Nthabiseng Thema

February 16, 2025

1 General data analysis

[560]:
```
#import libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

[562]:
```
#get the data
df = pd.read_csv("meeting_instances.csv")
```

[564]:
```
#show the head / first 5 rows of the data
df.head()
```

[564]:

	Person_Name	Person_ID	Meeting_ID	Went_Overtime_Flag	Meeting_Duration	Technical_Difficulties_Flag	Meeting_Sentiment
0	Tsepo Modise	1	108	No	2 hours	No	1.000000
1	Tsepo Modise	1	105	No	1.5 hours	Yes	1.000000
2	Tsepo Modise	1	102	No	1.5 hours	No	1.123476
3	Tsepo Modise	1	107	No	30 minutes	No	0.945200
4	Tsepo Modise	1	110	No	2 hours	No	0.831163

[566]:
```
#show the info pertaining the data, i.e get the number or entries, data types -␣
↪> this will help us check for any null values
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1117 entries, 0 to 1116
Data columns (total 7 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
```

```
 0   Person_Name                  1117 non-null   object
 1   Person_ID                    1117 non-null   int64
 2   Meeting_ID                   1117 non-null   int64
 3   Went_Overtime_Flag           1117 non-null   object
 4   Meeting_Duration             1117 non-null   object
 5   Technical_Difficulties_Flag  1117 non-null   object
 6   Meeting_Sentiment            1117 non-null   float64
dtypes: float64(1), int64(2), object(4)
memory usage: 61.2+ KB
```

[568]:
```
#describe the data to get the stats for numerical columns
df.describe()
```

[568]:

	Person_ID	Meeting_ID	Meeting_Sentiment
count	1117.000000	1117.000000	1117.000000
mean	4.740376	106.720680	0.525255
std	2.230995	3.992921	0.314438
min	1.000000	101.000000	-0.232314
25%	3.000000	103.000000	0.269752
50%	5.000000	106.000000	0.516097
75%	6.000000	111.000000	0.798676
max	8.000000	114.000000	1.299552

[570]:
```
#check if there are any duplicates
df.duplicated().sum()
```

[570]:
```
np.int64(19)
```

[572]:
```
#drop duplicates
df.drop_duplicates()
#initially we had 1117 entries, and now we have 1098 entries, meaning there were␣
 ↪19 duplicates
#we dropped those entries so we ensure accuracy and consistancy
```

[572]:

	Person_Name	Person_ID	Meeting_ID	Went_Overtime_Flag	Meeting_Duration	\
0	Tsepo Modise	1	108	No	2 hours	
1	Tsepo Modise	1	105	No	1.5 hours	
2	Tsepo Modise	1	102	No	1.5 hours	
3	Tsepo Modise	1	107	No	30 minutes	
4	Tsepo Modise	1	110	No	2 hours	
...	...	...	...	...	...	
1112	Lucy Ncube	8	113	Yes	1.5 hours	
1113	Lucy Ncube	8	111	Yes	45 minutes	
1114	Lucy Ncube	8	106	No	15 minutes	
1115	Lucy Ncube	8	101	No	15 minutes	
1116	Lucy Ncube	8	107	No	30 minutes	

	Technical_Difficulties_Flag	Meeting_Sentiment

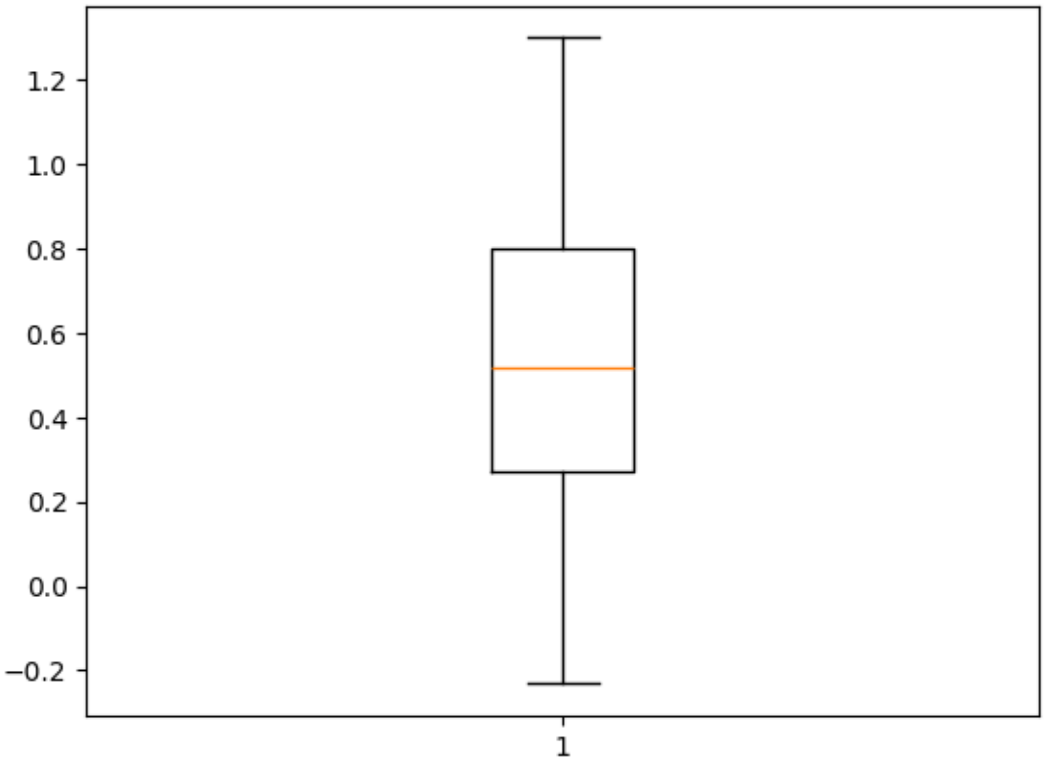
```
0                              No           1.000000
1                             Yes           1.000000
2                              No           1.123476
3                              No           0.945200
4                              No           0.831163
...                           ...                ...
1112                           No           0.128230
1113                           No           0.232723
1114                           No           0.807846
1115                           No           0.436845
1116                          Yes           0.251056

