**Report**

**Part 1 – Data Preprocessing**

In the first part, I first put the data in a dataframe with pandas. Then I displayed the first 5 rows of the data using the .head() function.

First 5 rows of dataset=

Date Screen Time (minutes) Number of Sessions Most Active Time \

0 2023-10-01 129.6 9 Night

1 2023-10-02 82.2 6 Evening

2 2023-10-03 175.5 13 Morning

3 2023-10-04 135.0 10 Afternoon

4 2023-10-05 82.6 7 Morning

Browsing (%) Posting (%) Messaging (%) Watching (%) Likes Comments \

0 48.2 12.2 11.0 28.6 33.0 8.0

1 46.9 13.7 23.6 15.8 22.0 4.0

2 57.0 7.5 16.3 19.1 33.0 6.0

3 44.2 10.0 17.4 28.4 22.0 6.0

4 57.4 7.1 14.7 20.8 13.0 4.0

Messages

0 11

1 5

2 7

3 12

4 6

Then I called the dataframe with the .tail() function to look at the last 5 rows of the data.

Last 5 rows of dataset=

Date Screen Time (minutes) Number of Sessions Most Active Time \

394 2024-10-26 84.8 8 Morning

395 2024-10-27 102.6 9 Morning

396 2024-10-28 47.2 4 Morning

397 2024-10-29 171.6 13 Afternoon

398 2024-10-30 154.8 12 Night

Browsing (%) Posting (%) Messaging (%) Watching (%) Likes Comments \

394 45.3 11.7 19.8 23.2 13.0 2.0

395 54.0 6.3 20.5 19.2 18.0 4.0

396 46.4 13.8 12.8 26.9 13.0 3.0

397 37.0 14.1 30.0 18.9 51.0 8.0

398 38.8 13.9 28.1 19.2 17.0 2.0

Messages

394 3

395 4

396 6

397 14

398 3

Then I called the dataframe with the .describe function to display the general data in the data such as mean, min, max 25% and count.

General statistics of dataset=

Date Screen Time (minutes) Number of Sessions \

count 5 5.000000 5.000000

mean 2024-10-28 00:00:00 112.200000 9.200000

min 2024-10-26 00:00:00 47.200000 4.000000

25% 2024-10-27 00:00:00 84.800000 8.000000

50% 2024-10-28 00:00:00 102.600000 9.000000

75% 2024-10-29 00:00:00 154.800000 12.000000

max 2024-10-30 00:00:00 171.600000 13.000000

std NaN 51.017252 3.563706

Browsing (%) Posting (%) Messaging (%) Watching (%) Likes \

count 5.000000 5.000000 5.000000 5.00000 5.000000

mean 44.300000 11.960000 22.240000 21.48000 22.400000

min 37.000000 6.300000 12.800000 18.90000 13.000000

25% 38.800000 11.700000 19.800000 19.20000 13.000000

50% 45.300000 13.800000 20.500000 19.20000 17.000000

75% 46.400000 13.900000 28.100000 23.20000 18.000000

max 54.000000 14.100000 30.000000 26.90000 51.000000

std 6.764614 3.310287 6.939957 3.51383 16.149303

Comments Messages

count 5.00000 5.000000

mean 3.80000 6.000000

min 2.00000 3.000000

25% 2.00000 3.000000

50% 3.00000 4.000000

75% 4.00000 6.000000

max 8.00000 14.000000

std 2.48998 4.636809

Then I called the dataframe with the .info function to look at the datatypes in the columns.

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 5 entries, 394 to 398

Data columns (total 11 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Date 5 non-null datetime64[ns]

1 Screen Time (minutes) 5 non-null float64

2 Number of Sessions 5 non-null int64

3 Most Active Time 5 non-null object

4 Browsing (%) 5 non-null float64

5 Posting (%) 5 non-null float64

6 Messaging (%) 5 non-null float64

7 Watching (%) 5 non-null float64

8 Likes 5 non-null float64

9 Comments 5 non-null float64

10 Messages 5 non-null int64

dtypes: datetime64[ns](1), float64(7), int64(2), object(1)

memory usage: 572.0+ bytes

Then, in order to use the data effectively and to prevent any corruption in the data during the examinations, I cleaned the data. To do this, I first found the missing cells in each column and put the average value in these missing cells. Secondly, I deleted the duplicate rows and ensured that each row was unique. At the same time, I made sure that the columns with dates were in accordance with the format using the to\_datetime function.

