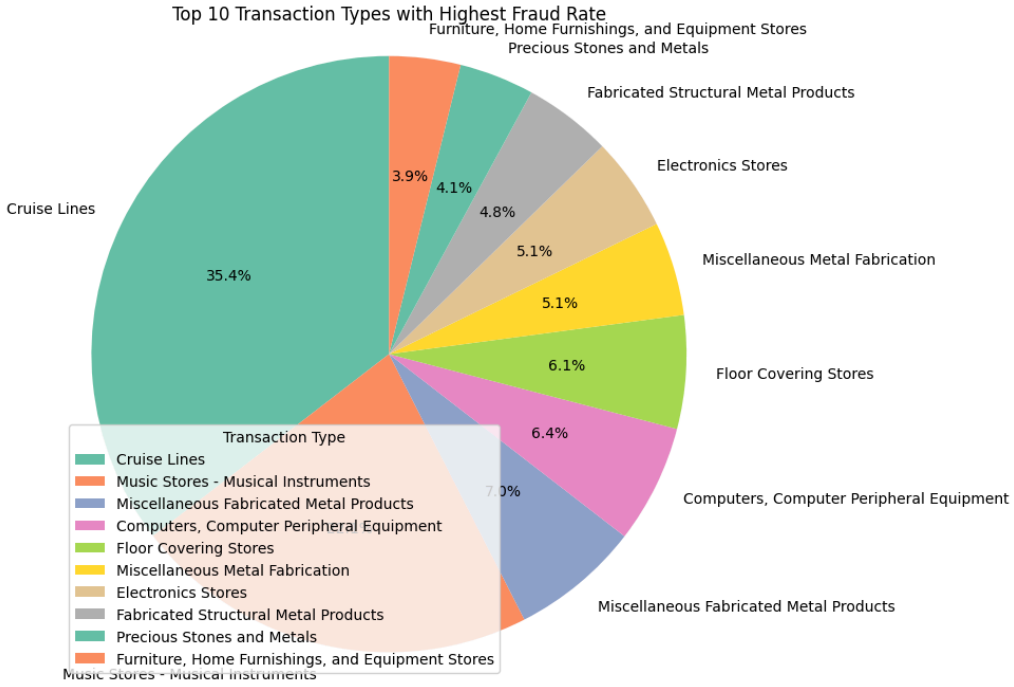


This table highlights fraud rates across those transaction types.

description	count	fraud_count	fraud_rate
Wholesale Clubs	403343	991	0.24569659074286648
Money Transfer	394401	725	0.18382306332894693
Drug Stores and P...	517400	479	0.09257827599536142
Taxicabs and Limo...	335586	300	0.08939586275947149
Fast Food Restaur...	334333	235	0.0702892026811592
Grocery Stores, S...	1066833	425	0.03983753783394402
Eating Places and...	669808	121	0.018064878293481117
Service Stations	955123	168	0.0175893576010629
Miscellaneous Foo...	979396	131	0.013375590670168145
Tolls and Bridge ...	451814	NULL	NULL

The pie chart illustrates the distribution of transaction types most prone to fraud.

- **Cruise Lines** lead with the highest fraud rate at 35.4%, despite being a smaller transaction category, indicating significant vulnerability in this sector.
- **Music Stores - Musical Instruments** follow at 37%, showing a notable fraud risk in this sector.