[1098 rows x 7 columns]
```

[574]:
```
#check if there are any ouliers
plt.boxplot(df["Meeting_Sentiment"])

#There are no outliers, all Meeting_
```

[574]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x216dfb9fd60>,
  <matplotlib.lines.Line2D at 0x216dfb9ffd0>],
 'caps': [<matplotlib.lines.Line2D at 0x216dfbaf2e0>,
  <matplotlib.lines.Line2D at 0x216dfbaf580>],
 'boxes': [<matplotlib.lines.Line2D at 0x216dfb9fac0>],
 'medians': [<matplotlib.lines.Line2D at 0x216dfbaf820>],
 'fliers': [<matplotlib.lines.Line2D at 0x216dfbafac0>],
 'means': []}
```



[576]:
```
#Merge in the person_info dataset
df2 = pd.read_csv("person_info.csv")
```

[578]:
```
#create a new dataframe and merge the two dataframes on the common column, which␣
↪is the Person_Name
df3 = pd.merge(df,df2, how="left", on="Person_Name").drop(columns = ["Unnamed:␣
↪0"])
```

[580]:
```
df3.head()
```

[580]:

	Person_Name	Person_ID	Meeting_ID	Went_Overtime_Flag	Meeting_Duration	Technical_Difficulties_Flag	Meeting_Sentiment	Person_Level
0	Tsepo Modise	1	108	No	2 hours	No	1.000000	Executive
1	Tsepo Modise	1	105	No	1.5 hours	Yes	1.000000	Executive
2	Tsepo Modise	1	102	No	1.5 hours	No	1.123476	Executive
3	Tsepo Modise	1	107	No	30 minutes	No	0.945200	Executive
4	Tsepo Modise	1	110	No	2 hours			

```
4                          No           0.831163    Executive

  Person_Daily_Time_Threshold
0                   6 hours
1                   6 hours
2                   6 hours
3                   6 hours
4                   6 hours
```

1. Calculate the average meeting sentiment for each "Person_Level" (staff, manager, executive).

[582]:
```
#use groupby() -> group rows and perform an aggregate
df3.groupby("Person_Level")["Meeting_Sentiment"].mean()
```

[582]:
```
Person_Level
Executive    0.620325
Manager      0.487013
Staff        0.510143
Name: Meeting_Sentiment, dtype: float64
```

Are there any noticeable differences in sentiment among these groups? Yes! The Executive has a higher sentiment than the Manager and Staff. But the Manager has lower sentiment than the Staff. This could be due to the number of meetings that a manager has/ the number of managers there as compared to the staff. We can find that:

[584]:
```
df3["Person_Level"].value_counts()

#We have more Staff than Managers which also explains the slight difference in␣
↪sentiment
```

[584]:
```
Person_Level
Staff        452
Manager      423
Executive    242
Name: count, dtype: int64
```

[ ]:
```

```

[587]:
```
# merge the meeting_cadance
df4 = pd.read_csv("meeting_metadata.csv")
```

[589]:
```
df5 = pd.
↪merge(df3,df4[["Meeting_ID","Meeting_Cadence","Meeting_Topics"]],how="left",on="Meeting_ID")
```

