X Data Analysis using API

Nguyen Duc Thanh Vinh - 23229582
Nguyen The Hung - 23229586
Nguyen Phuong Huyen - 23229585

Group 1

Table of Contents

Data Collection

```python
#Set up your Bearer Token
bearer_token = "AAAAAAAAAAAAAAAAAAAAAMgV4wEAAAAA09spRslKr%2BZUrWP0JzmFicMNxYo%3DZDzkxtTIzNn5A7RhwdfBPq1XESws44pYLqDwGZwZ7GeVKVDL6z"
headers = {"Authorization": f"Bearer {bearer_token}"}

#Get user ID from username (no @ symbol)
username = "Google"
user_url = f"https://api.twitter.com/2/users/by/username/{username}"
user_resp = requests.get(user_url, headers=headers).json()
#The .json() part at the end of this line converts the response from the API into a Python dictionary
```

Data checking, save to csv

*limit 100 tweets per time

```python
# Safety check before accessing 'data'
if "data" in user_resp:
    user_id = user_resp["data"]["id"]
# This line accesses the value associated with the key "id", which is inside the key "data" in t

    #Fetch recent tweets
    tweet_url = f"https://api.twitter.com/2/users/{user_id}/tweets"
    params = {
        "max_results": 100,
        "tweet.fields": "created_at,public_metrics,text"
     #Catch the value from the key "tweet.fields" , asking for tweet's creation time (created_at
    }
    tweet_resp = requests.get(tweet_url, headers=headers, params=params).json()

    if "data" in tweet_resp:
        #Load tweets into DataFrame
        df = pd.json_normalize(tweet_resp["data"])
        if "public_metrics.like_count" in df.columns:
            df["created_at"] = pd.to_datetime(df["created_at"])
            df["likes"] = df["public_metrics.like_count"]
            df["retweets"] = df["public_metrics.retweet_count"]
            df["hour"] = df["created_at"].dt.hour

            # Plot average likes by hour
            avg_likes_by_hour = df.groupby("hour")["likes"].mean()
            avg_likes_by_hour.plot(kind="line", marker="o", color="darkblue")
            plt.title(f"Average Likes by Hour of Day ({username})")
            plt.xlabel("Time of Day")
            plt.ylabel("Average Likes")
            plt.grid(True)
            plt.tight_layout()
            plt.show()
        else:
            print("Expected metrics not found in tweet data. Columns available:")
            print(df.columns)

    else:
        print("No tweet data returned. Full response:")
        print(tweet_resp)
else:
    print("User not found or token issue. Full response:")
    print(user_resp)
# Save to CSV in your Google Drive
df.to_csv('/content/drive/MyDrive/google_tweets.csv', index=False)
```

- fabrizio_tweets.csv
- google_tweets.csv
- mcdonald_tweets.csv
- starbucks_tweets.csv
- tesla_tweets.csv
- trump_tweets.csv
- wendys_tweets.csv

Data Processing

- rename ‘hour’ ⟶ ‘posting_time’
- drop two columns: ‘likes’ and ‘retweets’

```
 #   Column
---  ------
 0   text
 1   edit_history_tweet_ids
 2   created_at
 3   id
 4   public_metrics.retweet_count
 5   public_metrics.reply_count
 6   public_metrics.like_count
 7   public_metrics.quote_count
 8   public_metrics.bookmark_count
 9   public_metrics.impression_count
 10  likes
 11  retweets
 12  hour
 13  article.title
```

before

```
 #   Column
---  ------
 0   text
 1   edit_history_tweet_ids
 2   created_at
 3   id
 4   public_metrics.retweet_count
 5   public_metrics.reply_count
 6   public_metrics.like_count
 7   public_metrics.quote_count
 8   public_metrics.bookmark_count
 9   public_metrics.impression_count
 10  posting_time
```

