Data Mining (Phase 4)

Team Name: Trio-Chargers

Team Members:

1. Sumanth Reddy Desireddy (Team Head) (sdesi4@unh.newhaven.edu)
2. Lakshmi Kara Gupta Chandulooru (lchan3@unh.newhaven.edu)
3. Uday kiran Karumburi Arumugam (ukaru1@unh.newhaven.edu)

Research Question:

Account security: Detecting spam or legit, unusual login behavior and unauthorized access attempts through Machine learning algorithms.

About Question:

Account security is a critical concern for online platforms like social media websites and email services. Detecting spam or legitimate accounts, unusual login behavior, and unauthorized access attempts is essential to protect user data and maintain the integrity of the platform. Machine learning algorithms play a significant role in enhancing account security by automating the detection of these security threats. Here's an explanation of how machine learning can be applied to these aspects of account security:

1. Detecting Spam or Legitimate Accounts:

- Classification Models: Algorithms like logistic regression, decision trees, random forests, or more advanced techniques like neural networks can be used for classification.
- Training Data: Historical data containing labeled examples of legitimate and spam accounts are used to train the model.
- Feature Engineering: Machine learning models can be trained using a dataset like the one described earlier, with features like user activity, profile information, and behavior patterns.

2. Unusual Login Behavior Detection:

- Feature Extraction: Relevant features for detecting unusual login behavior include IP address, geolocation, device type, login time, and login frequency.

3. Unauthorized Access Attempts:

- Data Sources: Login attempts, failed login attempts, IP addresses, geolocation data, and authentication logs are valuable data sources.
- Real-time Detection: Machine learning models can operate in real-time, continuously monitoring login attempts and assessing their legitimacy. Suspicious patterns can trigger security alerts or block further access attempts.

4. Behavioral Biometrics:

- Behavioral Analysis: Machine learning can be used to analyze user behavior, including typing patterns, mouse movements, and even the way users interact with a website or app.

5. Adaptive Security:

- Feedback Loop: Security teams can provide feedback to the machine learning system, helping it improve over time and ensuring that false positives and negatives are minimized.

Selected Data Set:

1. https://www.kaggle.com/datasets/khajahussainsk/facebook-spam-dataset
2. https://www.kaggle.com/datasets/sheenabatra/facebook-data

The dataset can be used for building machine learning models. To collect the dataset, Facebook API and Facebook Graph API are used and the data is collected from public profiles by Kaggle.

The dataset consists of a total of 600 profiles, with 500 being legitimate and 100 being spam.

Here's a description of the features included in the dataset, along with a brief explanation of each:

1. **Number of Friends:** This feature represents the count of friends associated with the user's Facebook profile. It measures the user's social network size.
2. **Number of Followings:** This feature indicates the number of accounts or pages that the user is following on Facebook.
3. **Number of Community:** This feature likely pertains to the number of Facebook Groups or Communities that the user is a part of.
4. **Age of the User Account (in days):** This feature reflects how long the user's Facebook account has been active, measured in days. It can provide insights into the account's longevity.
5. **Total Number of Posts Shared:** This feature counts the total number of posts shared by the user on their Facebook timeline.
6. **Total Number of URLs Shared:** This feature represents the total count of URLs or web links shared by the user in their posts.
7. **Total Number of Photos/Videos Shared:** This feature counts the total number of photos and videos shared by the user on their Facebook profile.
8. **Fraction of Posts Containing URLs:** This feature calculates the ratio of posts containing URLs to the total number of posts shared. It indicates how frequently the user shares web links.
9. **Fraction of Posts Containing Photos/Videos:** Similar to the previous feature, this one calculates the ratio of posts containing photos or videos to the total number of posts. It shows the user's preference for multimedia content.
10. **Average Number of Comments per Post:** This feature calculates the average number of comments received on the user's posts. It can be an indicator of engagement with the user's content.
11. **Average Number of Likes per Post:** This feature calculates the average number of likes (or reactions) received on the user's posts. It reflects the user's popularity or the engagement level of their posts.
12. **Average Number of Tags in a Post (Rate of Tagging):** This feature calculates the average number of tags present in each of the user's posts. It indicates how often the user is tagged in other people's posts or how frequently they tag others in their posts.

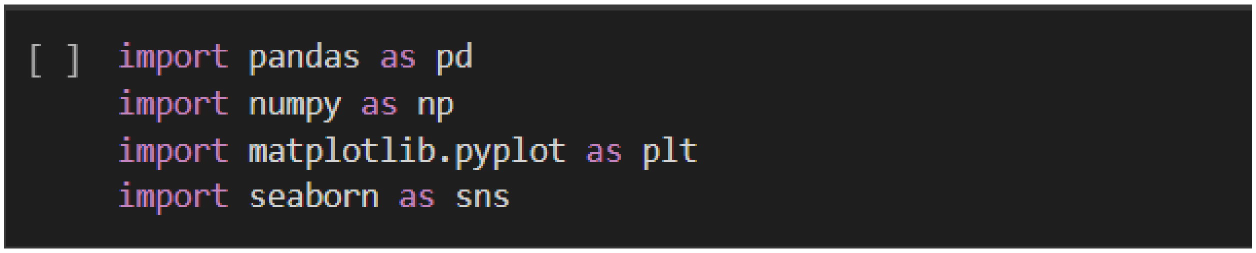
The "Label" column assigns a binary value to each profile, with "0" indicating a legitimate profile and "1" indicating a spam profile. This label is the target variable for classification tasks in machine learning.

Researchers and data scientists can use this dataset to develop and train machine learning models for the detection and classification of spam profiles on Facebook based on the provided features. The goal would be to build a predictive model that can automatically identify and flag spam profiles, helping maintain the integrity and security of online social networks.

Exploring the Dataset

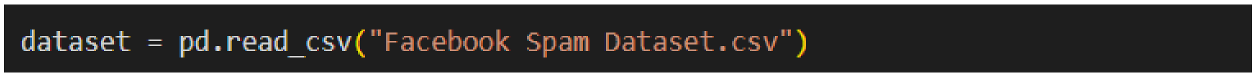
We will import a dataset upon which we will be doing the data analysis operations to comprehend the concepts and procedures involved in data analysis.