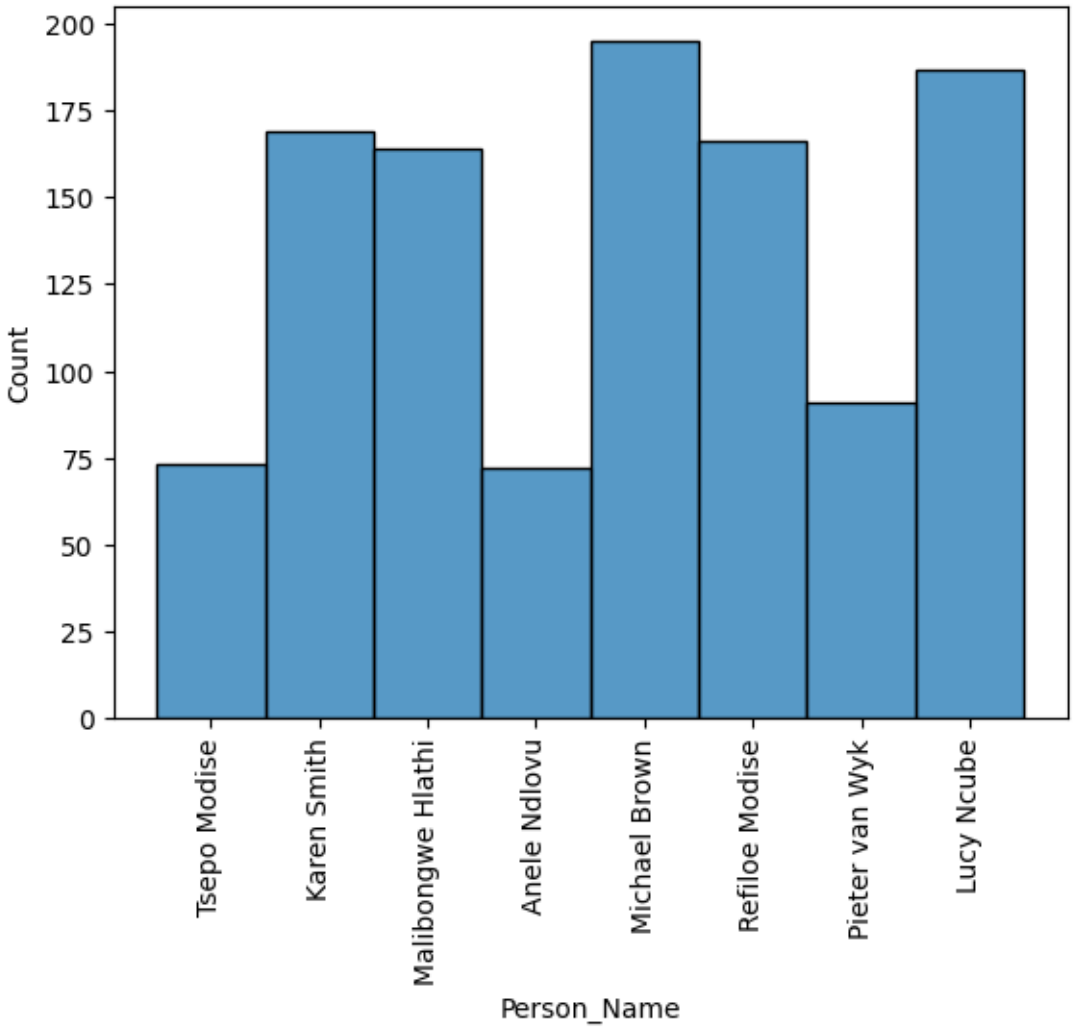
[591]:
```
df5.head()
```

[591]:
	Person_Name	Person_ID	Meeting_ID	Went_Overtime_Flag	Meeting_Duration	Technical_Difficulties_Flag	Meeting_Sentiment	Person_Level	Person_Daily_Time_Threshold	Meeting_Cadence	Meeting_Topics
0	Tsepo Modise	1	108	No	2 hours	No	1.000000	Executive	6 hours	Monthly	Strategy Discussion, Financial Planning, Marke...
1	Tsepo Modise	1	105	No	1.5 hours	Yes	1.000000	Executive	6 hours	Weekly	Project Progress, Milestone Review
2	Tsepo Modise	1	102	No	1.5 hours	No	1.123476	Executive	6 hours	Weekly	Team Performance, Sales Figures, Updating on P...
3	Tsepo Modise	1	107	No	30 minutes	No	0.945200	Executive	6 hours	Daily	Project Status as of today, and Roadblocks
4	Tsepo Modise	1	110	No	2 hours	No	0.831163	Executive	6 hours	Monthly	Employee Training, Skill Development

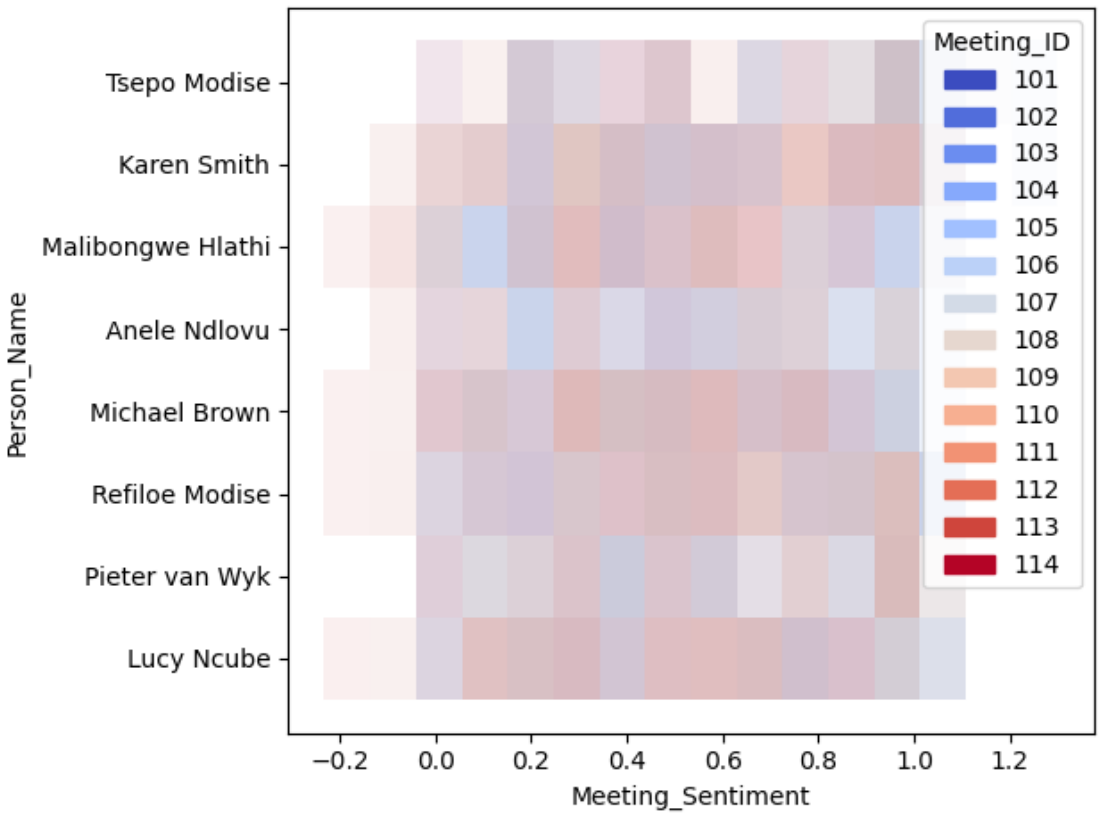
2. Which people are experiencing too many meetings and what meetings would you recommend they reduce?

[722]:
```
sns.histplot(df5, x="Person_Name"  )
plt.xticks(rotation=90)
```

[722]:
```
([0, 1, 2, 3, 4, 5, 6, 7],
 [Text(0, 0, 'Tsepo Modise'),
  Text(1, 0, 'Karen Smith'),
  Text(2, 0, 'Malibongwe Hlathi'),
  Text(3, 0, 'Anele Ndlovu'),
  Text(4, 0, 'Michael Brown'),
  Text(5, 0, 'Refiloe Modise'),
  Text(6, 0, 'Pieter van Wyk'),
  Text(7, 0, 'Lucy Ncube')])
```