after

Data Processing

removed tweets with ‘public_metrics.impression_count‘ == 0

```
# remove impression_count = 0
tweet_data = tweet_data[tweet_data['public_metrics.impression_count'] > 0]
```

calculate the engagement rate with the following formula:

$$\text{engagement rate} = \frac{\text{total interaction}}{\text{impression count}}$$

with total interaction = sum of reply, like, retweet, quote, bookmark

```
# Calculate engagement_rate
tweet_data['engagement_rate'] = (
    (tweet_data['public_metrics.reply_count'] +
     tweet_data['public_metrics.retweet_count'] +
     tweet_data['public_metrics.like_count'] +
     tweet_data['public_metrics.quote_count'] +
     tweet_data['public_metrics.bookmark_count'])
    / tweet_data['public_metrics.impression_count']
)
```

Hypothesis 1

Research question: How does the posting time (hour of the day) affect the engagement of tweets?

Null Hypothesis (H_0): The posting time (hour of the day) has no significant effect on tweet engagement.

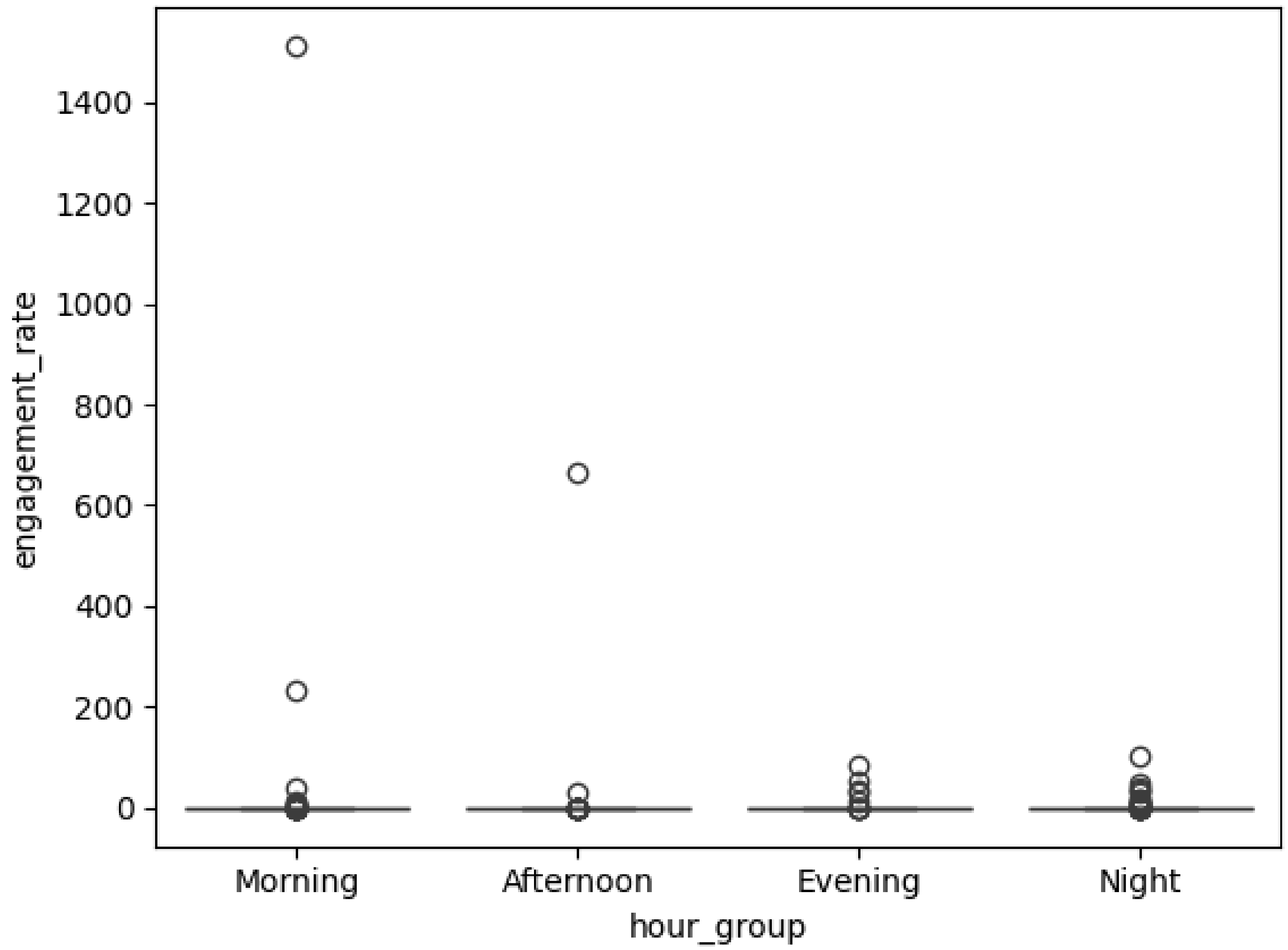
Alternative Hypothesis (H_1): The posting time (hour of the day) significantly improves tweet engagement.

Hypothesis 1

1. group posting_time into four groups: Morning, Afternoon, Evening, Night

```python
# Group posting_time
def categorize_hour(hour):
    if 5 <= hour <= 11:
        return 'Morning'
    elif 12 <= hour <= 17:
        return 'Afternoon'
    elif 18 <= hour <= 22:
        return 'Evening'
    else:
        return 'Night'

tweet_data['hour_group'] = tweet_data['posting_time'].apply(categorize_hour)
```

2. create box plot to check if there are outliers



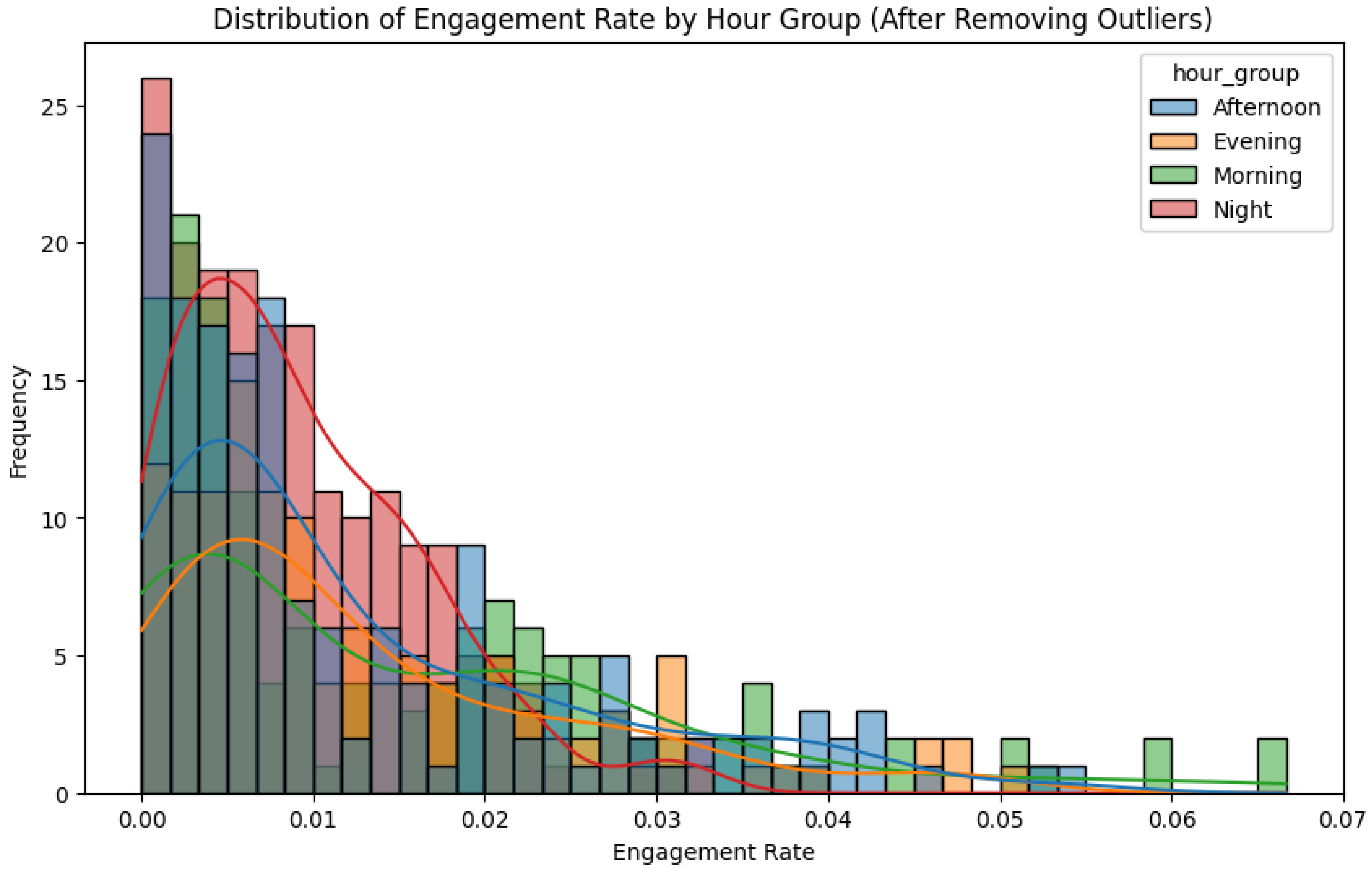
3. remove outliers using IQR method (1.5 x IQR rule)

```python
def remove_outliers_by_group(df, group_col, value_col):
    cleaned_df = pd.DataFrame()

    for group, data in df.groupby(group_col):
        Q1 = data[value_col].quantile(0.25)
        Q3 = data[value_col].quantile(0.75)
        IQR = Q3 - Q1

        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        filtered = data[(data[value_col] >= lower_bound) & (data[value_col] <= upper_bound)]
        cleaned_df = pd.concat([cleaned_df, filtered])

    return cleaned_df
tweet_clean = remove_outliers_by_group(tweet_data, 'hour_group', 'engagement_rate')

print("Before:", len(tweet_data))
print("After:", len(tweet_clean))
```