Bringing in Libraries to use



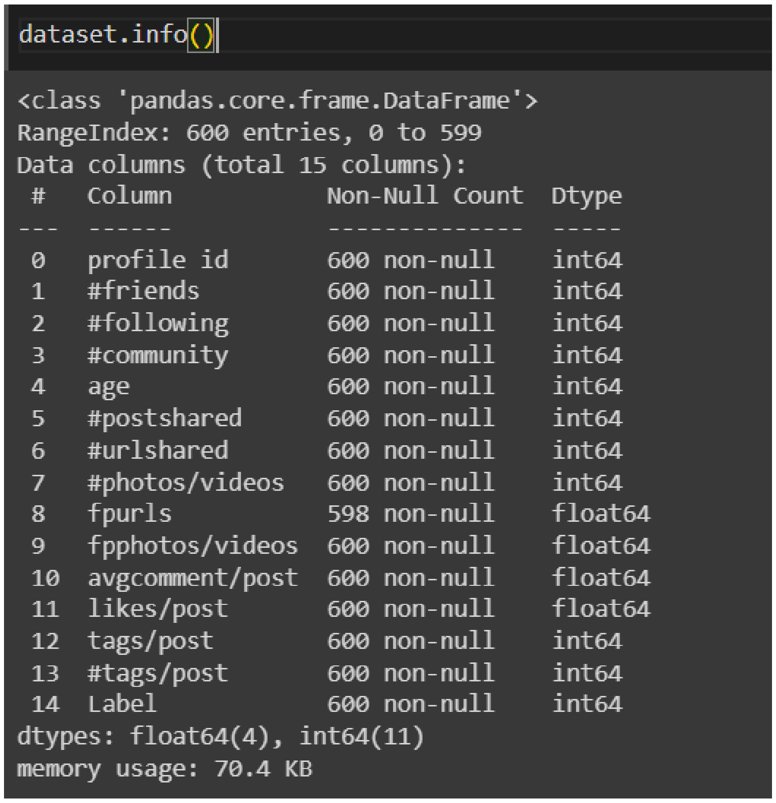
```
[ ]  import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
```

Importing the Dataset
Here, we will be using the "Facebook Spam Dataset.csv"



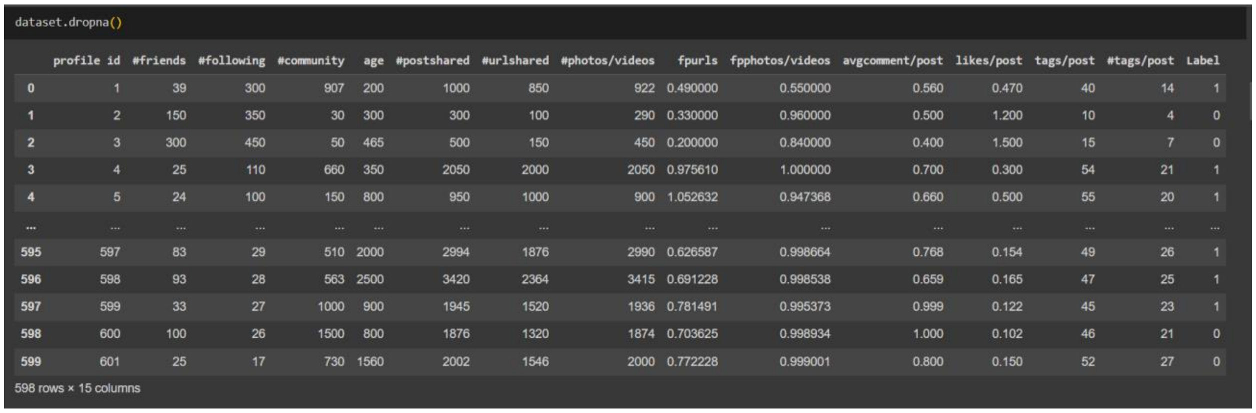
```
dataset = pd.read_csv("Facebook Spam Dataset.csv")
```

Here, there are 600 rows and 15 columns in the provided dataset. Here, many sorts of columns are provided. Let's now use the data frame's info method to obtain a summary of the data.



```
dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 600 entries, 0 to 599
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   profile id      600 non-null    int64
 1   #friends        600 non-null    int64
 2   #following      600 non-null    int64
 3   #community      600 non-null    int64
 4   age             600 non-null    int64
 5   #postshared     600 non-null    int64
 6   #urlshared      600 non-null    int64
 7   #photos/videos  600 non-null    int64
 8   fpurls          598 non-null    float64
 9   fpphotos/videos 600 non-null    float64
 10  avgcomment/post 600 non-null    float64
 11  likes/post      600 non-null    float64
 12  tags/post       600 non-null    int64
 13  #tags/post      600 non-null    int64
 14  Label           600 non-null    int64
dtypes: float64(4), int64(11)
memory usage: 70.4 KB
```

Here we are searching for null values in the dataset.

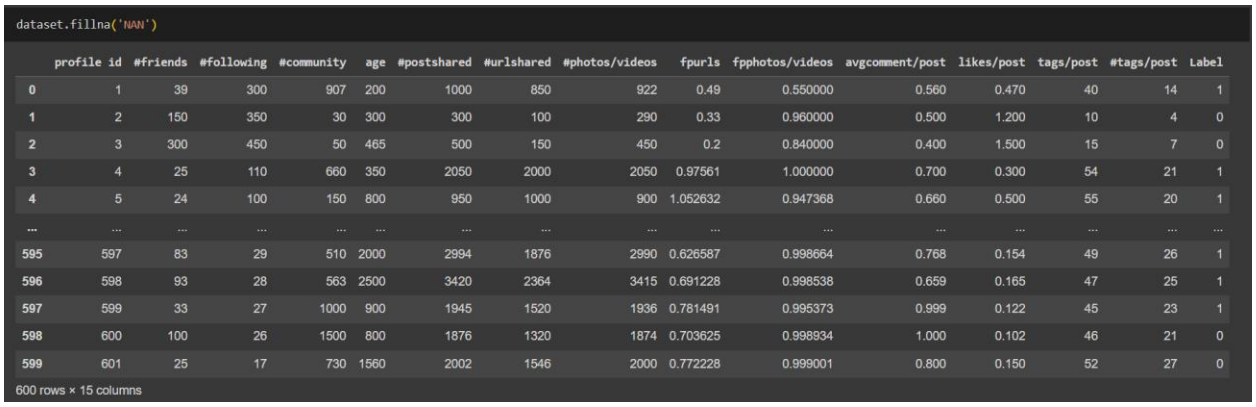


```
dataset.dropna()
```

	profile id	#friends	#following	#community	age	#postshared	#urlshared	#photos/videos	fpurls	fpphotos/videos	avgcomment/post	likes/post	tags/post	#tags/post	Label
0	1	39	300	907	200	1000	850	922	0.490000	0.550000	0.560	0.470	40	14	1
1	2	150	350	30	300	300	100	290	0.330000	0.960000	0.500	1.200	10	4	0
2	3	300	450	50	465	500	150	450	0.200000	0.840000	0.400	1.500	15	7	0
3	4	25	110	660	350	2050	2000	2050	0.975610	1.000000	0.700	0.300	54	21	1
4	5	24	100	150	800	950	1000	900	1.052632	0.947368	0.660	0.500	55	20	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
595	597	83	29	510	2000	2994	1876	2990	0.626587	0.998664	0.768	0.154	49	26	1
596	598	93	28	563	2500	3420	2364	3415	0.691228	0.998538	0.659	0.165	47	25	1
597	599	33	27	1000	900	1945	1520	1936	0.781491	0.995373	0.999	0.122	45	23	1
598	600	100	26	1500	800	1876	1320	1874	0.703625	0.998934	1.000	0.102	46	21	0
599	601	25	17	730	1560	2002	1546	2000	0.772228	0.999001	0.800	0.150	52	27	0

