1. Introduction

In today's digital age, online networks have become central to nearly every aspect of life, from communication and social engagement to commerce and financial transactions. With the explosion of platforms like social media, e-commerce websites, and online service providers, the scope of online interactions has expanded dramatically. However, with this growth comes an increasing risk of fraudulent behavior. Cybercriminals are constantly finding new ways to exploit the vulnerabilities of online platforms for malicious purposes such as identity theft, creating fake accounts, spreading misinformation, conducting phishing scams, and committing financial fraud.

These fraudulent activities can have far-reaching consequences, compromising user security, eroding trust in online platforms, and causing significant financial and reputational damage to both users and service providers. For instance, fake profiles and bots can manipulate public opinion on social media, while fraudulent transactions can lead to billions of dollars in financial losses on e-commerce platforms.

Traditional fraud detection methods, such as rules-based systems or basic transaction monitoring, are often inadequate in the face of these growing and evolving threats. These approaches tend to rely on predefined sets of rules or manually established patterns of fraud, which can easily be bypassed by sophisticated attackers who adapt their tactics to avoid detection. Moreover, as online networks continue to expand in size and complexity, fraudsters increasingly leverage the interconnected nature of users, forming fake networks or manipulating their interactions to blend in with legitimate activity.

Given these challenges, more advanced methods are required to effectively detect and prevent fraud in real time. Machine learning, particularly unsupervised learning techniques, provides an opportunity to tackle this problem in a more dynamic and adaptive manner. Unlike traditional approaches, machine learning can uncover hidden patterns and anomalies in network behavior that may indicate fraudulent activities, even when no labeled data is available.

Our project focuses on the application of unsupervised learning techniques, such as K-Means clustering, in combination with network analysis to detect fraudulent behavior in online networks. By analyzing network properties like user connections, interaction frequency, and centrality, we can identify suspicious patterns that deviate from normal behavior, enabling the early detection of potential fraud.

Additionally, we have also classified the suspicious behaviour using degree centrality in a directed graph, where we have specifically identified suspicious behaviour on the basis of high threshold criteria of in-degree and out-degree centrality.

2. Problem Statement

Detecting fraudulent users within large and complex online networks is a significant challenge due to the dynamic nature of user behavior and the increasing sophistication of fraud tactics. Fraudulent users often exhibit complex interaction patterns, blending in with normal activity to evade detection. They may create fake accounts, establish large networks

of connections, or exploit system vulnerabilities to avoid being flagged by conventional security measures. As fraudsters become more advanced, traditional methods such as rules-based systems or simple monitoring of financial transactions are insufficient for identifying these subtle yet malicious behaviors.

The goal of our project is to leverage machine learning, specifically unsupervised learning methods like K-Means clustering, combined with network analysis using degree centrality, to detect fraudulent behavior in an online network. This approach focuses on identifying anomalous user behavior based on the structure of their interactions and the properties of their network connections. Unlike traditional fraud detection techniques that require pre-labeled data or defined fraud patterns, this project aims to find suspicious users without prior knowledge of what constitutes fraud.

3. Objectives

- **Detection of Suspicious Activities:**
  To utilize network properties in order to identify potential fraudulent users based on their activities within the online network, analyzing interactions and behavioral anomalies.
- **Anomaly Detection:**
  To apply various anomaly detection methods, focusing on K-Means clustering, to classify users as either fraudulent or legitimate by comparing user behavior against established norms.
- **Graph-Based Features:**
  To implement graph-based features to enhance classification accuracy, constructing a user interaction graph that analyzes connectivity and centrality measures to differentiate between legitimate and fraudulent activities.
- **Visualization and Reporting:**
  To provide insightful visualizations of clustering results and user classifications, generating detailed reports to aid stakeholders in making informed decisions and implementing effective countermeasures.
- **Scalability and Future Implementation:**
  To assess the scalability of the methods used and explore their potential for real-time application in larger networks, ensuring adaptability to evolving user behaviors.

4. Approach

4.1 Anomaly Detection using K-Means Clustering

- Core Technique:Anomaly detection serves as a vital technique for identifying fraudulent users within the network. By leveraging clustering algorithms, we

effectively analyzed user behaviors and interactions to uncover patterns indicative of fraud.

- Understanding K-Means Clustering:
    - K-Means is an unsupervised machine learning algorithm commonly used for partitioning data into distinct groups (clusters) based on similarity. The algorithm works by:
        i. Selecting the Number of Clusters (k): The user specifies the number of clusters desired, which represents the different groups of similar data points.
        ii. Initializing Centroids: K-Means begins by randomly initializing 'k' centroids, which act as the central points of each cluster.
        iii. Assigning Data Points: Each data point is assigned to the nearest centroid based on the Euclidean distance, forming clusters.
        iv. Updating Centroids: The centroids are then recalculated by taking the mean of all data points within each cluster.
        v. Iterating: Steps 3 and 4 are repeated until the centroids no longer change significantly, indicating convergence.

- Implementation in the Project:
    - In our project, K-Means clustering is implemented on the dataset to group users based on their network activity and graph-based features. The goal is to identify unusual patterns in user behavior that may signal fraudulent activity.
    - Data Preparation: Before applying K-Means, the dataset was preprocessed by removing unnecessary columns and standardizing the feature set to ensure that all features contribute equally to the distance calculations.
    - Clustering Process:
        - The optimal number of clusters is determined using the Elbow Method, which involves plotting the Within-Cluster Sum of Squares (WCSS) against the number of clusters to find the point where adding more clusters yields diminishing returns.
        - Once the optimal number of clusters was established, K-Means was applied to the standardized dataset, resulting in the partitioning of users into distinct clusters.

- Identifying Outliers:
    - By partitioning users into clusters, K-Means allows for the identification of outliers—users whose behavior deviates significantly from the majority in

their cluster. These outliers are flagged for further investigation as potential fraudulent users.

- The clustering process not only helps in detecting these anomalies but also facilitates the understanding of user behavior patterns, which is critical for distinguishing legitimate users from those exhibiting suspicious activities.

- Post-Clustering Classification: After clustering, we focused on categorizing users into two distinct groups:
  - Legitimate Users: These users display activity and network properties consistent with typical patterns observed in the general user population. Their behavior aligns with established norms, indicating they are likely not engaging in fraudulent activities.
  - Fraudulent Users: In contrast, these users are characterized by their unusual behavior, atypical network connections, and distinct graph-based metrics. Their classification as outliers suggests potential fraudulent activity, prompting further analysis.