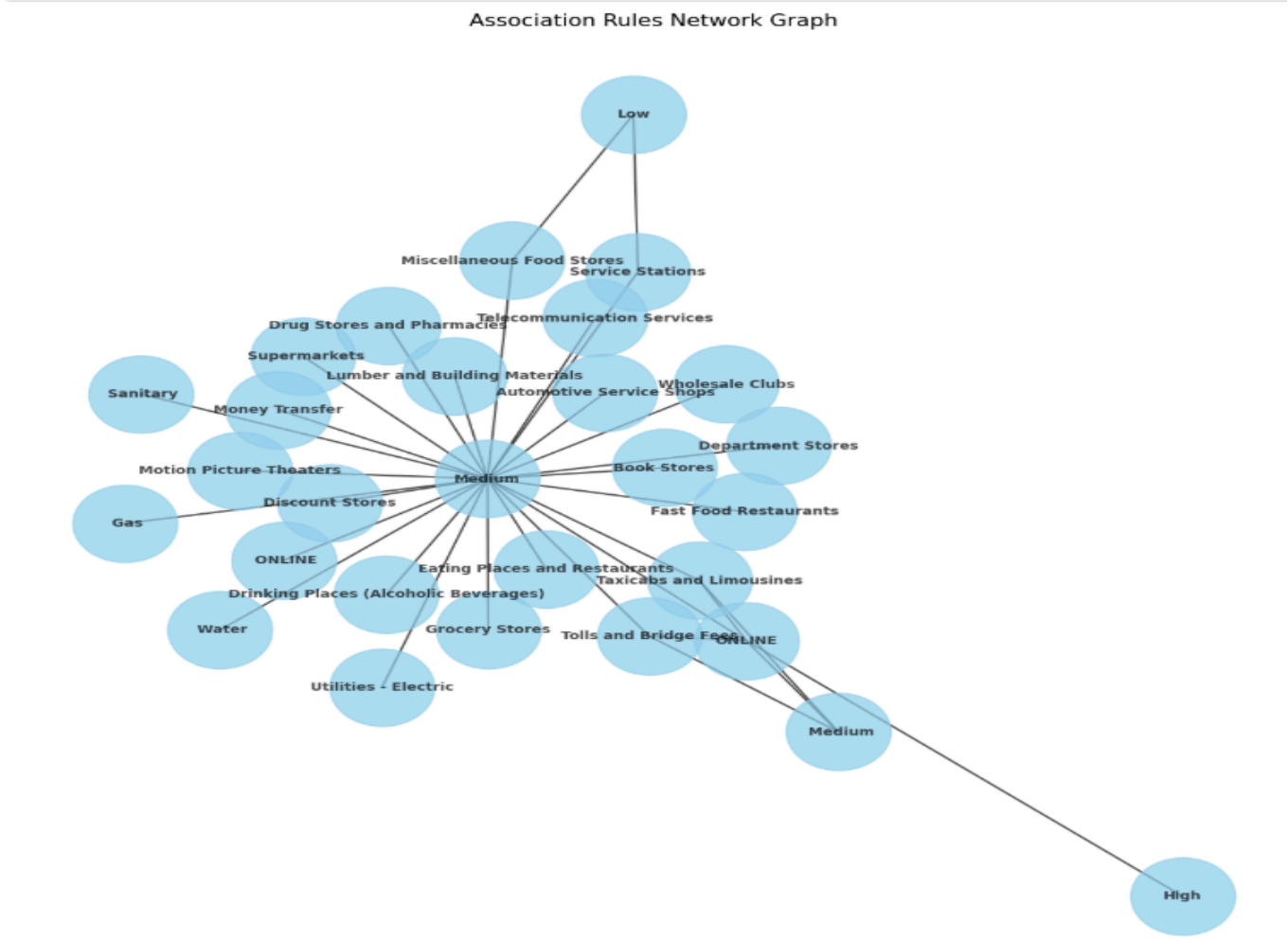


- Association of Transaction Features

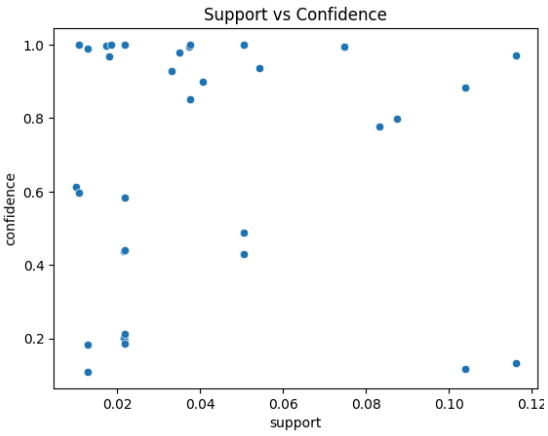
  1. We replaced the amount column with a flag to indicate transactions with high, low, or medium amounts.
  2. We got the association rules between merchant city, amount flag, and transaction

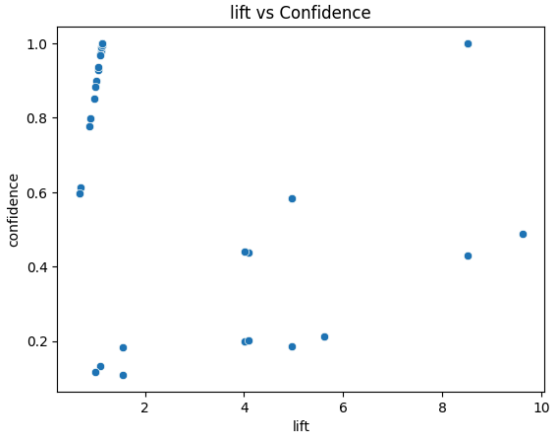
antecedent	consequent	confidence	lift	support
[Taxicabs and Limousines, ONLINE]	[Medium]	0.9999744673897502	1.1366653186581632	0.0219656548210015
[High]	[ONLINE]	0.1821651112615146	1.5498134080127555	0.012861298470896626
[Miscellaneous Food Stores]	[Medium]	0.797217877140605	0.9061930498339628	0.08758219187224894
[Miscellaneous Food Stores]	[Low]	0.1989808004116823	4.007379192564344	0.02185998977225144
[Telecommunication Services]	[Medium]	0.6138166352123848	0.6977218959225984	0.010069250988478584
[Wholesale Clubs]	[Medium]	0.8996015797968479	1.0225720252981456	0.040701010200491015
[Medium]	[Grocery Stores, Supermarkets]	0.131997600372822	1.103034611239523	0.011612409384088301
[Medium]	[ONLINE]	0.1181715656929816	1.0053729591168712	0.10396072311236737
[Eating Places and Restaurants]	[Medium]	0.9939728996966295	1.1298433706216735	0.07468017534116518
[Motion Picture Theaters]	[Medium]	0.9999381628173021	1.1366240514560935	0.01088316350836229
[Book Stores]	[Medium]	0.9982603895186863	1.1347169360413516	0.017443818891901178
[Automotive Service Shops]	[Medium]	0.9790925419757907	1.1129289522019155	0.035157969808736166
[Drinking Places (Alcoholic Beverages)]	[Medium]	0.9996275381473027	1.1362709661513961	0.018665024184620845
[Tolls and Bridge Fees, Medium]	[ONLINE]	1.0	8.507740023762603	0.05067659843344274
[Service Stations]	[Medium]	0.7769638046618079	0.8831703602563655	0.08324162422210839
[Service Stations]	[Low]	0.2030408648938409	4.089146970624386	0.021753203013854347
[Department Stores]	[Medium]	0.9270323473914559	1.053752423604066	0.033120720747803443
[Grocery Stores, Supermarkets]	[Medium]	0.9703880551126559	1.1030346112395233	0.011612409384088301
[Money Transfer]	[Medium]	0.8522518959130428	0.9687499075588419	0.037703914194596205
[ONLINE]	[High]	0.109420583758404	1.5498134080127555	0.012861298470896626

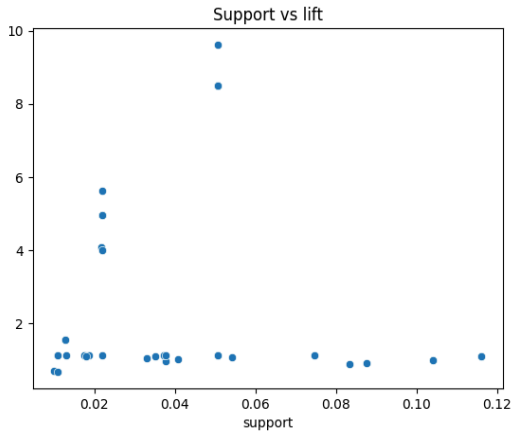
only showing top 20 rows



- **(Medium)**The "Medium" node is highly connected. This suggests "Medium transaction amount" is a common factor across various activities.
- **Strong Connections:** Categories like "Tolls and Bridge Fees" and "Online" have a direct, strong link (lift = 8.5 in the table), showing a high likelihood of co-occurrence.