598 rows × 15 columns

And here we are replacing with the NAN of the null values.



```
dataset.fillna('NAN')
```

	profile id	#friends	#following	#community	age	#postshared	#urlshared	#photos/videos	fpurls	fpphotos/videos	avgcomment/post	likes/post	tags/post	#tags/post	Label
0	1	39	300	907	200	1000	850	922	0.49	0.550000	0.560	0.470	40	14	1
1	2	150	350	30	300	300	100	290	0.33	0.960000	0.500	1.200	10	4	0
2	3	300	450	50	465	500	150	450	0.2	0.840000	0.400	1.500	15	7	0
3	4	25	110	660	350	2050	2000	2050	0.97561	1.000000	0.700	0.300	54	21	1
4	5	24	100	150	800	950	1000	900	1.052632	0.947368	0.660	0.500	55	20	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
595	597	83	29	510	2000	2994	1876	2990	0.626587	0.998664	0.768	0.154	49	26	1
596	598	93	28	563	2500	3420	2364	3415	0.691228	0.998538	0.659	0.165	47	25	1
597	599	33	27	1000	900	1945	1520	1936	0.781491	0.995373	0.999	0.122	45	23	1
598	600	100	26	1500	800	1876	1320	1874	0.703625	0.998934	1.000	0.102	46	21	0
599	601	25	17	730	1560	2002	1546	2000	0.772228	0.999001	0.800	0.150	52	27	0

600 rows × 15 columns

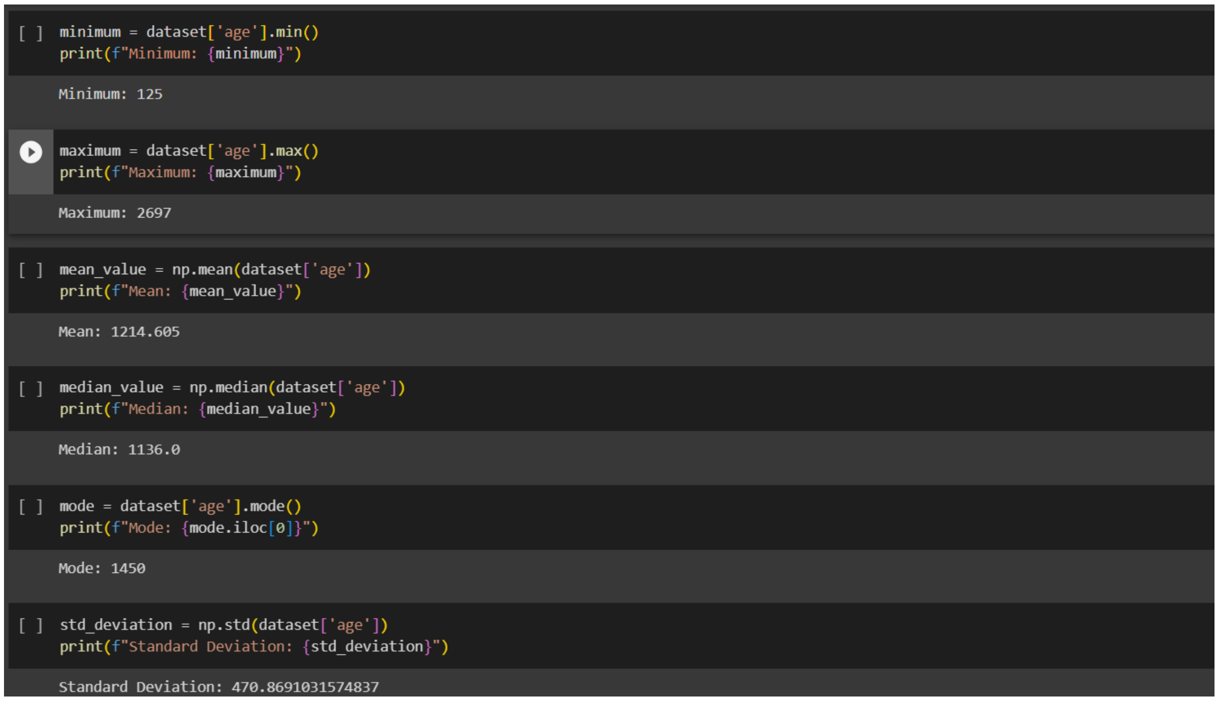
List of Exploration Techniques:

- Maximum, Mean, Median, Mode, Standard Deviation
- Bar Graph
- Pie Chart
- Histogram Plot
- Box Plot
- Count plot
- Grid plot
- Stacked Bar
- Scatter Plot
- Correlation Heatmap

Description of Data Explorations:

Univariate Analysis of Numerical Variables

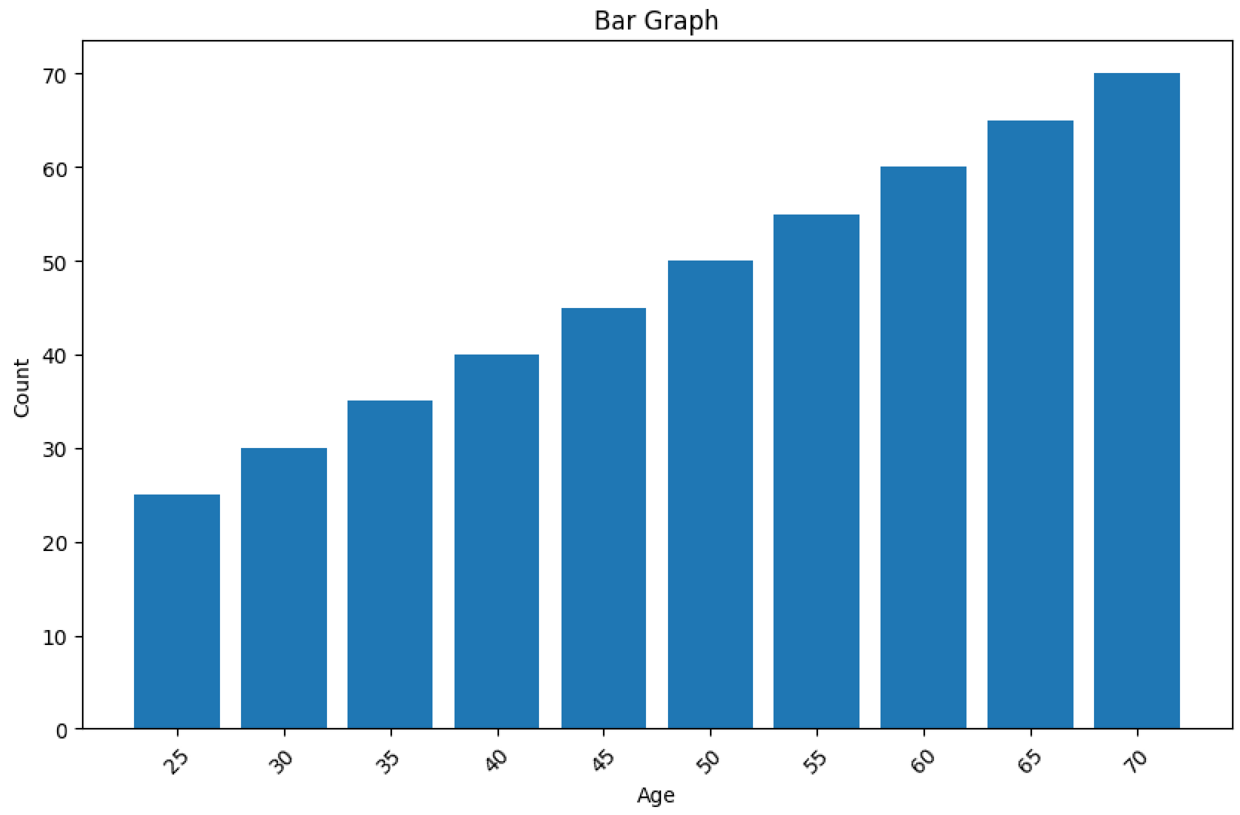
Maximum, Mean, Median, Mode, Standard Deviation



```
[ ] minimum = dataset['age'].min()
    print(f"Minimum: {minimum}")

    Minimum: 125

    maximum = dataset['age'].max()
    print(f"Maximum: {maximum}")

    Maximum: 2697

[ ] mean_value = np.mean(dataset['age'])
    print(f"Mean: {mean_value}")

    Mean: 1214.605

[ ] median_value = np.median(dataset['age'])
    print(f"Median: {median_value}")

    Median: 1136.0

[ ] mode = dataset['age'].mode()
    print(f"Mode: {mode.iloc[0]}")

    Mode: 1450

[ ] std_deviation = np.std(dataset['age'])
    print(f"Standard Deviation: {std_deviation}")

    Standard Deviation: 470.8691031574837
```

Here we just took min, max, mean, median, mode and standard deviation of the age column.

Bar Graph



The distribution of ages in our dataset is seen in the bar graph above. Here are a few significant findings from the graph:

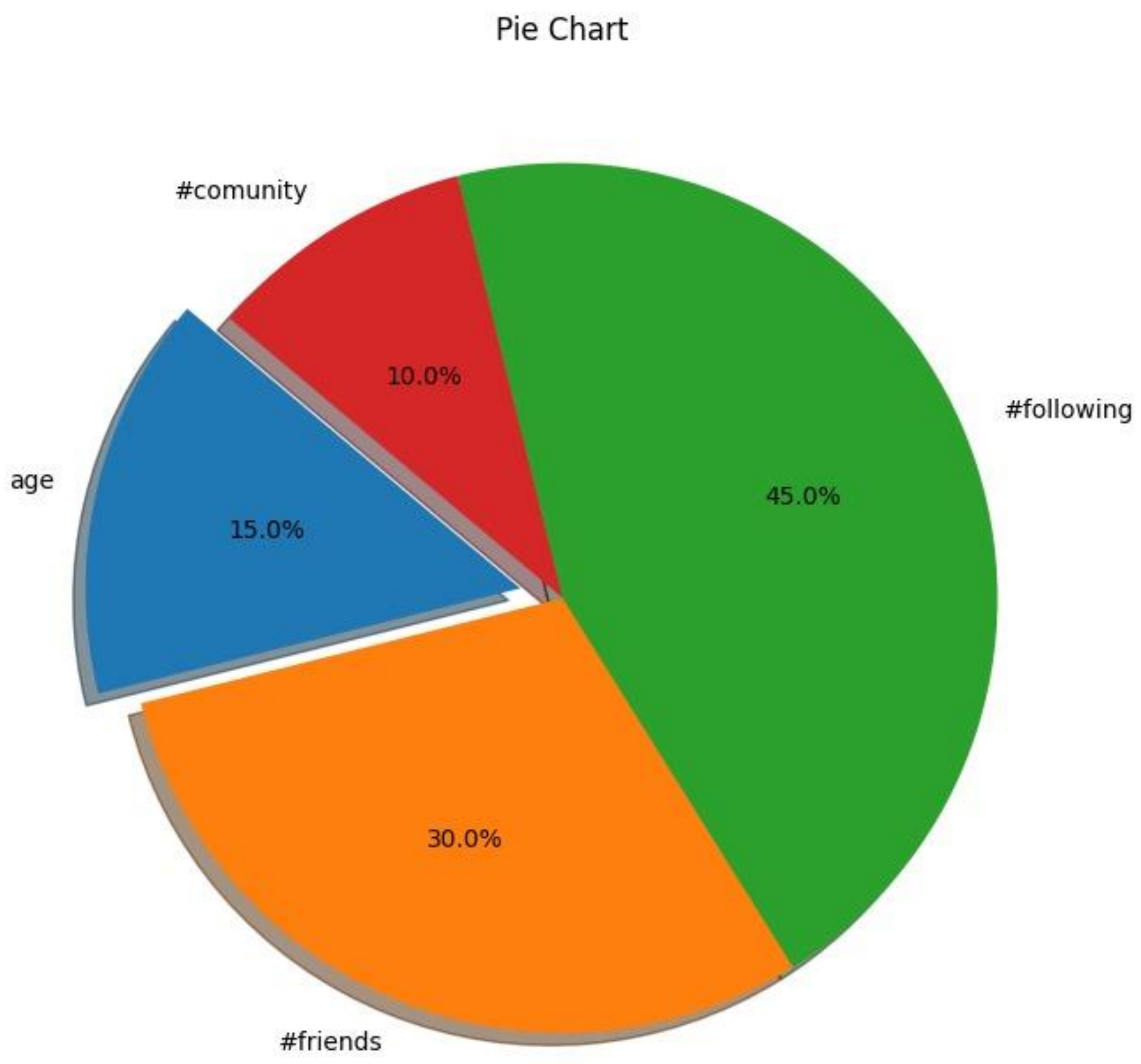
Age Range: The dataset includes individuals of all ages, with the youngest being 25 and the oldest being 70.

Most Common Age Groups: In this dataset, people between the ages of 40 and 50 have the largest representation, closely followed by those between the ages of 30 and 40.

Distribution Shape: The graph's form implies that ages are distributed rather evenly, with a small bump between 40 and 50, which denotes a disproportionately high number of people in this age group.

Outliers: The youngest (25–30) and oldest (65–70) age groups have fewer people, potentially due to outliers or a lower population in those age ranges.

Pie Chart



The distribution of data among age, #friends, #following, #comunity is shown in the pie chart above. Key conclusions from the chart are as follows:

Category Composition: Based on the four categories, the graphic clearly illustrates how the dataset is divided. Most of the data, or category C, comprises 45% of the total. Category B, which comprises 30% of the dataset, is next.

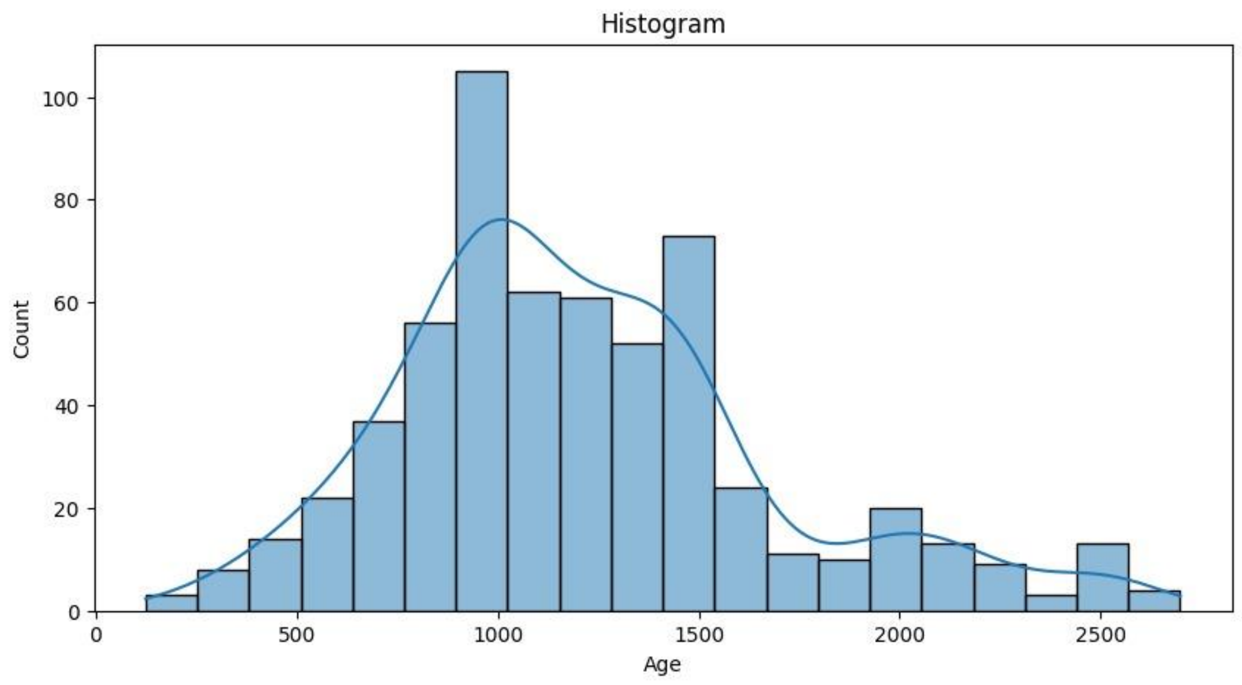
Category A: By significantly expanding it from the pie chart's center, we decided to highlight Category A. With a 15% share of the data, Category A has the least share of all the categories, as seen by this graphic effect.

Rest of Categories: With only 10% of the dataset, Category D is the smallest category.

Proportional Representation: For a fast reference on how the data are distributed proportionally in the graphic, look at the percentage labels next to each category.

Start Angle: To improve presentation, the chart's start angle is set to 140 degrees, which modifies the pie chart's first rotation.

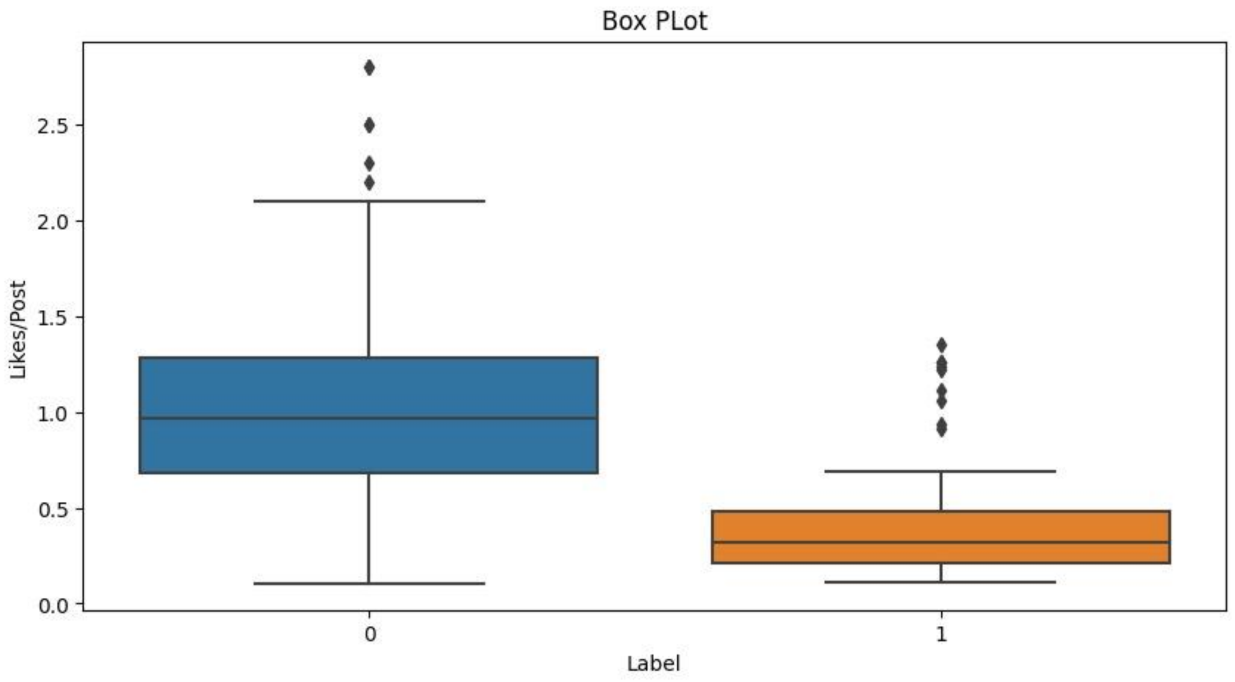
Histogram Graph



Distribution Plot (Histogram with KDE):

- sns.histplot(df['age'], bins=20, kde=True) creates a histogram of the 'age' column from your DataFrame (df) with 20 bins and a Kernel Density Estimate (KDE) overlaid on it.
- The histogram shows the count of people in each age group and the distribution of ages in your dataset. The probability density of the data is presented by the KDE in a smoothed form.

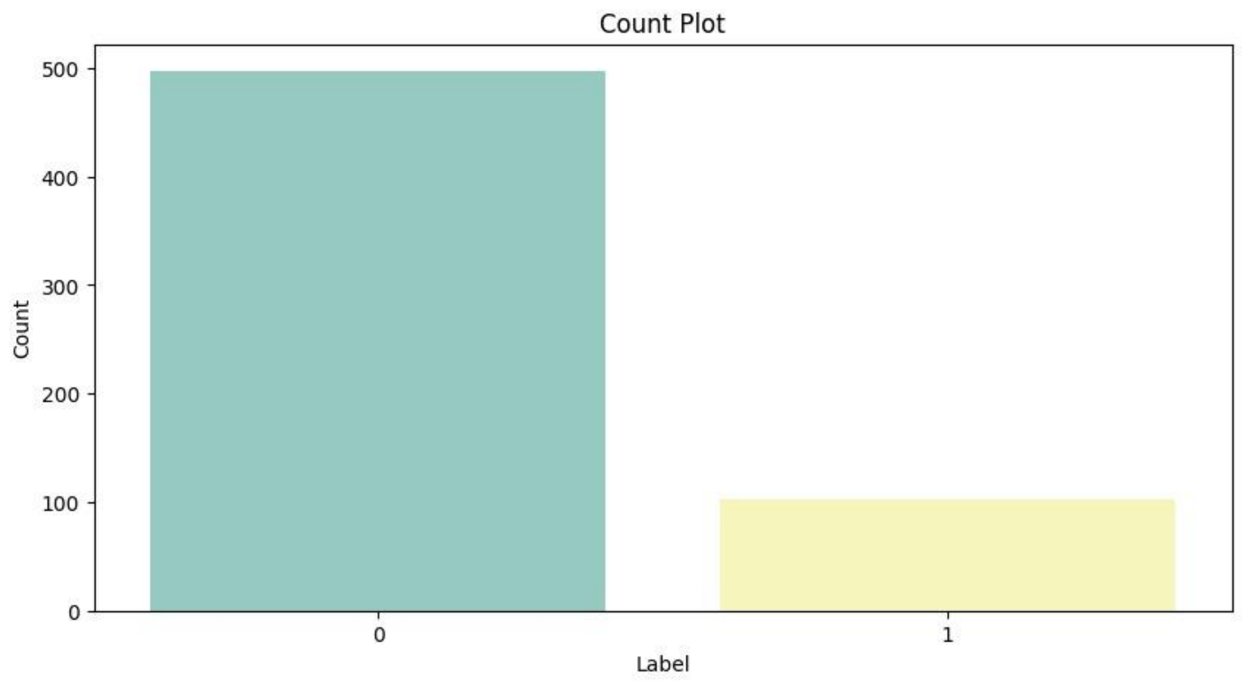
Box Plot



Box Plot:

- sns.boxplot(x='Label', y='likes/post', data=df) creates a box plot that compares the distribution of 'likes/post' for different categories or labels ('Label') in your dataset.
- Box plots allow you to compare the distribution of "likes/post" among various categories by displaying the median, quartiles, and probable outliers for each category.

Count Plot



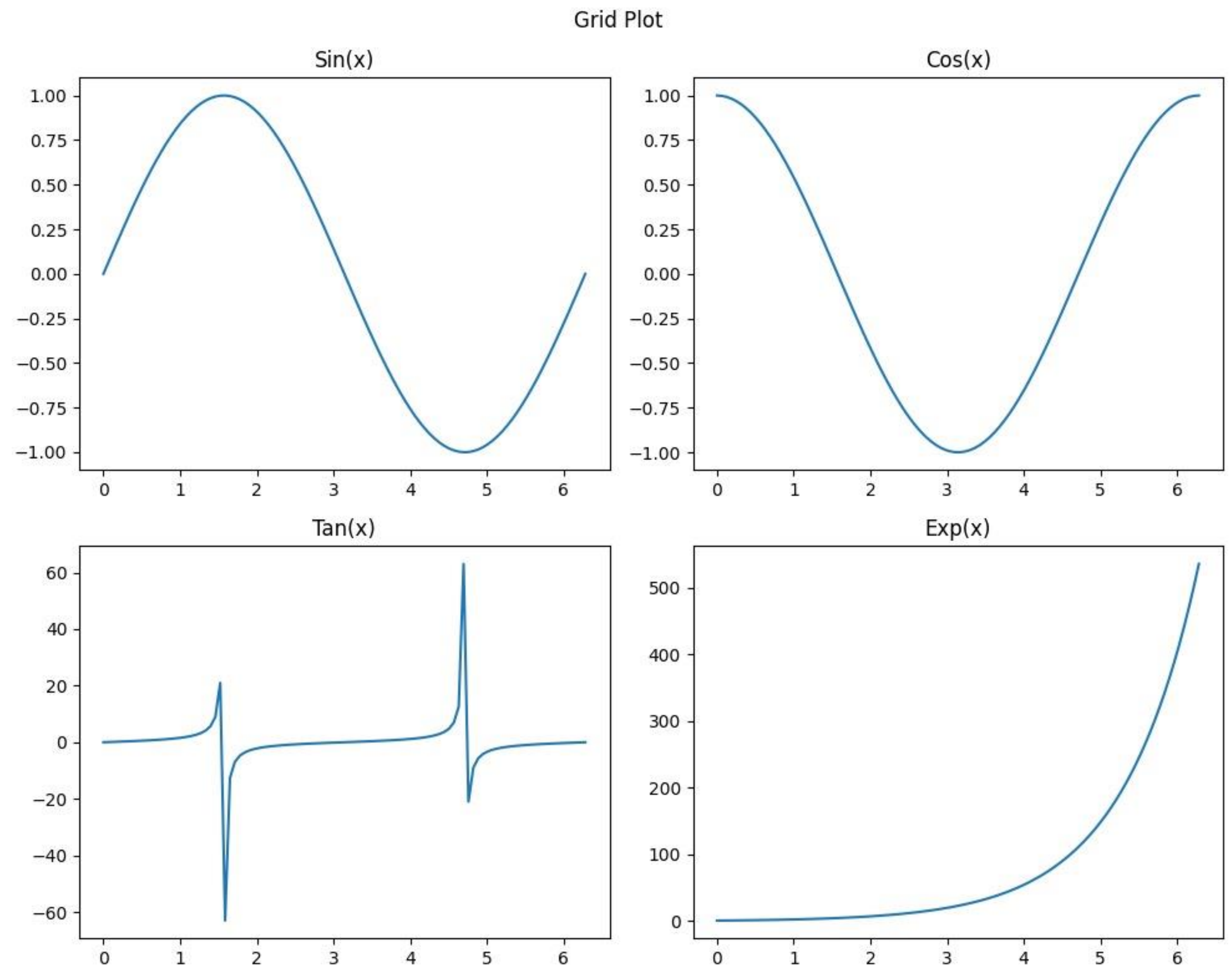
- plt.figure(figsize=(10, 5)) sets the size of the plot to be 10 inches in width and 5 inches in height, adjusting the aspect ratio.
- sns.countplot(x='Label', data=dataset, palette="Set3") creates a count plot using Seaborn. It plots the number of occurrences of each category in the 'Label' column, and the color palette "Set3" is used for coloring the bars.

How to Present the Plot:

The count plot is displayed using plt.show().

This count plot provides a simple means of comprehending the distribution of various categories inside the 'Label' column. You can easily understand the distribution and frequency of each "Label" thanks to the graphic depiction it offers of the number of data points that fall into each category. This kind of visualization is frequently used in exploratory data analysis to understand how categorical data are distributed.

Grid Plot

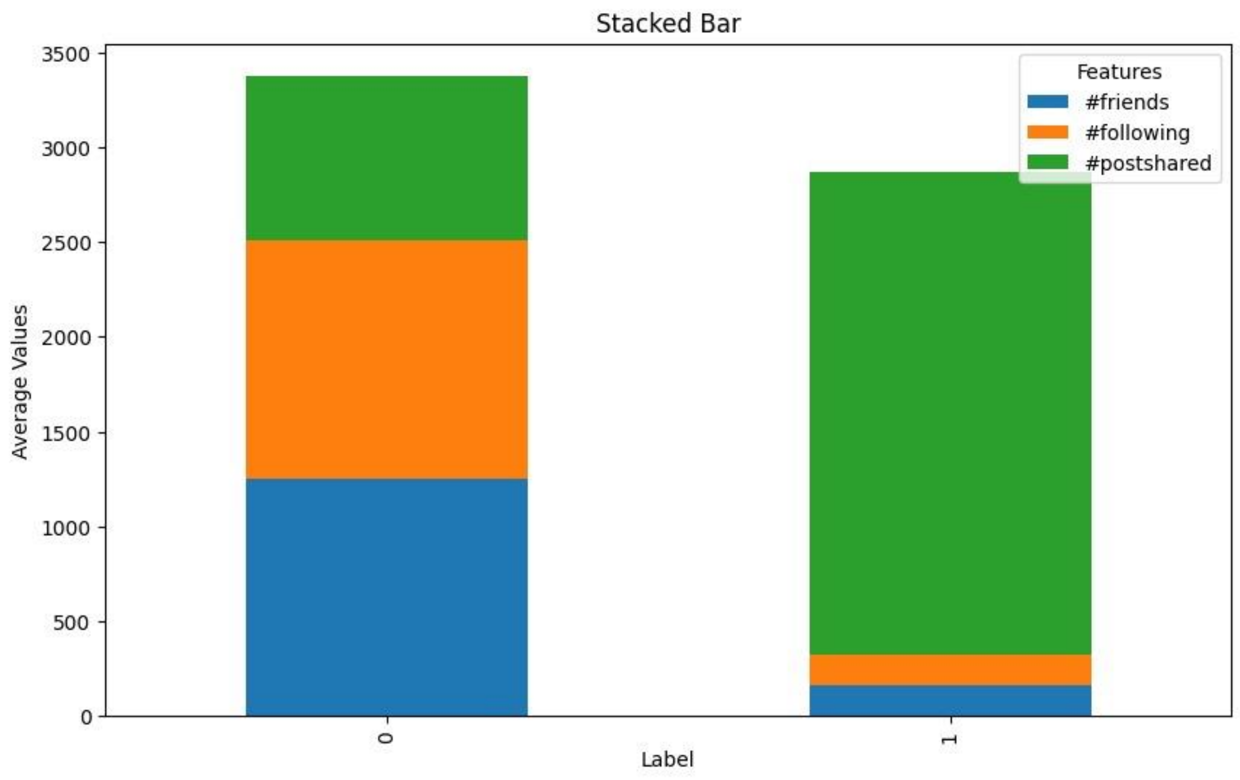


How to Display a Grid Plot:

The grid plot with the subplots' titles is displayed via plt.show().

This grid plot shows the behavior of several functions (including sine, cosine, tangent, and exponential) throughout the specified range of values. It makes it possible to compare these functions and their attributes quickly.

Stacked Bar



- The data in your DataFrame (df) is first grouped by the 'Label' column in the code, and the means of the chosen columns ('#friends', '#following', and '#postshared') are then computed for each group. The average values of these attributes for each of the 'Label' categories are then included in the stacked_data DataFrame.

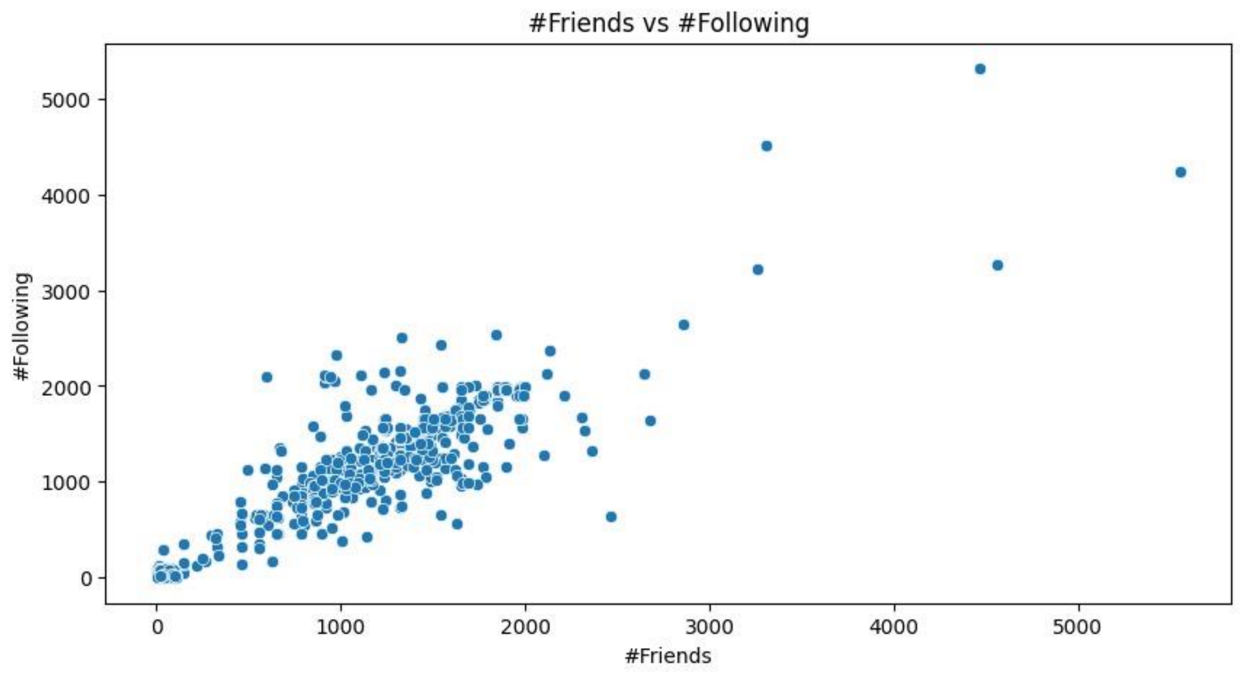
How to Present the Plot:

To display the stacked bar plot, use plt.show().

You may see the average numbers of "friends," "following," and "postshared" for each category or "Label" in your data using the stacked bar visualization. The feature averages for each 'Label' group may be easily compared by stacking the bars on top of one another. This visualization is helpful for exploratory data analysis and presenting summary statistics since it reveals patterns and variations in the chosen attributes across many categories.