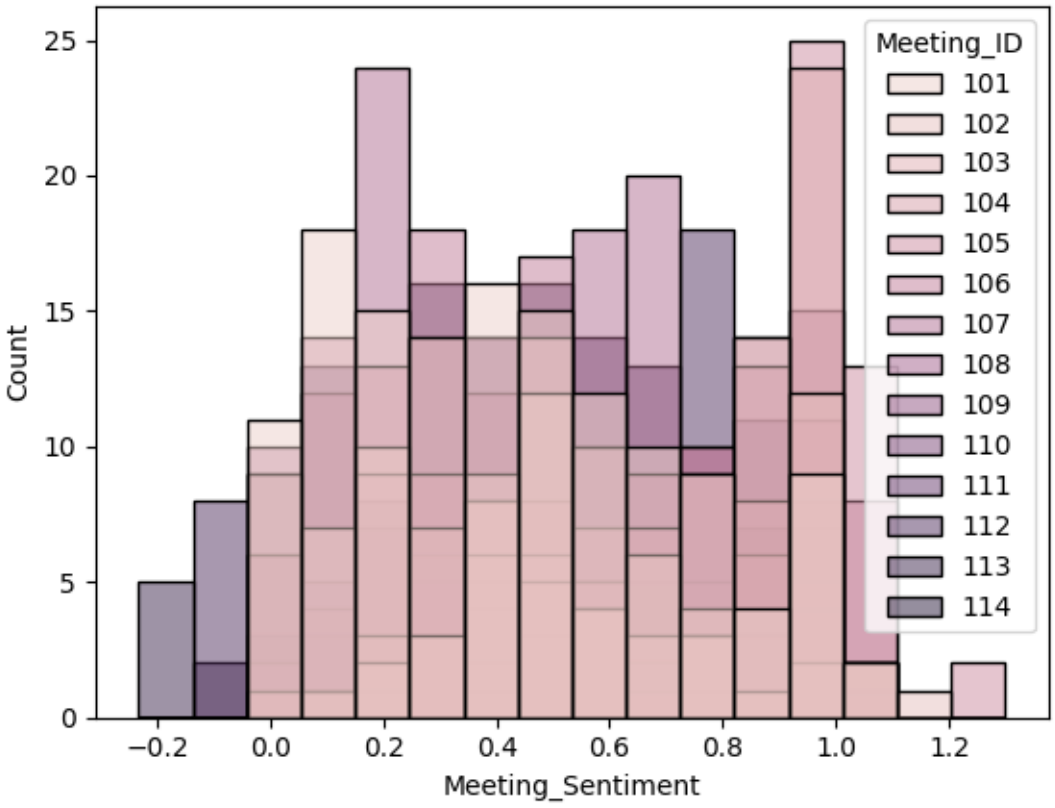
[738]:
```
sns.histplot(df5,x ="Meeting_Sentiment", y="Person_Name", hue="Meeting_ID",␣
↪alpha=0.2, palette="coolwarm")
plt.tight_layout()
```



[595]:
```
# on an attempt to reduce the daily meeting, check which ones have the lowest␣
↪sentiment

sns.histplot(df5,x ="Meeting_Sentiment", hue="Meeting_ID")
```

[595]: <Axes: xlabel='Meeting_Sentiment', ylabel='Count'>



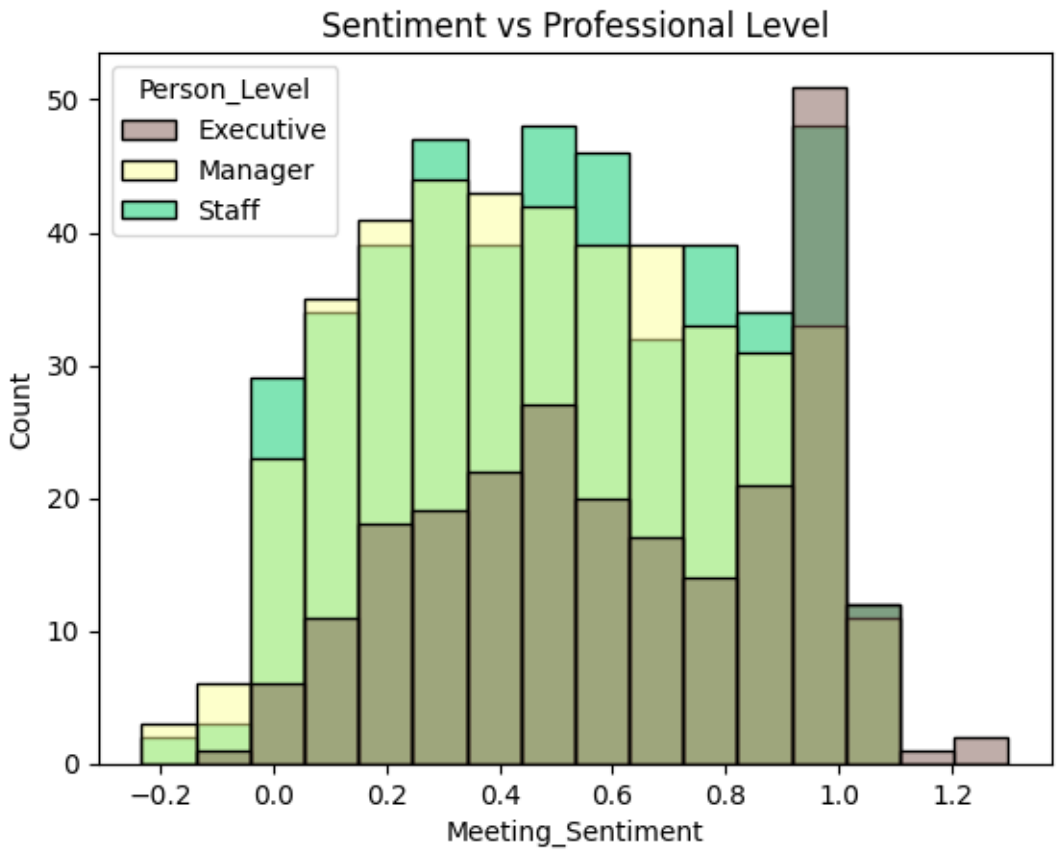
[597]:
```
df5.groupby(["Person_Name","Person_Level"])["Meeting_Topics"].count()
```

[597]:
```
Person_Name        Person_Level
Anele Ndlovu       Manager          72
Karen Smith        Executive       169
Lucy Ncube         Manager         187
Malibongwe Hlathi  Manager         164
Michael Brown      Staff           195
Pieter van Wyk     Staff            91
Refiloe Modise     Staff           166
Tsepo Modise       Executive        73
Name: Meeting_Topics, dtype: int64
```

3. Visualize the distribution of meeting sentiment scores using a histogram. Is there anything you can infer from the distribution?

