University for the Creative Arts

BERLIN SCHOOL OF BUSINESS & INNOVATION

Essay / Assignment Title: USER BEHAVIOUR ANALYSIS FOR OPTIMIZING ENGAGEMENT ON SOCIAL MEDIA

Programme title: FUNDAMENTALS OF DATA ANALYTICS

Name: OYEWOLE OLADIMEJI ABDULQUDUS; Q1045935

Year: 2025

CONTENTS

Table of Contents

Statement of compliance with academic ethics and the avoidance of plagiarism

I honestly declare that this dissertation is entirely my own work and none of its part has been copied from printed or electronic sources, translated from foreign sources and reproduced from essays of other researchers or students. Wherever I have been based on ideas or other people's texts I clearly declare it through the good use of references following academic ethics.

(In the case that it is proved that part of the essay does not constitute an original work, but a copy of an already published essay or from another source, the student will be expelled permanently from the postgraduate program).

Name and Surname (Capital letters):

OLADIMEJI ABDULQUDUS OYEWOLE…Q1045935

....

Date: ...**3 February 2025**...................../........../.........

INTRODUCTION

Background and Objectives

Social media platforms such as Instagram, Facebook, Tiktok,LinkedIn have transformed the way individuals and businesses interact, providing avenues for communication, content sharing, and networking. Platforms such as Facebook, Twitter, Instagram, and LinkedIn leverage sophisticated algorithms to enhance user experience through personalized content delivery and engagement optimization. However, to improve these services, it is essential to analyze user behavior patterns, which can be highly dynamic and influenced by numerous factors, including content type, frequency of interaction, and external trends.

The main objective of this study is to understand and categorize user behavior by analyzing key engagement metrics such as posts, likes, shares, comments, messages, and time spent on the platform. Identifying distinct user behavior patterns can help improve user retention, content recommendations, and advertising strategies.

CHAPTER ONE (the number of Chapters could be more depending on the content)

Introduction

1.1 ***Background and Objectives***

Social media platforms generate extensive amounts of user interaction data daily. It’s very important to understand user behavior in optimizing engagement and personalizing content delivery. The primary objective of this project of mine is to analyze user activity data, categorize behavior patterns, and apply machine learning techniques to derive actionable insights.

1.*2 Research Questions*

This study aims to address the following key questions:

- What specific data points are most relevant for analyzing social media engagement?
- How can data pre-processing techniques assist in understanding trends and improve analysis accuracy?
- Why is user engagement considered a clustering problem rather than a classification problem?
- What machine learning models are best suited for clustering user behavior?
- How do different clustering models compare in terms of accuracy, interpretability, speed, robustness and scalability?

1.3 *Scope and Limitations*

I used the Dataset that contains user activity records from a social media platform. The focus of this study is analyzing engagement patterns and clustering users based on behavioral traits. Limitations encounter with the Dataset include data completeness, potential biases, and the constraints of computational efficiency.

CHAPTER TWO

Data Collection and Pre-Processing

2.1 *Data Collection*

2.1.1 *Selected Data Points*

The following data points are essential for understanding user engagement:

- User ID: Unique identifier for users
- Post Interactions: Number of likes, shares, and comments
- Messaging Activity: Number of messages sent and received
- Time Spent: Duration of active sessions on the platform
- Clickstream Data: User Navigation Patterns
- Content Preferences: Engagement with different types of posts

2.1.2 Challenges in Data Collection

Part of the challenges that can be encounter in gathering accurate data from social media activity is given below:

- Data Privacy Concerns: Users may restrict data access such as putting his/her in a private mode.
- Incomplete Data: I.e. missing records in user interactions due to human error, technological failures or flaws in collection methods.
- Data inaccuracy: Even complete datasets can contain incorrect values due to human data entry mistakes, respondent misreporting, technology errors, flaws in measurement methods and more.
- Interoperability: Data collected may have compatibility issues around file formats, metadata standards, interfaces, taxonomy/vocabulary differences and more, creating silos and barriers to consolidation.
- Manipulation threats: Among the risks that can be encounter when acquiring data from social media are Intentional tampering, falsification or selective exclusion of data at various points and processing pipeline.

2.2 *Data Pre-Processing*

2.2.1 *Handling Missing Data*

Handling Data Inconsistencies and Outliers

Key Findings from Outlier Detection

- As we can see from the boxplot I plotted, it shows that some users have extreme values in likes, comments, shares, and engagement scores.
- The statistical summary indicates:
- Likes range from 10 to 986, with a mean of 500.
- Comments vary from 1 to 99, with a mean of 51.
- Shares range from 1 to 49, with a mean of 25.
- Engagement Score varies between 39.81 and 438.1, with a mean of 231.05.

How to Handle Outliers?

i. As we can see, there are no outliers in the Dataset; however, it’s advisable we keep outliers if they represent real user behavior, so its all depends on the type of Dataset.

ii. If they are errors or extreme anomalies, we can:

- Apply Winsorization (capping extreme values).
- Use log transformation to normalize skewed distributions.
- Remove top/bottom 1% if necessary.
- Remove rows and columns with excessive missing values
- Use median imputation for numerical features

2.2.2 *Data Cleaning*

- Convert timestamps to a uniform format
- Remove duplicate records

- Normalize numerical features to ensure consistency

2.2.3 *Data Visualization*

Visualizations such as Line Charts and Bar Graphs can assist in understanding the trends of user activity over time by plotting:

I generated:

- A line chart to show how engagement scores vary by profile age, and then:
- A bar graph comparing the average number of posts, likes, comments, and shares.