Hypothesis 1

4. check skewness

	hour_group	original_skew	log_skew
0	Afternoon	1.292876	1.270862
1	Evening	1.283973	1.259993
2	Morning	1.358648	1.320552
3	Night	0.967488	0.949055



after remove outliers and check skewness again --> distribution each groups is not normal --> we decide to use Kruskal-Wallis test (non-parametric tests)

```
stat, p = kruskal(
    *[group['engagement_rate'].values for name, group in tweet_data.groupby('hour_group')]
)
print(f"Kruskal-Wallis test: H={stat:.3f}, p={p:.4f}")

alpha = 0.05
if p < alpha:
    print("Since p-value < 0.05, we reject the null hypothesis.")
    print("There is a statistically significant difference in tweet engagement across posting time groups.")
else:
    print("Since p-value ≥ 0.05, we fail to reject the null hypothesis.")
    print("There is no statistically significant difference in tweet engagement across posting time groups.")
```

Conclusion for hypothesis 1

after running code of Kruskal–Wallis test, this is the result:

```
Kruskal-Wallis test: H=5.631, p=0.1310
```

Conclusion for Hypothesis 1: How does the posting time (hour of the day) affect the engagement of tweets?
Since the p_value > 0.05, we fail to reject the null hypothesis
⟶ This indicates that there is no statistically significant difference in tweet engagement across different posting time groups

Hypothesis 2

Research question: How does the number of bookmarks affect the number of likes on tweets?

Null Hypothesis (H_0): There is no significant difference in the number of likes between tweets with higher and lower bookmark counts.