[599]:
```
sns.
 ↪histplot(data=df3,x="Meeting_Sentiment",hue="Person_Level",palette="terrain_r")
plt.title("Sentiment vs Professional Level")
```

[599]:
```
Text(0.5, 1.0, 'Sentiment vs Professional Level')
```



The sentiment scores are mostly neutral, with a peak around 0.2-0.4

Executives have a more positive sentiment(1-1.3) possibly due to different engagement levels and experienced decision making

4. Compare the average "Meeting_Sentiment" for daily, weekly, fortnightly, and monthly meetings.

Are there any significant differences in sentiment across different cadences?

[606]:
```
df5.groupby("Meeting_Cadence")["Meeting_Sentiment"].mean()
```

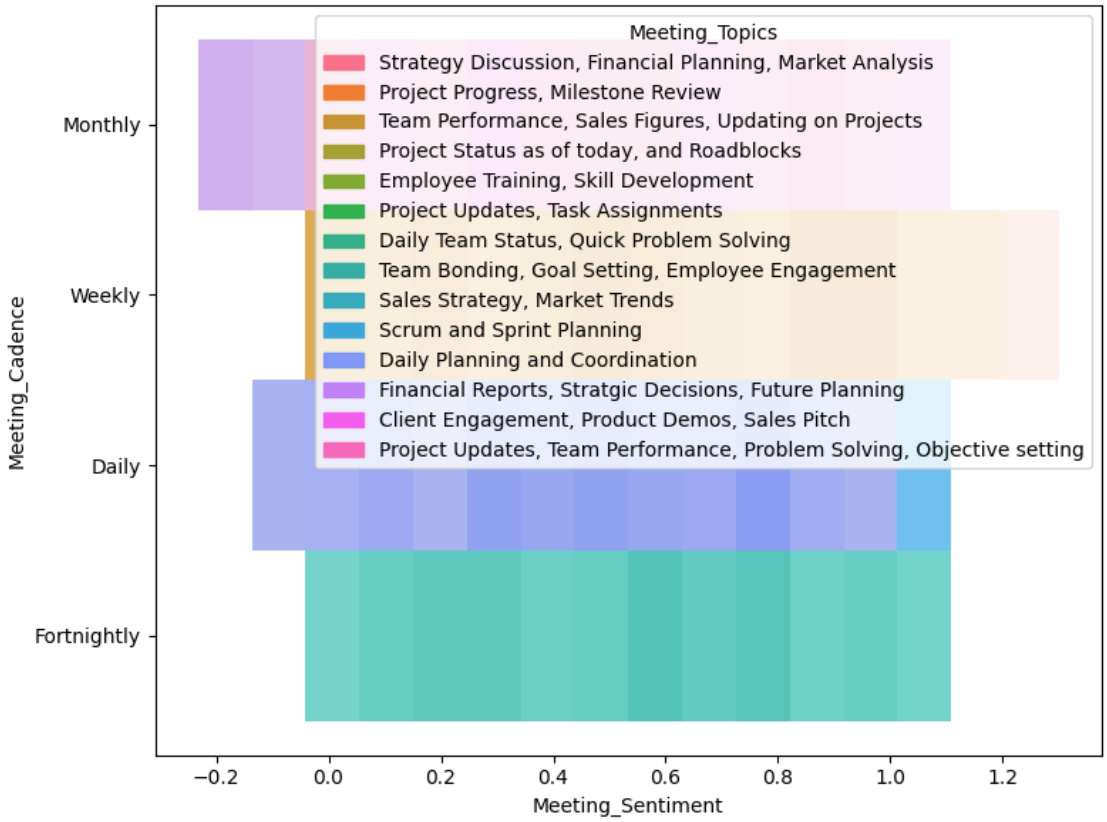
[606]:
```
Meeting_Cadence
Daily          0.471909
Fortnightly    0.601488
Monthly        0.526964
Weekly         0.626473
Name: Meeting_Sentiment, dtype: float64
```

Daily meetings have the lowest sentiment which suggests they are less productive -> maybe due to repetitiveness -> reduce the frequency of daily meetings

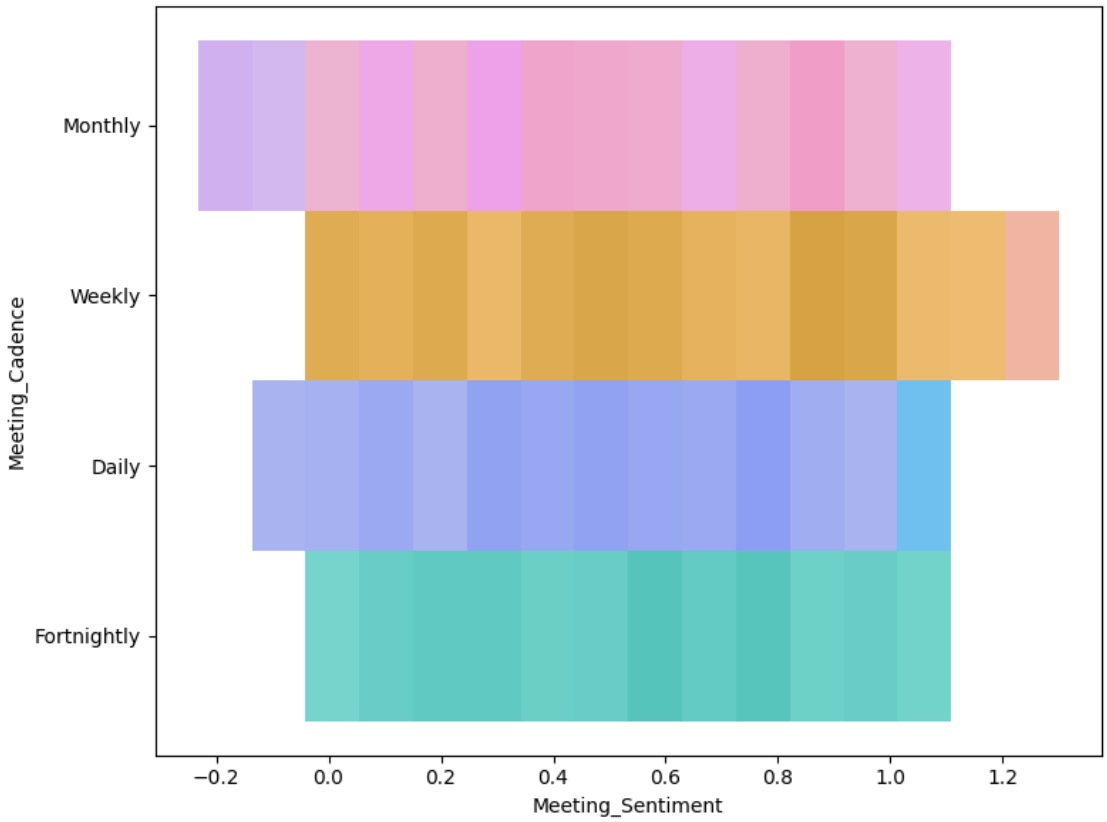
Weekly meetings have the highest sentiment(0.626473) -> employees feel more positive