Bivariate Analysis of Numerical& Numerical Variables

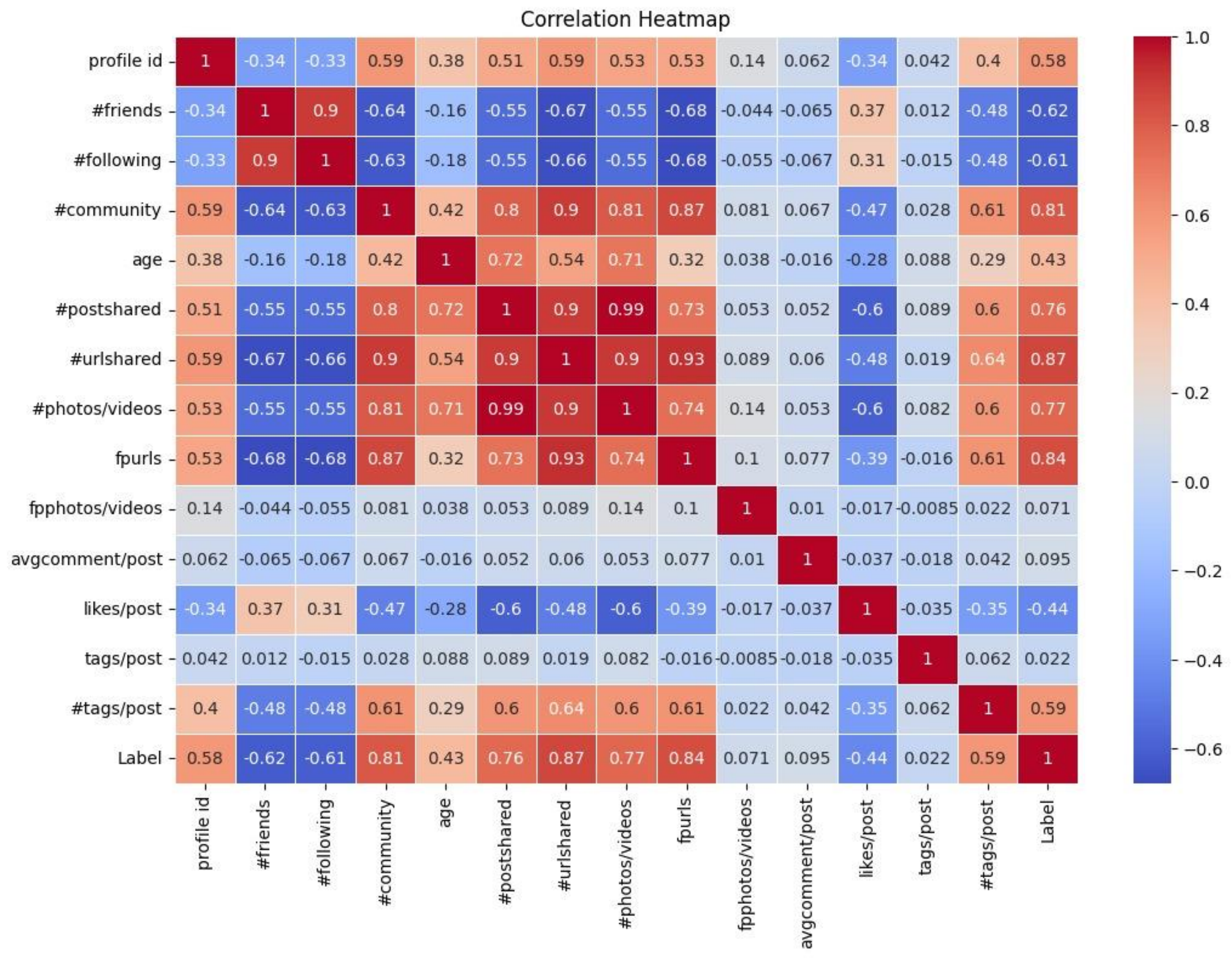
Scatter Plot



Scatter Plot:

- sns.scatterplot(x='#friends', y='#following', data=df) generates a scatter plot with the number of friends (#friends) on the x-axis and the number of people followed (#following) on the y-axis.
- For each data point, this graph aids in visualizing the relationship or correlation between the number of friends and the number of individuals being followed.

Correlation Heatmap



	profile id	#friends	#following	#community	age	#postshared	#urlshared	#photos/videos	fpurls	fpphotos/videos	avgcomment/post	likes/post	tags/post	#tags/post	Label
profile id	1	-0.34	-0.33	0.59	0.38	0.51	0.59	0.53	0.53	0.14	0.062	-0.34	0.042	0.4	0.58
#friends	-0.34	1	0.9	-0.64	-0.16	-0.55	-0.67	-0.55	-0.68	-0.044	-0.065	0.37	0.012	-0.48	-0.62
#following	-0.33	0.9	1	-0.63	-0.18	-0.55	-0.66	-0.55	-0.68	-0.055	-0.067	0.31	-0.015	-0.48	-0.61
#community	0.59	-0.64	-0.63	1	0.42	0.8	0.9	0.81	0.87	0.081	0.067	-0.47	0.028	0.61	0.81
age	0.38	-0.16	-0.18	0.42	1	0.72	0.54	0.71	0.32	0.038	-0.016	-0.28	0.088	0.29	0.43
#postshared	0.51	-0.55	-0.55	0.8	0.72	1	0.9	0.99	0.73	0.053	0.052	-0.6	0.089	0.6	0.76
#urlshared	0.59	-0.67	-0.66	0.9	0.54	0.9	1	0.9	0.93	0.089	0.06	-0.48	0.019	0.64	0.87
#photos/videos	0.53	-0.55	-0.55	0.81	0.71	0.99	0.9	1	0.74	0.14	0.053	-0.6	0.082	0.6	0.77
fpurls	0.53	-0.68	-0.68	0.87	0.32	0.73	0.93	0.74	1	0.1	0.077	-0.39	-0.016	0.61	0.84
fpphotos/videos	0.14	-0.044	-0.055	0.081	0.038	0.053	0.089	0.14	0.1	1	0.01	-0.017	-0.0085	0.022	0.071
avgcomment/post	0.062	-0.065	-0.067	0.067	-0.016	0.052	0.06	0.053	0.077	0.01	1	-0.037	-0.018	0.042	0.095
likes/post	-0.34	0.37	0.31	-0.47	-0.28	-0.6	-0.48	-0.6	-0.39	-0.017	-0.037	1	-0.035	-0.35	-0.44
tags/post	0.042	0.012	-0.015	0.028	0.088	0.089	0.019	0.082	-0.016	-0.0085	-0.018	-0.035	1	0.062	0.022
#tags/post	0.4	-0.48	-0.48	0.61	0.29	0.6	0.64	0.6	0.61	0.022	0.042	-0.35	0.062	1	0.59
Label	0.58	-0.62	-0.61	0.81	0.43	0.76	0.87	0.77	0.84	0.071	0.095	-0.44	0.022	0.59	1

- An effective technique for investigating the connections between numerical characteristics in your dataset is a correlation heatmap. You may use it to rapidly determine which qualities are associated with one another either favorably, unfavorably, or not at all. Warm hues denote strong positive correlations, cold colors denote strong negative correlations, and values near to zero denote weak or no connections.

- In data analysis and machine learning activities, this visualization can help you find patterns in your data, spot multicollinearity (strong correlations between predictor variables), and direct feature selection or dimensionality reduction.

GitHub Repository Link:

- https://github.com/sumanthreddy8910/Phase_4.git