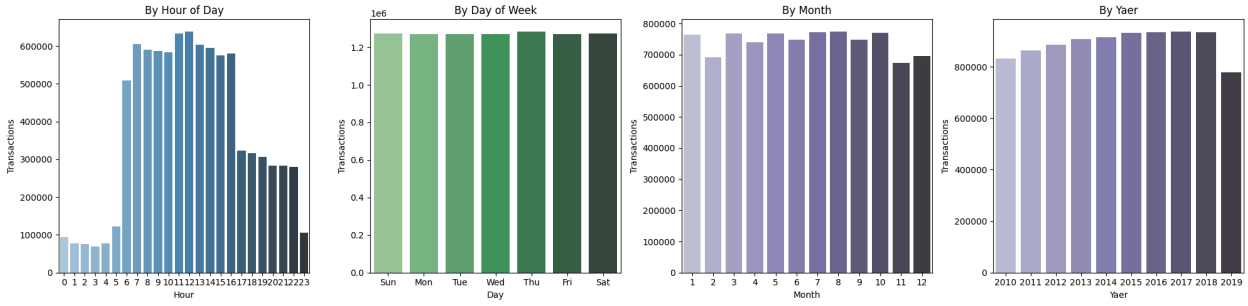


- Many rules are highly reliable (high confidence) but not very frequent (low support)
- Rules with high lift indicate strong associations, far more likely than if the items were independent. Lower lift values suggest weaker-than-expected associations despite decent confidence
- Rules with the highest lift aren't the most frequent but show a strong dependency. High support rules have a lift near 1. This suggests a trade-off: rare rules can have strong associations (high lift), while frequent rules are often less surprising (lift near 1).

Overall:

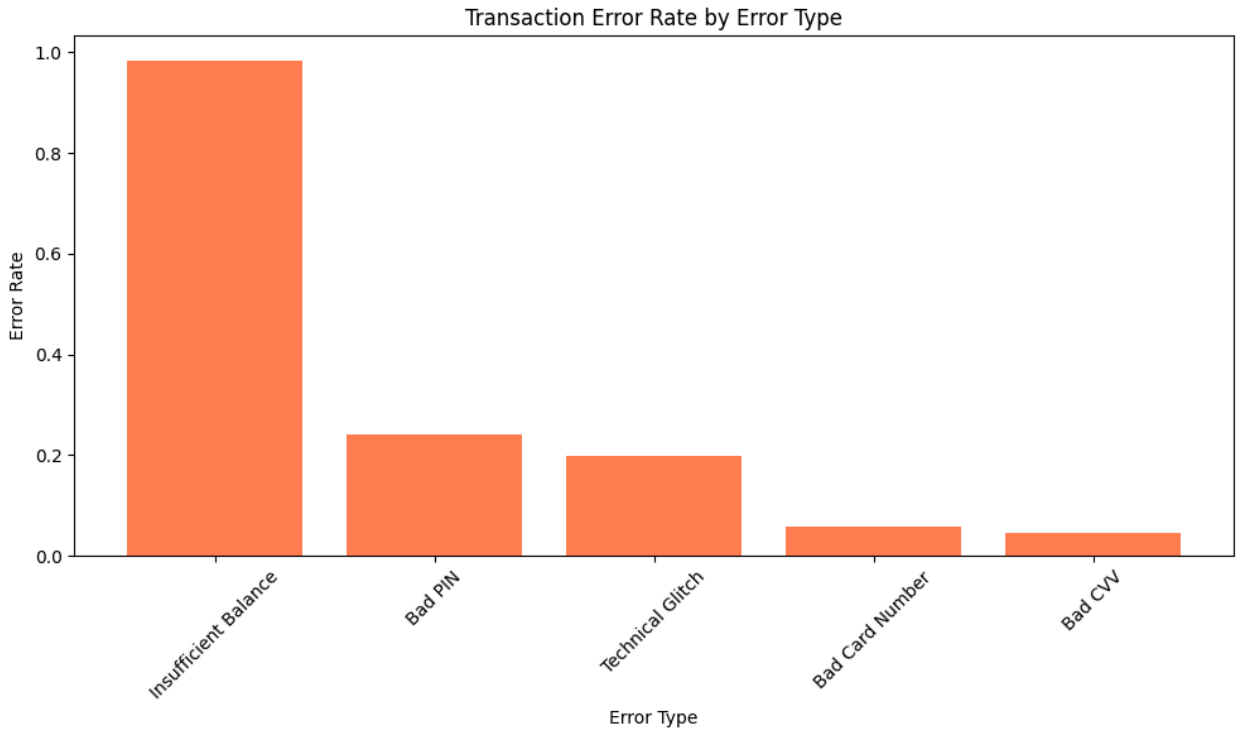
The graphs highlight many reliable but infrequent rules (high confidence, low support), some of which show strong associations (high lift). Meanwhile, frequent rules (high support) tend to have a lift near 1, reflecting common but less surprising patterns.

- Time-Based Transaction Behavior Analysis

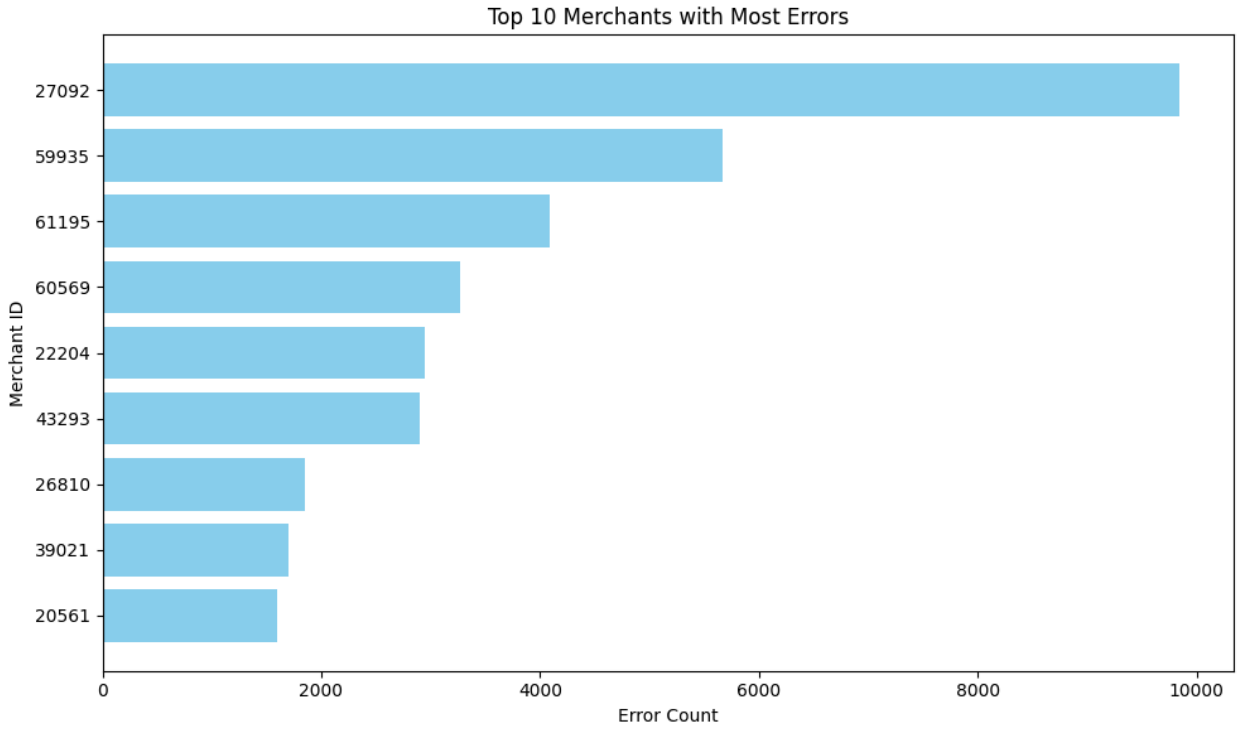


Transactions are most frequent during midday hours, slightly higher on weekends, stable across months with minor variations, and show a consistent upward trend over the years.

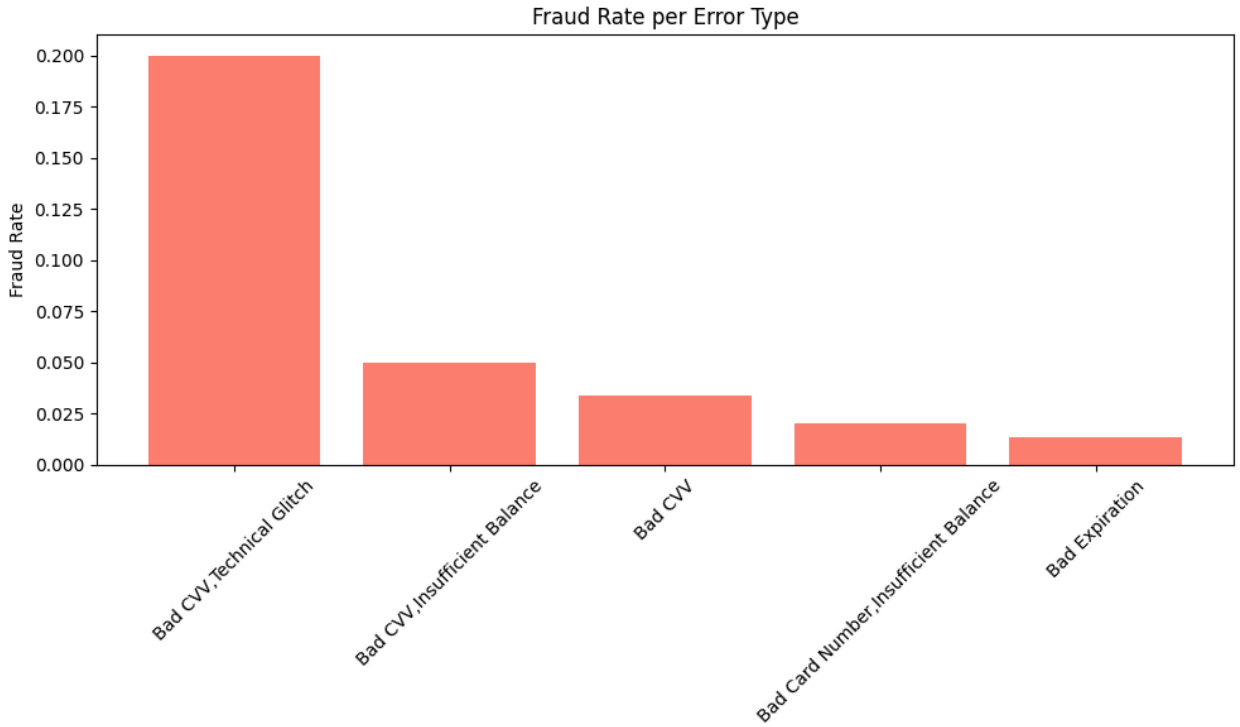
- Transaction Error Rates
    1. We got the error types that happen the most.



2. We got the merchants with the most errors



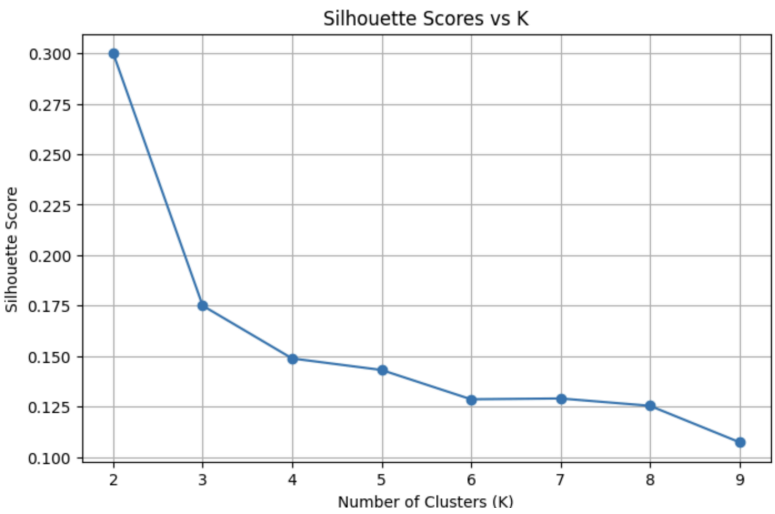
3. We calculated the fraud rate for each error type