Alternative Hypothesis (H_1): Tweets with higher bookmark counts have significantly more likes than those with lower bookmark counts.

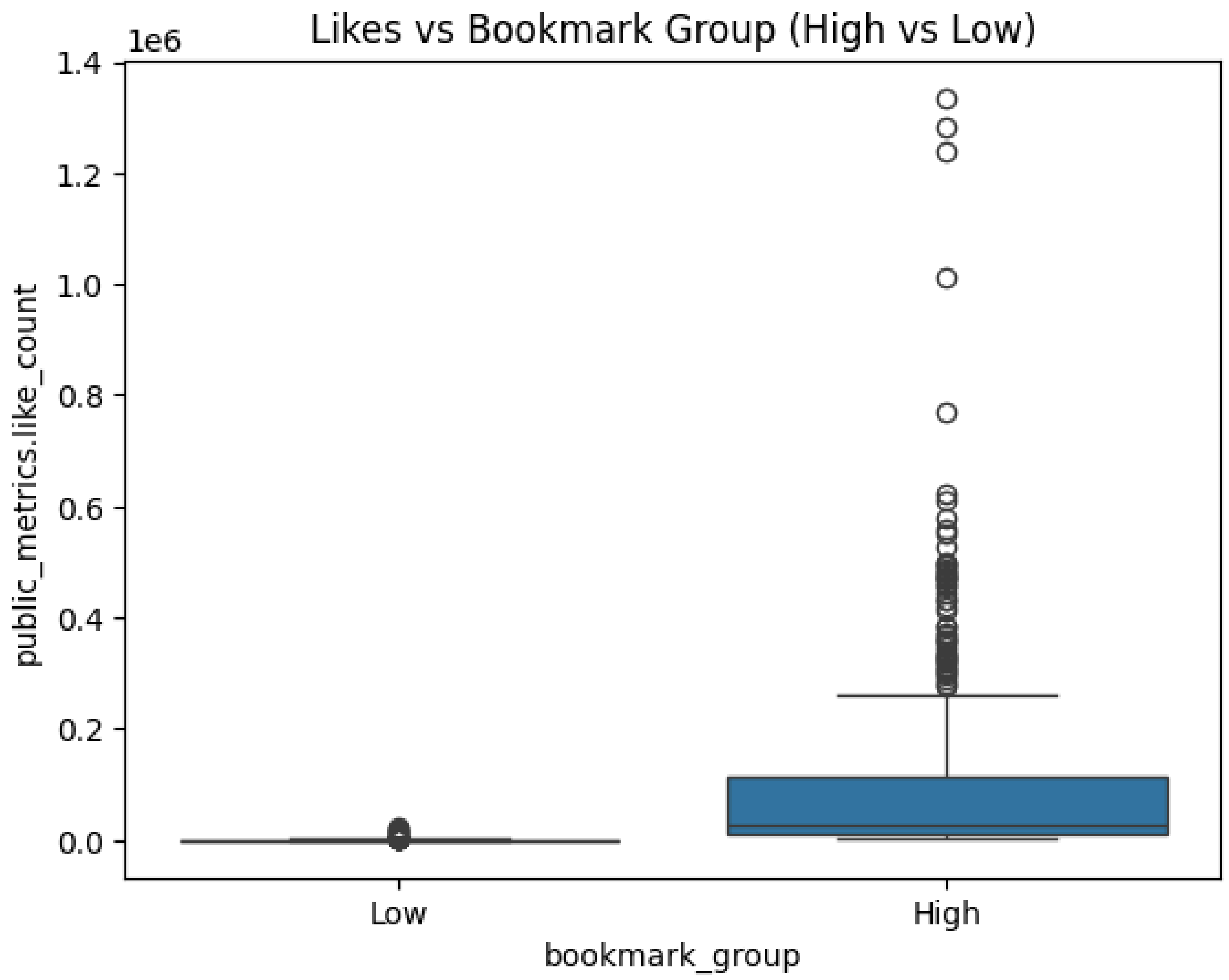
Hypothesis 2

1. divide high and low bookmarks according to median:

```
median_bookmarks = tweet_data['public_metrics.bookmark_count'].median()
tweet_clean['bookmark_group'] = np.where(
    tweet_clean['public_metrics.bookmark_count'] >= median_bookmarks, 'High', 'Low'
)
```

2. create a box plot

tweets with high bookmark counts tend to receive significantly more likes, suggesting a positive relationship between bookmarking and user appreciation.