Values missing from each column= Date 0

Screen Time (minutes) 0

Number of Sessions 0

Most Active Time 0

Browsing (%) 0

Posting (%) 0

Messaging (%) 3

Watching (%) 4

Likes 2

Comments 2

Messages 0

dtype: int64

Median value for each column put to missing cells

Deleted 3 same rows twice

Now data is cleaned. Final version=

Date Screen Time (minutes) Number of Sessions Most Active Time \

0 2023-10-01 129.6 9 Night

1 2023-10-02 82.2 6 Evening

2 2023-10-03 175.5 13 Morning

3 2023-10-04 135.0 10 Afternoon

4 2023-10-05 82.6 7 Morning

Browsing (%) Posting (%) Messaging (%) Watching (%) Likes Comments \

0 48.2 12.2 11.0 28.6 33.0 8.0

1 46.9 13.7 23.6 15.8 22.0 4.0

2 57.0 7.5 16.3 19.1 33.0 6.0

3 44.2 10.0 17.4 28.4 22.0 6.0

4 57.4 7.1 14.7 20.8 13.0 4.0

Messages

0 11

1 5

2 7

3 12

4 6

Then, to make the data more meaningful, I created the average session duration feature from the existing columns. To create this, I calculated the average session duration for each row by doing Screen Time / Number of sessions and this became a new column.

Screen Time (minutes) Number of Sessions average session duration

0 129.6 9 14.4

1 82.2 6 13.7

2 175.5 13 13.5

3 135.0 10 13.5

4 82.6 7 11.8

**Part 2 – Exploratory Data Analysis (EDA)**

In this section, I first examined the correlation between columns. I did this especially using the .corr function. First, I found a value of 0.96 by correlating screen time with number of sessions. Thus, as I had predicted, there was a strongly high positive correlation between screen time and number of sessions. Secondly, I looked at the correlation between Screen time and Like and found a value of 0.80, so this time I found a high positive correlation. Thirdly, I looked at the correlation between number of sessions and comments and found a value of 0.66. Although there was a positive correlation, this correlation started to become a weak positive correlation. Finally, when I looked at the correlation between Posting and screen time, I found a value of -0.03, so there is no correlation between these 2 values.

Correlation of screen time and number of sessions = 0.96

Correlation of screen time and likes = 0.80

Correlation of number of sessions and comments = 0.66

Correlation of posting and screen time: -0.03

Then, in this section, I calculated the daily pattern, weekly pattern, monthly patterns and printed them. At the same time, I calculated the aggregate pattern and printed them as a whole.