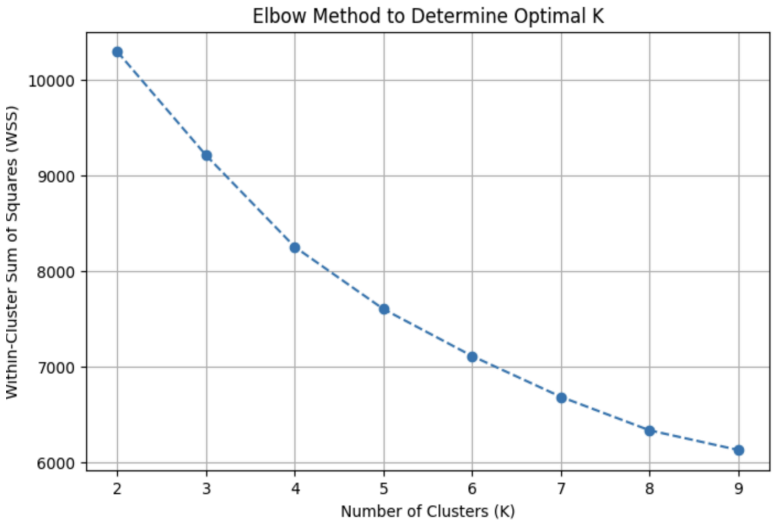


Models & Results

1. Customer Segmentation Based on Transaction Behavior

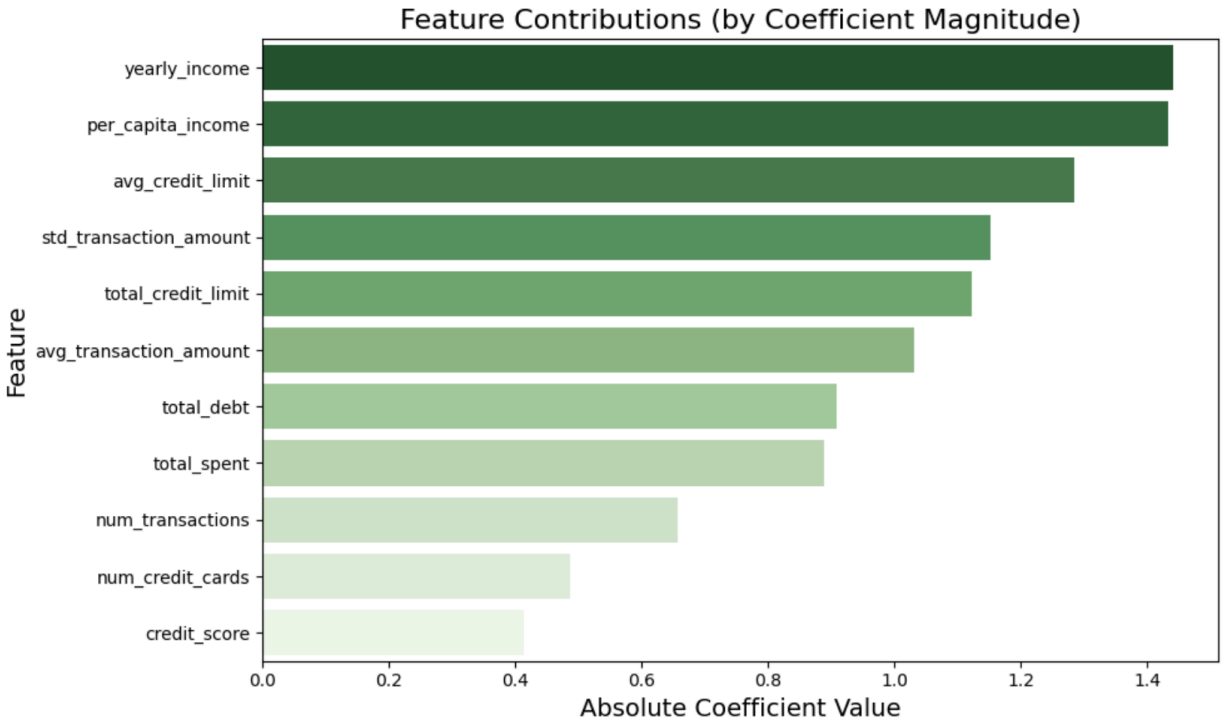
We tried to segment users' behavior by calculating total and average spending, transaction counts, spending variability, and credit card limits. Irrelevant or high-cardinality columns were dropped to focus on meaningful features.
Next, we assembled and scaled these features, then applied K-Means clustering to group users based on their transaction and credit patterns. We evaluated different cluster counts (k=2 to 9) using both training cost and silhouette score to determine the optimal number of user behavior groups.



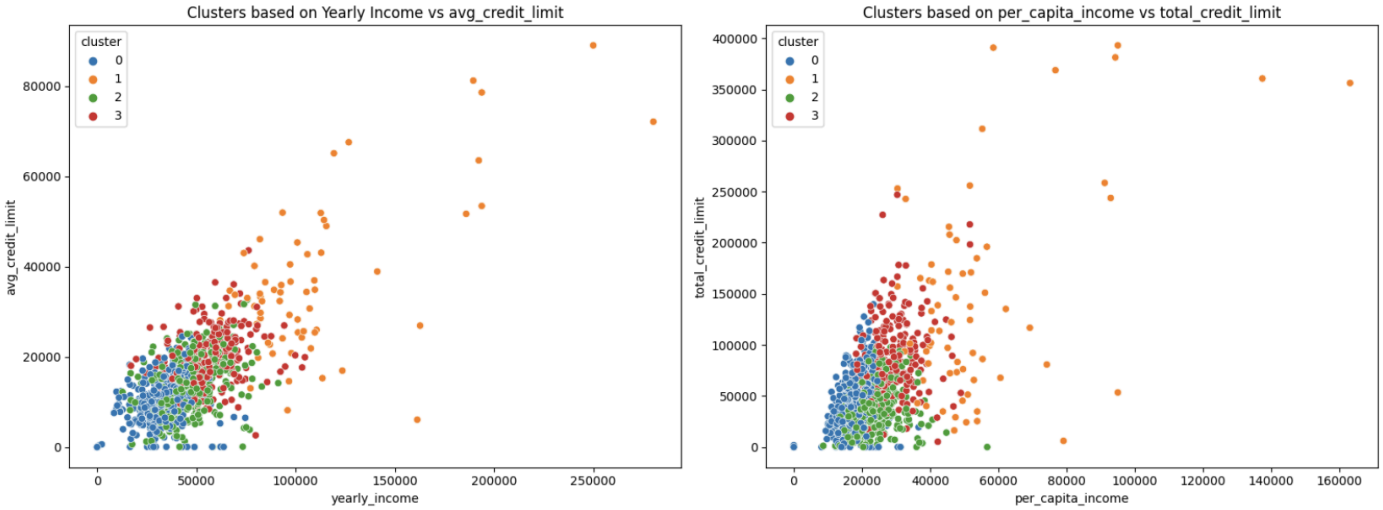


Observation:

k = 4 is the best choice

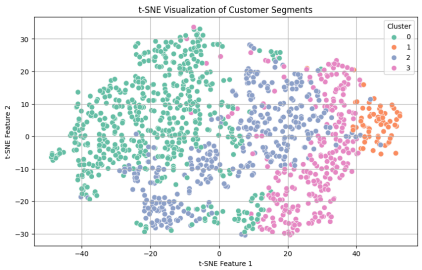
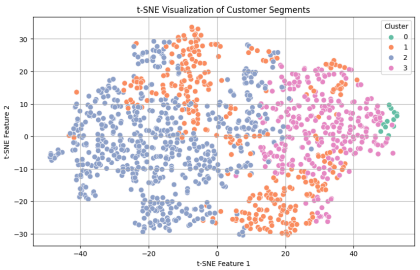


Cluster Visualization against some features (income,credit_limit)



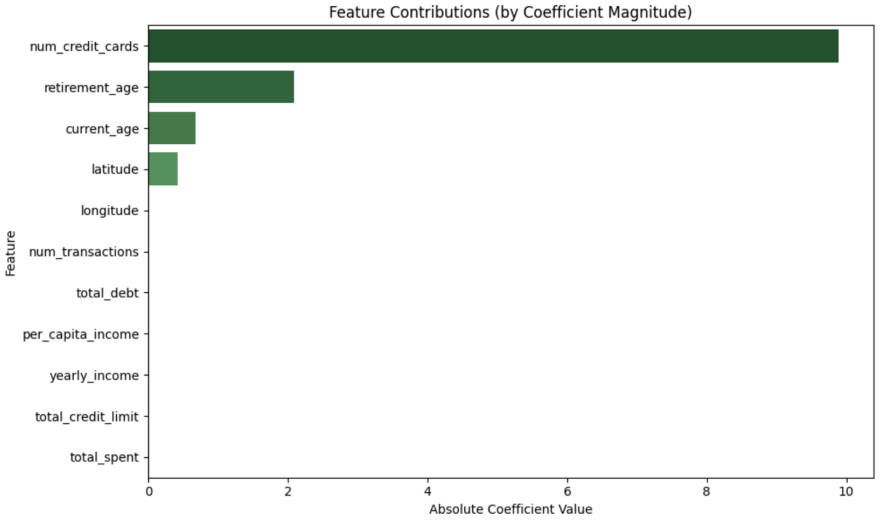
2. Customer Segmentation from Scratch using MapReduce

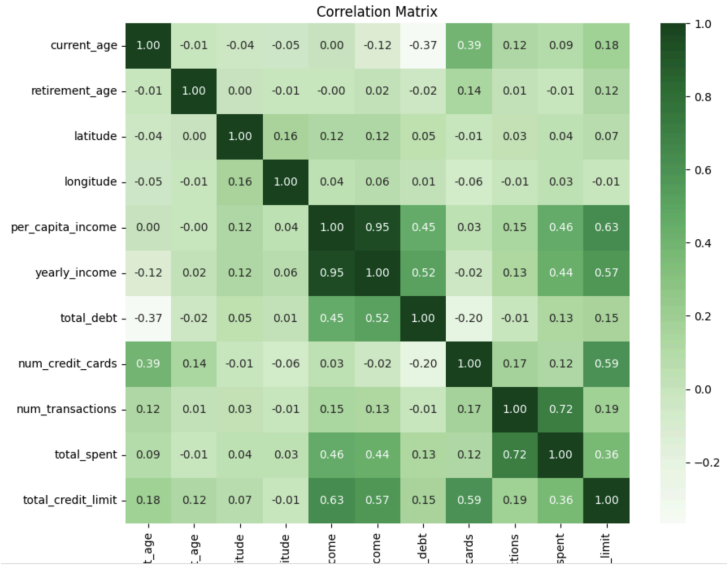
- Map: Processes input data in parallel and emits key-value pairs.
- Reduce: Aggregates the mapped data by key to produce the final output. In the context of K-means, MapReduce can be used to parallelize the assignment of points to clusters (Map) and the computation of new centroids (Reduce).

POC	Sklearn	Pyspark Map-Reduce From Scratch
Silhouette Score	0.24763889158278077	0.16460602446780384
Training Time	76.82719612121582	28.99254155158
results		

3. Predicting Customer Credit Scores

We used 90% of the 1219 users (approximately 1097 users) for training and 10% (approximately 122 users) for testing. The model applied is **Lasso Linear Regression (L1)**, which helps in feature selection and reduces the risk of overfitting by penalizing the coefficients of less important features.

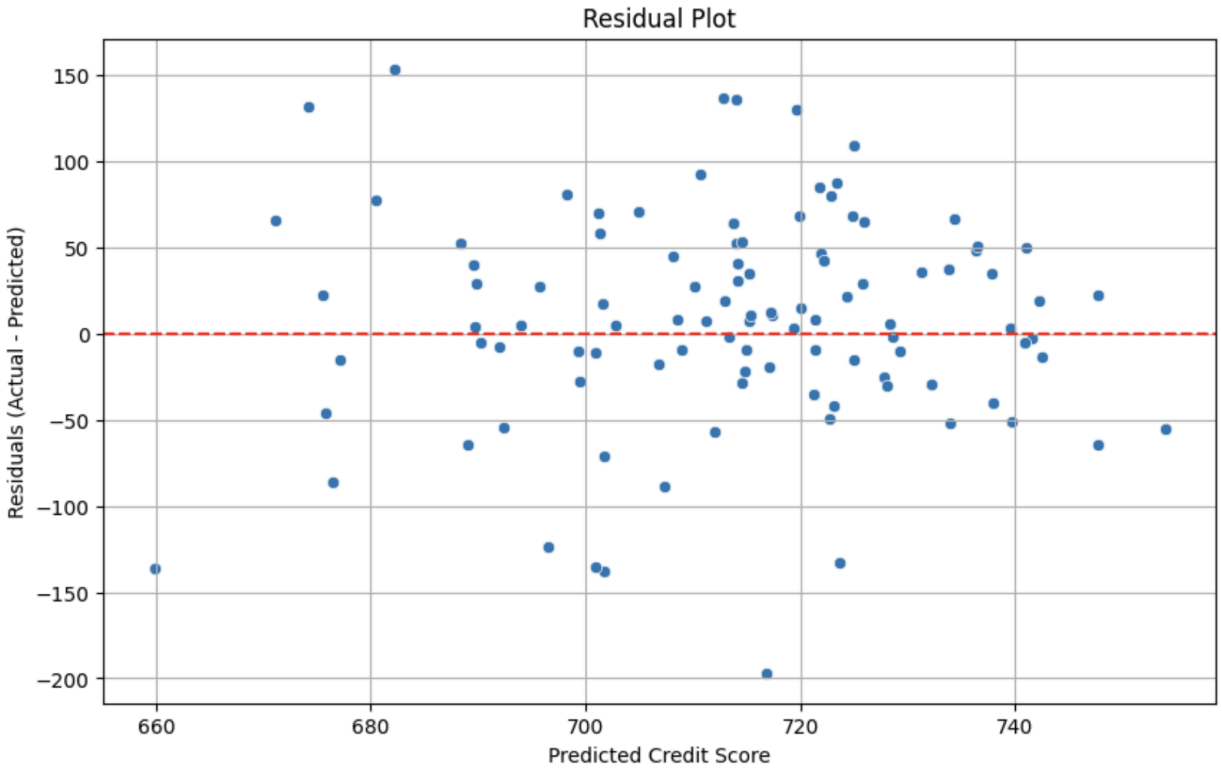




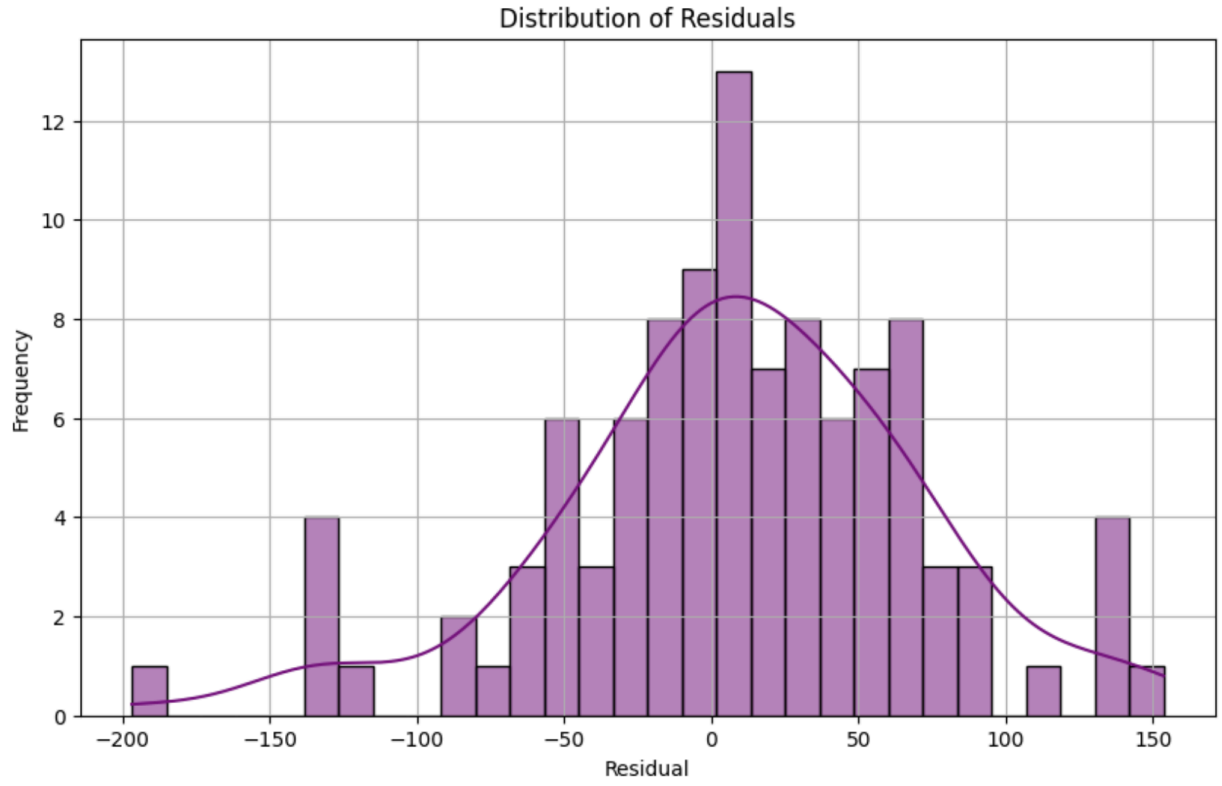
The most important features in predicting credit scores are:

- num_credit_cards: A higher number of credit cards reflects more financial activity and credit management.
- retirement_age: Users nearing retirement are typically more financially stable.
- current_age: Age impacts credit history length and financial stability. Younger users may have shorter credit histories, while older users may have fewer credit needs, affecting their credit risk.
- Latitude: Geographical location influences economic factors such as job stability and income levels.

Residuals Scatter Plot for testing points



Distribution of Residuals for testing points



4. KNN with MapReduce Implementation

- **Map Phase**: Each partition computes cosine similarity between a test point and training points, maintaining a heap of the (k=3) nearest neighbors (similarity, label pairs).
- **Reduce Phase**: Merges per-partition neighbor lists to find the global (k) nearest neighbors using heapq.nlargest.
- **Classification**: Predicts the label via majority voting on the global neighbors' labels.
- **Execution**: Training data is distributed across 6 partitions, enabling parallel similarity computations. Applied to 3,000 test points, it achieves 99.73% accuracy but a macro F1 of 49.93% due to class imbalance.