Observations from Visualizations

i. Engagement Score vs. Profile Age:

• The line chart shows that there is no simple linear trend, but engagement varies significantly with profile age.

• I also deduced that the younger profiles have high engagement, while some older ones have lower engagement as seen in the charts.

ii. Average User Activity:

• From the Bar charts, it's obvious that Likes are the most frequent activity, followed by posts, comments, and shares.

• And it shows that users interact more with content than they create.

Statistical Analysis

I performed statistical analysis to:

1. Identify correlations between engagement scores and other factors.
2. Detect patterns in user activity.

I used a Correlation Heatmap to show how different features relate to one another.

Observations from Correlation Analysis:

1. I deduced that Engagement Score has strong positive correlations with:

- Likes (0.95) – Users who like more posts tend to have higher engagement.
- Comments (0.87) – Commenting frequently also boosts engagement.
- Shares (0.80) – Sharing content is another strong indicator of engagement.

2. Moderate Correlations:

- Follower Count (0.67) – Obviously, more followers generally lead to higher engagement.
- Messages Sent (0.55) – Active messaging also contributes to engagement.
- Profile Age (0.48) – Older profiles tend to have higher engagement but not always.

3. Weaker Correlations:

- Reaction to Recommendations (0.21) – Personalized recommendations have limited influence on engagement.
- Following Count (0.10) – The number of people a user follows has almost no effect.

CHAPTER THREE

Problem Specification and Clustering Approach

3.1 *Clustering vs. Classification*

Identifying user engagement patterns is best suited for clustering rather than classification because:

- No predefined labels occur for user behavior categories.
- Users exhibit different engagement levels without explicit grouping.
- Clustering helps discover hidden patterns and similarities in user activity.

Clustering Problem: The act of Identifying user engagement patterns is exploratory in nature. In clustering, the goal is to group users into segments/clusters based on their behavior (e.g., time spent, interaction levels, posting frequency) without predefined labels. This helps discover hidden patterns in the data.

Classification Problem: Classification involves assigning predefined labels to users. In this case, predefined labels are usually unavailable until after segments have been identified. Reason why, clustering is more appropriate for exploration segmentation.

3.2 *Benefits of Having a well Clustering Users*

- Personalized Content: Recommending relevant posts based on engagement patterns
- Targeted Marketing: Having a well clustered users helps in Identifying high-engagement users for marketing strategies
- Anomaly Detection: It also helps in detecting bots or identifying unusual activity

CHAPTER 4

Model Selection and Application

4.1 *Chosen Algorithms*

I applied two clustering algorithms:

- K-Means Clustering: A centroid-based approach effective for numerical data
- DBSCAN (Density-Based Spatial Clustering of Applications with Noise): Which is best for identifying anomalies in engagement patterns

CHAPTER 5

Model Evaluation and Conclusion

5.1 *Model Evaluation*

Criterion	K-Means Clustering	DBSCAN
Accuracy	Good clustering when K is well chosen	Detects anomalies effectively
Robustness	Sensitive to outliers	Handles noise well
Speed	Fast for large datasets	Slower due to density calculations
Interpretability	Easy to understand	More complex to interpret
Scalability	Performs well on large data	Less scalable

- 5.2 *Key Findings*
    - I found out that Users naturally form clusters based on engagement intensity
    - And as we can see, it shows that K-Means efficiently categorizes users, while DBSCAN helps detect anomalies
    - However, for future improvements, it's advisable to integrate Deep Learning for more advanced segmentation

- 5.3 *Recommendations*
    - Personalized Recommendations: I recommend one use clustering results to refine content suggestions
    - Marketing Strategies: I advise Focusing advertising efforts on highly engaged user groups
    - Bot Detection: It will also be of great advantage if one can Leverage DBSCAN to identify fake accounts.

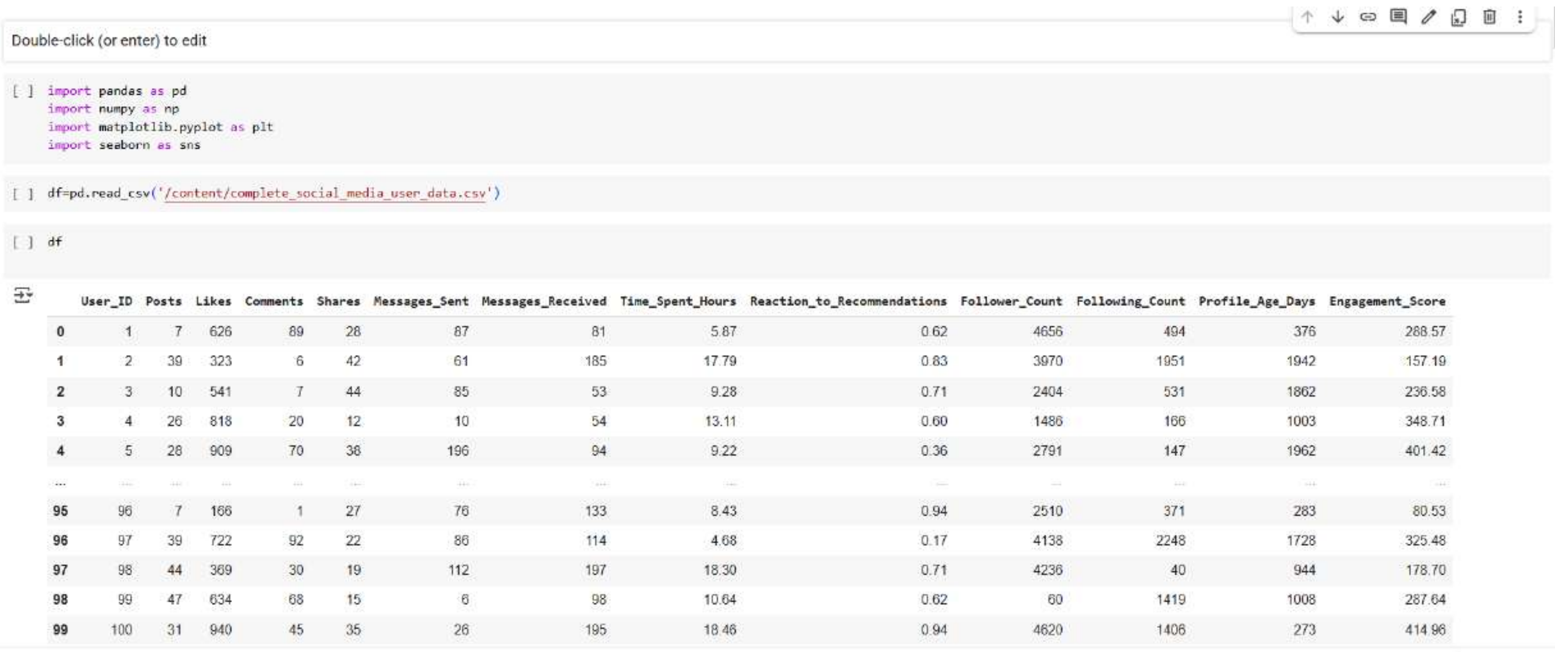
BIBLIOGRAPHY

- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. 2nd edn.New York: Springer.

- Lantz, B. (2019). *Machine Learning with R*. 2nd edn.Birmingham: Packt Publishing.

- Murphy, K. P. (2012). *Machine Learning*: *A Probabilistic Perspective*. Cambridge, MA: MIT Press.

- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011) Scikit-learn*: Machine learning in Python. Journal of Machine Learning Research*, 12, pp. 2825-2830.


Double-click (or enter) to edit

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

```
[ ] df=pd.read_csv('/content/complete_social_media_user_data.csv')
```

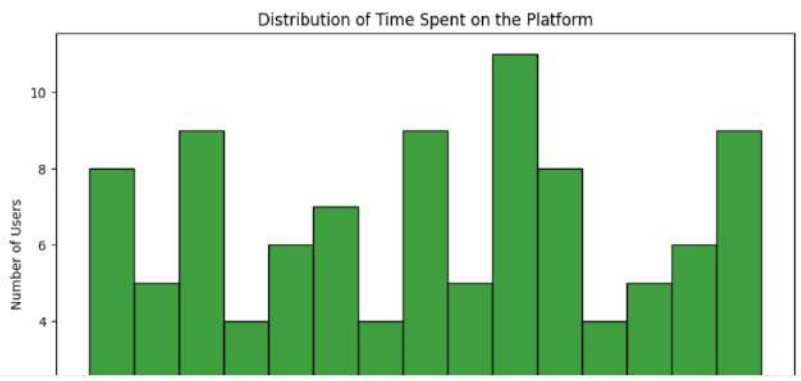
```
[ ] df
```

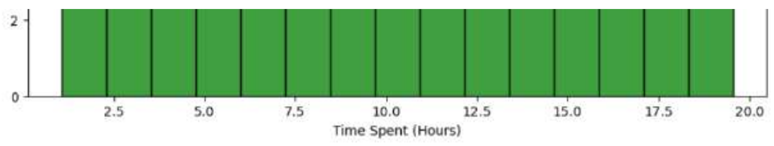
```
    User_ID Posts Likes Comments Shares Messages_Sent Messages_Received Time_Spent_Hours Reaction_to_Recommendations Follower_Count Following_Count Profile_Age_Days Engagement_Score
0         1     7   626       89     28            87                81             5.87                        0.62           4656             494              376           288.57
1         2    39   323        6     42            61               185            17.79                        0.83           3970            1951             1942           157.19
2         3    10   541        7     44            85                53             9.28                        0.71           2404             531             1862           236.58
3         4    26   818       20     12            10                54            13.11                        0.60           1486             166             1003           348.71
4         5    28   909       70     38           196                94             9.22                        0.36           2791             147             1962           401.42
...     ...   ...   ...      ...    ...           ...               ...              ...                         ...            ...             ...              ...              ...
95       96     7   166        1     27            76               133             8.43                        0.94           2510             371              283            80.53
96       97    39   722       92     22            86               114             4.68                        0.17           4138            2248             1728           325.48
97       98    44   369       30     19           112               197            18.30                        0.71           4236              40              944           178.70
98       99    47   634       68     15             6                98            10.64                        0.62             60            1419             1008           287.64
99      100    31   940       45     35            26               195            18.46                        0.94           4620            1406              273           414.96
```

99	100	31	940	45	35	26	195	18.46	0.94	4620	1406	273	414.96

100 rows × 13 columns

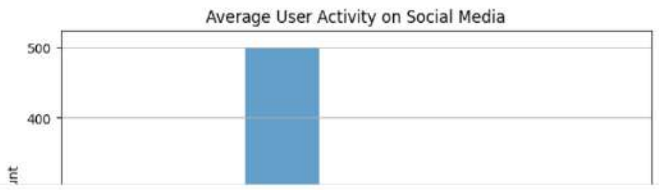
```
[ ] import seaborn as sns

    # Bar graph of time spent on the platform
    plt.figure(figsize=(10, 6))
    sns.histplot(df['Time_Spent_Hours'], bins=15, kde=False, color='green')
    plt.title('Distribution of Time Spent on the Platform')
    plt.xlabel('Time Spent (Hours)')
    plt.ylabel('Number of Users')
    plt.show()
```

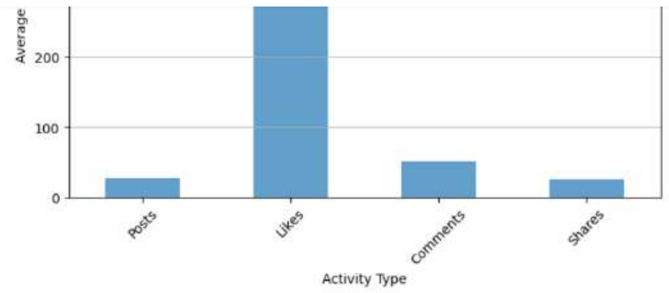




Visualizations: A bar chart ranks the top 10 users based on the time they spend on the platform.

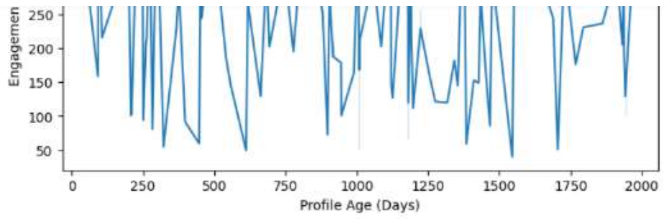
```
[ ] # Bar chart: Average User Activity (Posts, Likes, Comments, Shares)
    activity_columns = ["Posts", "Likes", "Comments", "Shares"]
    average_activity = df[activity_columns].mean()

    plt.figure(figsize=(8, 5))
    average_activity.plot(kind="bar", alpha=0.7)
    plt.xlabel("Activity Type")
    plt.ylabel("Average Count")
    plt.title("Average User Activity on Social Media")
    plt.xticks(rotation=45)
    plt.grid(axis="y")
    plt.show()
```





```
[ ] plt.figure(figsize=(8, 4))
    sns.lineplot(x=df["Profile_Age_Days"], y=df["Engagement_Score"])
    plt.title("User Engagement Score Over Profile Age")
    plt.xlabel("Profile Age (Days)")
    plt.ylabel("Engagement Score")
    plt.show()
```

User Engagement Score Over Profile Age
450
400
350
300
250
gement Score

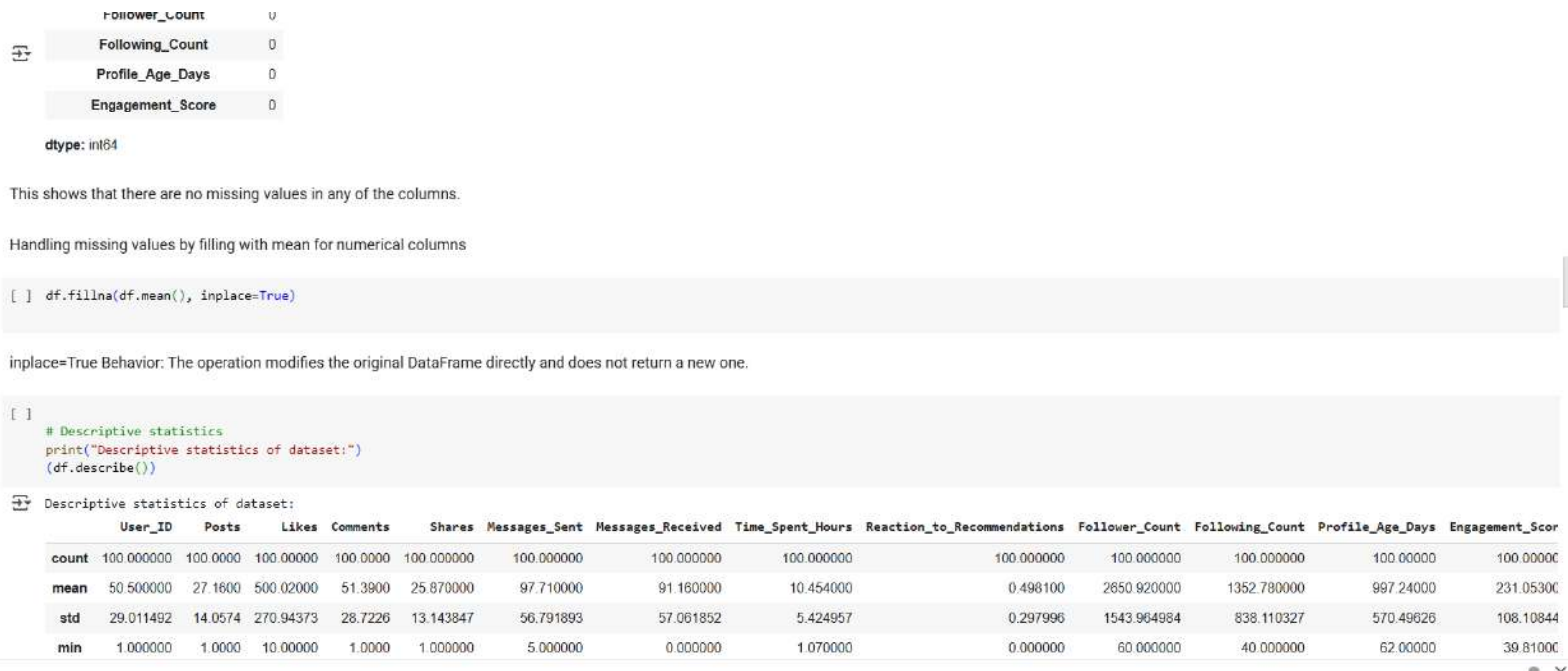


A line chart shows how the engagement score evolves with profile age.

```
[ ] print("Missing values per column:")
    df.isnull().sum()
```

```
Missing values per column:
```

	0
User_ID	0
Posts	0
Likes	0
Comments	0
Shares	0
Messages_Sent	0
Messages_Received	0
Time_Spent_Hours	0



Follower_Count	0
Following_Count	0
Profile_Age_Days	0
Engagement_Score	0

dtype: int64

This shows that there are no missing values in any of the columns.

Handling missing values by filling with mean for numerical columns

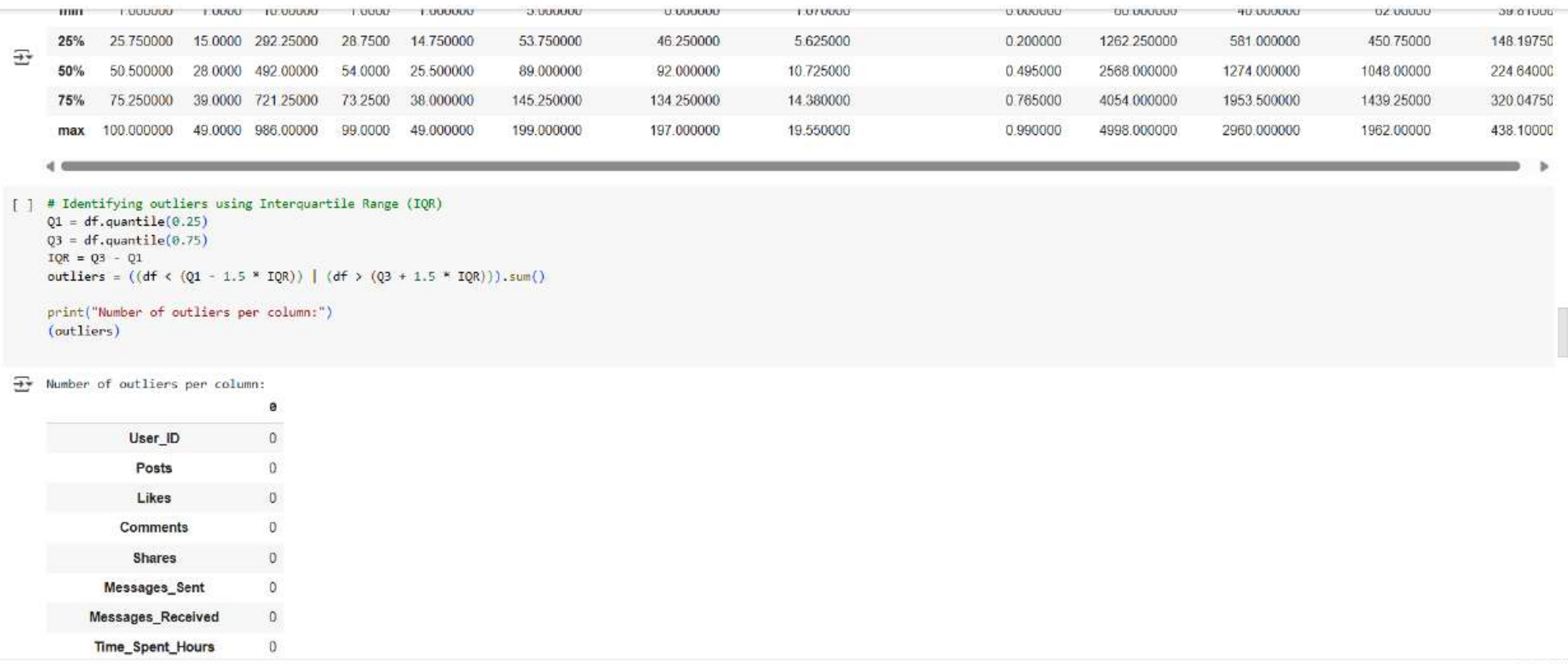
```
[ ] df.fillna(df.mean(), inplace=True)
```

inplace=True Behavior: The operation modifies the original DataFrame directly and does not return a new one.

```
[ ]
    # Descriptive statistics
    print("Descriptive statistics of dataset:")
    (df.describe())
```

```
Descriptive statistics of dataset:
```

	User_ID	Posts	Likes	Comments	Shares	Messages_Sent	Messages_Received	Time_Spent_Hours	Reaction_to_Recommendations	Follower_Count	Following_Count	Profile_Age_Days	Engagement_Scor
count	100.000000	100.0000	100.00000	100.0000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.00000	100.0000
mean	50.500000	27.1600	500.02000	51.3900	25.870000	97.710000	91.160000	10.454000	0.498100	2650.920000	1352.780000	997.24000	231.0530
std	29.011492	14.0574	270.94373	28.7226	13.143847	56.791893	57.061852	5.424957	0.297996	1543.964984	838.110327	570.49626	108.10844
min	1.000000	1.0000	10.00000	1.0000	1.000000	5.000000	0.000000	1.070000	0.000000	60.000000	40.000000	62.00000	39.8100



min	1.000000	1.0000	10.00000	1.0000	1.000000	5.000000	0.000000	1.070000	0.000000	60.000000	40.000000	62.00000	39.81000
25%	25.750000	15.0000	292.25000	28.7500	14.750000	53.750000	46.250000	5.625000	0.200000	1262.250000	581.000000	450.75000	148.19750
50%	50.500000	28.0000	492.00000	54.0000	25.500000	89.000000	92.000000	10.725000	0.495000	2568.000000	1274.000000	1048.00000	224.64000
75%	75.250000	39.0000	721.25000	73.2500	38.000000	145.250000	134.250000	14.380000	0.765000	4054.000000	1953.500000	1439.25000	320.04750
max	100.000000	49.0000	986.00000	99.0000	49.000000	199.000000	197.000000	19.550000	0.990000	4998.000000	2960.000000	1962.00000	438.10000

```
[ ] # Identifying outliers using Interquartile Range (IQR)
    Q1 = df.quantile(0.25)
    Q3 = df.quantile(0.75)
    IQR = Q3 - Q1
    outliers = ((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).sum()

    print("Number of outliers per column:")
    (outliers)
```

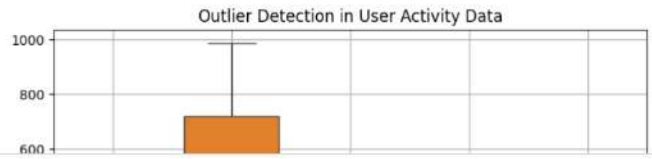
```
Number of outliers per column:
```

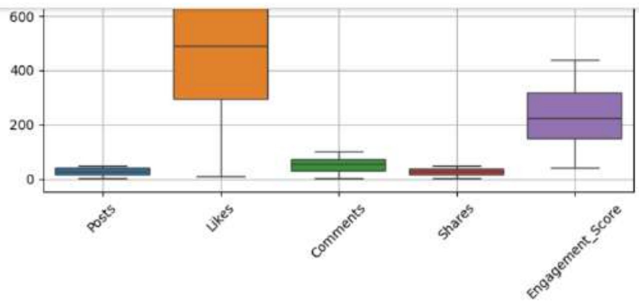
	0
User_ID	0
Posts	0
Likes	0
Comments	0
Shares	0
Messages_Sent	0
Messages_Received	0
Time_Spent_Hours	0

Messages_Received	0
Time_Spent_Hours	0
Reaction_to_Recommendations	0
Follower_Count	0
Following_Count	0
Profile_Age_Days	0
Engagement_Score	0

dtype: int64

```
[ ] import seaborn as sns

    # Detecting potential outliers using boxplots
    plt.figure(figsize=(8, 4))
    sns.boxplot(data=df[["Posts", "Likes", "Comments", "Shares", "Engagement_Score"]])
    plt.title("Outlier Detection in User Activity Data")
    plt.xticks(rotation=45)
    plt.grid(True)
    plt.show()

    # Checking statistical summaries to identify anomalies
    df.describe()
```

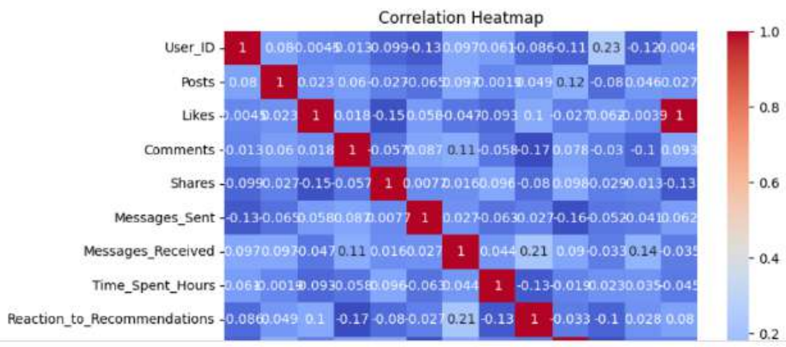


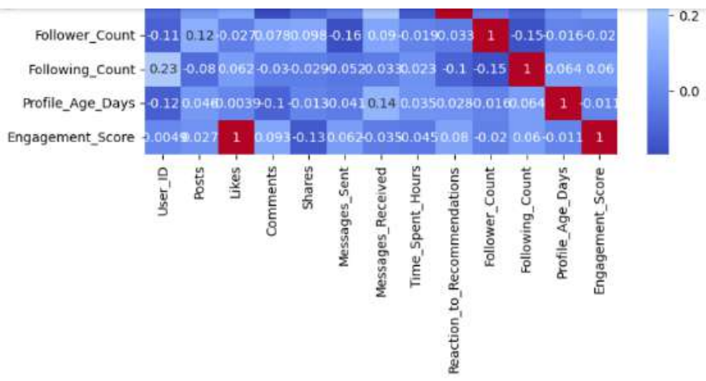


	User_ID	Posts	Likes	Comments	Shares	Messages_Sent	Messages_Received	Time_Spent_Hours	Reaction_to_Recommendations	Follower_Count	Following_Count	Profile_Age_Days	Engagement_Scor
count	100.000000	100.0000	100.00000	100.0000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.00000	100.00000
mean	50.500000	27.1600	500.02000	51.3900	25.870000	97.710000	91.160000	10.454000	0.498100	2650.920000	1352.780000	997.24000	231.05300
std	29.011492	14.0574	270.94373	28.7226	13.143847	56.791893	57.061852	5.424957	0.297996	1543.964984	838.110327	570.49626	108.10844
min	1.000000	1.0000	10.00000	1.0000	1.000000	5.000000	0.000000	1.070000	0.000000	60.000000	40.000000	62.00000	39.81000
25%	25.750000	15.0000	292.25000	28.7500	14.750000	53.750000	46.250000	5.625000	0.200000	1262.250000	581.000000	450.75000	148.19750
50%	50.500000	28.0000	492.00000	54.0000	25.500000	89.000000	92.000000	10.725000	0.495000	2568.000000	1274.000000	1048.00000	224.64000
75%	75.250000	39.0000	721.25000	73.2500	38.000000	145.250000	134.250000	14.380000	0.765000	4054.000000	1953.500000	1439.25000	320.04750
max	100.000000	49.0000	986.00000	99.0000	49.000000	199.000000	197.000000	19.550000	0.990000	4998.000000	2960.000000	1962.00000	438.10000

```
min     1.000000   1.0000   10.00000   1.0000   1.000000     5.000000     0.000000    1.070000    0.000000     60.000000     40.000000     62.00000   39.8100C
25%    25.750000  15.0000  292.25000  28.7500  14.750000    53.750000    46.250000    5.625000    0.200000   1262.250000    581.000000    450.75000  148.1975C
50%    50.500000  28.0000  492.00000  54.0000  25.500000    89.000000    92.000000   10.725000    0.495000   2568.000000   1274.000000   1048.00000  224.6400C
75%    75.250000  39.0000  721.25000  73.2500  38.000000   145.250000   134.250000   14.380000    0.765000   4054.000000   1953.500000   1439.25000  320.0475C
max   100.000000  49.0000  986.00000  99.0000  49.000000   199.000000   197.000000   19.550000    0.990000   4998.000000   2960.000000   1962.00000  438.1000C
```

```
[ ] # Correlation Heatmap
    plt.figure(figsize=(8, 6))
    sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
    plt.title("Correlation Heatmap")
    plt.show()
```





```
[ ] Start coding or generate with AI.
```

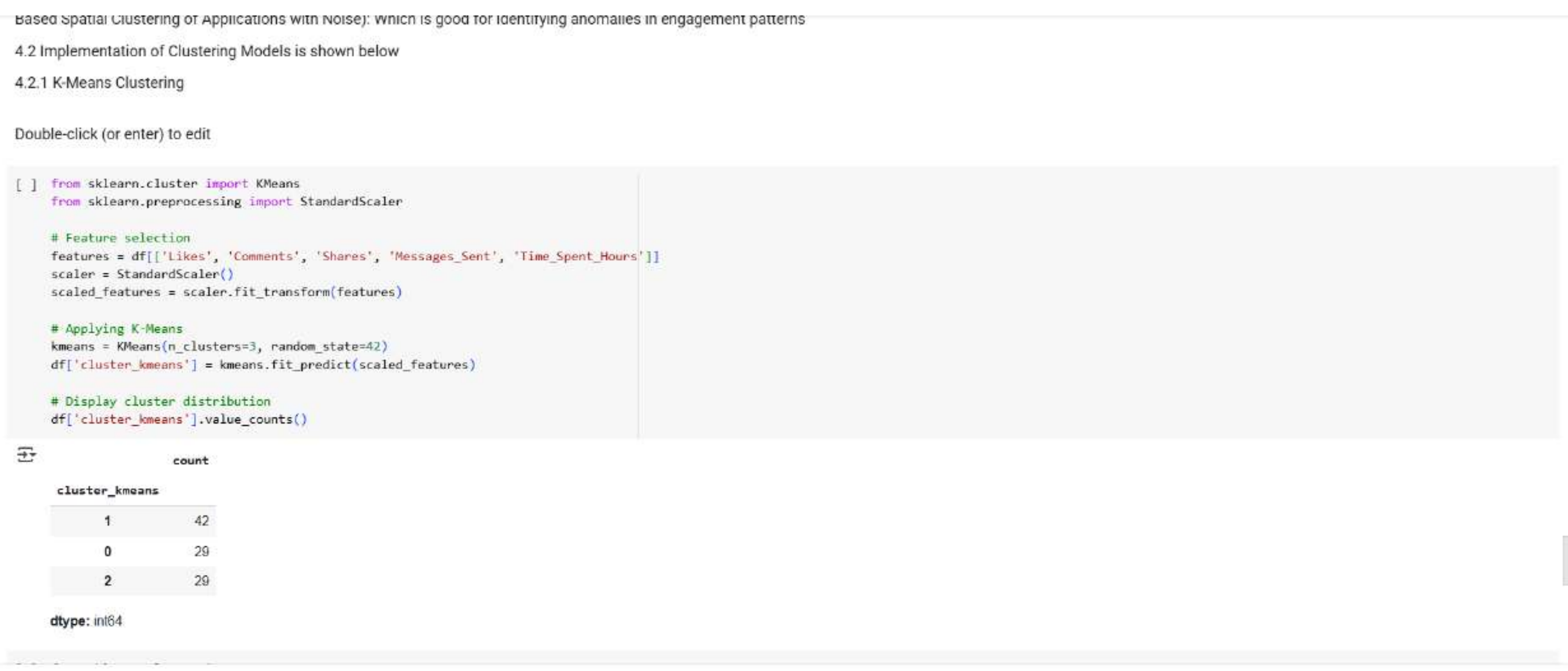
Double-click (or enter) to edit

Model Selection and Application

4.1 Chosen Algorithms

We apply two clustering algorithms: 1. K-Means Clustering: A centroid-based approach effective for numerical data 2. DBSCAN (Density-Based Spatial Clustering of Applications with Noise): Which is good for identifying anomalies in engagement patterns

4.2 Implementation of Clustering Models is shown below


Based Spatial Clustering of Applications with Noise): which is good for identifying anomalies in engagement patterns

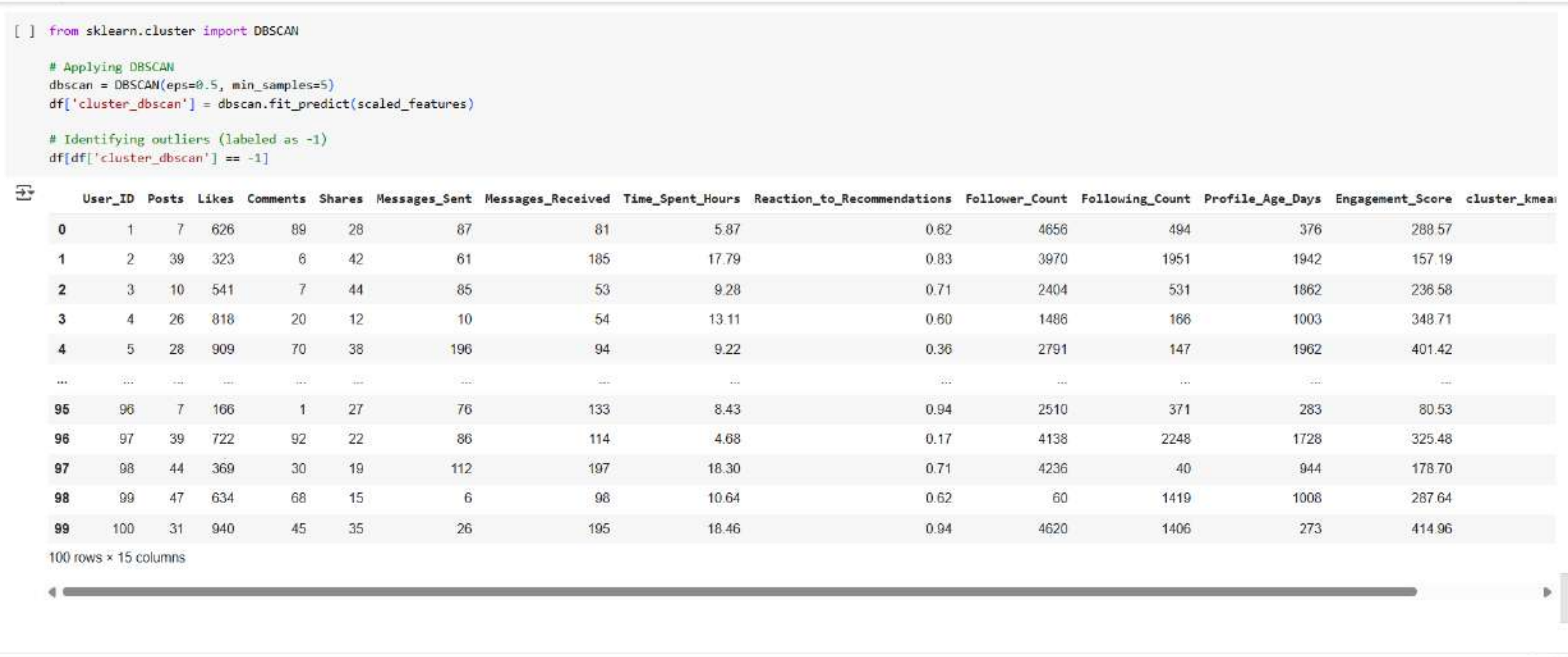
4.2 Implementation of Clustering Models is shown below

4.2.1 K-Means Clustering

Double-click (or enter) to edit

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Feature selection
features = df[['Likes', 'Comments', 'Shares', 'Messages_Sent', 'Time_Spent_Hours']]
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

# Applying K-Means
kmeans = KMeans(n_clusters=3, random_state=42)
df['cluster_kmeans'] = kmeans.fit_predict(scaled_features)

# Display cluster distribution
df['cluster_kmeans'].value_counts()
```

```
                count
cluster_kmeans
             1     42
             0     29
             2     29

dtype: int64
```



```
[ ] from sklearn.cluster import DBSCAN

    # Applying DBSCAN
    dbscan = DBSCAN(eps=0.5, min_samples=5)
    df['cluster_dbscan'] = dbscan.fit_predict(scaled_features)

    # Identifying outliers (labeled as -1)
    df[df['cluster_dbscan'] == -1]
```

	User_ID	Posts	Likes	Comments	Shares	Messages_Sent	Messages_Received	Time_Spent_Hours	Reaction_to_Recommendations	Follower_Count	Following_Count	Profile_Age_Days	Engagement_Score	cluster_kmea
0	1	7	626	89	28	87	81	5.87	0.62	4656	494	376	288.57	
1	2	39	323	6	42	61	185	17.79	0.83	3970	1951	1942	157.19	
2	3	10	541	7	44	85	53	9.28	0.71	2404	531	1862	236.58	
3	4	26	818	20	12	10	54	13.11	0.60	1486	166	1003	348.71	
4	5	28	909	70	38	196	94	9.22	0.36	2791	147	1962	401.42	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
95	96	7	166	1	27	76	133	8.43	0.94	2510	371	283	80.53	
96	97	39	722	92	22	86	114	4.68	0.17	4138	2248	1728	325.48	
97	98	44	369	30	19	112	197	18.30	0.71	4236	40	944	178.70	
98	99	47	634	68	15	6	98	10.64	0.62	60	1419	1008	287.64	
99	100	31	940	45	35	26	195	18.46	0.94	4620	1406	273	414.96	

100 rows × 15 columns