4.2 Classification using Degree Centrality

- Graph Representation:
  - Represent the social network as a directed graph (using NetworkX). Each user is a node, and directed edges represent messages sent and received.
- Centrality Metrics:
  - In-Degree: Number of messages received (incoming edges).
  - Out-Degree: Number of messages sent (outgoing edges).
  - High in-degree or out-degree centrality indicates abnormal behavior, such as potential fraud or manipulation.
- Threshold Calculation:
  - Calculate the 90th percentile for both in-degree and out-degree centrality to identify users with extreme behaviors.
- Fraudulent User Identification:
  - Flag users with in-degree or out-degree centrality above the 90th percentile as suspicious.
- Visualization:
  - Visualize the network with suspicious users colored red and high centrality users colored orange.

5. Dataset

We created two synthetic datasets to simulate user behavior in an online network. This dataset aims to mimic real-world interactions while incorporating various patterns of legitimate and fraudulent behaviors. By generating a synthetic dataset, we explored user dynamics, tested algorithms for detecting fraud, and validated our approach without the ethical concerns associated with real user data.

Dataset1 Link -> https://drive.google.com/file/d/1WGMIHdqAEfjI--Ky-hKgfrncg1FTCydw/view?usp=sharing

- User ID: Unique identifier for each user.
- Friends Count: Number of friends the user has on the social network.
- Groups Joined: Number of groups the user is part of.
- Posts Per Week: Average number of posts made by the user each week.
- Likes Received: Total number of likes received on the user's posts.
- Messages Sent: Total number of private messages sent by the user.
- Login Frequency: Number of logins made by the user per week.
- Time Between Interactions (hours): Average time (in hours) between two interactions.
-

Dataset2 Link -> https://drive.google.com/file/d/12NpDOtJZ9mnDi9FAsghBfsa4tRXNJXMf/view?usp=drive_link

- User ID: Unique identifier for each user.
- Friends Count: List of friends.
- Messages Sent: Total number of private messages sent by the user.

The datasets consists of several key features that help characterize user behavior within the network. Each feature plays a critical role in analyzing and identifying fraudulent users.

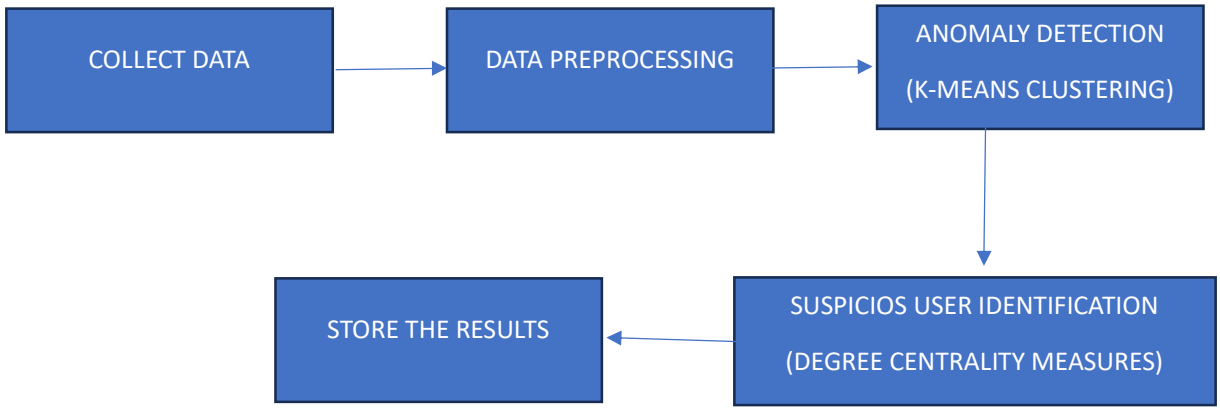
Dataset Characteristics

- User Diversity: The dataset includes a diverse range of users, encompassing both legitimate and fraudulent behaviors. This diversity is crucial for training and testing algorithms intended to detect fraudulent activities.
- Fraudulent User Patterns: Fraudulent users tend to exhibit specific behavioral patterns, such as:
    - Fewer Connections: They may have a lower friends count but engage in high-volume messaging, especially to users who are not friends.
    - Unusual Activity: They may demonstrate suspicious patterns, like a high number of messages sent relative to their number of friends, or engage in spam-like behaviors, such as posting multiple times in rapid succession.

This dataset serves as a foundation for applying network analysis techniques to identify potential fraudulent users, allowing you to explore patterns and test various anomaly detection methods effectively.

6. System Design

Conceptual Model



7. Implementation

DATASET 1

7.1. Setup and Data Loading

from google.colab import files

import pandas as pd

This section sets up the environment, specifying the encoding format and importing necessary libraries.

- google.colab.files: This library is used to handle file uploads and downloads in Google Colab.
- pandas: A powerful library for data manipulation and analysis, particularly for working with structured data.

7.2. Load Data

df = pd.read_csv('data7.csv')

df.head()

- Loading the dataset: The dataset named data7.csv is loaded into a Pandas DataFrame (df).
- df.head(): This function displays the first five rows of the DataFrame, allows to inspect the structure and contents of the dataset.

7.3. Data Summary

df.describe()

- Summary statistics: describe() provides a summary of the numerical features in the dataset, including count, mean, standard deviation, min, max, and quartiles. This helps identify data distribution and potential anomalies.

7.4. Data Preprocessing

from sklearn.preprocessing import StandardScaler

df_clean = df.drop(columns=['User ID'], errors='ignore')

- StandardScaler: Imported to standardize features by removing the mean and scaling to unit variance.
- Data Cleaning: The column User ID is dropped from the DataFrame as it is not needed for analysis. The errors='ignore' argument ensures that no error is raised if the column is not present.

7.5. Data Standardization

scaler = StandardScaler()

X_scaled = scaler.fit_transform(df_clean)

- Initialization and Transformation: A StandardScaler object is created, and the data (df_clean) is standardized using fit_transform(), resulting in a new array (X_scaled) with scaled values (mean=0, variance=1).

7.6. Data Visualization

import seaborn as sns

```
import matplotlib.pyplot as plt

sns.pairplot(df_clean)

plt.show()
```

- Visualization Libraries: Imports seaborn and matplotlib.pyplot for data visualization.
- Pair Plot: Creates a pair plot of the features in df_clean, showing pairwise relationships and distributions. This visualization helps to identify patterns or clusters visually.

7.7. K-Means Clustering

```
from sklearn.cluster import KMeans

wcss = []

for i in range(1, 11):

    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=42)

    kmeans.fit(X_scaled)

    wcss.append(kmeans.inertia_)
```

- KMeans Import: Imports the KMeans clustering algorithm from sklearn.
- WCSS List: Initializes an empty list to store the Within-Cluster Sum of Squares (WCSS) for each number of clusters.
- Loop through cluster counts: For i from 1 to 10, the KMeans model is initialized and fitted to the scaled data. The WCSS (inertia) is calculated and appended to the wcss list. WCSS measures the compactness of the clusters; lower values indicate better clustering.