Hypothesis 2

3. The Mann-Whitney U test was used to compare the distributions of engagement levels between the two groups.

```
# Mann-Whitney U test
high = tweet_clean[tweet_clean['bookmark_group'] == 'High']['public_metrics.like_count']
low = tweet_clean[tweet_clean['bookmark_group'] == 'Low']['public_metrics.like_count']

stat, p = mannwhitneyu(high, low, alternative='greater')

print(f"Mann-Whitney U statistic: {stat:.4f}, p-value: {p:.4e}")

if p < 0.05:
    print("Tweets with higher bookmarks tend to have more likes (significant difference).")
else:
    print("No significant difference in likes between high and low bookmark groups.")
```

Conclusion for hypothesis 2

After running code of Mann - Whitney U test, this is the result:

```
Mann-Whitney U statistic: 93334.0000, p-value: 4.2782e-97
```

Conclusion for Hypothesis 2: How does the number of bookmarks affect the number of likes on tweets?
Since the p_value < 0.05, we reject the null hypothesis
⟶ Tweets with higher bookmarks tend to have more likes (significant difference)

Hypothesis 3

Research question: How does tweet length affect user engagement on Twitter?

- **Null Hypothesis (H_0): Longer tweets do not receive more likes than shorter tweets.**

- **Alternative Hypothesis (H_1): Longer tweets tend to receive more likes than shorter tweets.**

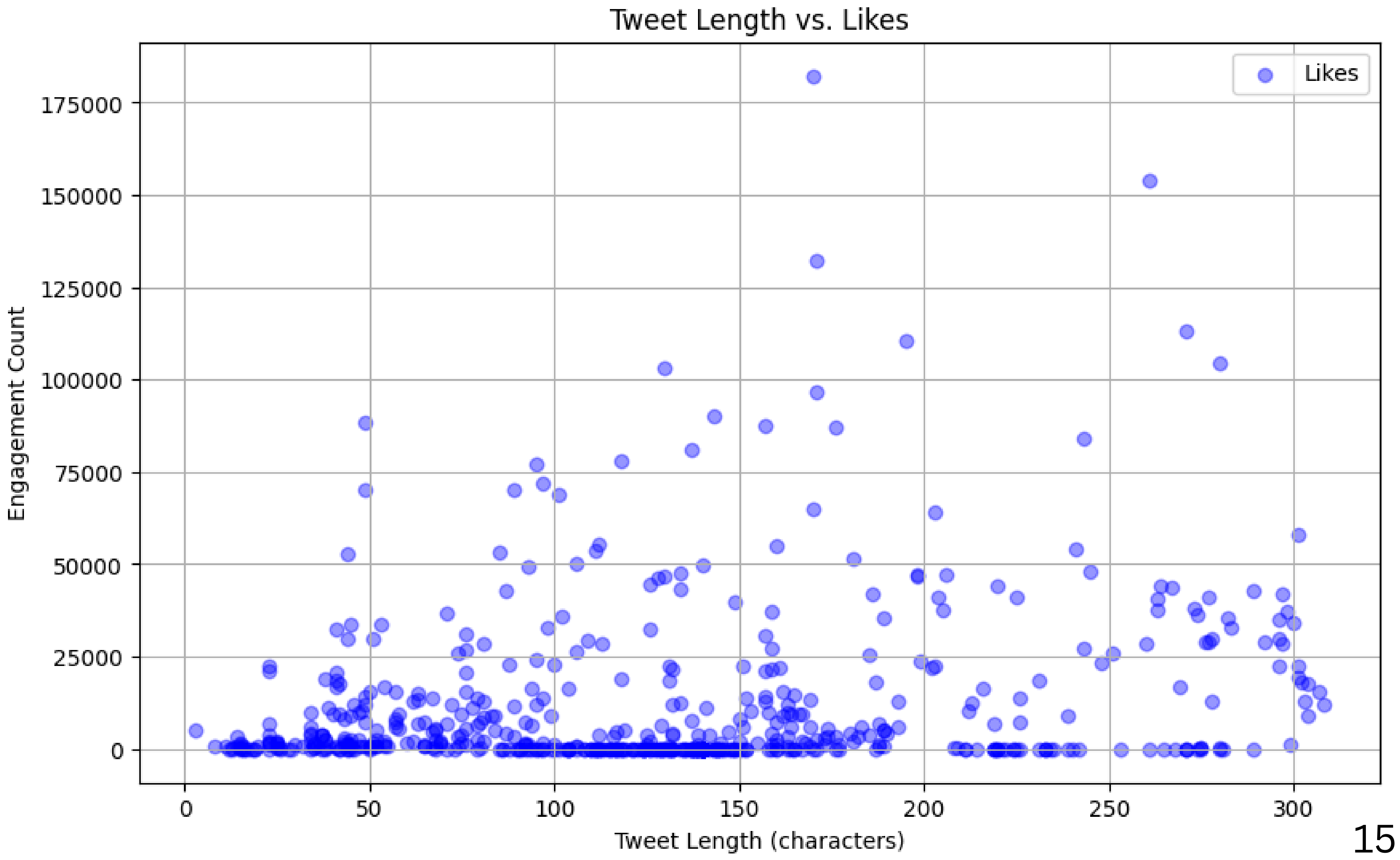
Hypothesis 3

1. add a new column representing of each tweet (in characters)

```
df['tweet_length'] = df['text'].astype(str).apply(len)
```

2. create a scatter plot

tweets with moderate length tend to get higher likes
⟶ use one-tailed test



Hypothesis 3

```
likes_corr = df['tweet_length'].corr(df['likes'])
```

3. check the correlation between tweet length and likes/retweets

```
print(f"Correlation with Likes: {likes_corr:.3f}")
```

Correlation with likes = 0.237
⟶ Longer tweets slightly tend to receive more likes, but the relationship is weak.

4. Split tweets into two groups

- **Short tweets: shorter than or equal to mean**
- **Long tweets: longer than mean**

```
df['tweet_length_mean']=df['tweet_length'].mean()
short_tweets = df[df['tweet_length'] <= df['tweet_length_mean']]
long_tweets = df[df['tweet_length'] > df['tweet_length_mean']]
```

Hypothesis 3

5. Test whether tweet length affects the number of likes by using a <u>one-tailed test</u>

```
from scipy.stats import ttest_ind

t_likes = ttest_ind(long_tweets['likes'], short_tweets['likes'], equal_var=False, alternative ='greater')

print(f"T-test for Likes → t-statistic: {t_likes.statistic:.5f}, p-value: {t_likes.pvalue:.5f}")
```

Conclusion for hypothesis 3

After running the code of t-test, this is the result:

```
T-test for Likes → t-statistic: 3.62649, p-value: 0.00016
```

Since the p-value is less than 0.05, we reject the null hypothesis.

⟶ Longer tweets tend to receive more likes and on average than shorter ones, suggesting that tweet length positively influences user engagement.

References

- Kruskal, W. H., & Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis. Journal of the American Statistical Association, 47(260), 583–621.
- Mann, H. B., & Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. The Annals of Mathematical Statistics, 18, 50–60.
- Yuen, K. K. (1974). The two-sample trimmed t for unequal population variances. Biometrika, 61(1), 165–170.

Thank You