Daily patterns result=

Day

Friday 98.714286

Monday 99.403509

Saturday 105.726786

Sunday 111.571930

Thursday 106.857143

Tuesday 109.094737

Wednesday 110.857895

Name: Screen Time (minutes), dtype: float64

Week patterns result=

Week

1 249.0

2 156.0

3 111.0

4 162.0

5 144.0

6 152.0

7 146.0

8 142.0

9 191.0

10 120.0

11 152.0

12 153.0

13 196.0

14 158.0

15 159.0

16 156.0

17 132.0

18 108.0

19 111.0

20 105.0

21 138.0

22 142.0

23 132.0

24 231.0

25 101.0

26 134.0

27 198.0

28 138.0

29 158.0

30 129.0

31 142.0

32 151.0

33 79.0

34 89.0

35 113.0

36 140.0

37 204.0

38 105.0

39 199.0

40 275.0

41 279.0

42 294.0

43 253.0

44 287.0

45 199.0

46 125.0

47 172.0

48 181.0

49 133.0

50 133.0

51 133.0

52 167.0

Name: Likes, dtype: float64

Monthly patterns =

Month

April 8.666667

August 7.032258

December 8.548387

February 8.758621

January 9.612903

July 8.677419

June 8.133333

March 9.483871

May 7.225806

November 8.966667

October 8.524590

September 7.900000

Name: Number of Sessions, dtype: float64

All Daily Summaries =

Screen Time (minutes) Likes Comments Messages

Day

Friday 98.714286 20.071429 4.178571 6.392857

Monday 99.403509 21.228070 4.315789 6.596491

Saturday 105.726786 20.589286 4.053571 6.517857

Sunday 111.571930 22.140351 4.526316 6.754386

Thursday 106.857143 19.517857 3.714286 5.892857

Tuesday 109.094737 21.964912 4.403509 6.526316

Wednesday 110.857895 22.157895 4.192982 5.982456

Aggregate patterns =

Screen Time (minutes) Likes

Month Day

April Friday 106.075 23.75

Monday 91.000 16.40

Saturday 127.850 22.75

Sunday 105.925 23.25

Thursday 138.825 31.75

... ... ...

September Saturday 125.575 36.75

Sunday 122.160 23.40

Thursday 99.100 20.25

Tuesday 101.775 16.75

**Part 3 – Visualization**

In this section, I prepared a Line chart using matplotlib using the Date data as the x axis and the Screen time data as the y axis. With this visual, we can examine the screen time data of each date as a graph.

A graph showing a blue line

Description automatically generated

In this part, I used matplotlib to keep them in a list using the ["Browsing (%)", "Posting (%)", "Messaging (%)", "Watching (%)"] in the data. Then, I calculated the mean of each one, converted them into percentages and graphed it as a bar chart. According to the bar chart, the most activity I did on Instagram was Browsing.

A graph of different colored squares

Description automatically generated

In this section, I created a scatterplot graph using matplotlib depending on the screen time and like correlation in the data. According to this graph, I saw that my liking rate increased as screen time increased. This shows positive correlation.

A graph with blue dots

Description automatically generated

In this section, I calculated the percentages in the data according to the screen time on the days of the week in April using matplotlib. Then, I created a pie chart with these percentages. When you look at the Pie Chart, you can see that I used Instagram the most on Thursdays in April. The reason for this was that I had fewer classes on Thursdays.

A pie chart with numbers and a few days of the week

Description automatically generated

Then I created a heatmap to visualize the correlation between the days of the week and the hours of the day. According to this heatmap, the average screen time is found for the hours of the days of the week. If a certain hour and a certain day of the week are very correlated, it becomes redder. According to the heatmap, my screen time is highest on Sunday at 21:00, and lowest on Friday at 9:00 in the morning.

A chart of different shades of red

Description automatically generated

**Part 4 – Machine Learning**

In this section, first I selected the attributes to create the cluster ["Screen Time (minutes)", "Browsing (%)", "Posting (%)", "Messaging (%)", "Watching (%)"]. After defining StandardScaler, I scaled these attributes and applied K-means cluster. I created a cluster for each attribute.

Cluster staticstics =

Screen Time (minutes) Browsing (%) Posting (%) Messaging (%) \

cluster

0 109.348993 47.824161 9.314765 17.607383

1 103.002685 52.539597 12.313423 20.378859

2 105.665306 41.468367 10.655102 27.675510

Watching (%)

cluster

0 25.193289

1 14.713423

2 20.212245

Then I created the cluster chart for this cluster with sns.scatterplot for screen time and browsing attributes.

A graph showing a cluster of dots

Description automatically generated with medium confidence

Then I set up a decision tree regression model with these attributes. To do this, I assigned the data attributes to the x value and our goal to the y value. I split the data into train and test using train\_test\_split. We defined our regressor with DecisionTreeRegressor and defined train and tests to the model with .fit. Then we completed the training of our model with .predict and obtained our mse and r2\_score values ​​as follows.

Decision tree model evaluation =

Mean squared error = 307.62

R squared = 0.84

I did same thing for RandomForestRegressor and got:

Random forest model evaluation =

Mean squared error = 276.48

R squared = 0.86

With a mean squared error of 276.48 vs. 307.62 and an R-squared value of 0.86 vs. 0.84, the Random Forest model does a little better than the Decision Tree model. Over 84% of the variation in screen time can be explained by both models, which show that they are very good at making predictions. Compared to the single Decision Tree model, the Random Forest model's group method probably does a better job of generalizing and reducing overfitting.