7.8. Elbow Method Visualization

```
plt.figure(figsize=(8, 5))

plt.plot(range(1, 11), wcss, marker='o')

plt.title('Elbow Method to Find Optimal Number of Clusters')

plt.xlabel('Number of Clusters')

plt.ylabel('WCSS')

plt.show()
```

- Plotting the Elbow Graph: This block creates a plot of the number of clusters (1 to 10) against the corresponding WCSS values.
- Purpose: The Elbow Method helps determine the optimal number of clusters by identifying the "elbow" point where WCSS begins to decrease at a slower rate.

7.9. Apply K-Means Clustering

kmeans = KMeans(n_clusters=2, init='k-means++', max_iter=300, n_init=10, random_state=42)

df['Cluster'] = kmeans.fit_predict(X_scaled)

- Clustering Application: K-Means is applied with n_clusters=2 (based on the Elbow Method results).
- Fitting and Predicting: The model is fitted to the scaled data, and predictions (cluster labels) are stored in a new column, Cluster, in the original DataFrame.

7.10. Check Clustering Result

df.head()

- Review Clustering Results: Displays the first five rows of the DataFrame, now including the Cluster column, to check the clustering assignments.

7.11. Cluster Centers Calculation

cluster_centers = pd.DataFrame(scaler.inverse_transform(kmeans.cluster_centers_), columns=df_clean.columns)

print("Cluster Centers:\n", cluster_centers)

- Cluster Centers: The cluster centers are calculated and inverse transformed back to the original scale.
- Display: The centers are stored in a new DataFrame (cluster_centers) and printed. This shows the average feature values for each cluster, allowing for interpretation of the clusters.

7.12. Identify Suspicious Patterns

suspicious_cluster = cluster_centers[(cluster_centers['Friends Count'] < 100) & (cluster_centers['Messages Sent'] > 500)]

print("Suspicious Cluster Centers:\n", suspicious_cluster)

- Suspicious Clusters Definition: Criteria are defined to identify suspicious patterns. Here, clusters with fewer than 100 friends and more than 500 messages sent are considered as suspicious.
- Print Results: The suspicious cluster centers are printed for review.

7.13. Filter Suspicious Users

suspicious_cluster_index = suspicious_cluster.index

potential_fraud_users = df[df['Cluster'].isin(suspicious_cluster_index)]

print("Potential Fraudulent Users:\n", potential_fraud_users)

- Suspicious Cluster Index: The index of the suspicious clusters is obtained.
- User Filtering: Users belonging to suspicious clusters are filtered from the original DataFrame and stored in potential_fraud_users.
- Output: The potential fraudulent users are printed for further examination.

7.14. Visualize Clusters

sns.pairplot(df, hue='Cluster', palette=['green', 'red'], diag_kind='kde')

plt.show()

- Cluster Visualization: A pair plot is created, coloring points by their cluster assignment. This visually represents how different clusters are distributed in the feature space.

7.15. Save Results

df.to_csv('clusters.csv', index=False)

from google.colab import files

files.download('clusters.csv')

- Save to CSV: The DataFrame, now containing cluster labels, is saved to a CSV file named clusters.csv.
- File Download: The saved CSV file is made available for download to your local machine.

DATASET 2

7.16. Directed Graph

```
G = nx.DiGraph()

suspicious_nodes = []

for i, row in df.iterrows():
    user_id = row['User ID']
    friends = row['Friends']
    if isinstance(friends, (int, np.integer)):
        friends = [friends]
    elif isinstance(friends, str):
        friends = eval(friends)
    for friend in friends:
        G.add_edge(user_id, friend)
    num_friends = len(friends)
    num_messages = row['Messages Sent']
    if num_friends < 3 and num_messages > 180:
        suspicious_nodes.append(user_id)

df['Num_Friends'] = df['Friends'].apply(lambda x: len(eval(x)) if isinstance(x, str) else len(x))

df.head()
```

- Create Directed Graph: Initialized a directed graph G using nx.DiGraph().
- Track Suspicious Users: Created an empty list of suspicious_nodes to store user IDs meeting suspicious criteria.
- Iterate Through Users: Loop over each row in the DataFrame df, extracting User ID and Friends.
- Handle Friends Data: Converted friends to a list if it's an integer or a string.
- Add Directed Edges: For each user, added directed edges from the user to each friend in the graph G.

- Calculate Friends and Messages: Calculated num_friends (length of friends list) and extract num_messages from the DataFrame.
- Flag Suspicious Users: Added users with fewer than 3 friends and more than 180 messages to the suspicious_nodes list.
- Add Friends Count to DataFrame: Added a new column Num_Friends to the DataFrame using apply() to calculate the number of friends.
- Display Updated DataFrame: Displayed the updated DataFrame with the Num_Friends column using df.head().

7.17. Degree Centrality

in_degree_centrality = nx.in_degree_centrality(G)

out_degree_centrality = nx.out_degree_centrality(G)

print("In degree centrality = ",in_degree_centrality)

print("Out degree centrality = ",out_degree_centrality)

in_degree_threshold = np.percentile(list(in_degree_centrality.values()), 90)

out_degree_threshold = np.percentile(list(out_degree_centrality.values()), 90)

print("In degree threshold = ",in_degree_threshold)

print("Out degree threshold = ",out_degree_threshold)

- In-degree Centrality Calculation: Measures how many incoming edges a node has influence.
- Out-degree Centrality Calculation: Measures how many outgoing edges a node has influence.
- Print Centralities: Display in-degree and out-degree centrality values for each node.
- 90th Percentile Threshold for In-degree: Identifies nodes with high in-degree centrality, suggesting importance in the network.
- 90th Percentile Threshold for Out-degree: Identifies nodes with high out-degree centrality, indicating activity or influence.
- Print Thresholds: Displays the calculated 90th percentile thresholds.