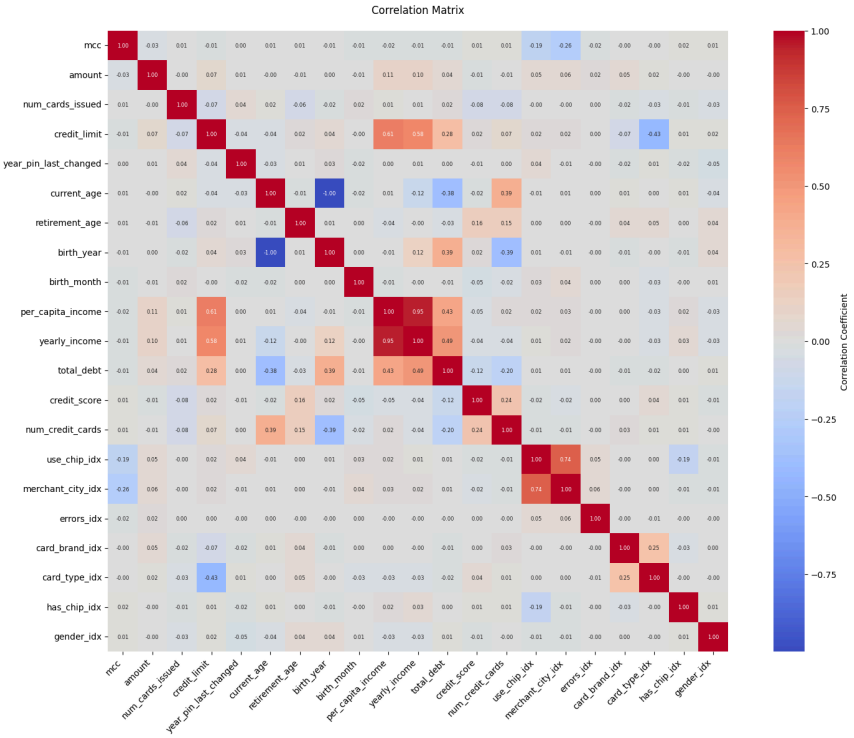
5.predictive model for fraud detection

Data Preprocessing:

- Dropped Columns: Removed irrelevant or sensitive columns (e.g., id, card_number, address, latitude, longitude).
- Handled Merchant City: Simplified merchant_city to "online" or "other" for consistency.
- Missing Values: Filled missing categorical values with "Unknown" and numerical values with 0.
- Label Encoding: Converted the target column (target) to numerical labels (label) using StringIndexer (No=0, Yes=1).

Feature Engineering:

- Created a feature vector using VectorAssembler with all columns except target and label as input features.
- Split the data into training (80%) and test (20%) sets with a random seed for reproducibility.
- Computed a Pearson correlation matrix for the feature vector using Correlation.corr.(it is appereant that features are mostly uncorreleated)



We trained using different models but the best was XGBoost

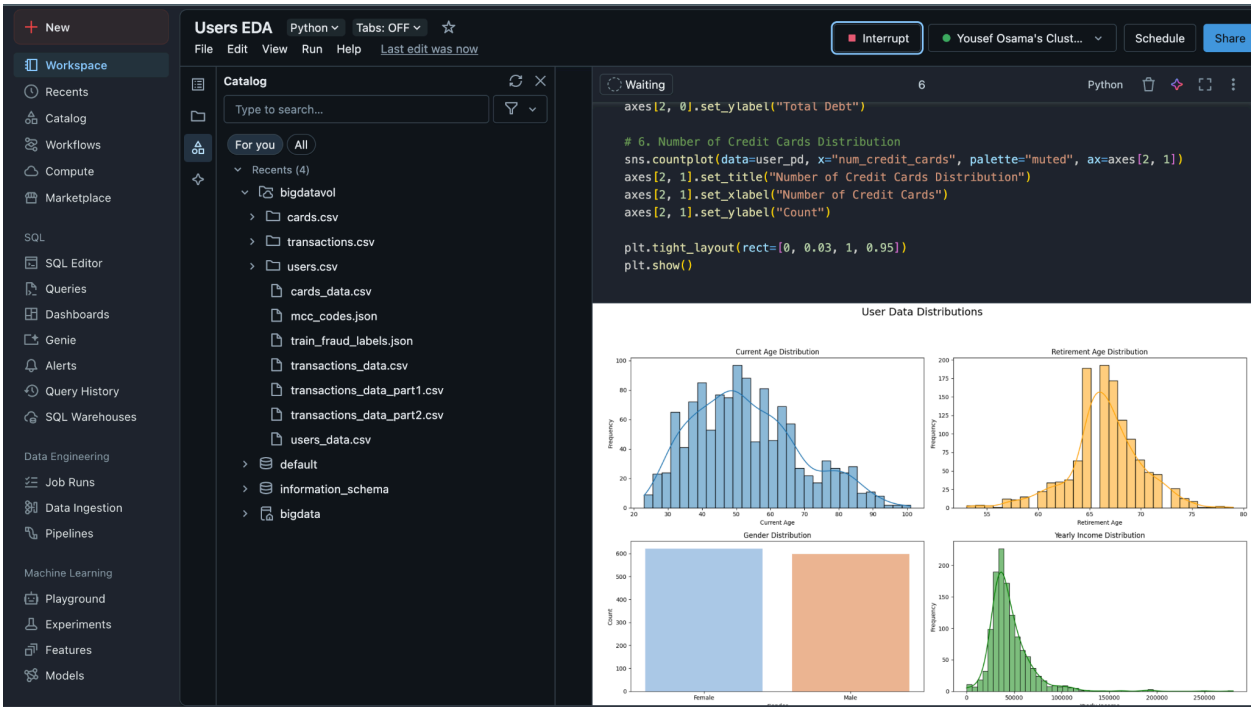
Results:

Metric	Logistic regression	Random Forest	XgBoost	SVM
Accuracy	0.841	0.842	0.952	0.821
Weighted Recall	0.9984	0.9985	0.9987	0.9984
Weighted Precision	0.9973	0.9985	0.9985	0.9969
F1-score	0.99769	0.9979	0.9986	0.9984

Extra work (bouns)

1 - working with cloud platform (azure databricks).

2 - working in fully distributed mode (more than 1 machine).



Unsuccessful trials

Direct Use of XGBoost Without Class Weights
Our first trials using `SparkXGBClassifier` without adjusting for class imbalance resulted in models biased toward predicting the majority class ("non-fraud")

Using Manhattan Distance Instead of Cosine Similarity
We experimented with Manhattan distance (L1 norm) for KNN instead of cosine similarity. However, it led to lower accuracy

High k-values in KNN
We tried setting `k = 10, 15, 20` instead of 3 to see if considering more neighbors would smooth out noisy predictions, but it took too long.

Future Work

Based on our findings and challenges, we propose the following future improvements and extensions:

- **Advanced Fraud Detection Techniques**
  Apply more robust imbalance handling methods like:
    - SMOTE (Synthetic Minority Over-sampling Technique) or its variants like SMOTE-ENN
- **Explainable AI for Fraud Detection**
- **Real-Time Fraud Detection Pipeline**