Fortnightly meetings also have a higher sentiment(0.601488)

[610]:
```
# on an attempt to reduce the daily meeting, check which ones have the lowest␣
 ↪sentiment
plt.figure(figsize=(8, 6))
sns.histplot(df5,x ="Meeting_Sentiment", y="Meeting_Cadence",␣
 ↪hue="Meeting_Topics",legend=(1,1))
plt.tight_layout()
```



[612]:
```
# on an attempt to reduce the daily meeting, check which ones have the lowest␣
 ↪sentiment
plt.figure(figsize=(8, 6))
sns.histplot(df5,x ="Meeting_Sentiment", y="Meeting_Cadence",␣
 ↪hue="Meeting_Topics",legend=False)
plt.tight_layout()
```



2 TEXT ANALYSIS

[617]:
```
dt = pd.read_csv("meeting_metadata.csv")
```

[619]:
```
# #split the meeting topics into columns
# df_topics = dt["Meeting_Topics"].str.split(",",expand=True).T #expand creates␣
 ↪new columns
# #convert my rows into columns
# df_topics = df_topics.melt(var_name="Topic_number",␣
 ↪value_name="Meeting_Topics").dropna()
```

[621]:
```
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
```

[623]:
```
# Download NLTK stopwords and wordnet for lemmatization (run once)
# import nltk
# nltk.download('stopwords')
# nltk.download('punkt')
```

```
# nltk.download('wordnet')
```

[625]:
```
stop_words = set(stopwords.words('english'))
```

[627]:
```
lemmatizer = WordNetLemmatizer()
```

[629]:
```
#convert topics to lowercase to ensure consistence
dt["Meeting_Topics"] = dt["Meeting_Topics"].str.lower()
dt["Meeting_Name"] = dt["Meeting_Name"].str.lower()
```

[631]:
```
dt.head()
```

[631]:

	Meeting_ID	Meeting_Name	Meeting_Cadence	Meeting_Duration	Meeting_Topics
0	101	daily standup meeting	Daily	15 minutes	project updates, task assignments
1	102	manager weekly report	Weekly	1.5 hours	team performance, sales figures, updating on p...
2	103	team huddle	Fortnightly	1 hour	daily team status, quick problem solving
3	104	department meeting	Monthly	1 hour	project updates, team performance, problem sol...
4	105	project review	Weekly	1.5 hours	project progress, milestone review

[633]:
```
#remove characters
dt["Meeting_Topics"] = dt["Meeting_Topics"].apply(lambda x: re.sub(r'[^a-z\s]',␣
↪'', x))
dt["Meeting_Name"] = dt["Meeting_Name"].apply(lambda x: re.sub(r'[^a-z\s]', '',␣
↪x))
```

[635]:
```
dt.head()
```

[635]:

	Meeting_ID	Meeting_Name	Meeting_Cadence	Meeting_Duration	Meeting_Topics
0	101	daily standup meeting	Daily	15 minutes	project updates task assignments
1	102	manager weekly report	Weekly	1.5 hours	team performance sales figures updating on pro...
2	103	team huddle	Fortnightly	1 hour	daily team status quick problem solving
3	104	department meeting	Monthly	1 hour	project updates team performance problem solvi...
4	105	project review	Weekly	1.5 hours	project progress milestone review

[ ]:

[638]:
```
# Tokenization -. split the text into individual words
dt['Meeting_Name_Tokens'] = dt['Meeting_Name'].apply(word_tokenize)
dt['Meeting_Topics_Tokens'] = dt['Meeting_Topics'].apply(word_tokenize)
```

[ ]:

[641]:
```
# Stopword removal -> remove common words
dt['Meeting_Name_Tokens'] = dt['Meeting_Name_Tokens'].apply(lambda x: [word for␣
↪word in x if word not in stop_words])
dt['Meeting_Topics_Tokens'] = dt['Meeting_Topics_Tokens'].apply(lambda x: [word␣
↪for word in x if word not in stop_words])
```

[643]:
```
dt.head()
```

[643]:
```
   Meeting_ID           Meeting_Name Meeting_Cadence Meeting_Duration  \
0         101  daily standup meeting           Daily       15 minutes
1         102  manager weekly report          Weekly        1.5 hours
2         103            team huddle     Fortnightly           1 hour
3         104     department meeting         Monthly           1 hour
4         105         project review          Weekly        1.5 hours

                                      Meeting_Topics  \
0                  project updates task assignments
1  team performance sales figures updating on pro...
2            daily team status quick problem solving
3  project updates team performance problem solvi...
4                 project progress milestone review

         Meeting_Name_Tokens  \
0  [daily, standup, meeting]
1  [manager, weekly, report]
2             [team, huddle]
3     [department, meeting]
4         [project, review]

                              Meeting_Topics_Tokens
0              [project, updates, task, assignments]
1  [team, performance, sales, figures, updating, ...
2     [daily, team, status, quick, problem, solving]
3  [project, updates, team, performance, problem,...
4             [project, progress, milestone, review]
```