7.18. Visualization

high_in_degree_nodes = [node for node, centrality in in_degree_centrality.items() if centrality >= in_degree_threshold]

high_out_degree_nodes = [node for node, centrality in out_degree_centrality.items() if centrality >= out_degree_threshold]

plt.figure(figsize=(12, 8))

pos = nx.spring_layout(G)

nx.draw_networkx_nodes(G, pos, node_color='green', node_size=50)

nx.draw_networkx_nodes(G, pos, nodelist=suspicious_nodes, node_color='red', node_size=100)

nx.draw_networkx_nodes(G, pos, nodelist=high_in_degree_nodes, node_color='orange', node_size=60)

nx.draw_networkx_nodes(G, pos, nodelist=high_out_degree_nodes, node_color='purple', node_size=50)

nx.draw_networkx_edges(G, pos, arrowstyle='->', arrowsize=10, alpha=0.15)

nx.draw_networkx_labels(G, pos, font_size=8, font_color="black")

plt.title("Directed Social Network Graph with In-degree and Out-degree Centrality")

plt.show()

print("Suspicious Users Based on Friends and Messages:\n", suspicious_nodes)

print("High In-Degree Centrality Users (90th percentile or above):\n", high_in_degree_nodes)

print("High Out-Degree Centrality Users (90th percentile or above):\n", high_out_degree_nodes)

- High In-Degree Nodes: Identifies users with many incoming interactions, indicating influence or centrality in the network.
- High Out-Degree Nodes: Identifies users with many outgoing interactions, indicating activity or the ability to initiate interactions.
- Suspicious Users: Flagged based on having fewer than 3 friends and sending more than 180 messages.
- Node Colors:
    - Green: Normal users
    - Red: Suspicious users
    - Orange: High in-degree users
    - Purple: High out-degree users
- Directed Edges: Arrows show the direction of interaction or influence.
- Graph Layout: Nodes are positioned using spring_layout() for better visualization.
- Thresholding: 90th percentile is used to determine which users have high in-degree or out-degree centrality.
- Fraud Detection: High in-degree or out-degree users could be hubs or active spreaders of fraudulent activity.

8. Results

Code1 -> https://colab.research.google.com/drive/1vgPfjAyvdhTRM4QE0iU479IuUvCr_xSC?usp=drive_link

	User ID	Friends Count	Groups Joined	Posts Per Week	Likes Received	Messages Sent	Login Frequency	Time Between Interactions (hours)
0	1	250	5	10	200	30	5	2.0
1	2	400	10	20	1500	50	7	1.0
2	3	10	1	40	50	600	6	0.5
3	4	500	15	5	1000	20	3	4.0
4	5	3	0	45	30	800	7	1.0

Figure 1: First five rows of the dataset

The above Figure1 displays the first five rows from our dataset to quickly understand the structure.

	User ID	Friends Count	Groups Joined	Posts Per Week	Likes Received	Messages Sent	Login Frequency	Time Between Interactions (hours)
count	50.00000	50.000000	50.000000	50.00000	50.000000	50.000000	50.000000	50.000000
mean	25.50000	178.240000	5.300000	24.48000	522.800000	356.600000	5.420000	1.890000
std	14.57738	171.550262	4.803273	16.66628	483.323443	435.921169	1.566095	1.217601
min	1.00000	0.000000	0.000000	5.00000	5.000000	9.000000	2.000000	0.400000
25%	13.25000	15.000000	1.000000	11.25000	41.250000	23.500000	4.000000	0.925000
50%	25.50000	112.500000	4.000000	17.50000	520.000000	37.500000	5.500000	1.650000
75%	37.75000	332.500000	9.000000	39.50000	800.000000	865.000000	6.750000	2.575000
max	50.00000	510.000000	16.000000	60.00000	1500.000000	1100.000000	8.000000	5.000000

Figure 2: Summary statistics

The above Figure2 shows the summary statistics of the overall data, namely:-

- The dataset contains 50 users.
- The average values for each variable provide a general overview of the user behavior. Here, average user has 178 friends, joins 5 groups, posts 24 times per week, and receives 522 likes per week, approximately.
- The standard deviation (std) shows the spread of the data. Here, the standard deviation for Friends Count is 171.55, indicating that there is a significant variation in the number of friends users have.
- The minimum and maximum values provide the range of each variable. Here, the number of friends ranges from 0 to 510.

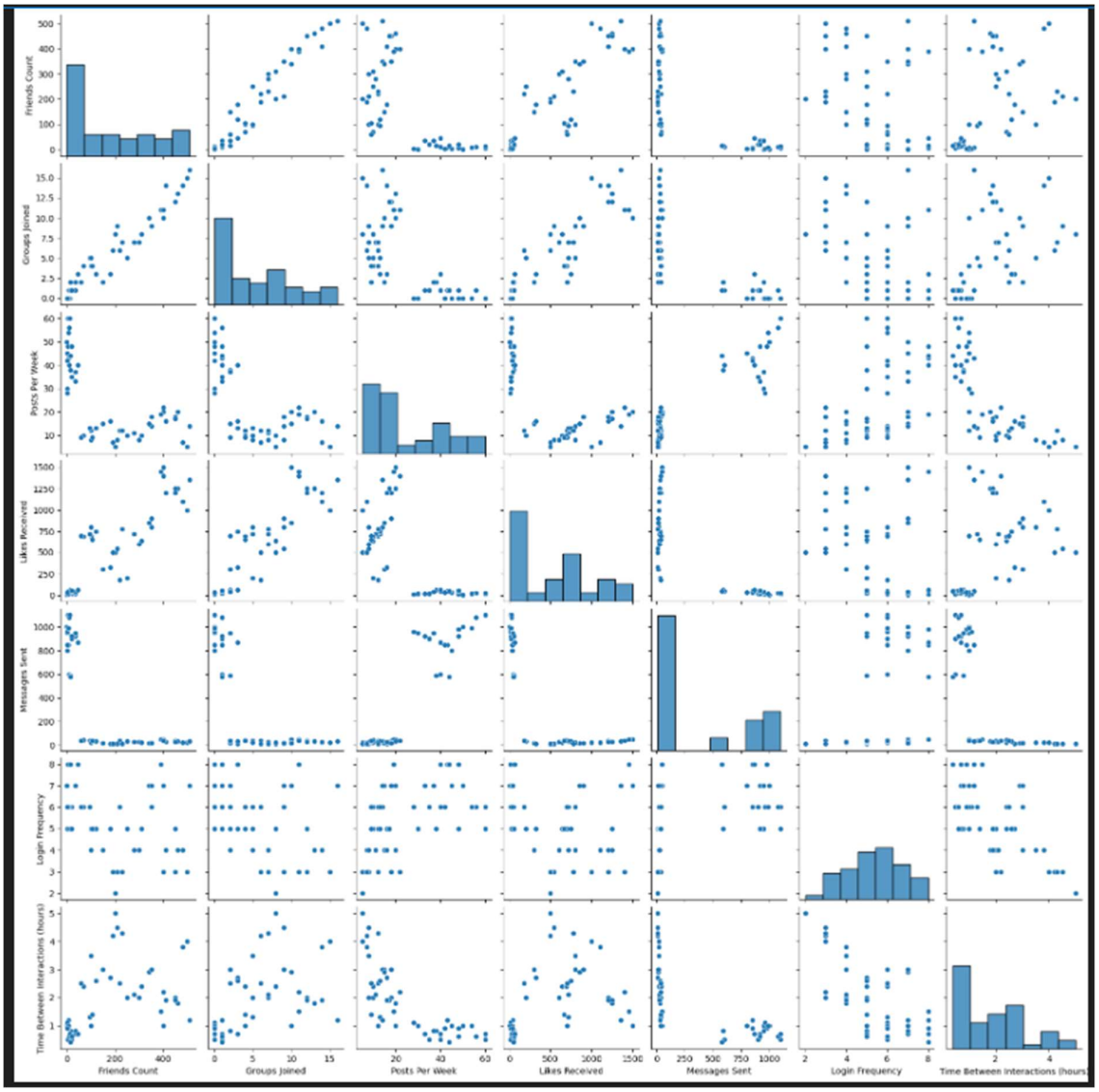


Figure 3: Distribution features

In the above Figure3 provided pair plot is a visualization tool that displays the relationships between all pairs of variables in a dataset. Each subplot in the matrix represents the relationship between two variables, with scatter plots, histograms, and density plots showing the distribution of each variable.

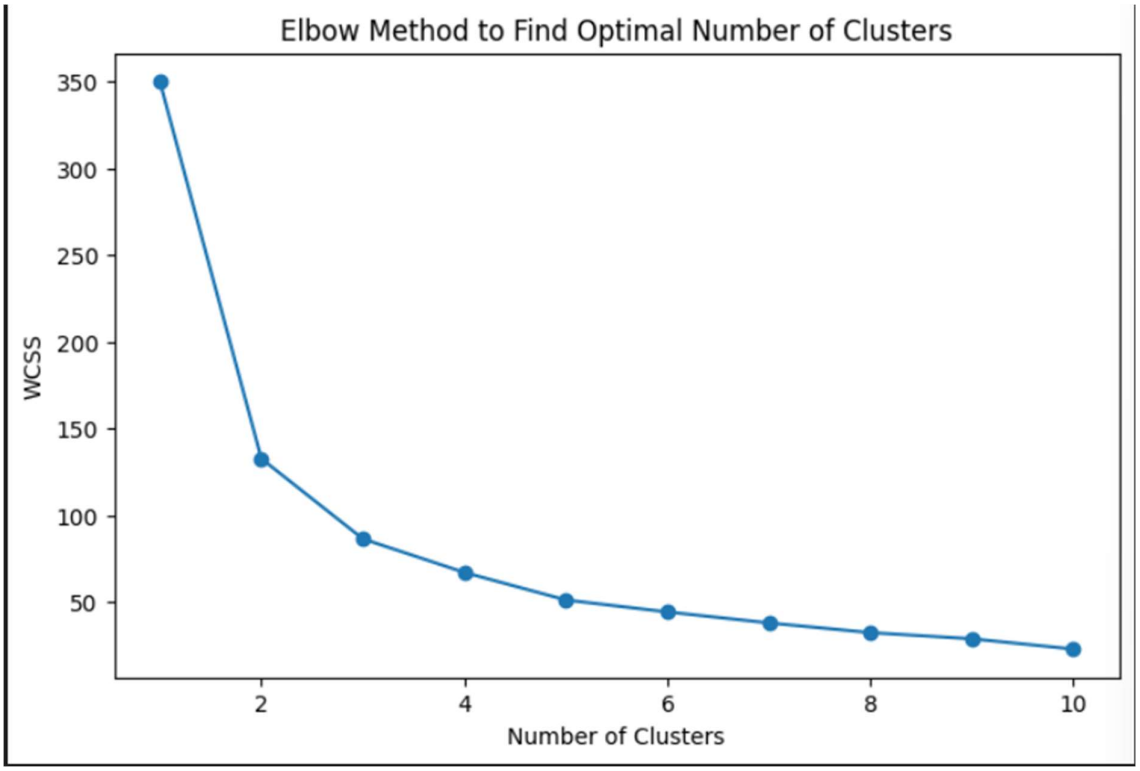


Figure 4: Elbow Graph

In the above Figure4 we observed:-

- Elbow Point: The "elbow" in the plot is the point where the rate of decrease in WCSS starts to slow down significantly. This is often considered the optimal number of clusters.
- Decreasing WCSS: As the number of clusters increases, the WCSS generally decreases. This is because each data point is assigned to a closer cluster center, reducing the overall variance within the clusters.

	User ID	Friends Count	Groups Joined	Posts Per Week	Likes Received	Messages Sent	Login Frequency	Time Between Interactions (hours)	Cluster
0	1	250	5	10	200	30	5	2.0	0
1	2	400	10	20	1500	50	7	1.0	0
2	3	10	1	40	50	600	6	0.5	1
3	4	500	15	5	1000	20	3	4.0	0
4	5	3	0	45	30	800	7	1.0	1

Figure 5: Cluster result after applying K-Means

In the above Figure5 we observedd:-

- Cluster 0: Users in this cluster have a higher number of friends, groups joined, and likes received. They also post and message more frequently. This suggests a cluster of highly engaged and socially active users.

- Cluster 1: Users in this cluster have a lower number of friends and groups joined, but they post and message significantly more than those in Cluster 0. This might indicate a cluster of highly active users who are focused on content creation and communication within a smaller social circle.

```
Cluster Centers:
    Friends Count  Groups Joined  Posts Per Week  Likes Received  \
0      279.354839       8.096774       12.677419      824.193548
1       13.263158       0.736842       43.736842       31.052632

   Messages Sent  Login Frequency  Time Between Interactions (hours)
0      27.096774         4.806452                           2.567742
1     894.210526         6.421053                           0.784211
```

Figure 6: Cluster Centers

In the above Figure6 we observed:-

Cluster Analysis

- Cluster 0:
    - Has a high average number of friends (279.354839).
    - Joins an average of 8.896774 groups.
    - Posts an average of 12.677419 times per week.
    - Receives an average of 824.193548 likes per week.
    - Sends an average of 27.096774 messages per week.
    - Logs in an average of 4.806452 times per week.
    - Has an average time between interactions of 2.567742 hours.
- Cluster 1:
    - Has a low average number of friends (13.263158).
    - Joins an average of 0.736842 groups.
    - Posts an average of 43.736842 times per week.
    - Receives an average of 31.052632 likes per week.
    - Sends an average of 894.210526 messages per week.
    - Logs in an average of 6.421053 times per week.
    - Has an average time between interactions of 0.784211 hours.

Key Differences Between Clusters:

- Cluster 0 appears to represent users with a larger social network, moderate posting activity, and relatively frequent interactions.

- Cluster 1 seems to represent users with a smaller social network, but significantly higher posting and messaging activity. They also log in more frequently and have shorter time between interactions.

Potential Insights:

- Engagement Patterns: Cluster 1 users might be highly engaged and active on the platform, despite having a smaller social network.
- Communication Styles: The difference in messaging activity between the clusters could indicate different communication styles. Cluster 1 users might be more active in group chats or one-on-one messaging.
- Platform Usage: Cluster 1 users might be using the platform more intensively, as evidenced by higher login frequency and shorter time between interactions.

```
Suspicious Cluster Centers:
    Friends Count  Groups Joined  Posts Per Week  Likes Received  \
1       13.263158       0.736842       43.736842       31.052632

   Messages Sent  Login Frequency  Time Between Interactions (hours)
1     894.210526         6.421053                           0.784211
Potential Fraudulent Users:
     User ID  Friends Count  Groups Joined  Posts Per Week  Likes Received  \
2          3             10              1              40              50
4          5              3              0              45              30
6          7             20              1              35              20
9         10              0              0              50               5
11        12              5              0              60              10
13        14              1              0              30              15
17        18             15              2              38              60
19        20              8              1              43              45
21        22             35              1              33              18
24        25              5              0              48              10
26        27             10              1              56              25
28        29              2              0              28              12
31        32             15              1              44              50
34        35             12              0              60              25
37        38             35              2              37              40
40        41              8              0              54              20
42        43              3              0              42              35
...
40           990              6                              1.0        1
42           850              6                              0.9        1
```

Figure 7: Suspicious clusters

In the above Figure7 we observed:-

i. Low Engagement

Low engagement among potential fraudulent users is characterized by:

- Limited Posts: Users with a small number of posts may indicate a lack of genuine interaction with the platform. In contrast, legitimate users often share content regularly.

- Few Likes: Low numbers of likes received can also point to a lack of engagement, as active users typically receive more likes on their posts. This could suggest that the user is either posting irrelevant content or is not engaged enough to attract likes.
- Minimal Messages: A low count of messages sent could imply that the user does not actively communicate with others on the platform. This lack of interaction can be a red flag, especially if the user is logged in frequently.

ii. Frequent Logins

Frequent logins despite low engagement might indicate:

- Suspicious Behavior: Users who log in often but do not engage with content or others may be utilizing automated tools or bots to mimic user activity. This behavior could be an attempt to maintain account visibility without genuine interaction.
- Account Manipulation: High login frequency can also be associated with attempts to manipulate platform algorithms. For example, a user might be logging in to check for notifications or messages without contributing to the community.

iii. Inconsistent Activity

Variability in user activity levels can manifest in several ways:

- Friend Count and Group Membership: Some potential fraudulent users may have a disproportionately high number of friends or groups joined relative to their engagement. This inconsistency can indicate that these users are trying to appear more socially connected than they actually are.
- Erratic Posting Patterns: Fluctuations in posting frequency—such as a sudden increase in posts followed by a long period of inactivity—can be indicative of bot activity or manipulation strategies aimed at gaining visibility.

Potential Indicators of Fraud

i. High Login Frequency with Low Engagement

This pattern is a strong indicator of possible fraudulent activity, as it suggests that:

- Automated Tools or Bots: Users may be using scripts or software to log in automatically without engaging in meaningful interactions. Monitoring such patterns can help identify non-human behavior on the platform.
- Fake Accounts: These accounts may be created for the purpose of inflating friend counts or manipulating group dynamics without genuine user activity.

ii. Inconsistent Activity Patterns

Activity inconsistencies, such as abrupt spikes or drops, can point to:

- Manipulated Engagement: A sudden increase in posting or liking behavior may be an attempt to boost visibility or engagement artificially. Conversely, a drop could

indicate that the user is retreating due to detection or potential flagging by the platform.

- Behavioral Analysis: Employing machine learning models to analyze user behavior over time can help detect unusual patterns indicative of fraud, such as sudden shifts in engagement frequency.

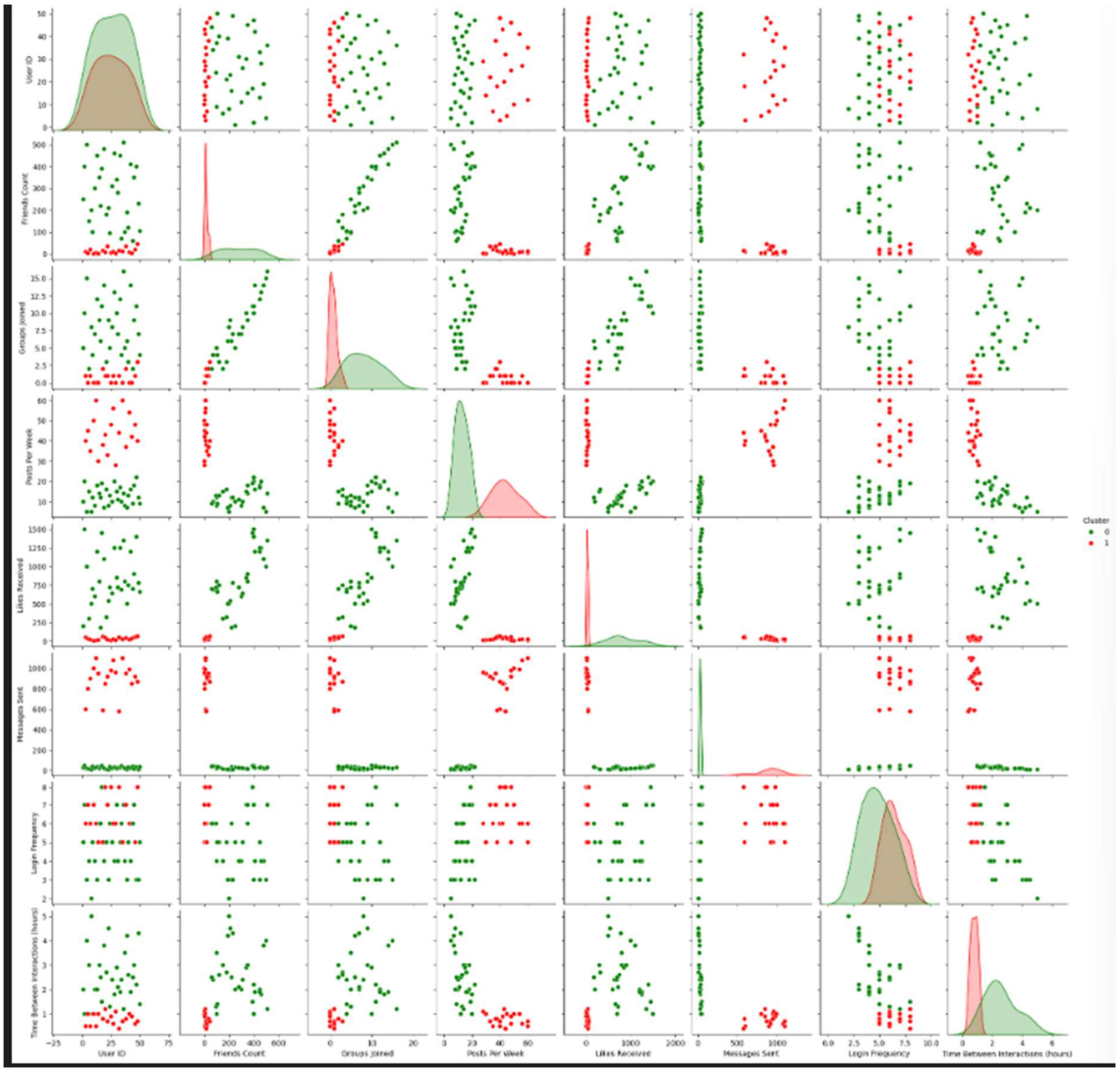


Figure 8: fraudulent users(red) and legitimate users(green)

In the above Figure8 we observed:-

Positive Correlations

i. Friends Count and Posts Per Week: This indicates that individuals with a higher number of friends tend to post more frequently. This could imply that having a larger social network encourages users to share more content, possibly to engage their friends or keep them updated.

ii. Friends Count and Messages Sent: Similar to posts, individuals with more friends may feel more inclined to communicate with them via messages. This suggests an active engagement with their social network.

iii. Likes Received and Messages Sent: A positive correlation here could indicate that users who receive more likes on their posts may be more motivated to send messages, possibly to thank friends for their engagement or to maintain connections.

iv. Login Frequency and Messages Sent: This correlation suggests that users who log in more often are also more active in sending messages. This may reflect a higher level of engagement with the platform, where frequent users are more likely to interact with others.

Negative Correlations

i. Time Between Interactions and Posts Per Week: A slight negative correlation indicates that as the time between interactions increases, the frequency of posts may decrease. This could imply that users who spend longer periods without interacting may not feel motivated to post updates.

ii. Time Between Interactions and Messages Sent: Similarly, if the time between interactions increases, it could lead to a decrease in the number of messages sent. This might suggest that users who are less engaged over time become less likely to communicate with their friends.

Skewness

i. Friends Count: If the distribution of friends count is positively skewed, it means that a small number of users have a very high number of friends, while most have fewer friends. This could reflect the social media phenomenon where a few users become highly popular.

ii. Groups Joined: A skewed distribution here might indicate that most users join only a few groups, while a small subset of users are very active and join many groups. This could be related to niche interests or user engagement levels.

Outliers

i. Visible Outliers in Scatter Plots: Outliers can often be identified in scatter plots, where certain data points stand apart from the general trend. For example, a user with an exceptionally high number of posts but very few friends might be an outlier. Analyzing these outliers can provide insights into unique user behaviors or profiles that deviate from the norm.

ii. Impact of Outliers: Outliers can significantly affect correlation coefficients and other statistical measures. It's essential to investigate these points to understand whether they represent genuine user behavior or data entry errors.

Our analysis suggests meaningful relationships between social media interaction variables. Positive correlations indicate that increased social connections and activity may lead to more engagement, while negative correlations point to the possibility that reduced interaction frequency can lead to decreased user activity. Skewness and outliers highlight the uneven

distribution of data points and the presence of exceptional cases, which can provide valuable insights into user behavior. Exploring these relationships further we could enhance understanding of user engagement dynamics on social media platforms.

	A	B	C	D	E	F	G	H	I
1	User ID	Friends Co	Groups Joi	Posts Per \	Likes Rece	Messages	Login Freq	Time Betw	Cluster
2	1	250	5	10	200	30	5	2	0
3	2	400	10	20	1500	50	7	1	0
4	3	10	1	40	50	600	6	0.5	1
5	4	500	15	5	1000	20	3	4	0
6	5	3	0	45	30	800	7	1	1
7	6	150	2	15	300	15	4	3	0
8	7	20	1	35	20	900	6	0.5	1
9	8	200	8	5	500	10	2	5	0
10	9	100	4	12	700	40	5	1	0
11	10	0	0	50	5	1000	7	1	1
12	11	300	7	8	600	25	4	2	0
13	12	5	0	60	10	1100	6	0.5	1
14	13	450	12	18	1200	35	3	2	0
15	14	1	0	30	15	950	5	1	1
16	15	350	9	14	800	20	6	3	0
17	16	220	6	12	180	35	6	2.5	0
18	17	390	11	19	1450	48	8	1.5	0
19	18	15	2	38	60	590	5	0.8	1
20	19	480	14	7	1100	22	4	3.8	0
21	20	8	1	43	45	850	8	1.2	1
22	21	180	3	16	320	12	5	2.7	0
23	22	35	1	33	18	920	7	0.7	1
24	23	210	9	8	540	9	3	4.5	0
25	24	95	5	13	720	38	6	1.3	0

Figure 9: Dataset post anomaly detection

In the above Figure9 we can see a part of the dataset with an additional cluster column having flag values 0 and 1. In this cluster column 0 represents the legitimate users whereas 1 represents the fraudulent user. The full dataset can be found in the below link.

Final Dataset Link ->
https://drive.google.com/file/d/1mHqyOdZ09i6SGuuxp80JqKI6y2XJhpcR/view?usp=sharing

Code2 ->
https://colab.research.google.com/drive/1fpVF4j5PNjaAKtKvaNKNCtwSGSTT2Jio?usp=sharing#scrollTo=KlqCP_bVI4Ni

	User ID	Friends	Messages Sent
0	user_1	['user_16', 'user_20', 'user_7', 'user_8']	167
1	user_2	['user_16']	185
2	user_3	['user_19', 'user_31', 'user_42', 'user_25', '...	133
3	user_4	['user_46', 'user_45', 'user_43', 'user_6', 'u...	129
4	user_5	['user_48', 'user_17', 'user_12', 'user_31', '...	121

Figure 10: First 5 rows of dataset 2

In the above figure10 we can observe a fraction of the dataset 2 for further analysis.

	User ID	Friends	Messages Sent	Num_Friends
0	user_1	['user_16', 'user_20', 'user_7', 'user_8']	167	4
1	user_2	['user_16']	185	1
2	user_3	['user_19', 'user_31', 'user_42', 'user_25', '...	133	9
3	user_4	['user_46', 'user_45', 'user_43', 'user_6', 'u...	129	8
4	user_5	['user_48', 'user_17', 'user_12', 'user_31', '...	121	8

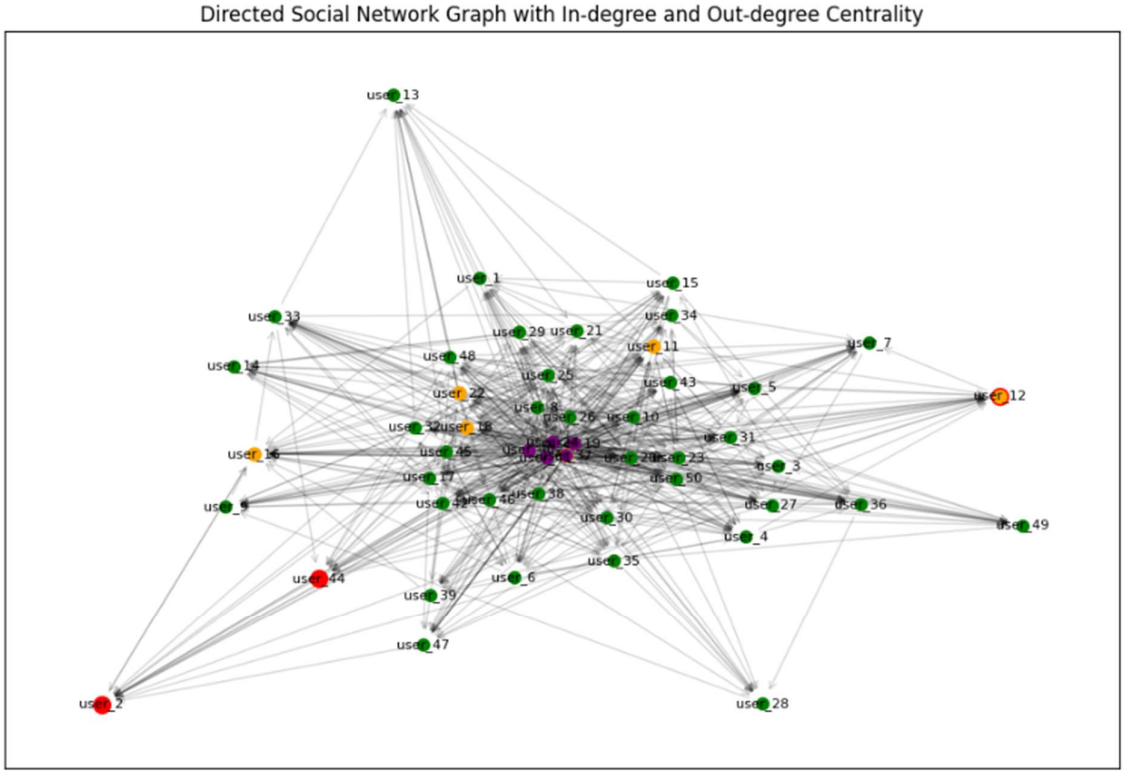
Figure 11: Calculation of no. of friends

We had calculated the total number of friends of each user and printed them.

```
In degree centrality =  {'user_1': 0.24489795918367346, 'user_16': 0.3877551020408163, 'user_20': 0.28571428
Out degree centrality =  {'user_1': 0.08163265306122448, 'user_16': 0.16326530612244897, 'user_20': 0.367346
In degree threshold =  0.3469387755102041
Out degree threshold =  0.467346938775511
```

Figure 12: Calculation of degree centrality

We had calculated the in degree and out degree centrality of each node along with the in degree and out degree threshold to analyse the suspicious pattern.




```
Suspicious Users Based on Friends and Messages:
 ['user_2', 'user_12', 'user_44']
High In-Degree Centrality Users (90th percentile or above):
 ['user_16', 'user_12', 'user_37', 'user_22', 'user_11', 'user_18']
High Out-Degree Centrality Users (90th percentile or above):
 ['user_19', 'user_24', 'user_40', 'user_37', 'user_41']
```

Figure 13: Visualization of the directed graph

Here the green nodes represent the potential users, red nodes represent the suspicious users who have total friends less than 3 and messages sent is more than 180, orange for users with suspiciously high in-degree centrality and purple for users with suspiciously high out-degree centrality.

We can also observe that user 12 is red as well as orange which indicates it shows suspicious behaviour along with high in-degree centrality and user 37 is orange as well as purple which indicates it shows high in-degree centrality along with high out-degree centrality.

This analysis confirms that user 12 and user 37 are confirmed as fraud users.

9. Conclusion

In this project, we successfully developed a system for detecting fraudulent behavior in online networks by leveraging network properties and applying anomaly detection techniques. The primary objective was to identify suspicious activities among users based on their network interactions, utilizing K-Means clustering to classify users into legitimate and fraudulent categories. The project demonstrates how data analysis and machine learning can be effectively integrated to enhance security measures in online platforms.

The project began with a thorough analysis of the dataset, which involved loading the data, cleaning it by removing unnecessary columns, and performing exploratory data analysis. This initial phase allowed us to understand the structure of the data and identify key features relevant to the analysis. By preparing the data meticulously, we set the stage for effective model training and evaluation.

We employed the K-Means clustering algorithm to segment users into different clusters based on their network activity. By implementing the Elbow Method, we determined the optimal number of clusters, which was found to be two. This allowed us to differentiate between potentially fraudulent users and legitimate ones effectively. The choice of K-Means and the determination of the optimal cluster number were essential in establishing a reliable classification system.

By analyzing the cluster centers, we established criteria for identifying suspicious users—specifically, those with fewer than 100 friends and more than 500 messages sent. This targeted approach enabled us to pinpoint individuals exhibiting potentially fraudulent behavior based on their interaction patterns. Defining such criteria illustrates how domain knowledge can enhance the effectiveness of anomaly detection models.

The use of visualizations, including pair plots and elbow graphs, provided insightful representations of the data and clustering results. These visual tools facilitated a better understanding of how users were distributed in the feature space and highlighted the characteristics of different clusters. Visualizations are a powerful means of communicating complex data findings, making them accessible to stakeholders and decision-makers.

Our further implementation identifies fraud users in a social network based on a combination of their number of friends, messages sent, in-degree and out-degree centrality. We pinpoint suspiciously influential active users who may play a role in spreading fraudulent activity. Visualizing the network with directed edges and color-coded nodes highlights users with significant centrality, providing insights for potential fraud detection and further investigation. The use of 90th percentile thresholds helps in isolating extreme behavior, enabling targeted monitoring of high-risk individuals.

The system successfully identified a set of users classified as potentially fraudulent, which could be subject to further investigation. This demonstrates the practical application of our model in real-world scenarios, where detecting suspicious activities is crucial for maintaining the integrity of online networks. The identification of potential fraudsters illustrates the tangible benefits of applying machine learning techniques to network analysis.

The findings from our project have significant implications for online platforms seeking to enhance their security measures against fraudulent behavior. By implementing similar anomaly detection methodologies, organizations can proactively identify and mitigate risks associated with fraudulent activities. This proactive approach is essential for safeguarding user trust and ensuring the long-term success of online services.