[645]:
```
# nltk.download('omw-1.4')
```

[647]:
```
# Stemming/Lemmatization - >normalize words by reducing them to their base root
dt['Meeting_Name_Tokens'] = dt['Meeting_Name_Tokens'].apply(lambda x:␣
↪[lemmatizer.lemmatize(word) for word in x])
dt['Meeting_Topics_Tokens'] = dt['Meeting_Topics_Tokens'].apply(lambda x:␣
↪[lemmatizer.lemmatize(word) for word in x])
```

[649]:
```
dt[['Meeting_Name', 'Meeting_Name_Tokens', 'Meeting_Topics',␣
↪'Meeting_Topics_Tokens']]
```

[649]:

	Meeting_Name	Meeting_Name_Tokens	\
0	daily standup meeting	[daily, standup, meeting]	
1	manager weekly report	[manager, weekly, report]	
2	team huddle	[team, huddle]	
3	department meeting	[department, meeting]	
4	project review	[project, review]	
5	daily scrum meeting	[daily, scrum, meeting]	
6	daily status meeting	[daily, status, meeting]	
7	executive strategy meeting	[executive, strategy, meeting]	
8	client presentation	[client, presentation]	
9	training workshop	[training, workshop]	
10	staff team building	[staff, team, building]	
11	daily planning meeting	[daily, planning, meeting]	
12	board meeting	[board, meeting]	
13	sales strategy meeting	[sale, strategy, meeting]	

	Meeting_Topics	\
0	project updates task assignments	
1	team performance sales figures updating on pro...	
2	daily team status quick problem solving	
3	project updates team performance problem solvi...	
4	project progress milestone review	
5	scrum and sprint planning	
6	project status as of today and roadblocks	
7	strategy discussion financial planning market ...	
8	client engagement product demos sales pitch	
9	employee training skill development	
10	team bonding goal setting employee engagement	
11	daily planning and coordination	
12	financial reports stratgic decisions future pl...	
13	sales strategy market trends	

	Meeting_Topics_Tokens
0	[project, update, task, assignment]
1	[team, performance, sale, figure, updating, pr...
2	[daily, team, status, quick, problem, solving]
3	[project, update, team, performance, problem, ...
4	[project, progress, milestone, review]

```
5                          [scrum, sprint, planning]
6                 [project, status, today, roadblock]
7   [strategy, discussion, financial, planning, ma...
8    [client, engagement, product, demo, sale, pitch]
9            [employee, training, skill, development]
10  [team, bonding, goal, setting, employee, engag...
11                    [daily, planning, coordination]
12  [financial, report, stratgic, decision, future...
13                    [sale, strategy, market, trend]
```

[ ]:

[652]:
```
#Keyword matching tokenized data
project_update_keywords = ['project', 'update','status', 'progress', 'meeting',␣
 ↪'review', 'report']

# Function to check if any keyword is present in the tokens
def contains_keywords(tokens, keywords):
    return any(word in tokens for word in keywords)
```

[654]:
```
# Check if meeting name or meeting topics contain any of the project update␣
 ↪keywords
dt['Related_To_Project_Updates'] = dt['Meeting_Name_Tokens'].apply(lambda x:␣
 ↪contains_keywords(x, project_update_keywords)) | \
                                    dt['Meeting_Topics_Tokens'].apply(lambda x:␣
 ↪contains_keywords(x, project_update_keywords))

# Filter DataFrame to show only meetings related to project updates
relevant_meetings = dt[dt['Related_To_Project_Updates']]

# Display the filtered DataFrame
print("Relevant Meetings Related to Project Updates:")
print(relevant_meetings[['Meeting_Name', 'Meeting_Topics']])
```

```
Relevant Meetings Related to Project Updates:
                  Meeting_Name  \
0        daily standup meeting
1        manager weekly report
2                  team huddle
3           department meeting
4               project review
5          daily scrum meeting
6         daily status meeting
7   executive strategy meeting
11      daily planning meeting
12                board meeting
13      sales strategy meeting
```

```
                                    Meeting_Topics
0                   project updates task assignments
1   team performance sales figures updating on pro...
2             daily team status quick problem solving
3   project updates team performance problem solvi...
4                  project progress milestone review
5                           scrum and sprint planning
6           project status as of today and roadblocks
7   strategy discussion financial planning market ...
11                    daily planning and coordination
12  financial reports stratgic decisions future pl...
13                       sales strategy market trends
```

[656]:
```
dt.head()
```

[656]:
```
   Meeting_ID           Meeting_Name Meeting_Cadence Meeting_Duration  \
0         101  daily standup meeting           Daily       15 minutes
1         102  manager weekly report          Weekly        1.5 hours
2         103             team huddle     Fortnightly           1 hour
3         104     department meeting         Monthly           1 hour
4         105         project review          Weekly        1.5 hours

                                    Meeting_Topics  \
0                   project updates task assignments
1   team performance sales figures updating on pro...
2             daily team status quick problem solving
3   project updates team performance problem solvi...
4                  project progress milestone review

        Meeting_Name_Tokens  \
0  [daily, standup, meeting]
1  [manager, weekly, report]
2             [team, huddle]
3     [department, meeting]
4         [project, review]

                               Meeting_Topics_Tokens  \
0                [project, update, task, assignment]
1  [team, performance, sale, figure, updating, pr...
2     [daily, team, status, quick, problem, solving]
3  [project, update, team, performance, problem, ...
4             [project, progress, milestone, review]

   Related_To_Project_Updates
0                        True
1                        True
2                        True
```

```
3                        True
4                        True
```

1. How much time is spent in meetings relating to providing updates on projects?

[660]:
```
# convert the Meeting duration column to a standard -> minutes to hours and␣
↪convert o float as currently they are objects
```

[662]:
```
data = pd.merge(df5, dt[['Meeting_ID', 'Related_To_Project_Updates']],␣
↪how="left", on='Meeting_ID')
```

[664]:
```
data.head()
```

[664]:

	Person_Name	Person_ID	Meeting_ID	Went_Overtime_Flag	Meeting_Duration	Technical_Difficulties_Flag	Meeting_Sentiment	Person_Level	Person_Daily_Time_Threshold	Meeting_Cadence	Meeting_Topics	Related_To_Project_Updates
0	Tsepo Modise	1	108	No	2 hours	No	1.000000	Executive	6 hours	Monthly	Strategy Discussion, Financial Planning, Marke...	True
1	Tsepo Modise	1	105	No	1.5 hours	Yes	1.000000	Executive	6 hours	Weekly	Project Progress, Milestone Review	True
2	Tsepo Modise	1	102	No	1.5 hours	No	1.123476	Executive	6 hours	Weekly	Team Performance, Sales Figures, Updating on P...	True
3	Tsepo Modise	1	107	No	30 minutes	No	0.945200	Executive	6 hours	Daily	Project Status as of today, and Roadblocks	True
4	Tsepo Modise	1	110	No	2 hours	No	0.831163	Executive	6 hours	Monthly	Employee Training, Skill Development	False

[666]:
```
# Filter rows with 'minutes'
mask = data['Meeting_Duration'].str.contains('minutes', case=False, na=False)

# Remove 'minutes' from the filtered rows
data.loc[mask, 'Meeting_Duration'] = data.loc[mask, 'Meeting_Duration'].str.
 ↪replace('minutes', '', case=False)

# Convert filtered rows to float and then to hours
data.loc[mask, 'Meeting_Duration'] = data.loc[mask, 'Meeting_Duration'].
 ↪astype(float) / 60
```

[667]:
```
# now remove hours and convert to float
data['Meeting_Duration'] = data['Meeting_Duration'].astype(str)  # Ensure all␣
 ↪values are strings
data['Meeting_Duration'] = data['Meeting_Duration'].str.replace('hours', '',␣
 ↪case=False)

data['Meeting_Duration'] = data['Meeting_Duration'].str.replace('hour', '',␣
 ↪case=False).astype(float)
```

[668]:
```
data.head()
```

[668]:

	Person_Name	Person_ID	Meeting_ID	Went_Overtime_Flag	Meeting_Duration	\
0	Tsepo Modise	1	108	No	2.0	
1	Tsepo Modise	1	105	No	1.5	
2	Tsepo Modise	1	102	No	1.5	
3	Tsepo Modise	1	107	No	0.5	
4	Tsepo Modise	1	110	No	2.0	

	Technical_Difficulties_Flag	Meeting_Sentiment	Person_Level	\
0	No	1.000000	Executive	
1	Yes	1.000000	Executive	
2	No	1.123476	Executive	
3	No	0.945200	Executive	
4	No	0.831163	Executive	

	Person_Daily_Time_Threshold	Meeting_Cadence	\
0	6 hours	Monthly	
1	6 hours	Weekly	
2	6 hours	Weekly	
3	6 hours	Daily	
4	6 hours	Monthly	

	Meeting_Topics	\
0	Strategy Discussion, Financial Planning, Marke...	
1	Project Progress, Milestone Review	
2	Team Performance, Sales Figures, Updating on P...	

```
3         Project Status as of today, and Roadblocks
4               Employee Training, Skill Development

   Related_To_Project_Updates
0                        True
1                        True
2                        True
3                        True
4                       False
```

[672]:
```
relevant_meetings = data[data['Related_To_Project_Updates']]
```

[674]:
```
#now we can claculate the total time spent in providing updates
total_time_spent = relevant_meetings['Meeting_Duration'].sum()
print("The total time spent giving updates to project is : ", total_time_spent ,␣
↪"hours")
```

```
The total time spent giving updates to project is :  804.75 hours
```

2. How many people are involved in planning and strategy meetings?

[677]:
```
# Define the stemmed keywords for planning and strategy
planning_strategy_keywords = ['plan', 'strategi', 'coordin', 'organ', 'develop',␣
↪'analyz', 'analys', 'forecast', 'object', 'schedul', 'sale']

# Filter the DataFrame to include only meetings related to planning and strategy
dt['Is_Planning_Strategy'] = dt['Meeting_Topics'].apply(lambda x: any(keyword in␣
↪x for keyword in planning_strategy_keywords))

# Count the number of unique Meeting_IDs for planning and strategy meetings
planning_strategy_meetings = dt[dt['Is_Planning_Strategy']]
num_people_involved = planning_strategy_meetings['Meeting_ID'].nunique()

print(f"Number of people (unique meetings) involved in planning and strategy␣
↪meetings: {num_people_involved}")
```

```
Number of people (unique meetings) involved in planning and strategy meetings: 9
```

[690]:
```
dt_new = pd.merge(data,dt, how="left", on="Meeting_ID")
```

[706]:
```
# Define the stemmed keywords for planning and strategy
planning_strategy_keywords = ['plan', 'strategi', 'coordin', 'organ', 'develop',␣
↪'analyz', 'analys', 'forecast', 'object', 'schedul', 'sale']

# Filter the DataFrame to include only meetings related to planning and strategy
dt_new['Is_Planning_Strategy'] = dt_new['Meeting_Topics_y'].apply(lambda x:␣
↪any(keyword in x for keyword in planning_strategy_keywords))

# Count the number of unique Meeting_IDs for planning and strategy meetings
```

```
planning_strategy_meetings = dt_new[dt_new['Is_Planning_Strategy']]
num_people_involved = planning_strategy_meetings['Meeting_ID'].nunique()

# Get unique names and levels of people involved
people_involved = planning_strategy_meetings[['Person_Name', 'Person_Level']].
 ↪drop_duplicates()

# Print the number of unique meetings
print(f"Number of people (unique meetings) involved in planning and strategy␣
 ↪meetings: {num_people_involved}\n")

# Print the names and levels of people involved
print("People involved in planning and strategy meetings with their levels:")
for index, row in people_involved.iterrows():
    print(f"{row['Person_Name']} - {row['Person_Level']}")
```

```
Number of people (unique meetings) involved in planning and strategy meetings: 9

People involved in planning and strategy meetings with their levels:
Tsepo Modise - Executive
Karen Smith - Executive
Malibongwe Hlathi - Manager
Anele Ndlovu - Manager
Michael Brown - Staff
Refiloe Modise - Staff
Pieter van Wyk - Staff
Lucy Ncube - Manager
```

[ ]: