EXPLORATORY DATA ANALYSIS REPORT



**Name: Vasupally Tarun Dorai Raj**

**Email: Vasupally.dorairaj.21063@iitgoa.ac.in**

Contents

[EDA ON EMOTION DATA: 3](#_Toc146829902)

[Calculation of overall score assigning an assumed weight: 18](#_Toc146829903)

[EDA ON TRANSCRIPTS\_DATA 21](#_Toc146829904)

[Calculation of overall score assigning an assumed weight: 41](#_Toc146829905)

[EDA ON TRANSCRIPTS TEXT 45](#_Toc146829906)

[EDA ON GAZE DATA: 49](#_Toc146829907)

[EDA/CALCULATIONS ON OVERALL CANDIDATES’ PERFORMANCE: 55](#_Toc146829908)

# EDA ON EMOTION DATA:

For each candidate:

* Contains histogram plot of each emotion
* Contains boxplot
* Contains dominant emotion histogram (dominant emotion could be observed via histogram)

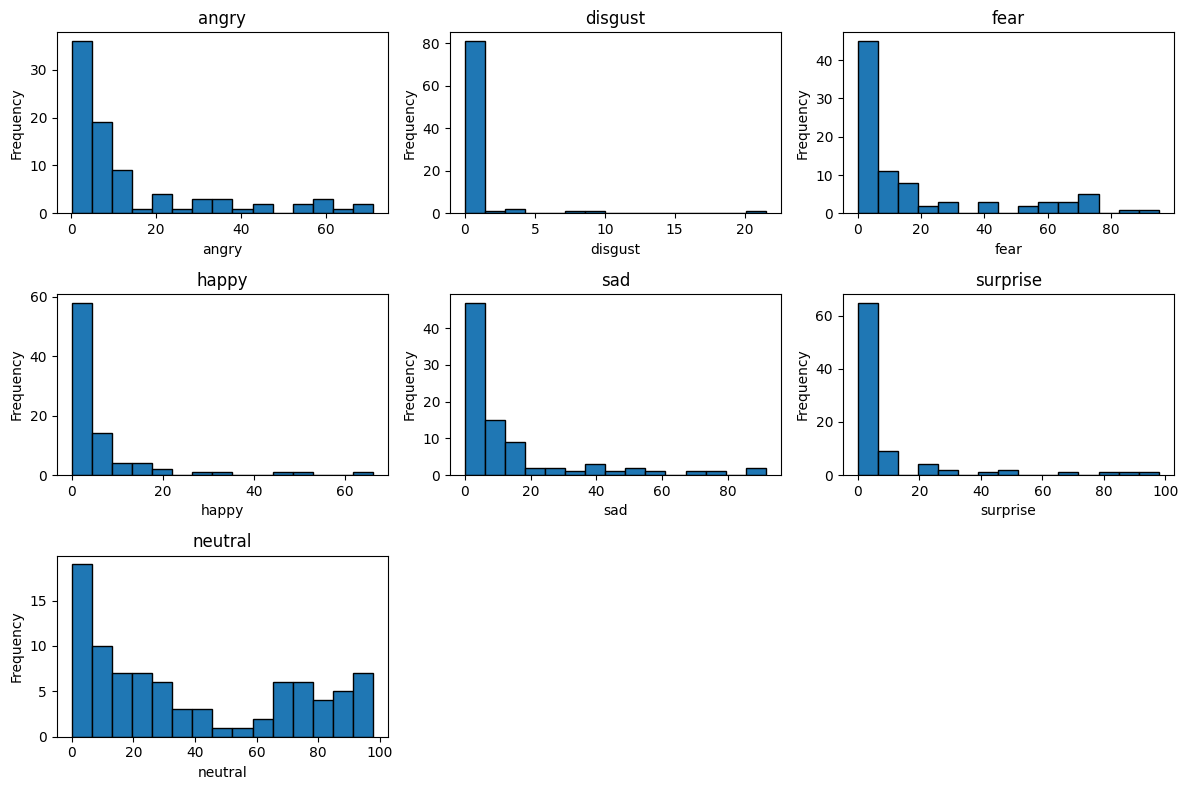
Insights:

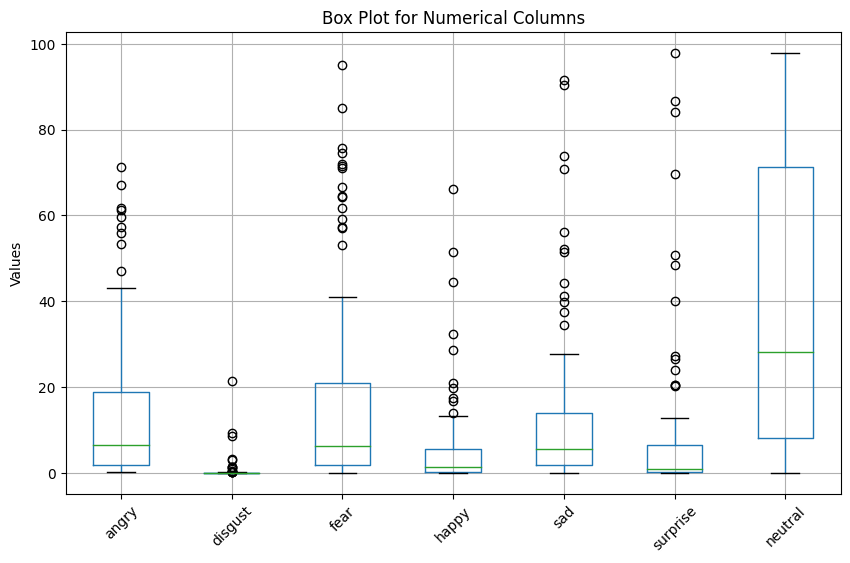
* Can observe the frequency of the scores in a range (range = bin size = 15 divisions)
* Boxplot offers insights into the statistical characteristics of these emotions.

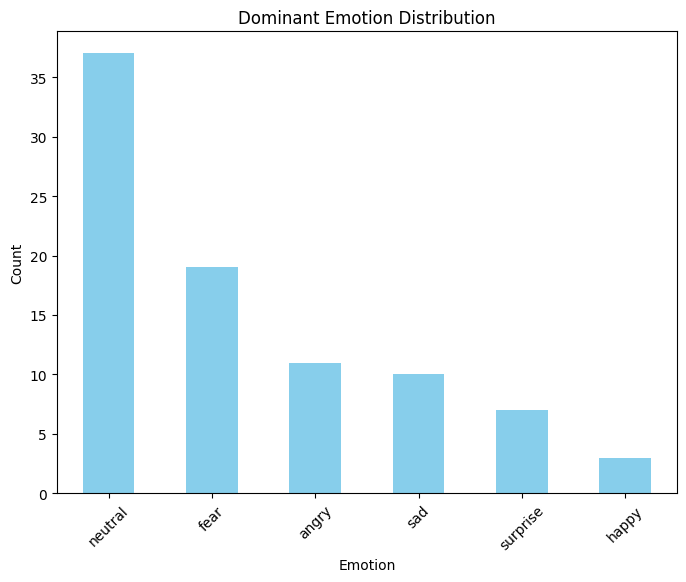
Median (Line Inside the Box**):** The line inside the box represents the median value of the emotion data. It's the middle point when all values are arranged in ascending order. It gives you an idea of the central tendency of the emotion.

Outliers (Data Points Beyond the Whiskers): Data points that fall beyond the whiskers are considered outliers. These are values that significantly deviate from the majority of the data. Outliers may indicate extreme emotions or unusual responses. In our data, there are many outliners for a few candidates' emotions show us that there is no such dependency.

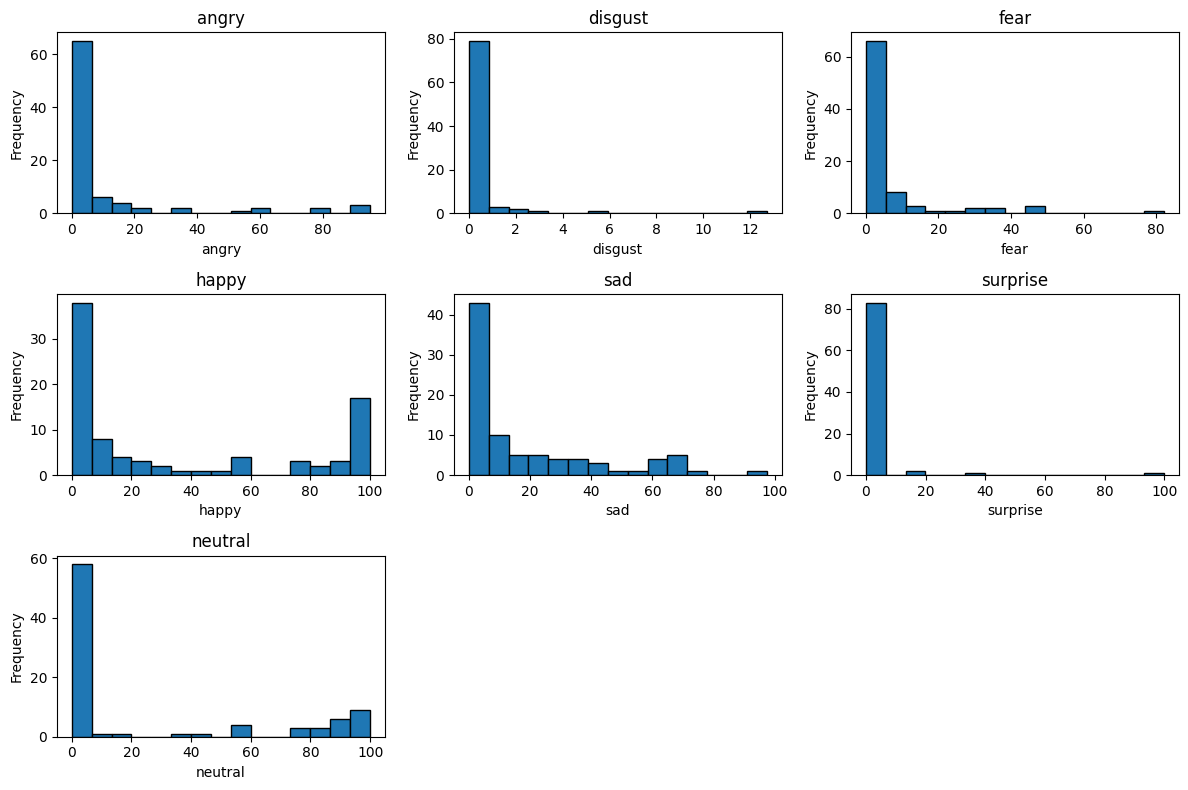
Candidate 1:

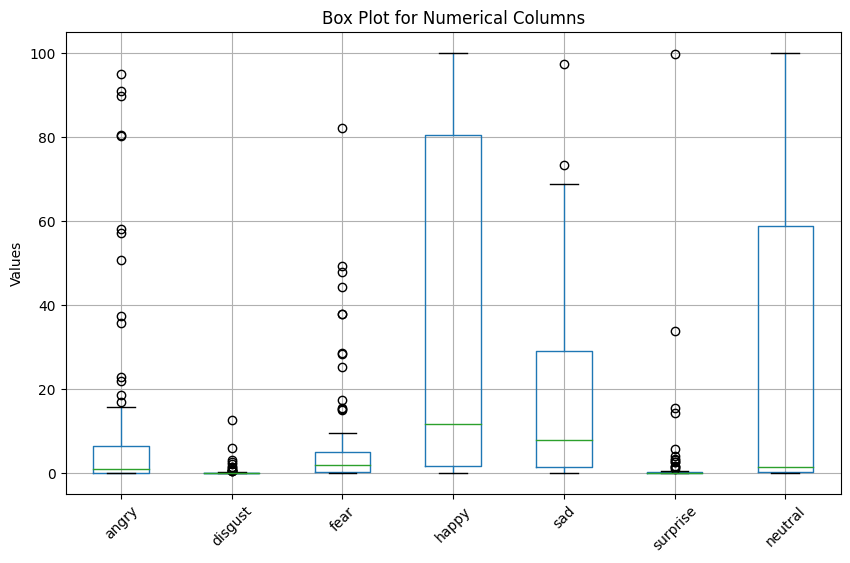


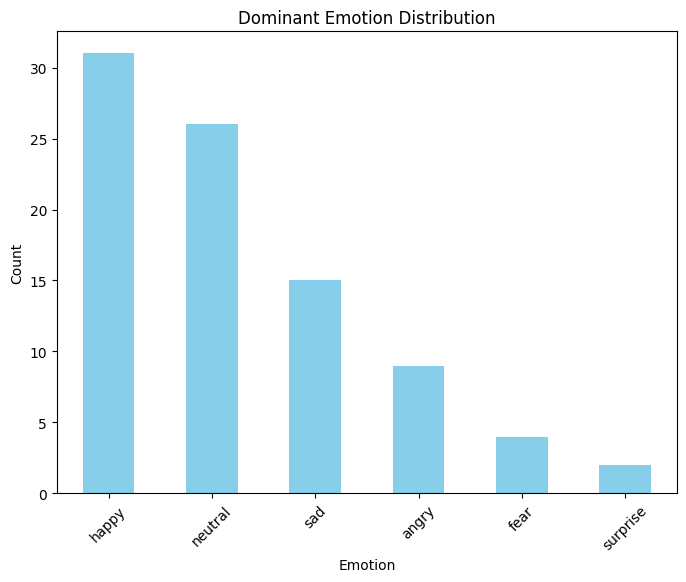




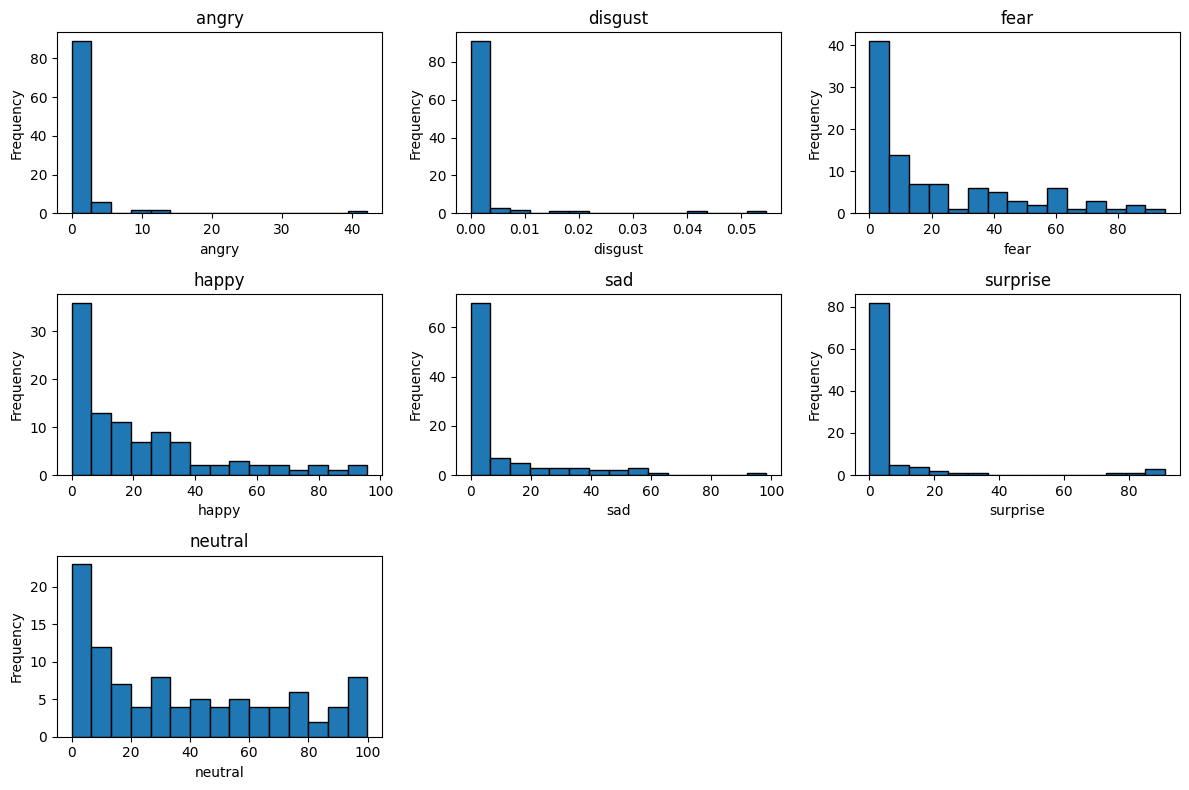
Candidate 2:

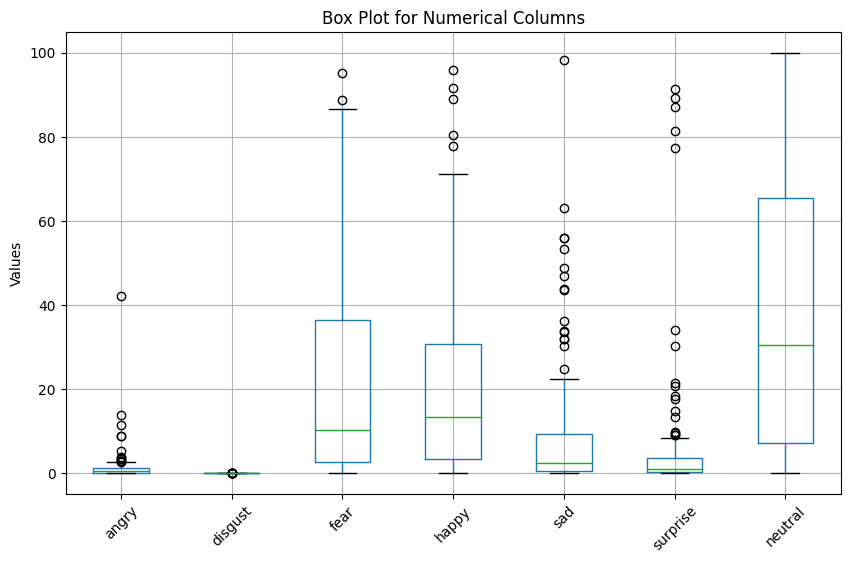


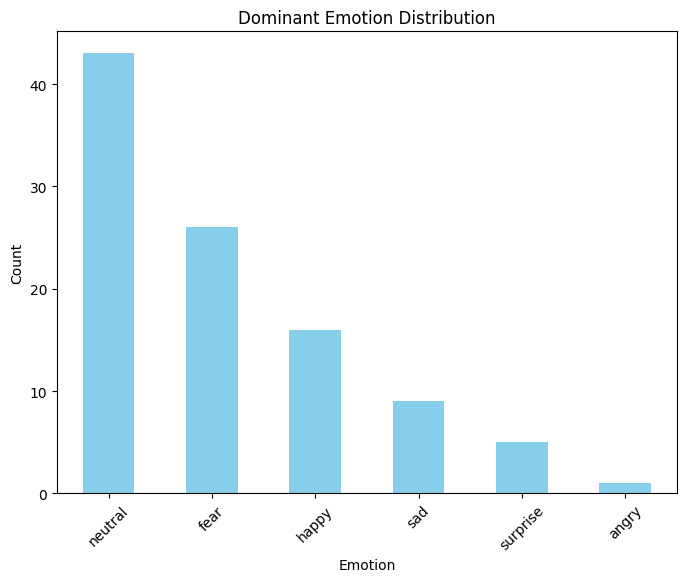




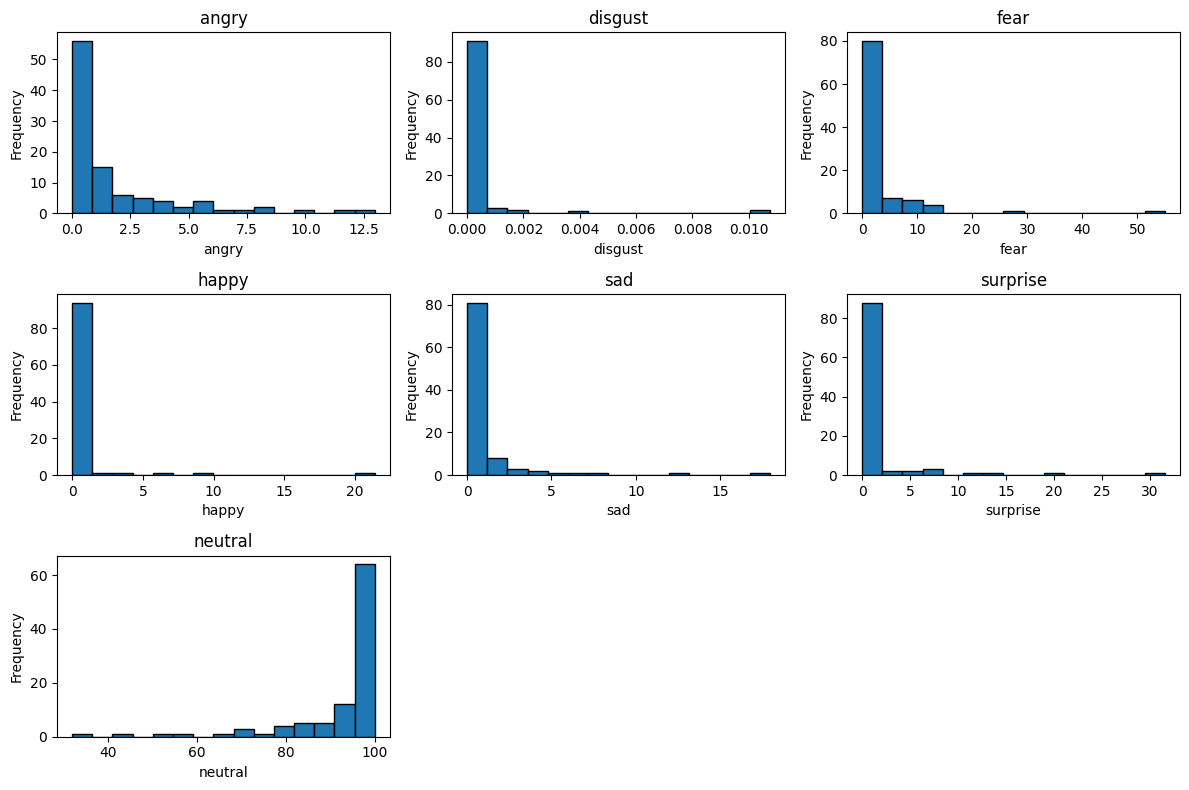
Candidate 3:

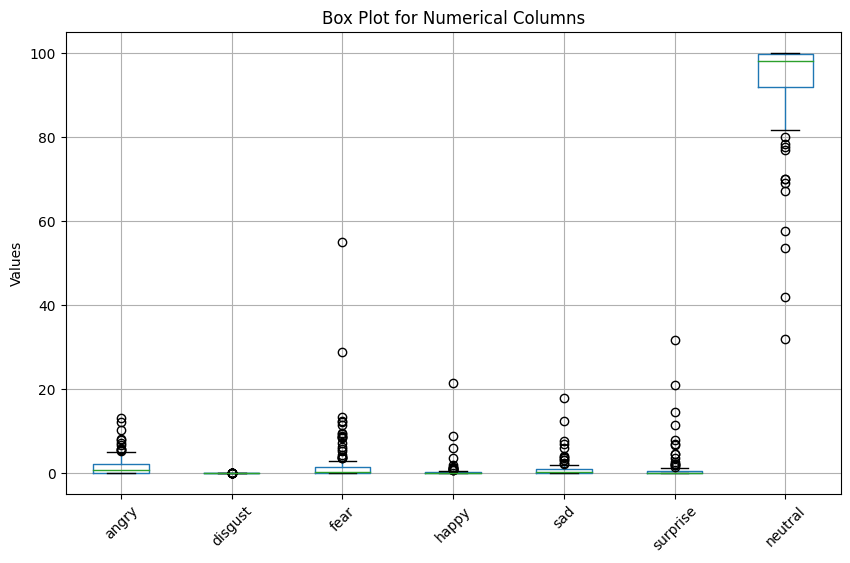






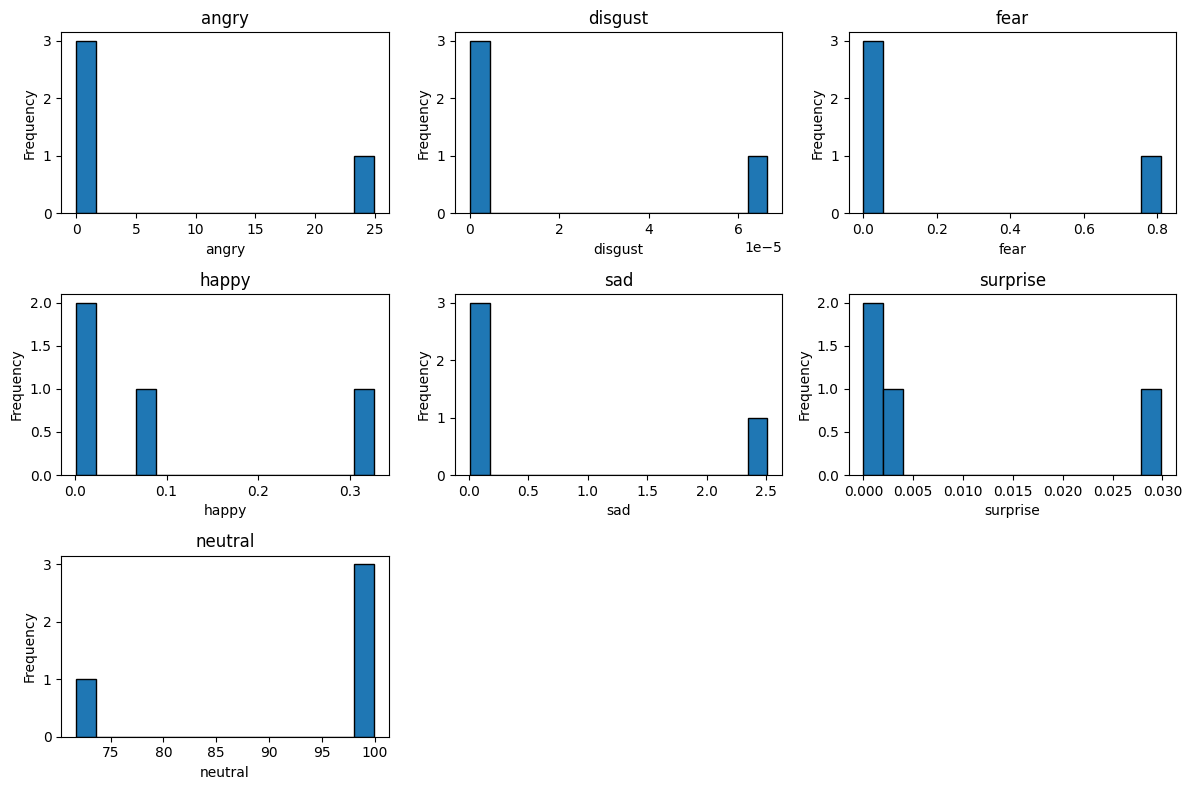
Candidate 4:

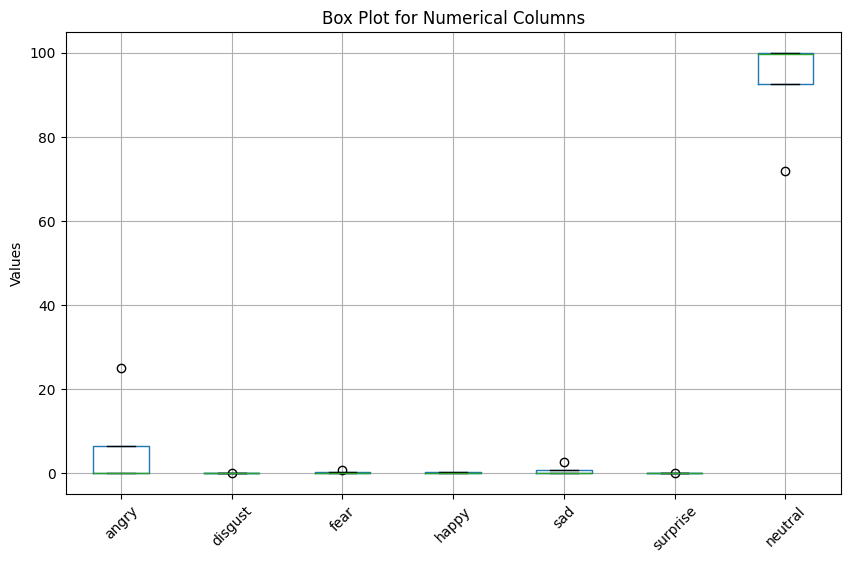


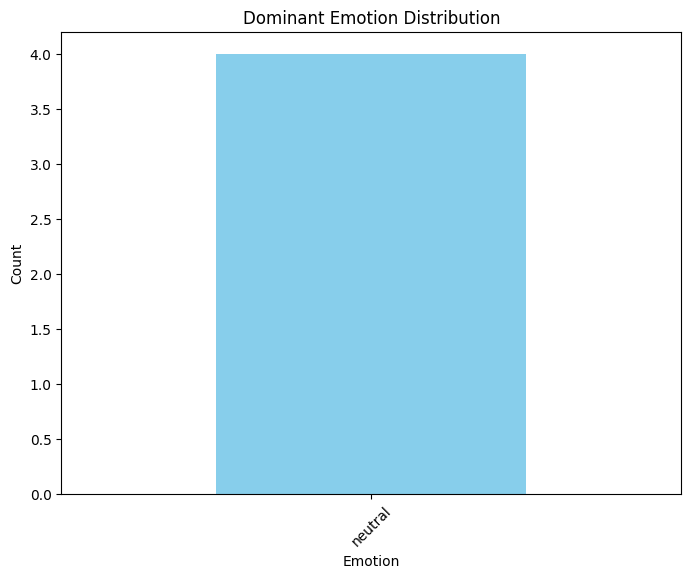




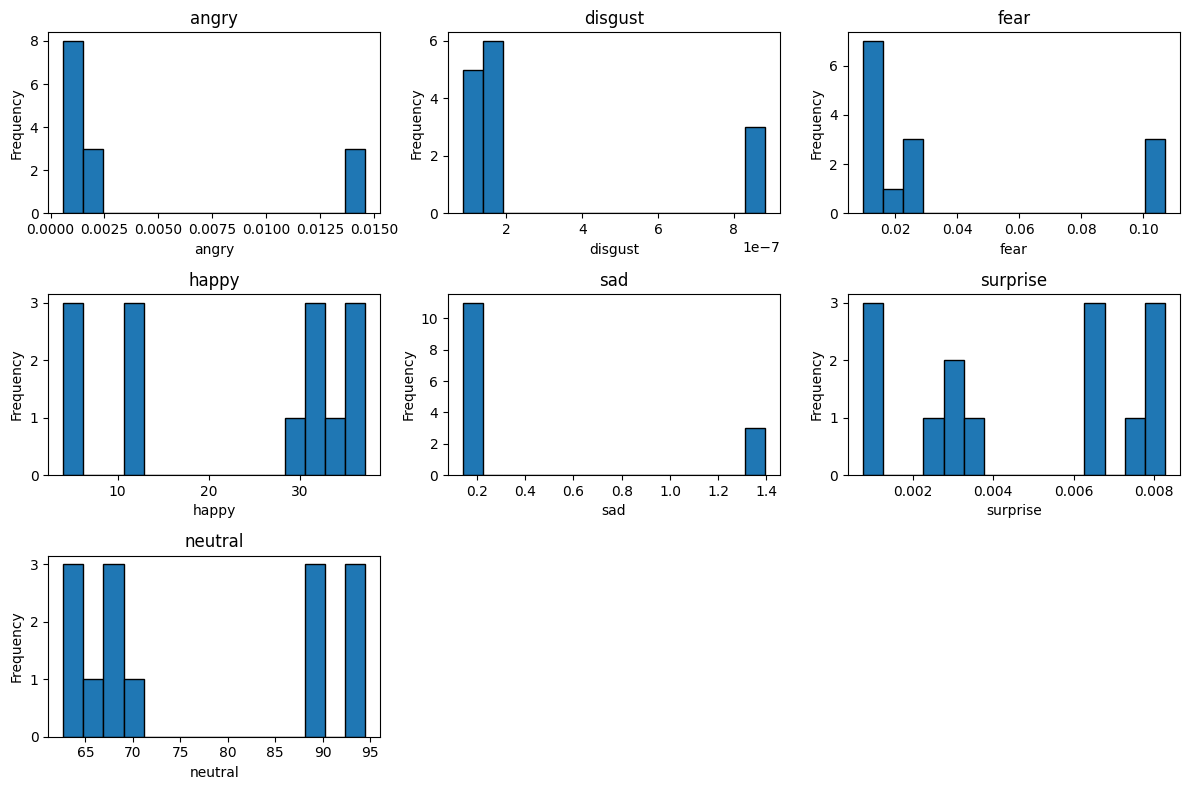
Candidate 5:

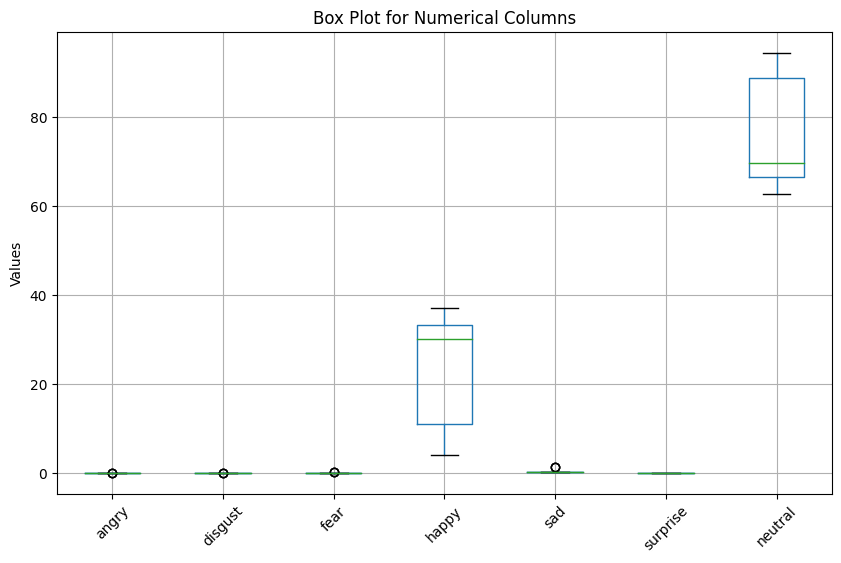


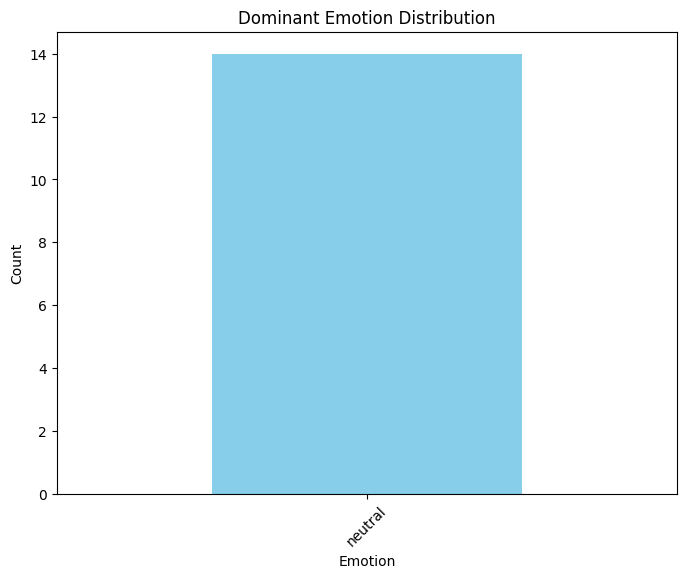




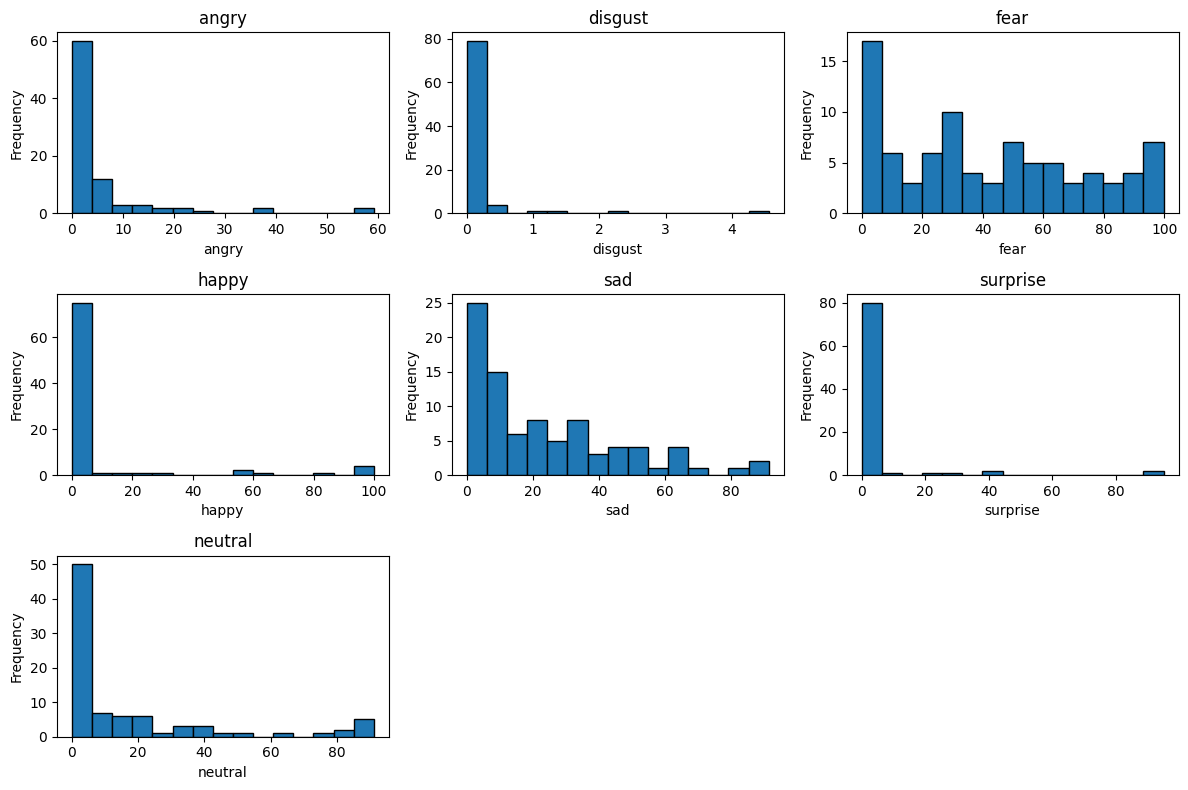
Candidate 6:

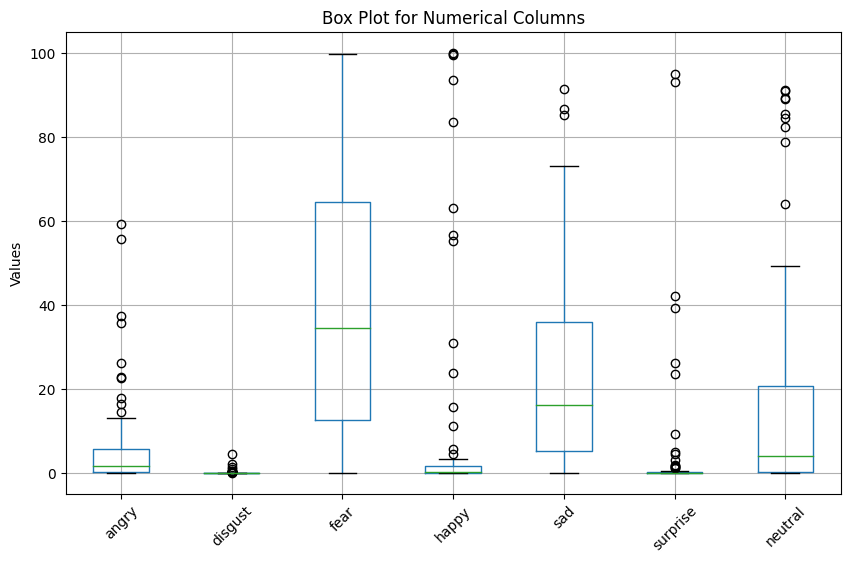


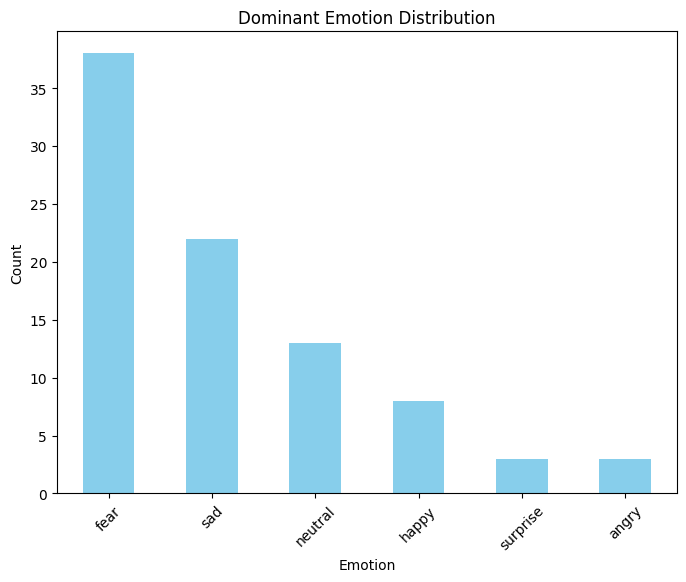




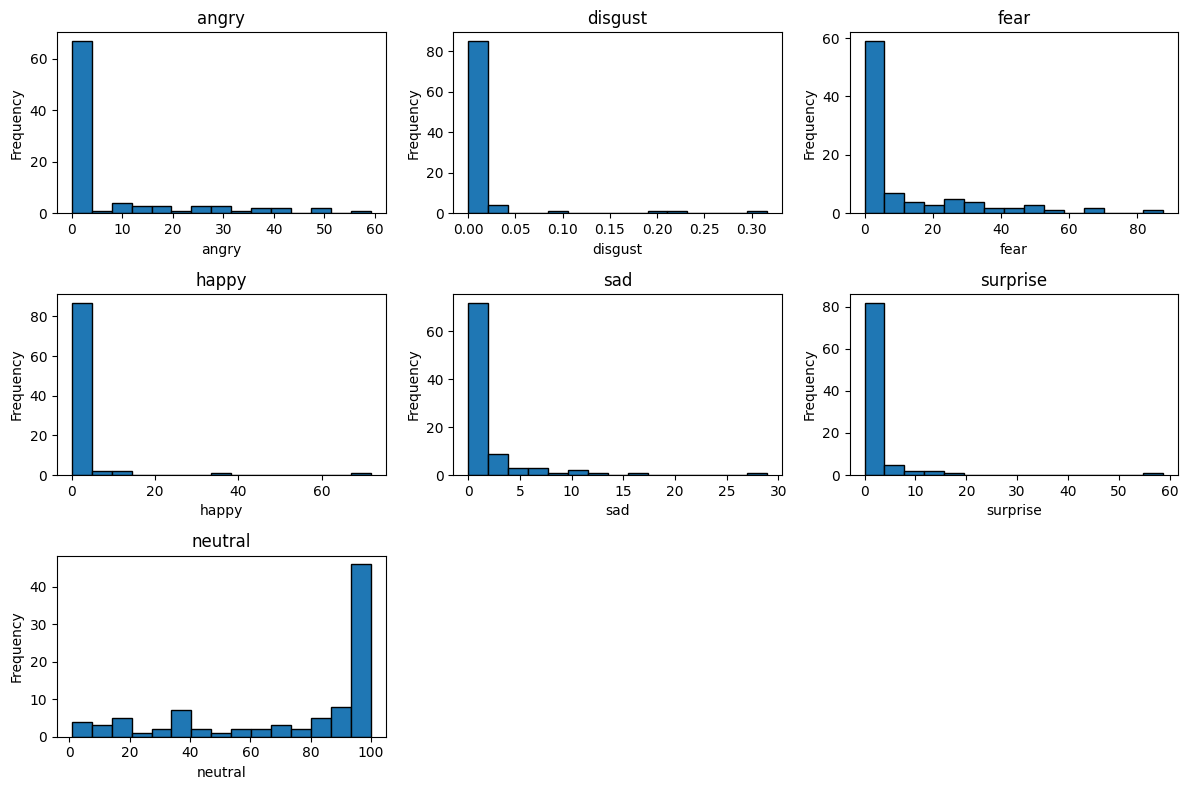
Candidate 7:

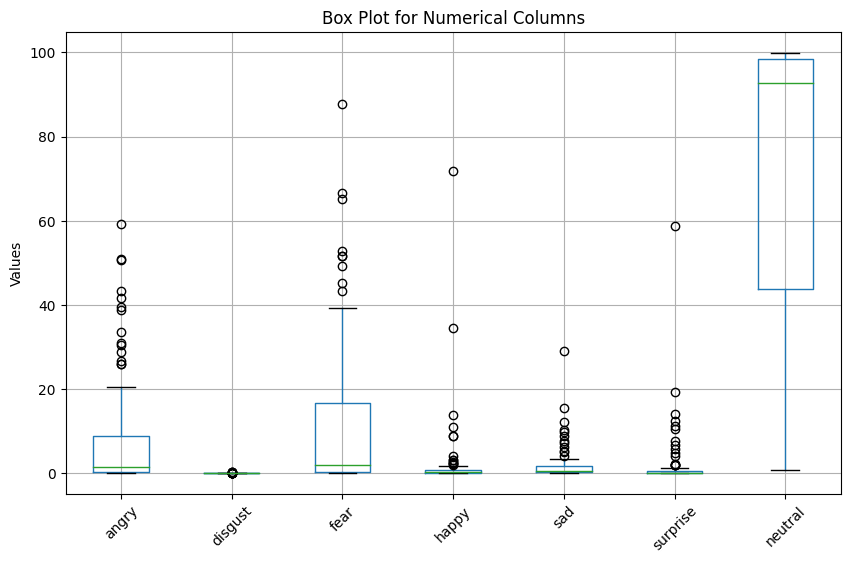


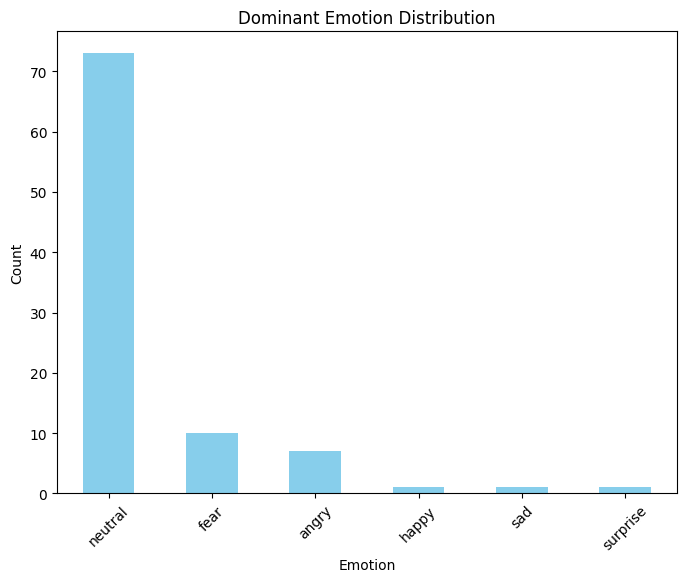




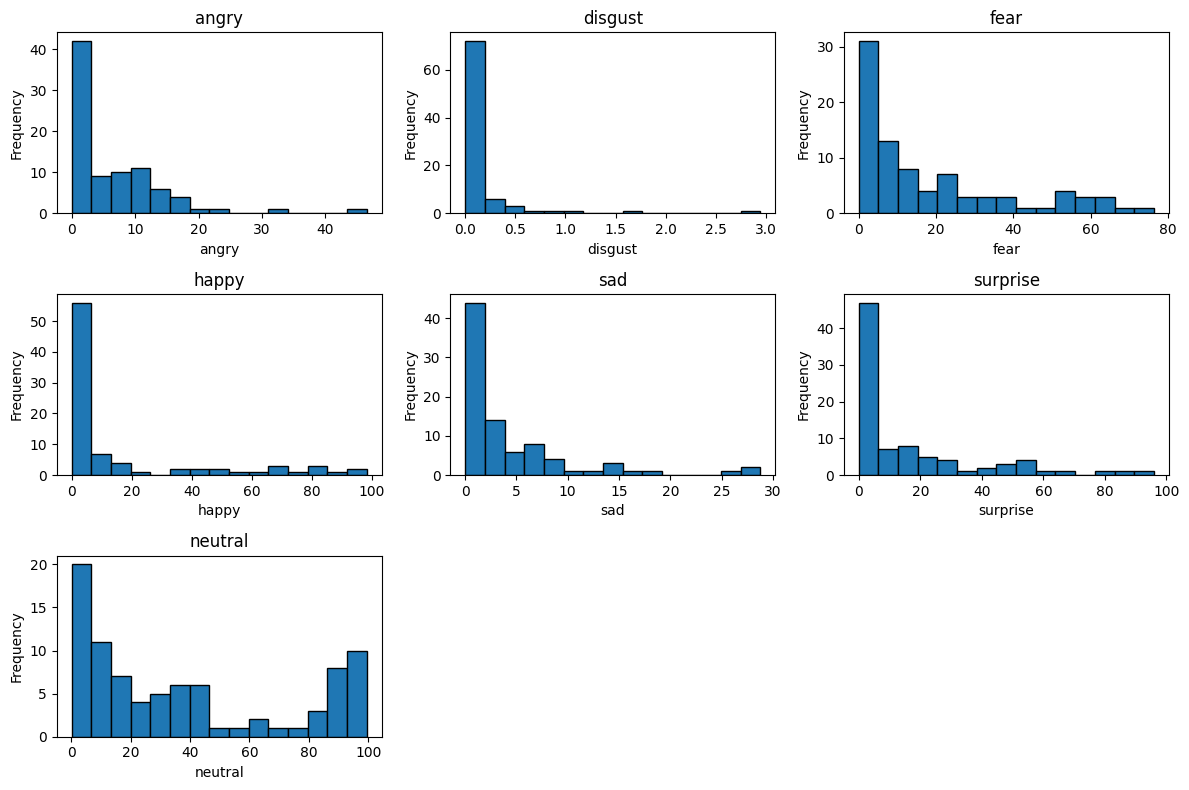
Candidate 8:

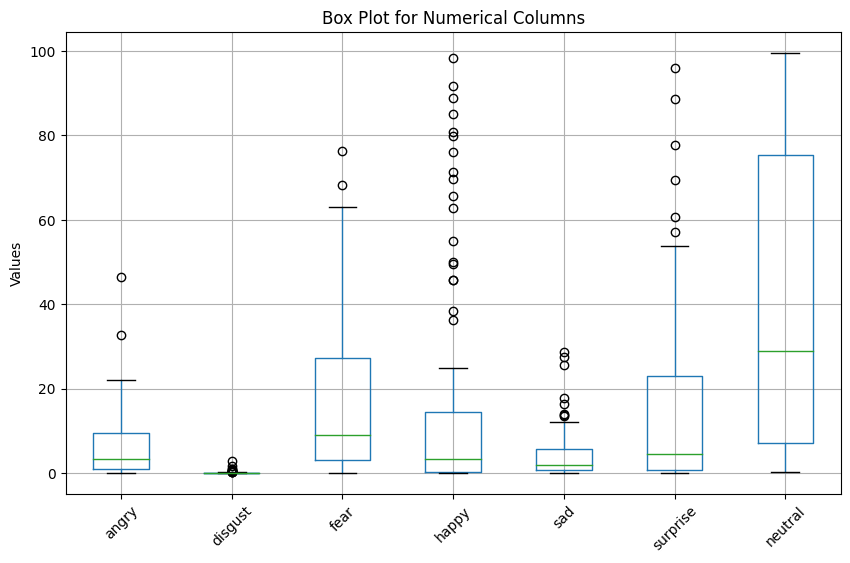


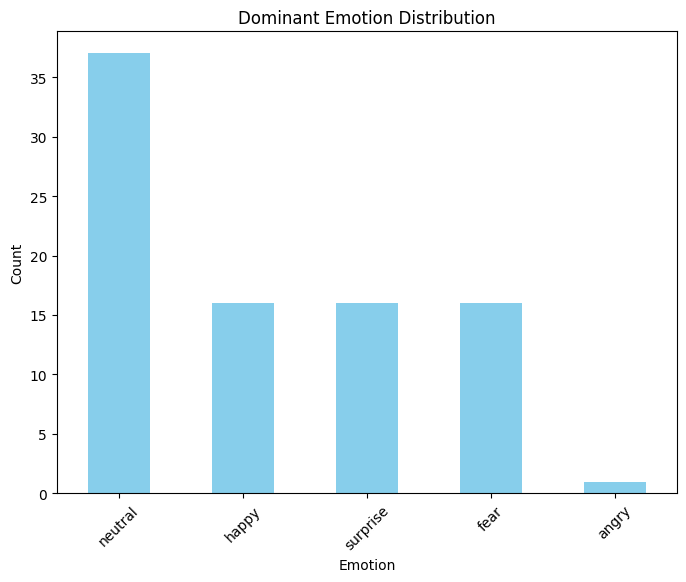




Candidate 9:

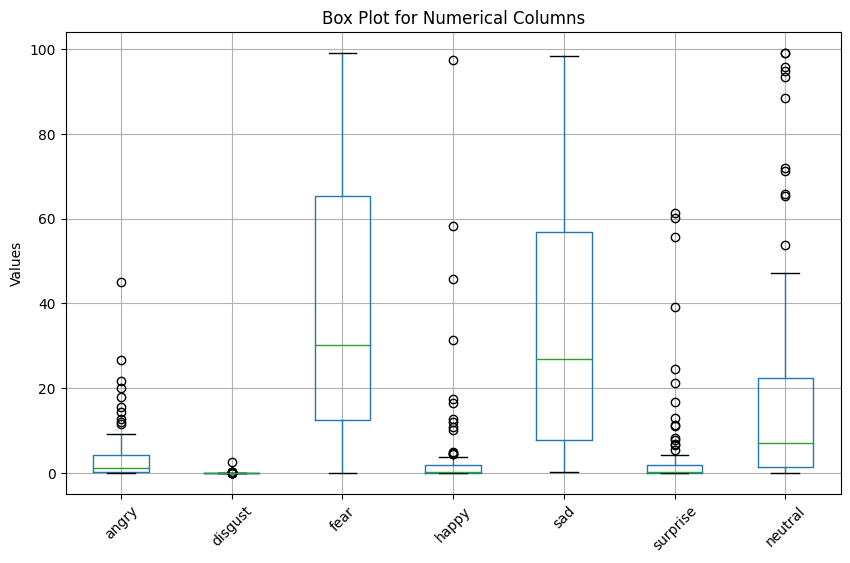


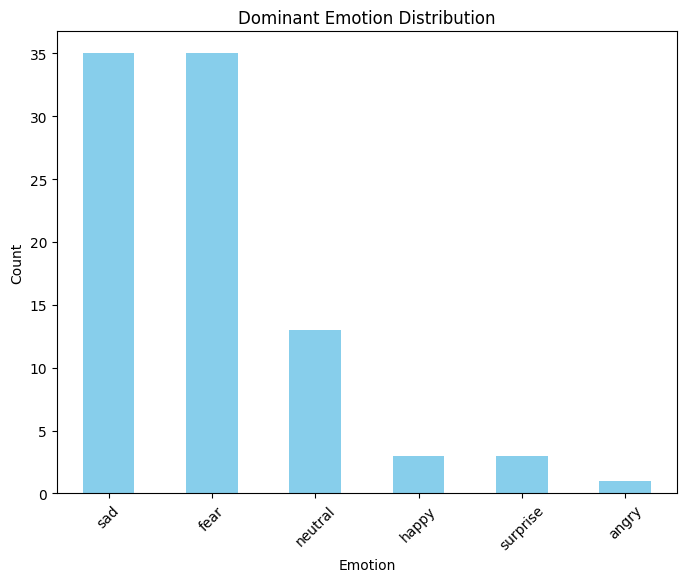




Candidate 10:







## Calculation of overall score assigning an assumed weight:

Here we have assigned assumed weights for the emotions

# Define weights for each emotion

weights = {

    'angry': 1,

    'disgust': 1,

    'fear': 1,

    'happy': 3,

    'sad': 2,

    'surprise': 2,

    'neutral': 3

}

# Calculate weighted scores for each student

df=pd.DataFrame()

We have multipied the weights with the emotion scores:

weighted\_score0=1\*r0[1]+1\*r0[2]+1\*r0[3]+3\*r0[4]+2\*r0[5]+2\*r0[6]+3\*r0[7]

print(weighted\_score0,"1")

weighted\_score1=1\*r1[1]+1\*r1[2]+1\*r1[3]+3\*r1[4]+2\*r1[5]+2\*r1[6]+3\*r1[7]

print(weighted\_score1,"2")

weighted\_score2=1\*r3[1]+1\*r3[2]+1\*r3[3]+3\*r3[4]+2\*r3[5]+2\*r3[6]+3\*r3[7]

print(weighted\_score2,"3")

weighted\_score3=1\*r4[1]+1\*r4[2]+1\*r4[3]+3\*r4[4]+2\*r4[5]+2\*r4[6]+3\*r4[7]

print(weighted\_score3,"4")

weighted\_score4=1\*r5[1]+1\*r5[2]+1\*r5[3]+3\*r5[4]+2\*r5[5]+2\*r5[6]+3\*r5[7]

print(weighted\_score4,"5")

weighted\_score5=1\*r6[1]+1\*r6[2]+1\*r6[3]+3\*r6[4]+2\*r6[5]+2\*r6[6]+3\*r6[7]

print(weighted\_score5,"6")

weighted\_score6=1\*r7[1]+1\*r7[2]+1\*r7[3]+3\*r7[4]+2\*r7[5]+2\*r7[6]+3\*r7[7]

print(weighted\_score6,"7")

weighted\_score7=1\*r8[1]+1\*r8[2]+1\*r8[3]+3\*r8[4]+2\*r8[5]+2\*r8[6]+3\*r8[7]

print(weighted\_score7,"8")

weighted\_score8=1\*r9[1]+1\*r9[2]+1\*r9[3]+3\*r9[4]+2\*r9[5]+2\*r9[6]+3\*r9[7]

print(weighted\_score8,"9")

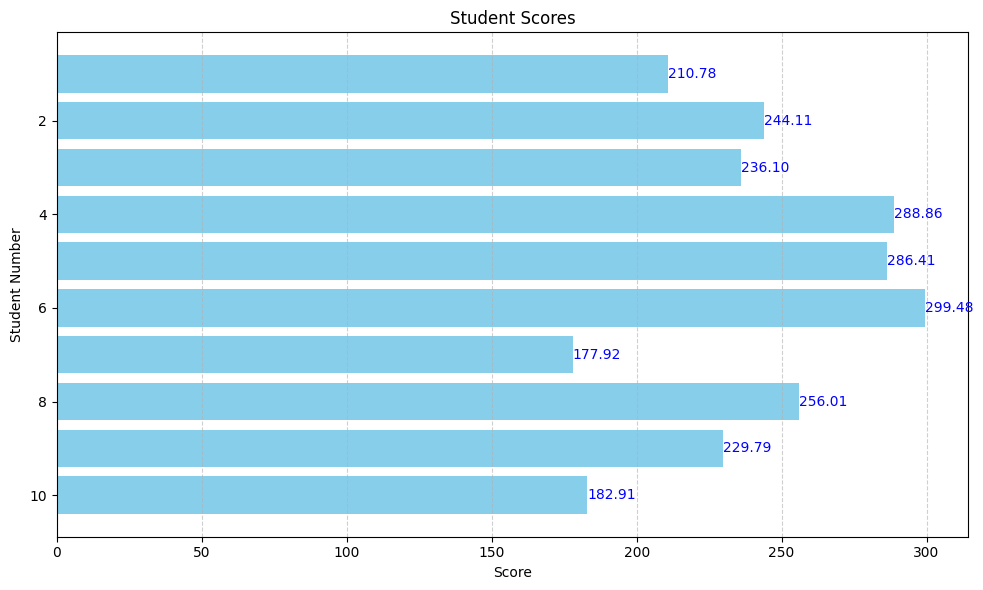
weighted\_score9=1\*r10[1]+1\*r10[2]+1\*r10[3]+3\*r10[4]+2\*r10[5]+2\*r10[6]+3\*r10[7]

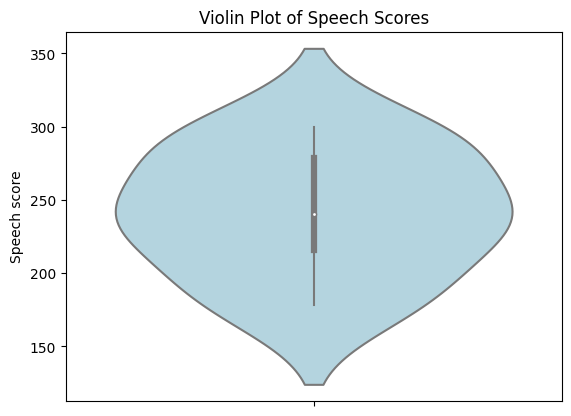
print(weighted\_score9,"10")

here r0, r1, r2 ….. r10 are the mean values of the respective emotions of each of the candidate

* 210.77823699836864 1
* 244.10549888402204 2
* 236.10291955049618 3
* 288.8648272225419 4
* 286.4087061038691 5
* 299.4796434348859 6
* 177.9238883899423 7
* 256.011384222267 8
* 229.78589032343052 9
* 182.91470031850952 10

Plot of the following data:





The violin plot depicts the distribution of speech scores, showing the density of scores across different values, along with key summary statistics(like the median).

# EDA ON TRANSCRIPTS\_DATA

Contains:

* Box plot of sentiment distribution vs sentiment score
* Emotion scores overtime
* Correlation heatmap
* Text vs speech speed
* Time duration for the text to be spoken vs text

Insights:

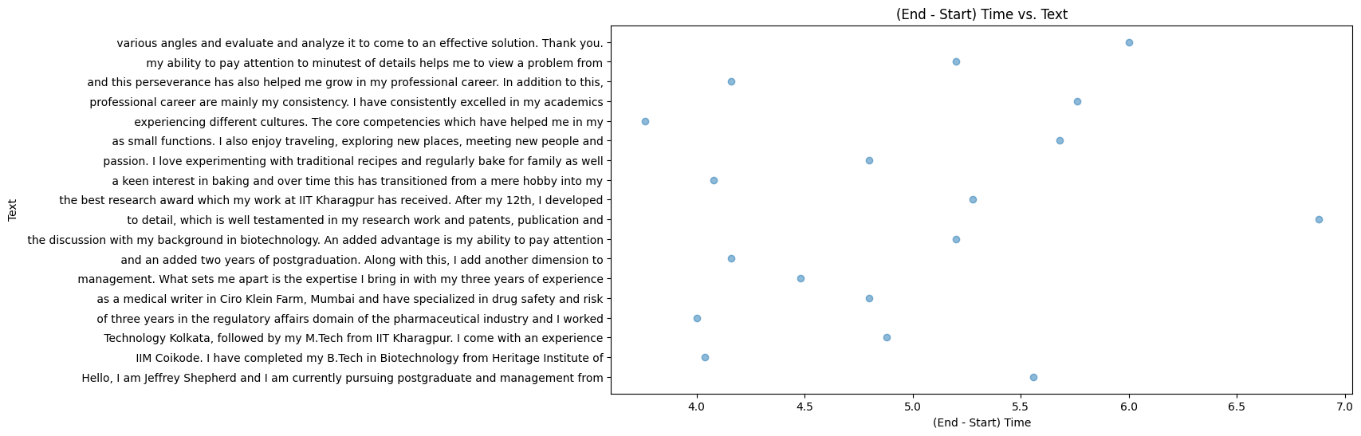
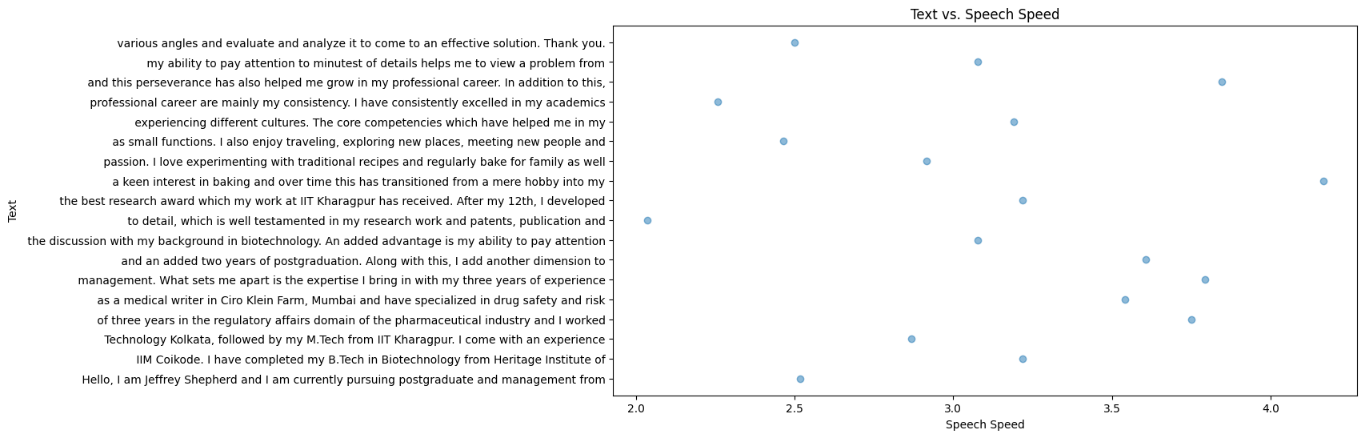
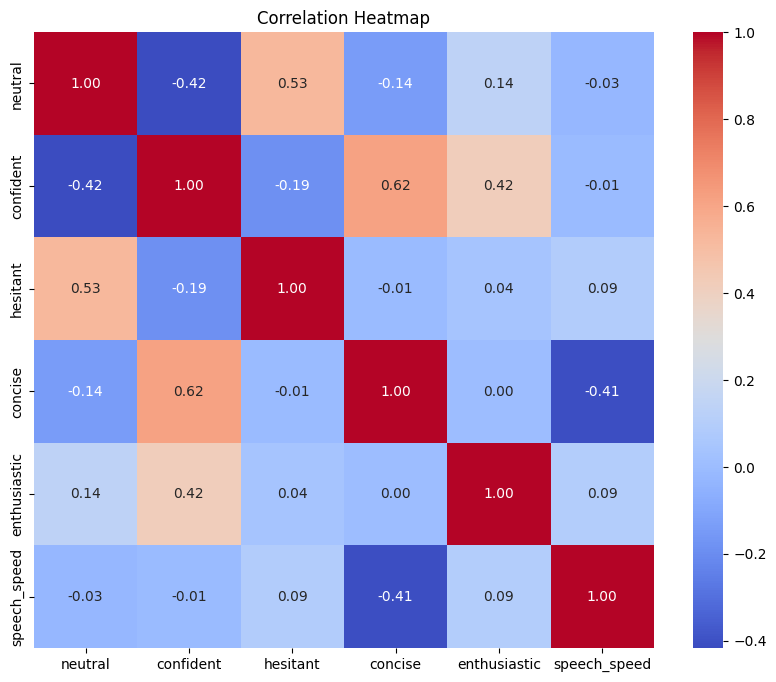
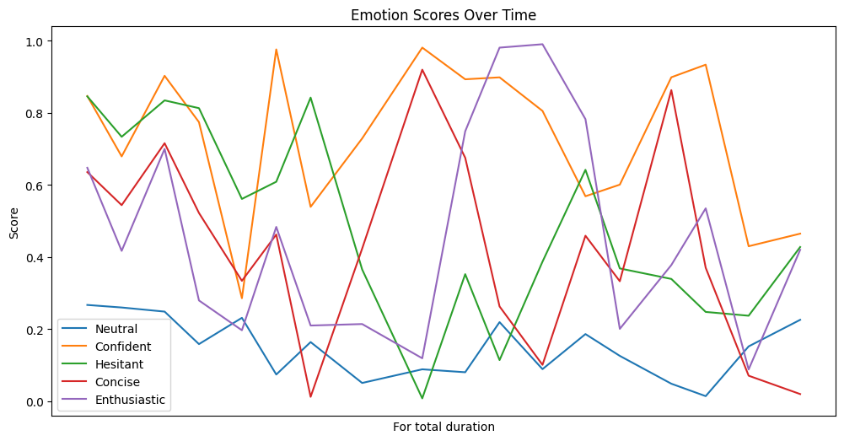
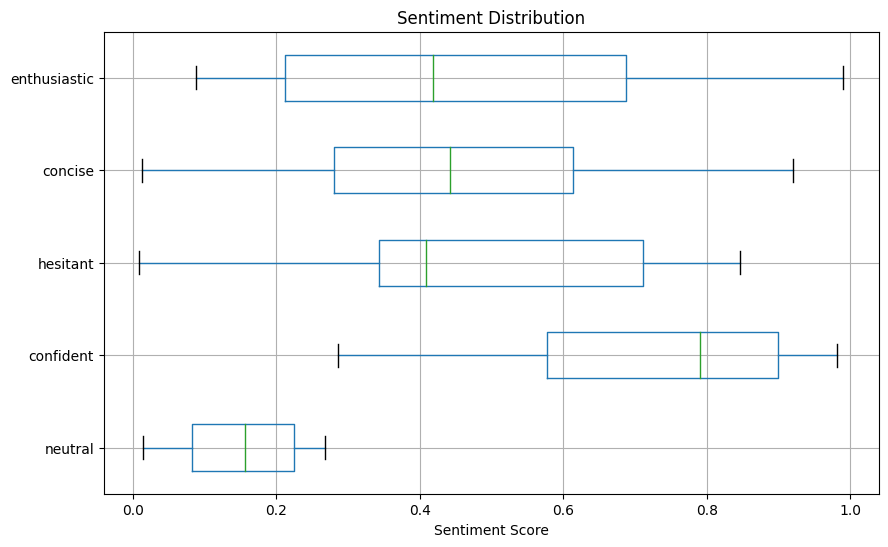
* From the box plot we can know where the maximum data lies for a particular emotion (it is easier for a interviewer to screen, for example, if an interviewer is looking a confident candidate then he can just look at all the 10 candidates' boxplot and screen accordingly)
* From heatmap in case of:

Positive Correlation: When two attributes have a positive correlation (value close to 1), it means that as one attribute increases, the other tends to increase as well. For example, if 'confident' and 'enthusiastic' have a strong positive correlation, it suggests that when someone is confident, they are also likely to be enthusiastic.

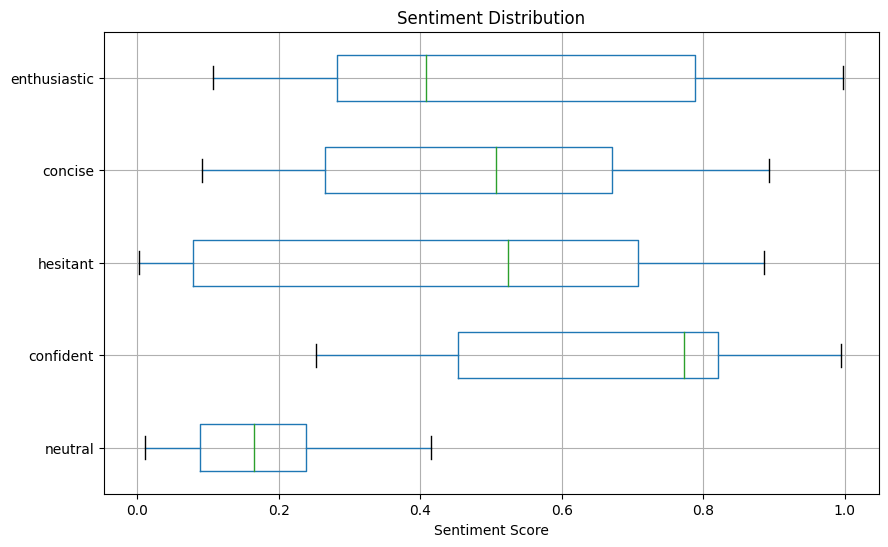
Negative Correlation: A negative correlation (value close to -1) indicates that as one attribute increases, the other tends to decrease. For instance, if 'hesitant' and 'concise' have a strong negative correlation, it implies that when someone is hesitant, they are less likely to be concise in their speech.

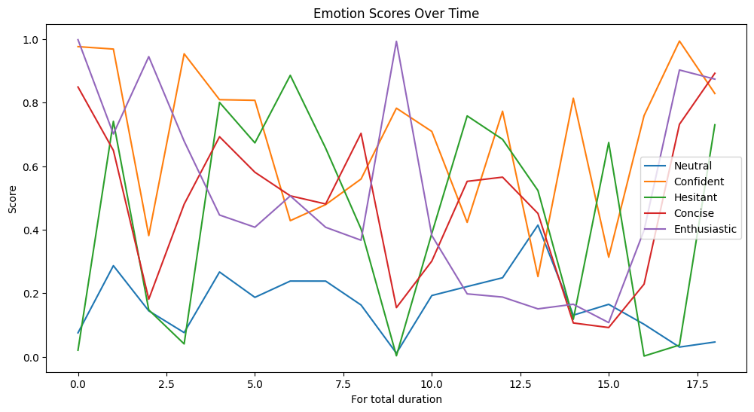
* From the scatter plots of text, we can observe the candidate's time taken for a sentence

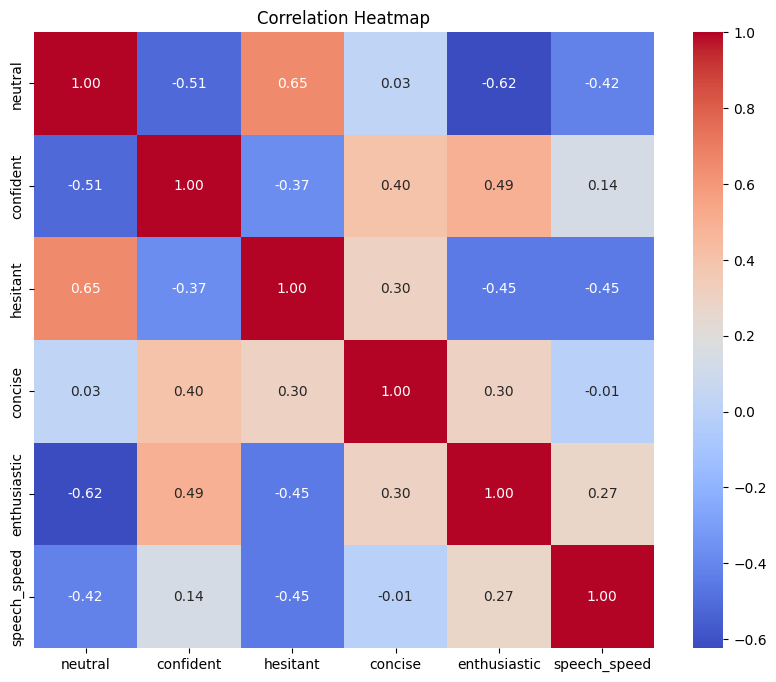
1)

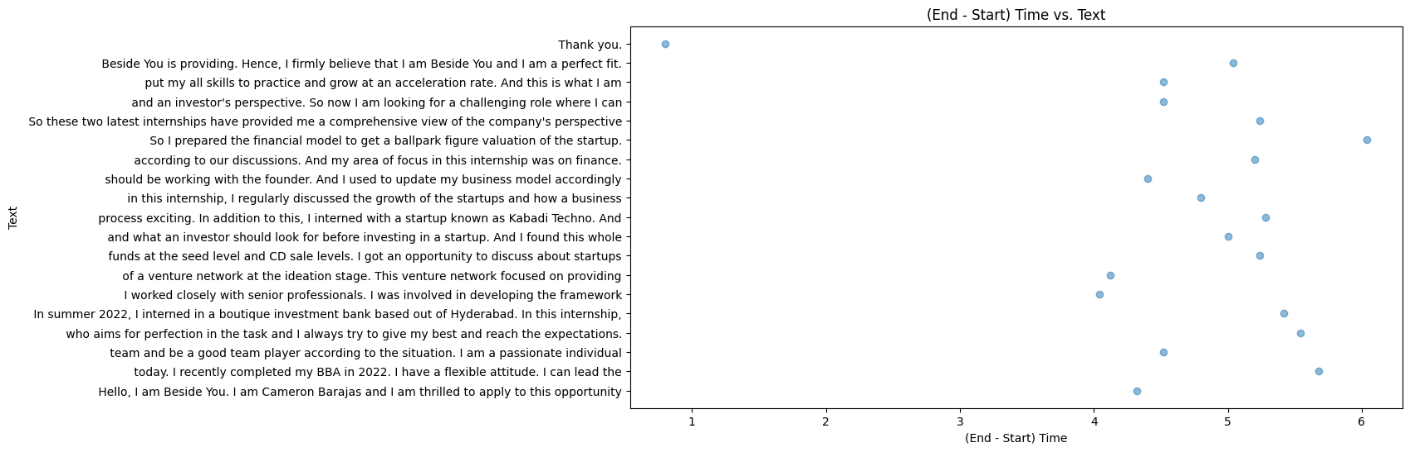
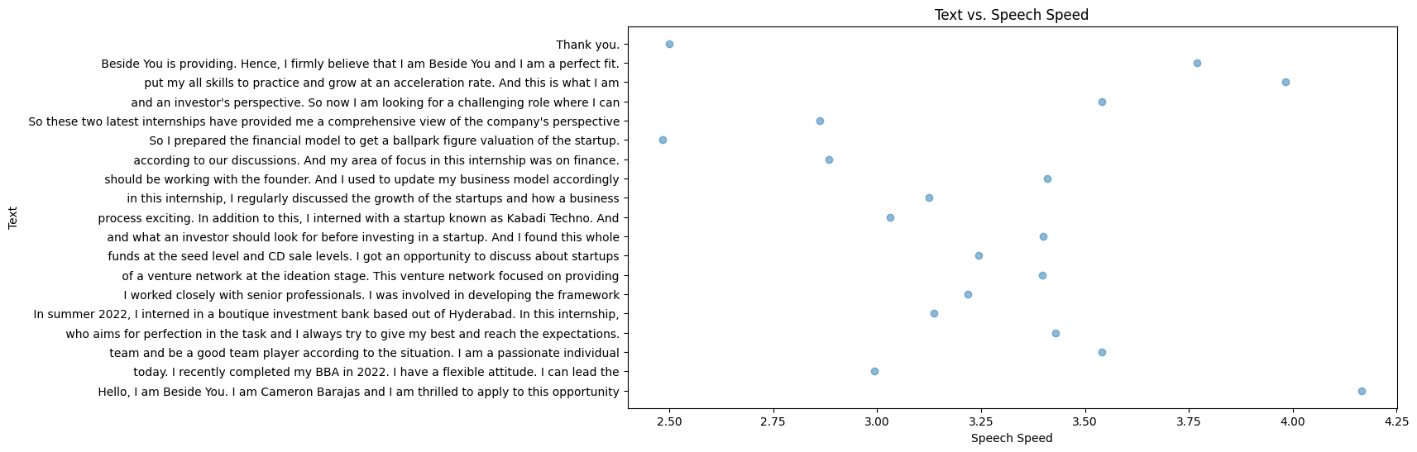


2)

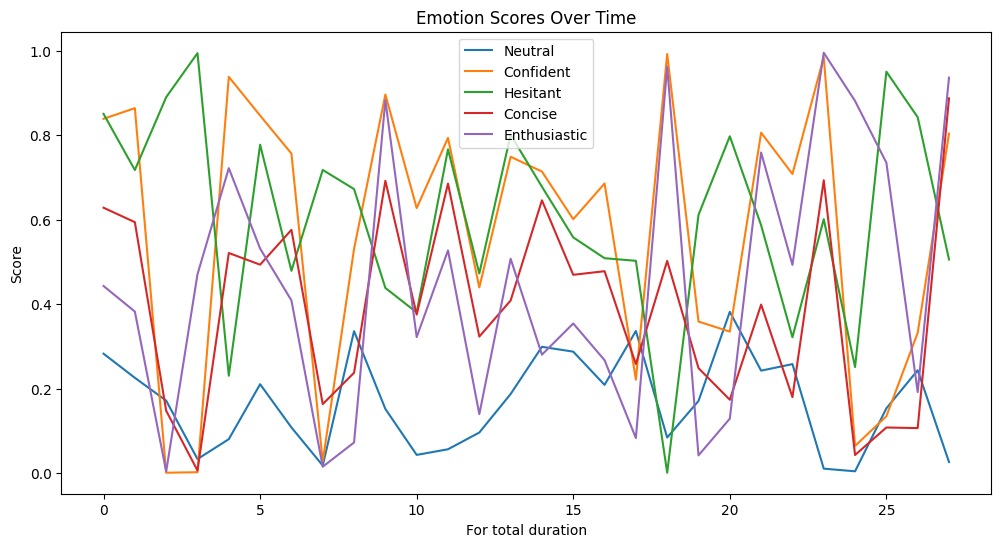
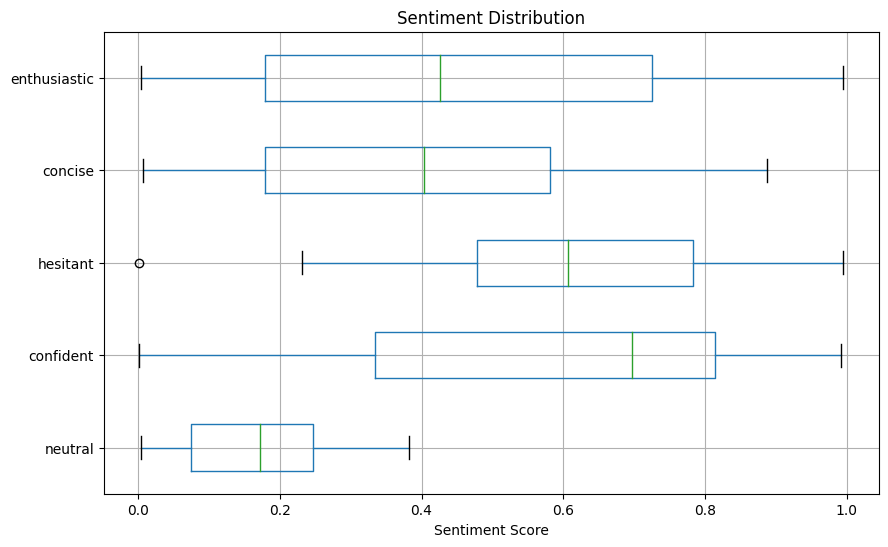


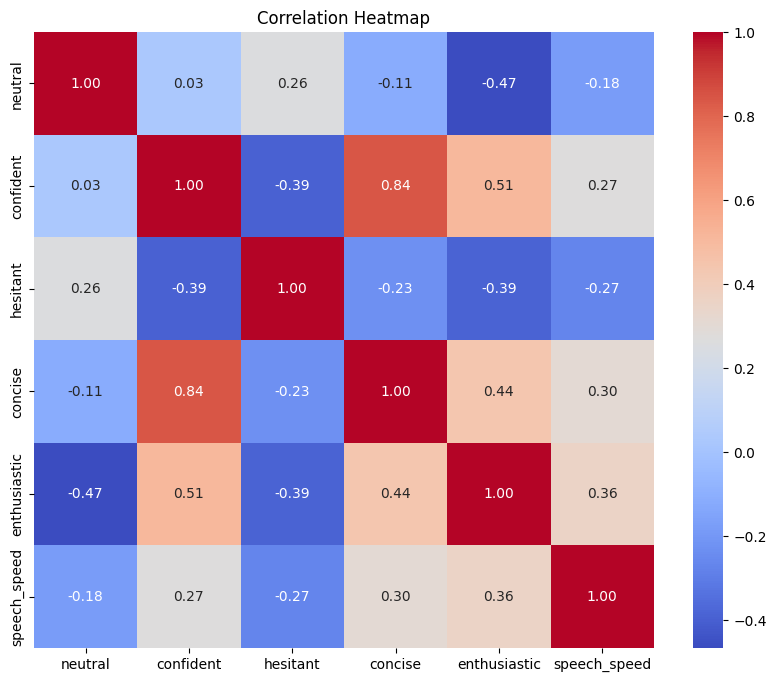


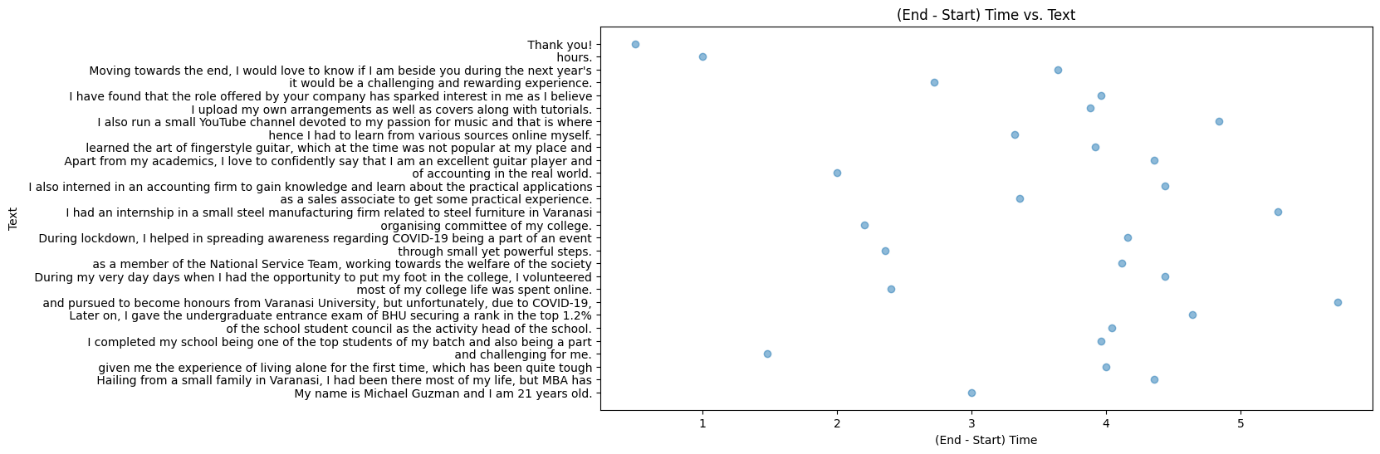


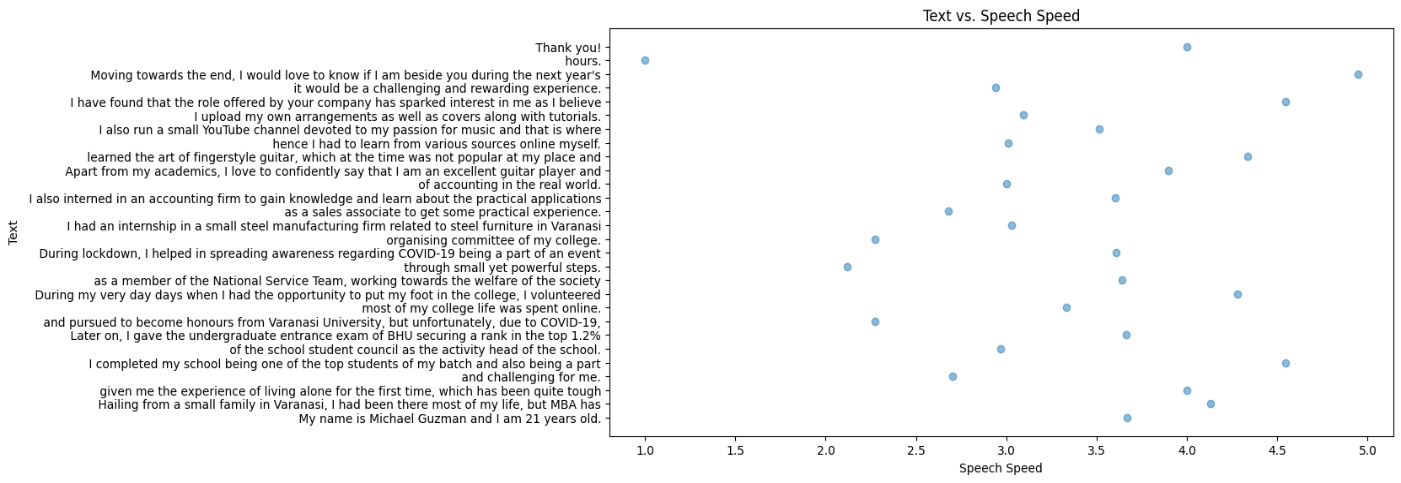


3)

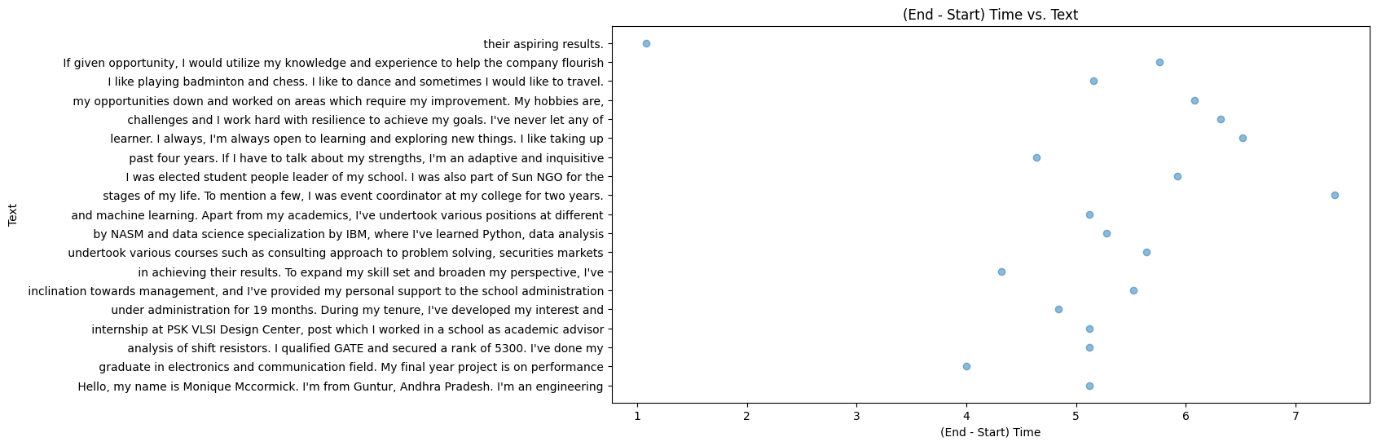
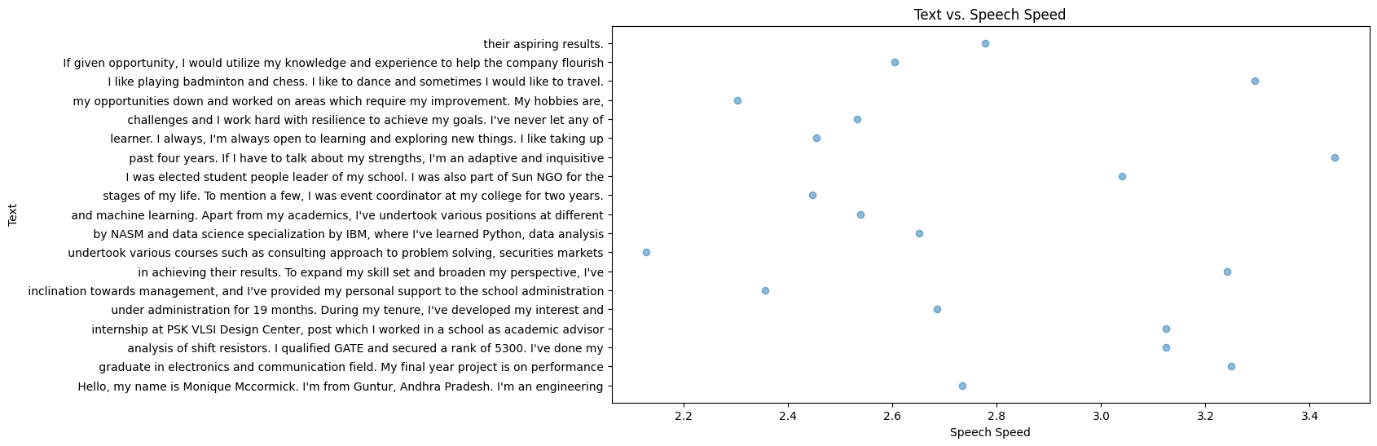
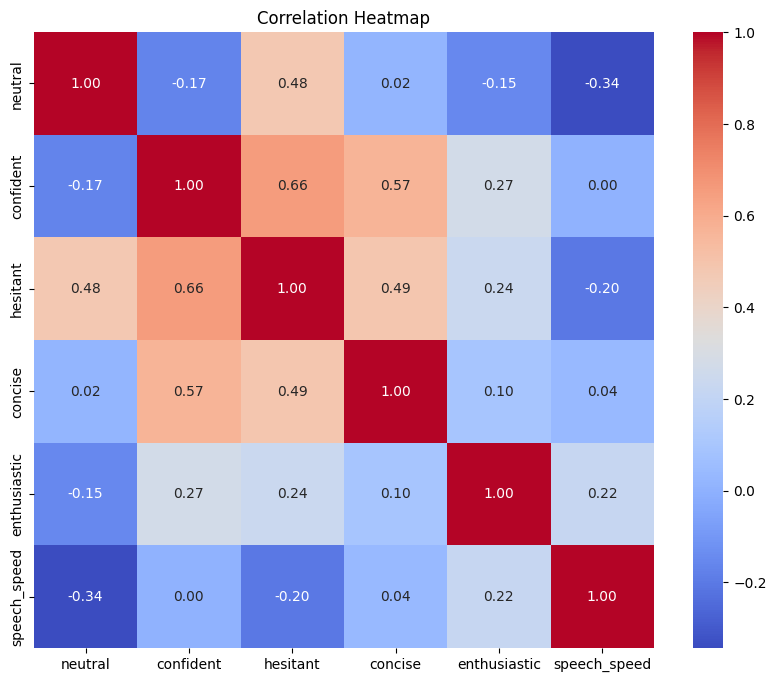
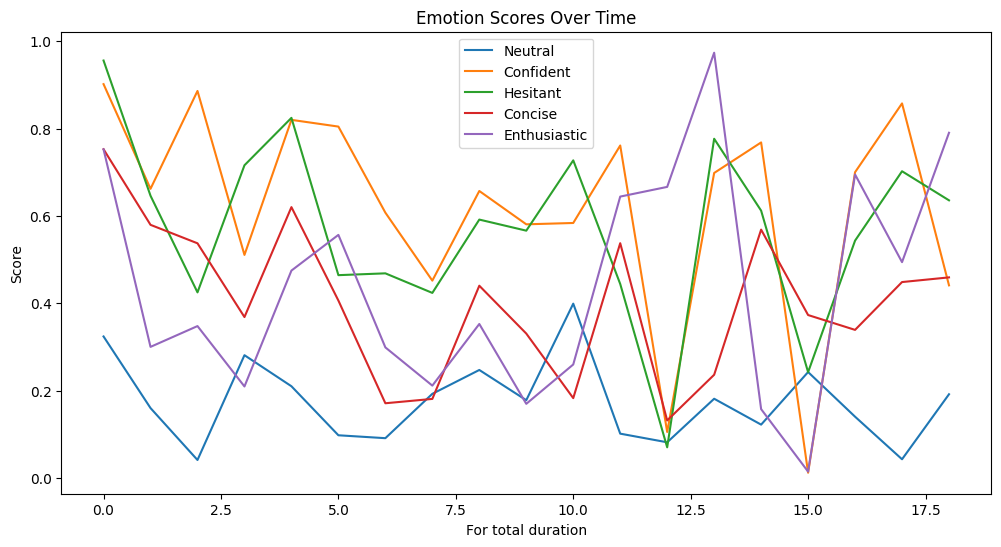
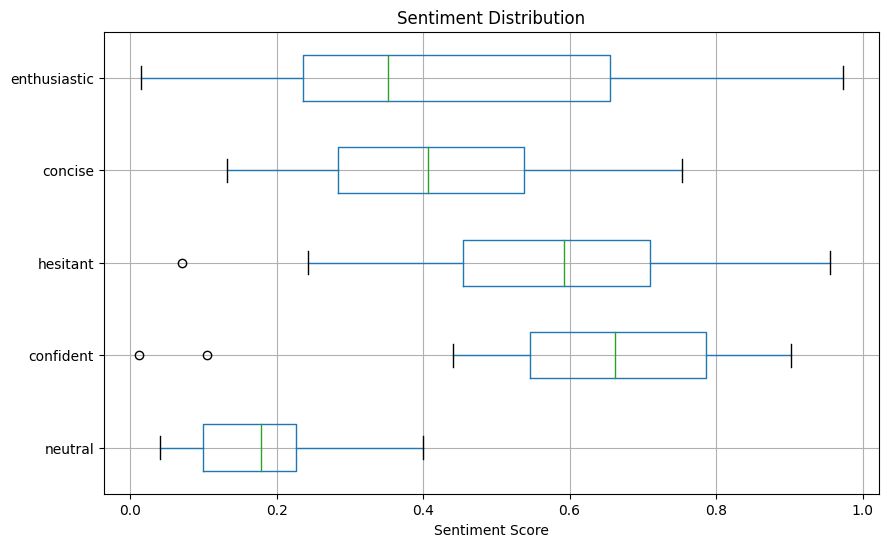




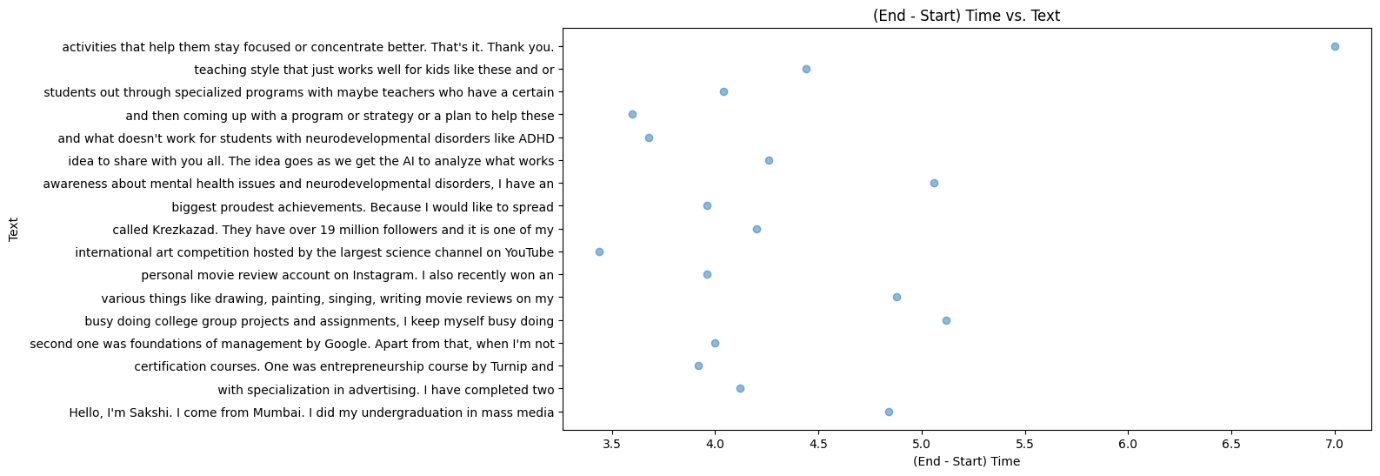
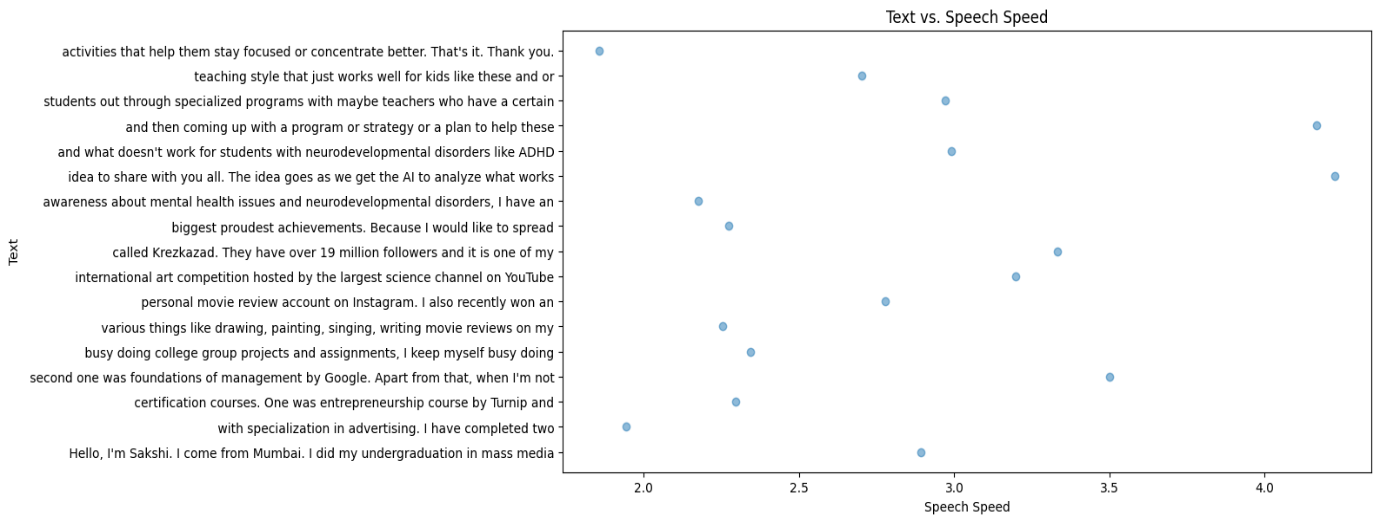
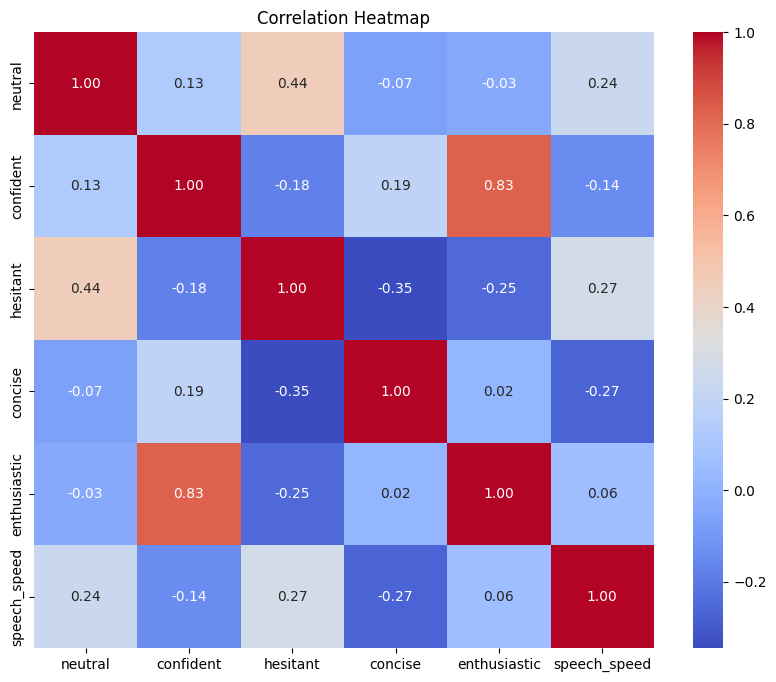
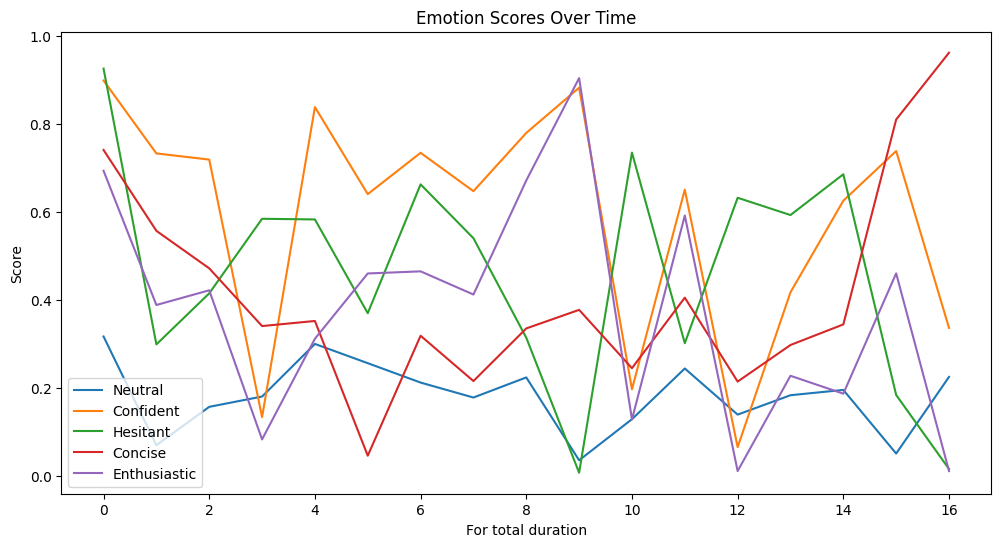
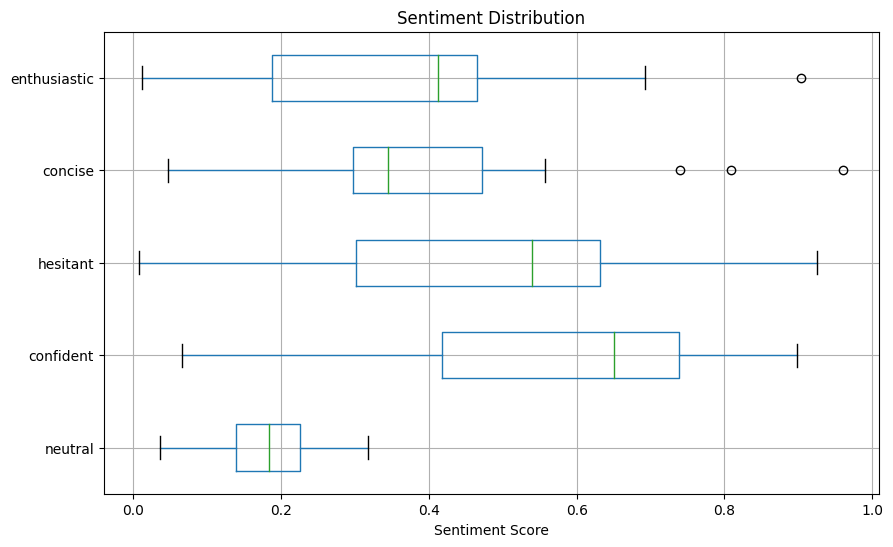




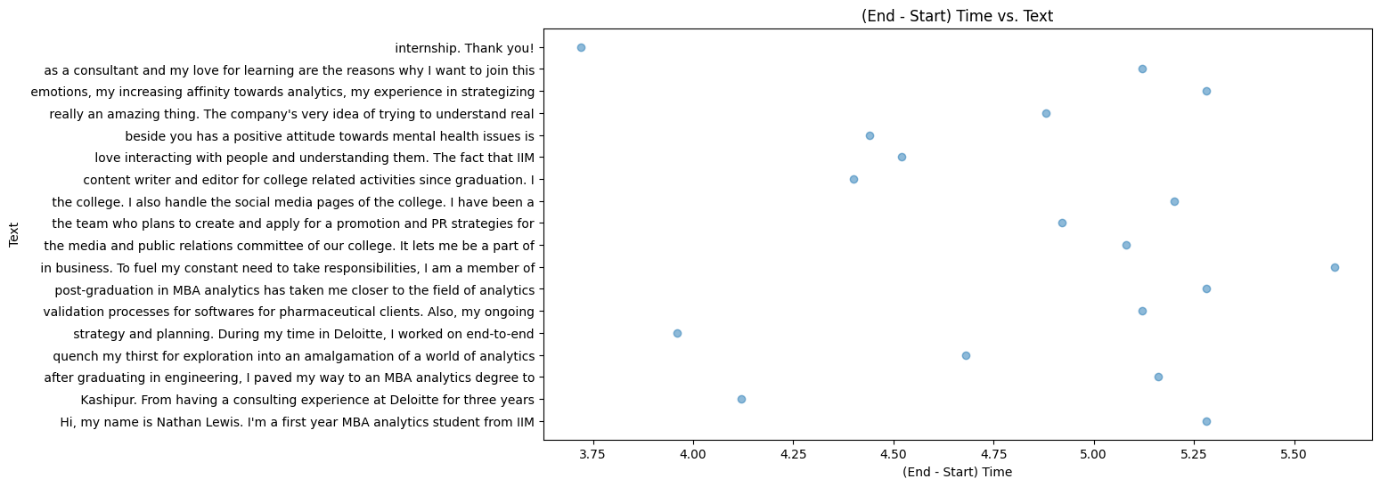
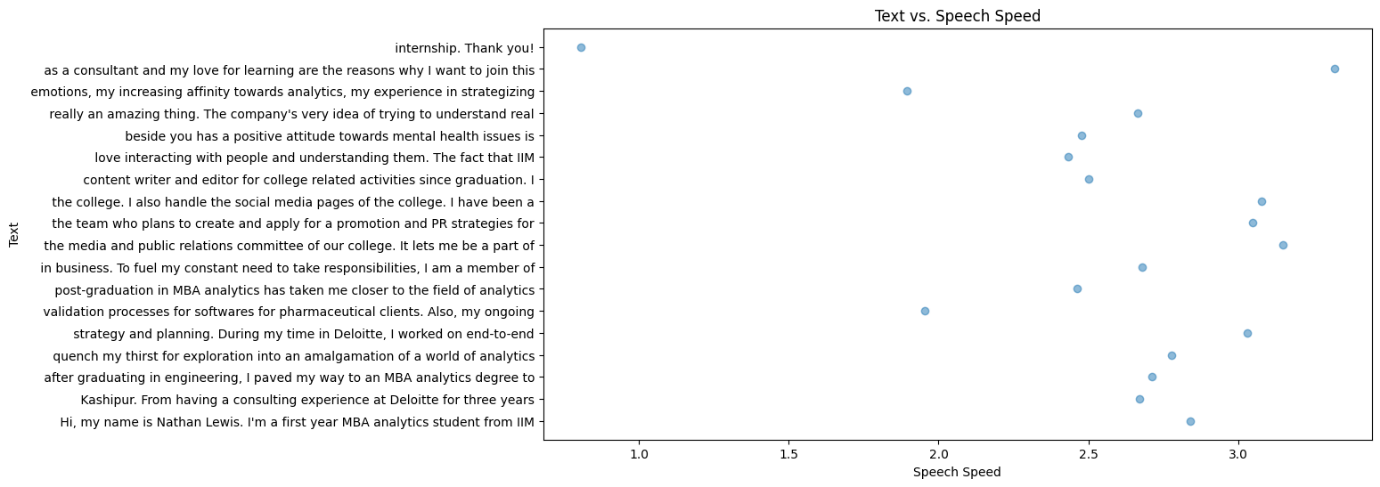
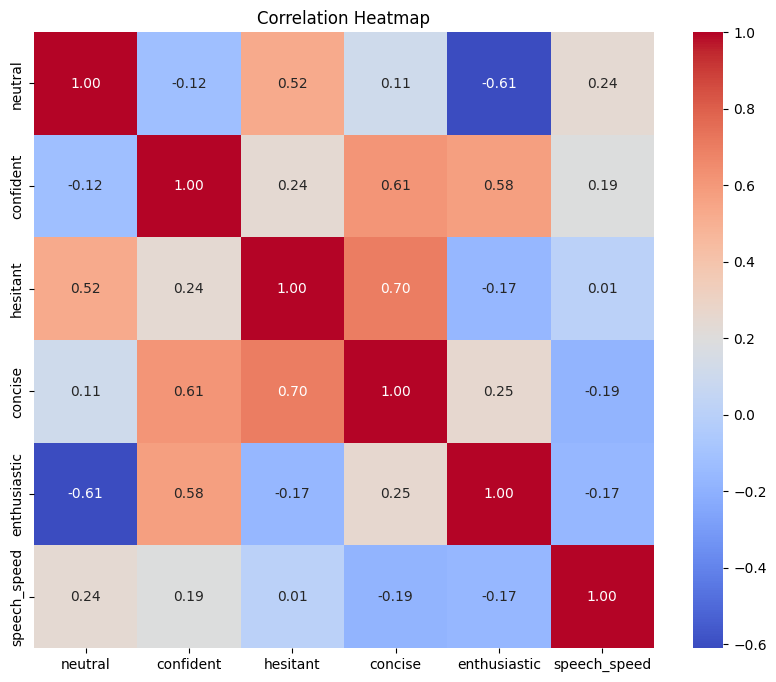
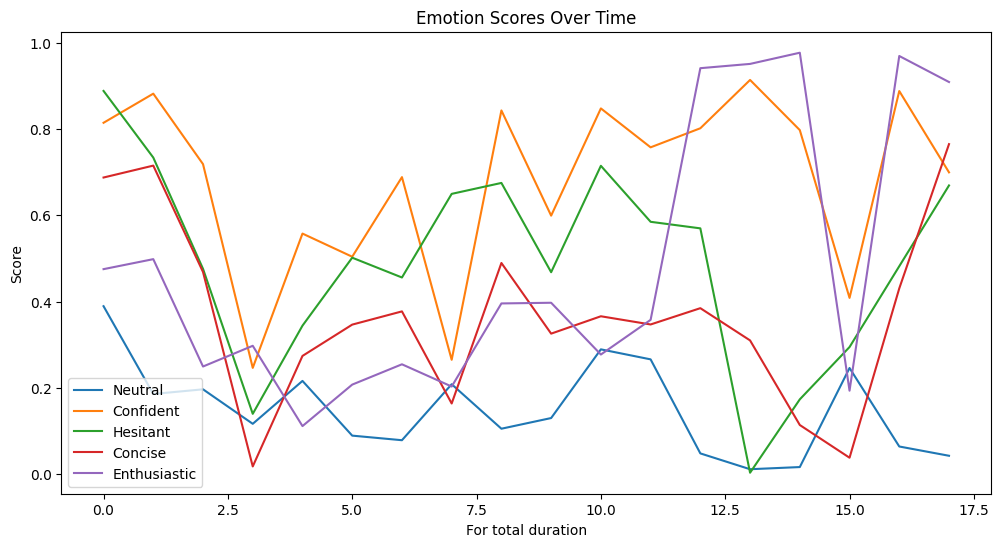
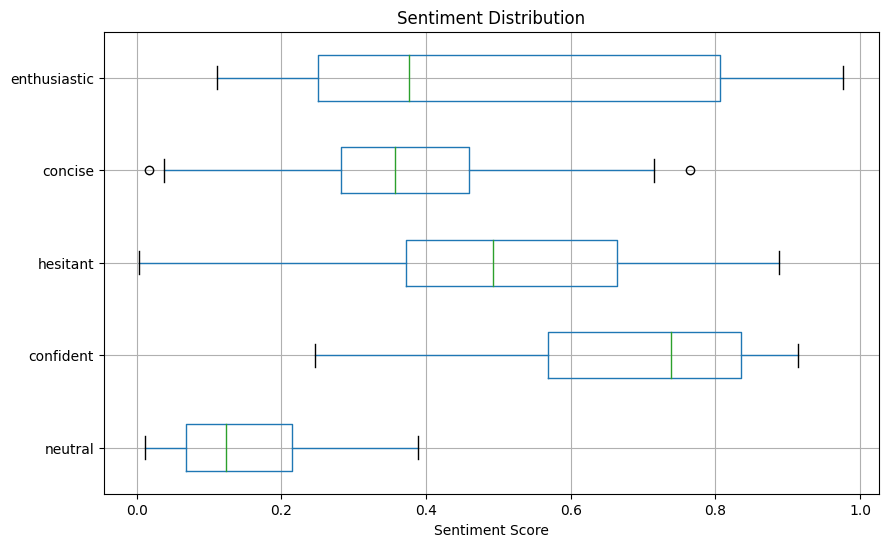
4)



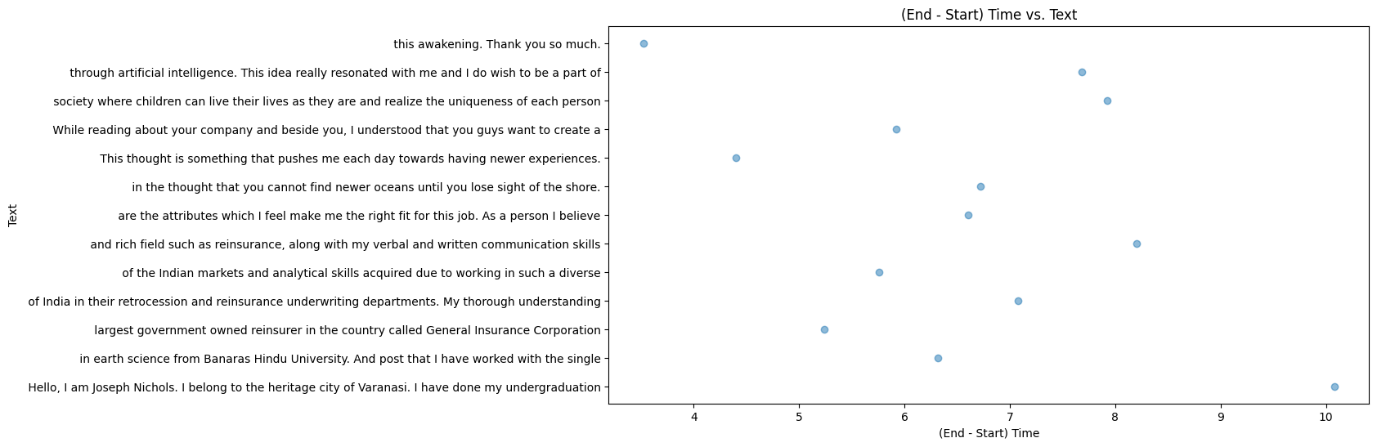
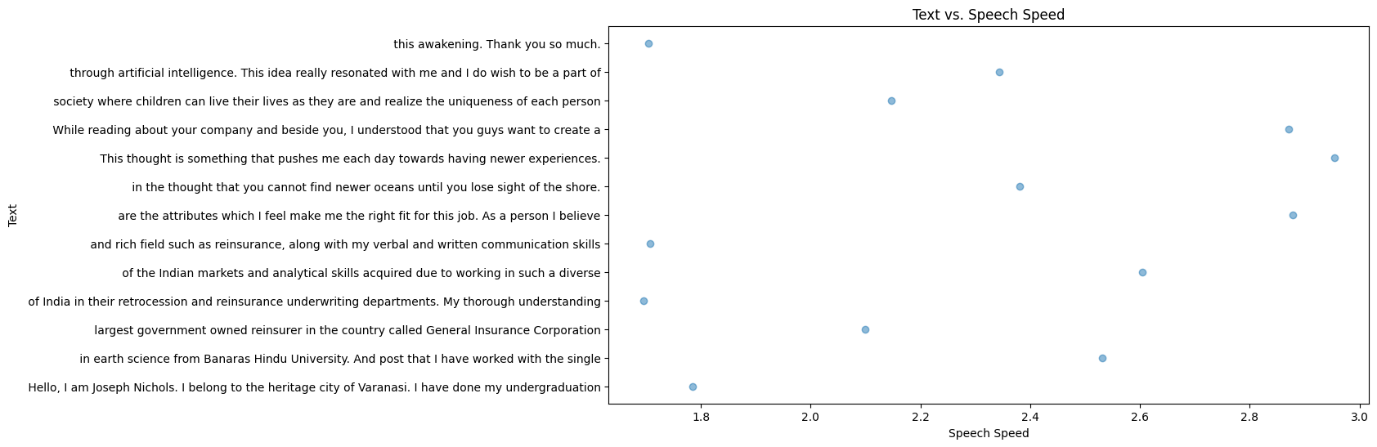
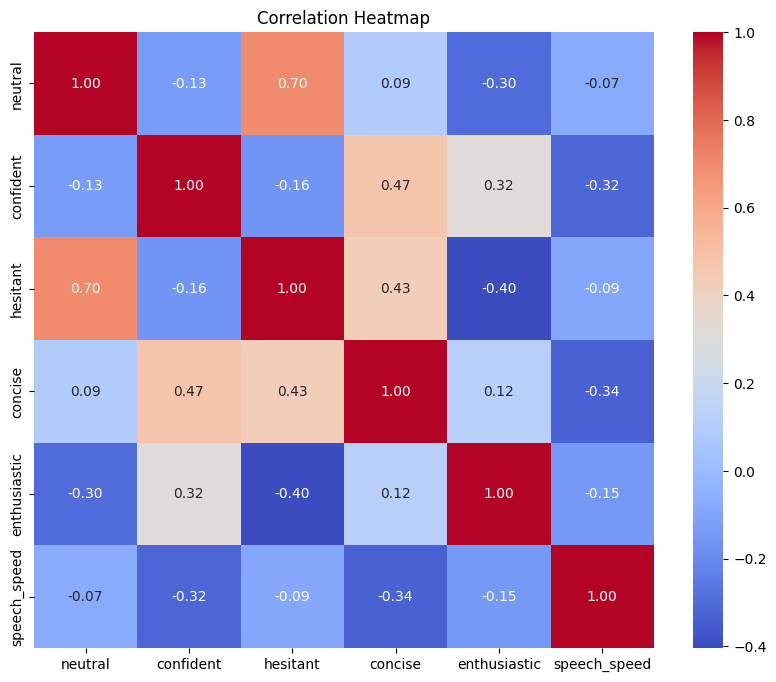
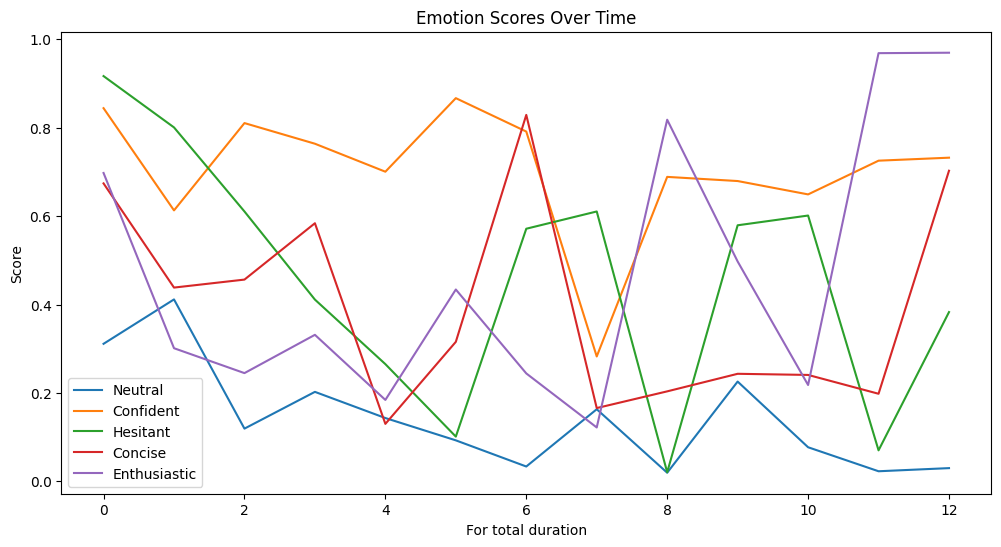
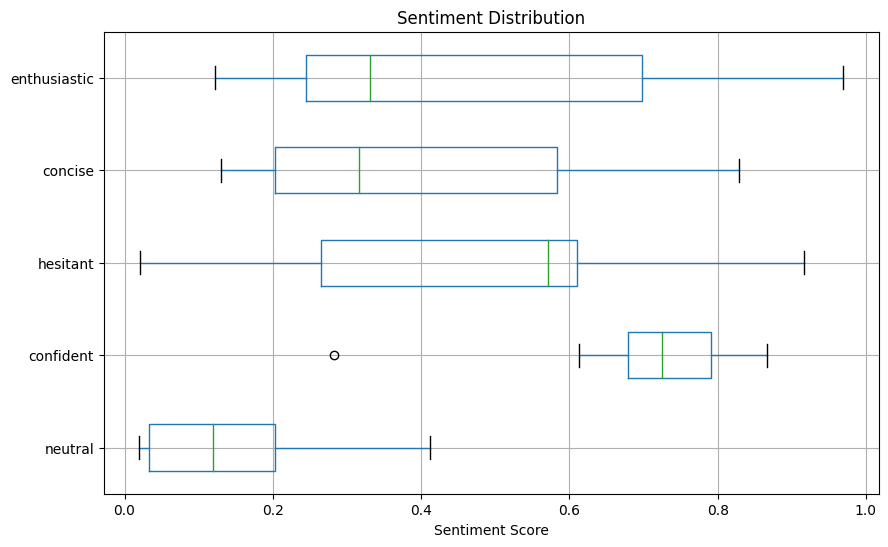
5)



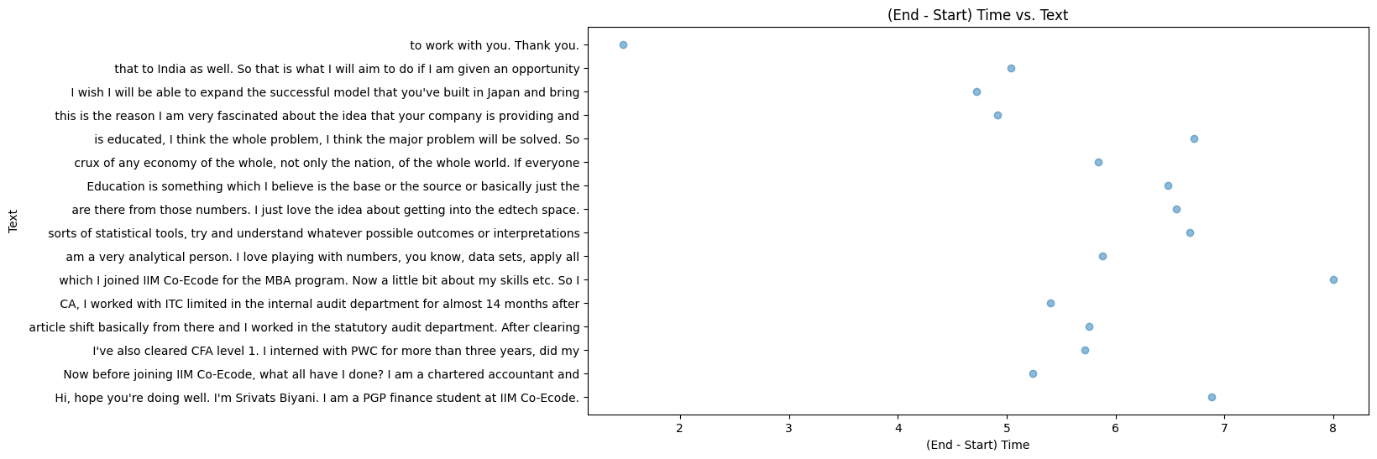
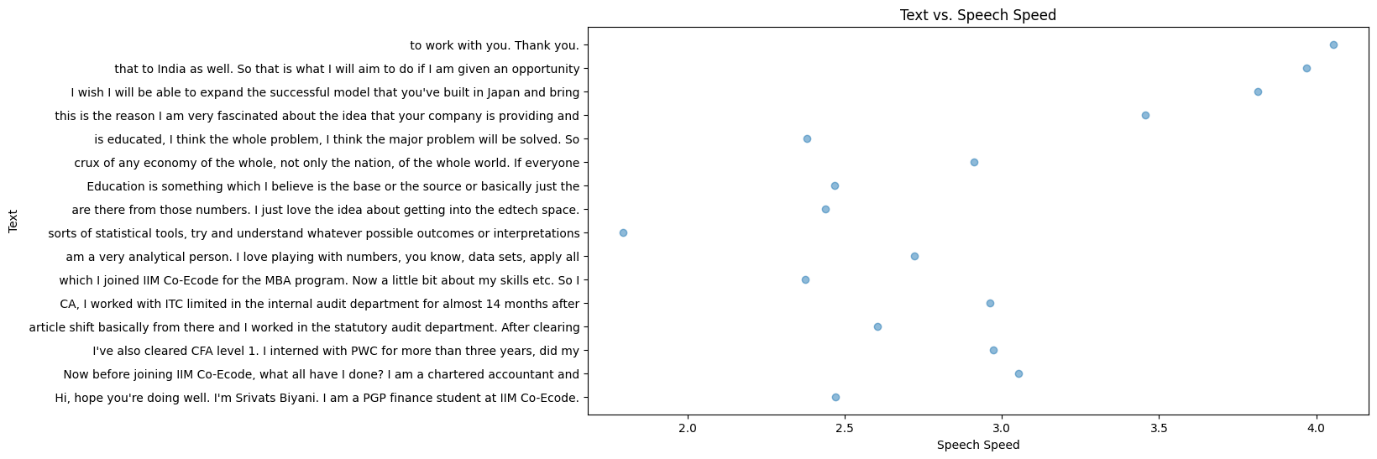
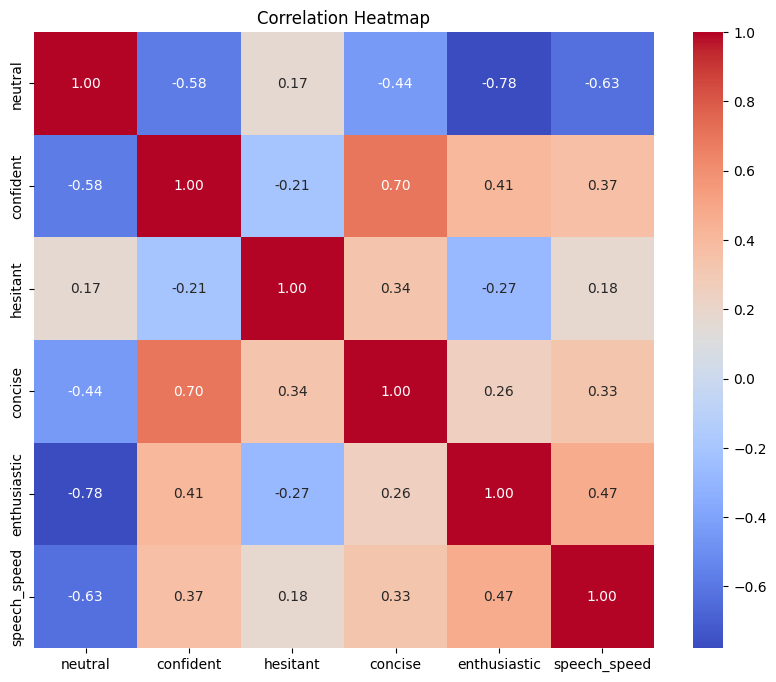
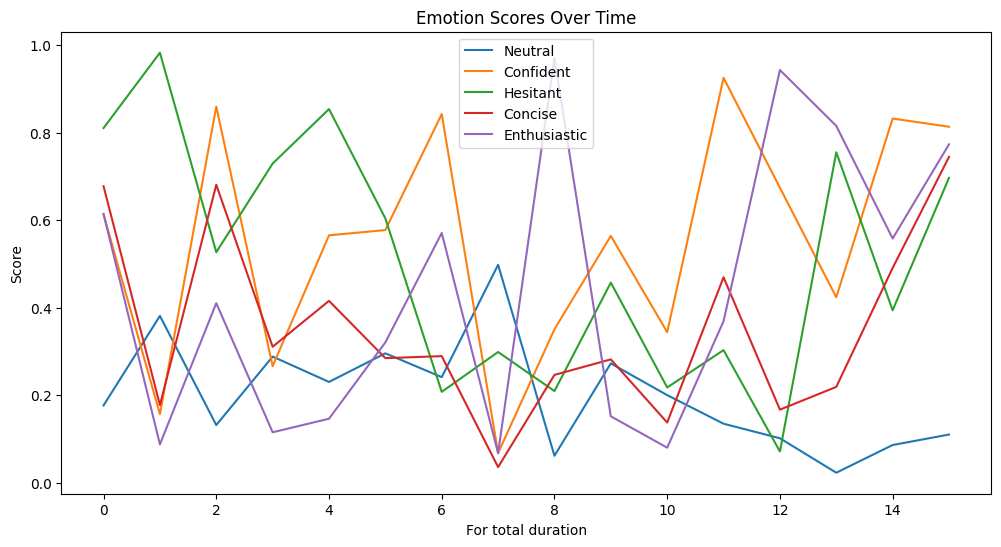
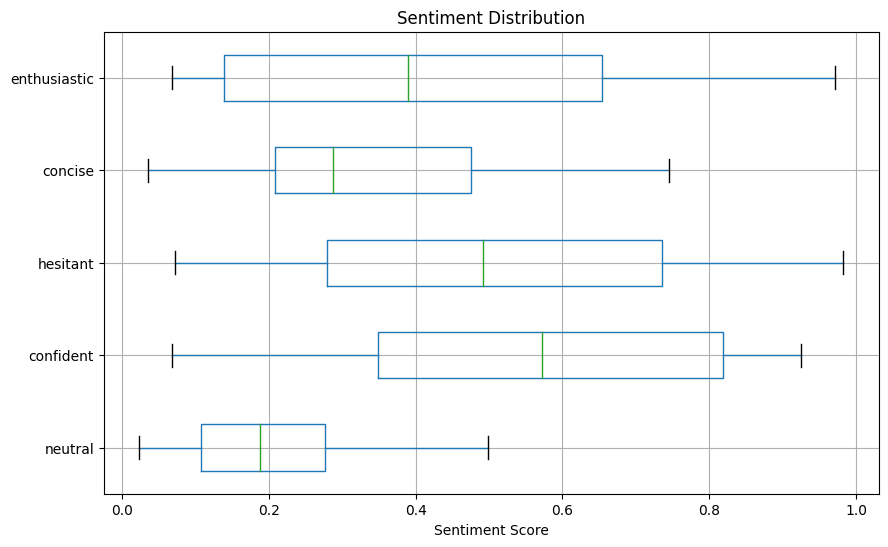
6)



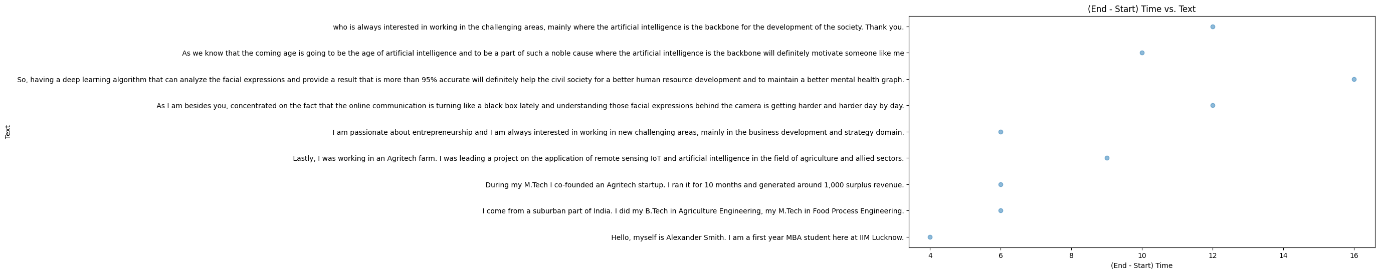
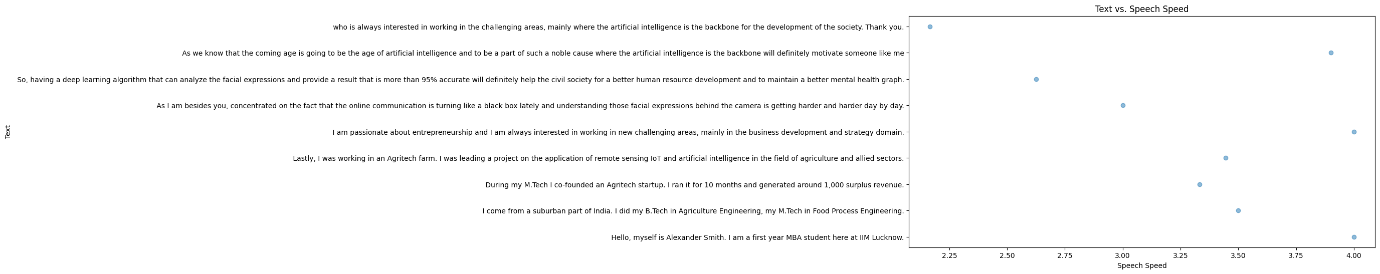
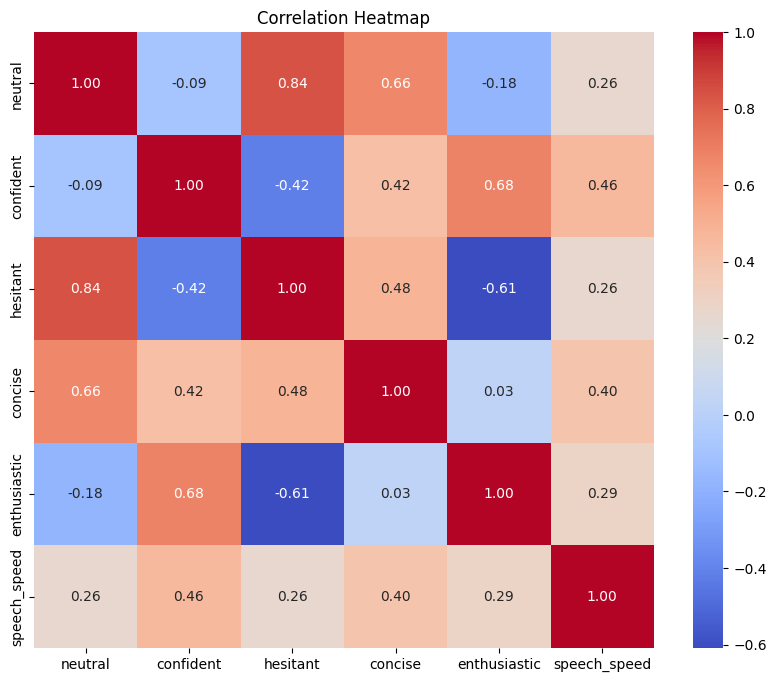
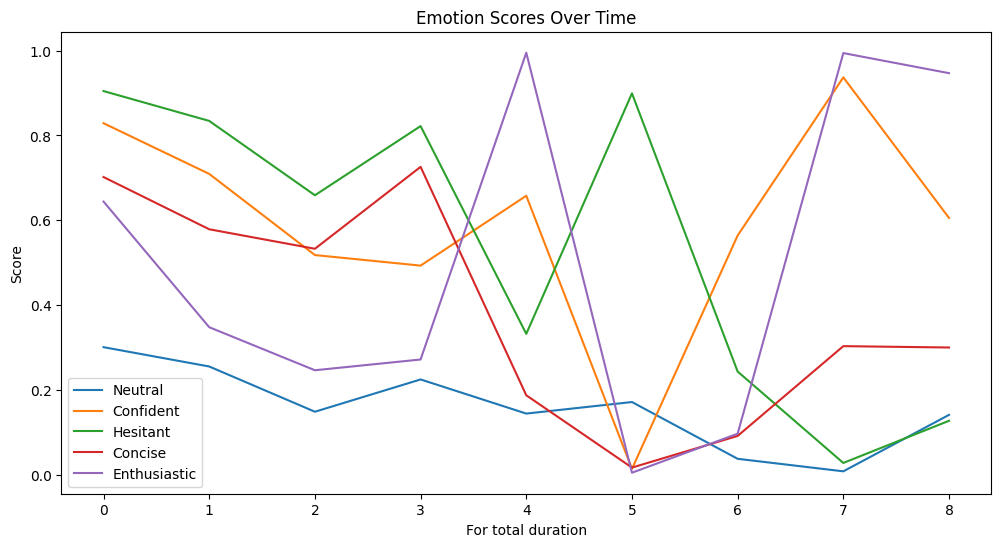
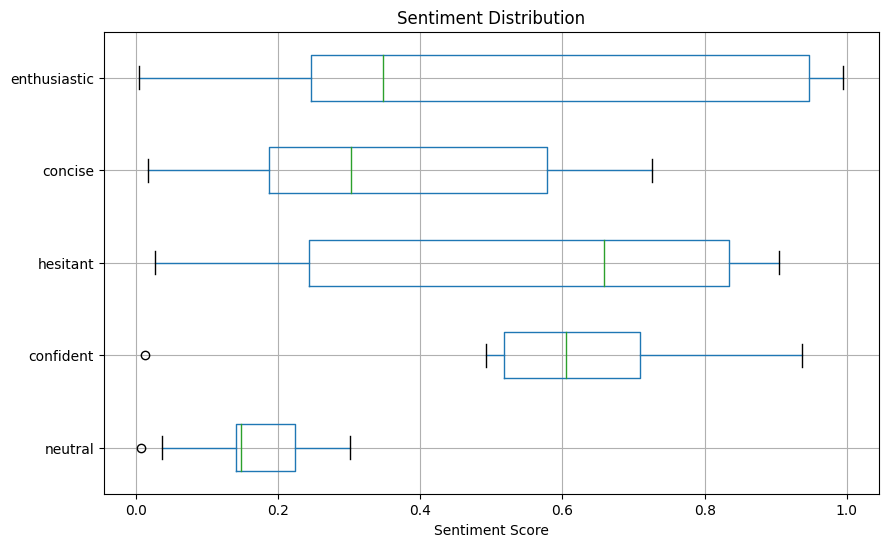
7)



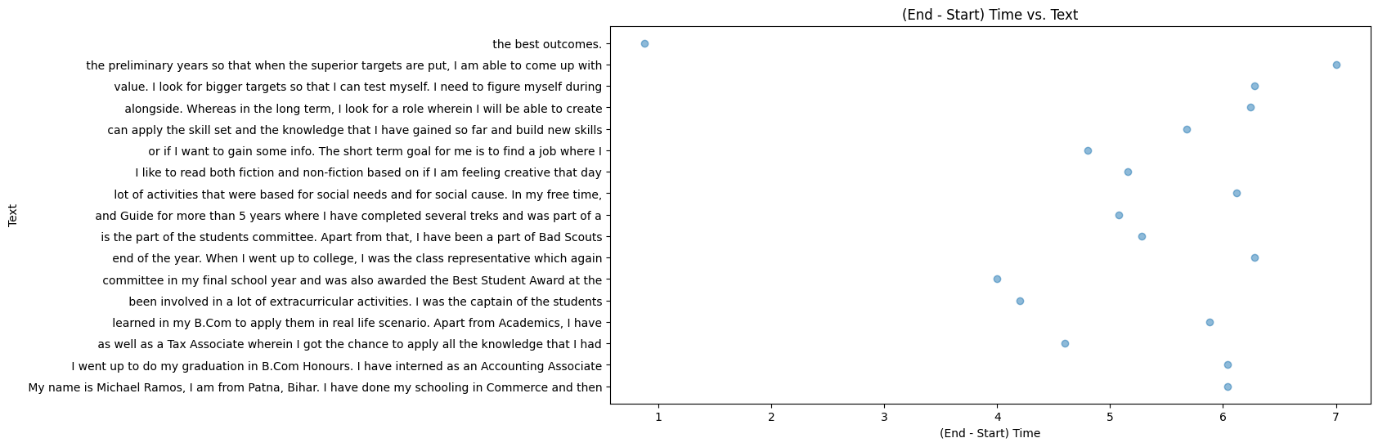
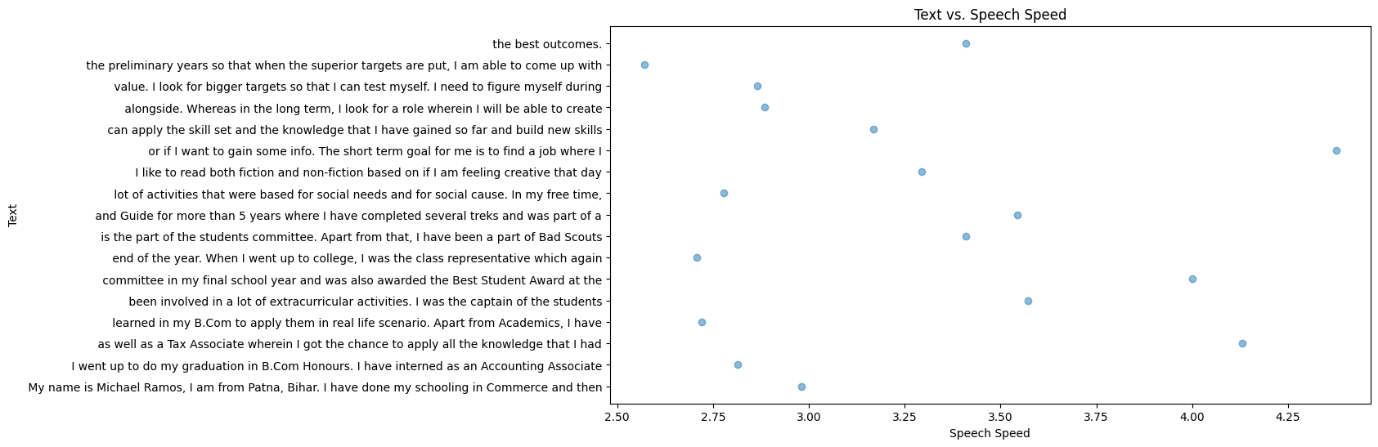
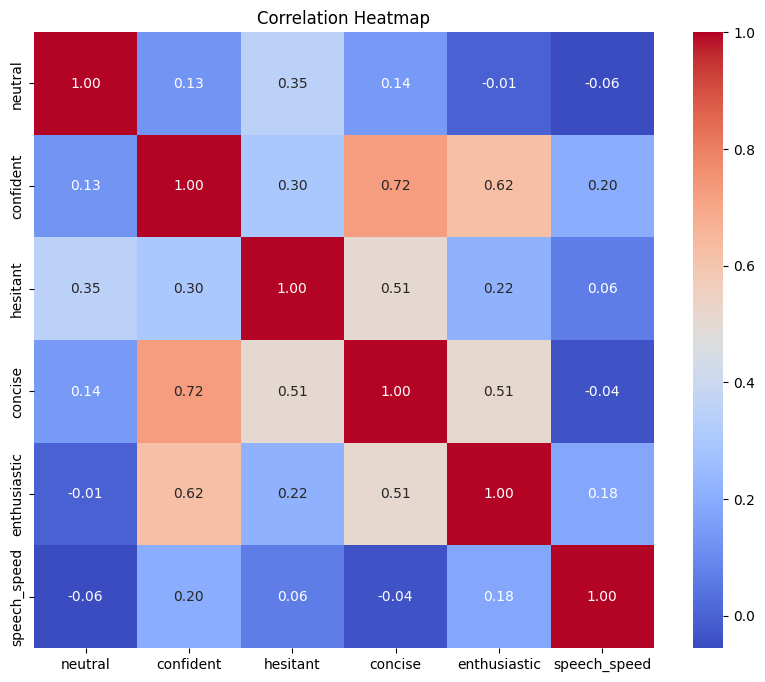
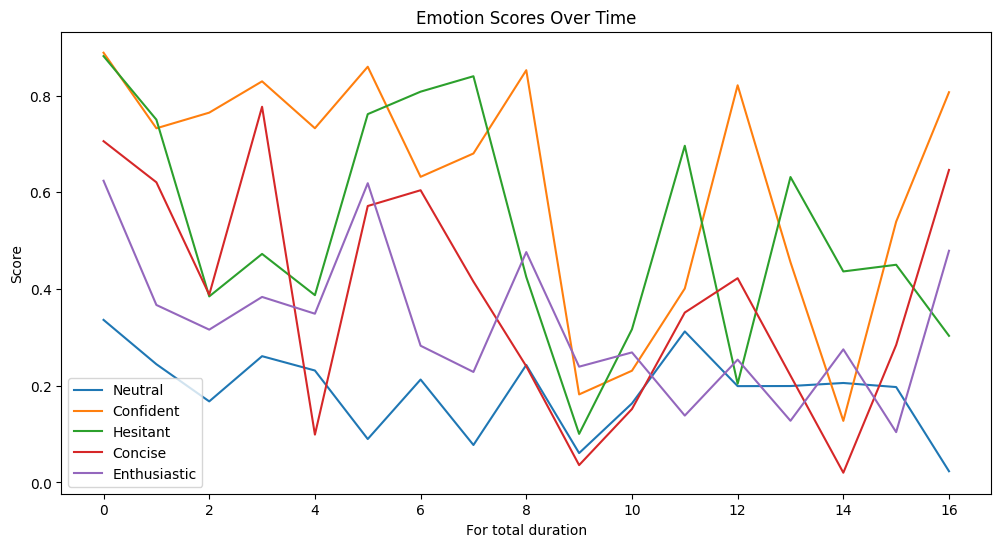
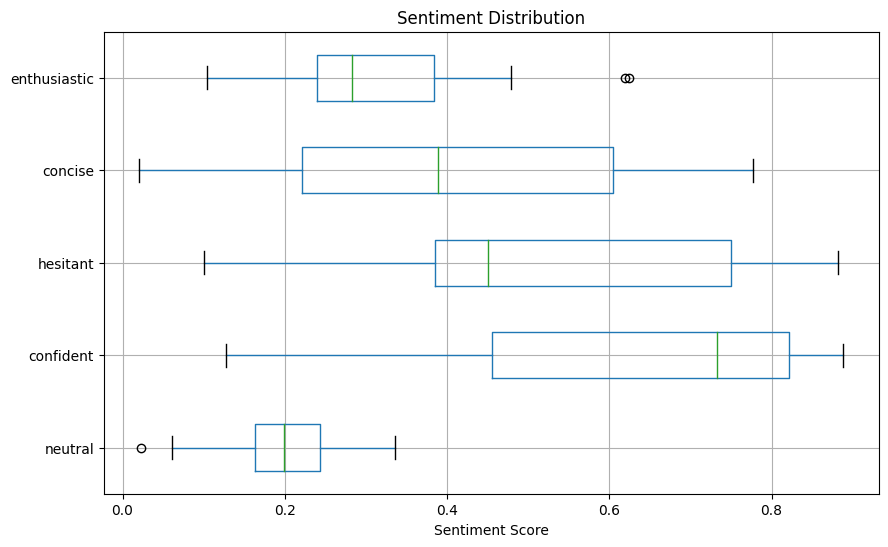
8)



9)



10)



## Calculation of overall score assigning an assumed weight:

import pandas as pd

d1=pd.read\_csv("/content/drive/MyDrive/I\_am\_beside\_u\_project/im beside you/1.csv")

d2=pd.read\_csv("/content/drive/MyDrive/I\_am\_beside\_u\_project/im beside you/2.csv")

d3=pd.read\_csv("/content/drive/MyDrive/I\_am\_beside\_u\_project/im beside you/3.csv")

d4=pd.read\_csv("/content/drive/MyDrive/I\_am\_beside\_u\_project/im beside you/4.csv")

d5=pd.read\_csv("/content/drive/MyDrive/I\_am\_beside\_u\_project/im beside you/5.csv")

d6=pd.read\_csv("/content/drive/MyDrive/I\_am\_beside\_u\_project/im beside you/6.csv")

d7=pd.read\_csv("/content/drive/MyDrive/I\_am\_beside\_u\_project/im beside you/7.csv")

d8=pd.read\_csv("/content/drive/MyDrive/I\_am\_beside\_u\_project/im beside you/8.csv")

d10=pd.read\_csv("/content/drive/MyDrive/I\_am\_beside\_u\_project/im beside you/10.csv")

r1=list(d1.mean())

r2=list(d2.mean())

r3=list(d3.mean())

r4=list(d4.mean())

r5=list(d5.mean())

r6=list(d6.mean())

r7=list(d7.mean())

r8=list(d8.mean())

r10=list(d10.mean())

Next, we will be calculating the weighted scores:

we have assigned weights for each of the emotions and multiplied them with the total emotion score of that particular emotion.

weighted\_score1=3\*r1[8]+1\*r1[9]+3\*r1[10]+1\*r1[11]+3\*r1[12]+3\*r1[13]+3\*r1[14]

weighted\_score2=3\*r2[8]+1\*r2[9]+3\*r2[10]+1\*r2[11]+2\*r2[12]+3\*r2[13]+3\*r2[14]

weigthed\_score3=3\*r3[8]+1\*r3[9]+3\*r3[10]+1\*r3[11]+2\*r3[12]+3\*r3[13]+3\*r3[14]

weighted\_score4=3\*r4[8]+1\*r4[9]+3\*r4[10]+1\*r4[11]+2\*r4[12]+3\*r4[13]+3\*r4[14]

weighted\_score5=3\*r5[8]+1\*r5[9]+3\*r5[10]+1\*r5[11]+2\*r5[12]+3\*r5[13]+3\*r5[14]

weighted\_score6=3\*r6[8]+1\*r6[9]+3\*r6[10]+1\*r6[11]+2\*r6[12]+3\*r6[13]+3\*r6[14]

weighted\_score7=3\*r7[8]+1\*r7[9]+3\*r7[10]+1\*r7[11]+2\*r7[12]+3\*r7[13]+3\*r7[14]

weighted\_score8=3\*r8[8]+1\*r8[9]+3\*r8[10]+1\*r8[11]+2\*r8[12]+3\*r8[13]+3\*r8[14]

weighted\_score10=3\*r10[8]+1\*r10[9]+3\*r10[10]+1\*r10[11]+2\*r10[12]+3\*r10[13]+3\*r3[14]

For the evaluation of communication skills, we can analyze the speech speed, a good speech speed is around 140 – 160 words per second. Thus we have taken the weights accordingly.

If speech speed is in between 2.5 to 3 then we have assigned its weight to be 1 and for the rest to be 0.5. And further in the below code we have added this data to analyse the final weighted score.

good\_speed\_min = 2.5

good\_speed\_max = 3.0

if 2.5<=r1[15]<=3.0:

  weight=1

else:weight=0.5

weighted\_score1=3\*r1[8]+1\*r1[9]+3\*r1[10]+1\*r1[11]+3\*r1[12]+3\*r1[13]+3\*r1[14]+r1[15]\*weight

print(weighted\_score1,"1")

if 2.5<=r2[15]<=3.0:

  weight=1

else:weight=0.5

weighted\_score2=3\*r2[8]+1\*r2[9]+3\*r2[10]+1\*r2[11]+2\*r2[12]+3\*r2[13]+3\*r2[14]+r2[15]\*weight

print(weighted\_score2,"2")

if 2.5<=r3[15]<=3.0:

  weight=1

else:weight=0.5

weigthed\_score3=3\*r3[8]+1\*r3[9]+3\*r3[10]+1\*r3[11]+2\*r3[12]+3\*r3[13]+3\*r3[14]+r3[15]\*weight

print(weigthed\_score3,"3")

if 2.5<=r4[15]<=3.0:

  weight=1

else:weight=0.5

weighted\_score4=3\*r4[8]+1\*r4[9]+3\*r4[10]+1\*r4[11]+2\*r4[12]+3\*r4[13]+3\*r4[14]+r4[15]\*weight

print(weighted\_score4,"4")

if 2.5<=r5[15]<=3.0:

  weight=1

else:weight=0.5

weighted\_score5=3\*r5[8]+1\*r5[9]+3\*r5[10]+1\*r5[11]+2\*r5[12]+3\*r5[13]+3\*r5[14]+r5[15]\*weight

print(weighted\_score5,"5")

if 2.5<=r6[15]<=3.0:

  weight=1

else:weight=0.5

weighted\_score6=3\*r6[8]+1\*r6[9]+3\*r6[10]+1\*r6[11]+2\*r6[12]+3\*r6[13]+3\*r6[14]+r6[15]\*weight

print(weighted\_score6,"6")

if 2.5<=r7[15]<=3.0:

  weight=1

else:weight=0.5

weighted\_score7=3\*r7[8]+1\*r7[9]+3\*r7[10]+1\*r7[11]+2\*r7[12]+3\*r7[13]+3\*r7[14]+r7[15]\*weight

print(weighted\_score7,"7")

if 2.5<=r8[15]<=3.0:

  weight=1

else:weight=0.5

weighted\_score8=3\*r8[8]+1\*r8[9]+3\*r8[10]+1\*r8[11]+2\*r8[12]+3\*r8[13]+3\*r8[14]+r8[15]\*weight

print(weighted\_score8,"8")

if 2.5<=r10[15]<=3.0:

  weight=1

else:weight=0.5

weighted\_score10=3\*r10[8]+1\*r10[9]+3\*r10[10]+1\*r10[11]+2\*r10[12]+3\*r10[13]+3\*r3[14]+r10[15]\*weight

print(weighted\_score10,"10")

Output(depicts weighted score, corresponding candidate)

9.151547941227388 1

8.979378582417203 2

8.474014169792463 3

9.73212821861804 4

9.331647825299882 5

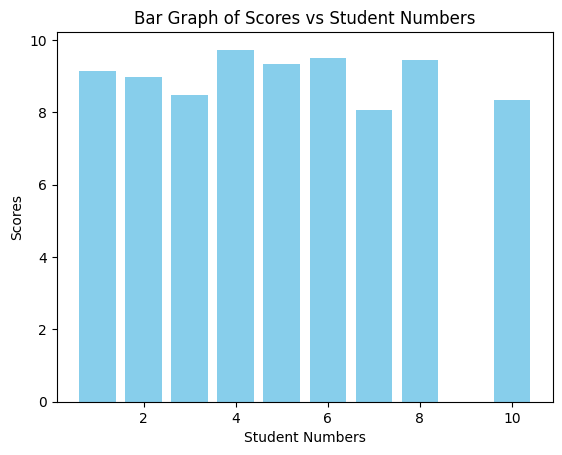
9.513114706303854 6

8.067367322933022 7

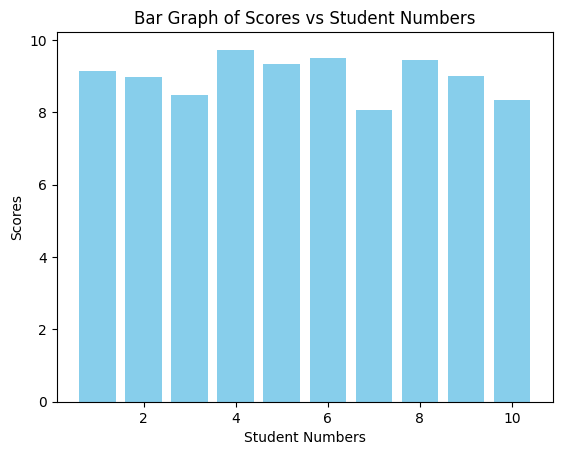
9.456855186928024 8

8.344604034190048 10

Plot of the above result:



Here we can observe the calculated weights and can see candidate 4 standing out among all. Here candidate 9 has no sufficient data. But for the final calculations, we have taken the weighted score of candidate 9 to be the **average score(**9.0056286653011**)** of all the candidate scores. If interviewer is strict then we can simply take other data for consideration.



# EDA ON TRANSCRIPTS TEXT

(Contains area of expertise and grading of communication skills)

Candidate 1:

Jeffrey Shepherd:

Score: 9/10

Remarks: Jeffrey Shepherd demonstrates strong communication skills and provides a detailed overview of his education, work experience, and personal interests. He emphasizes his ability to set and achieve goals, which adds to his profile positively.

Candidate 2:

Cameron Barajas:

Score: 7/10

Remarks: Cameron Barajas presents a concise overview of his qualifications and internship experiences. While he mentions his flexibility and passion, his response could benefit from more specific examples of setting and achieving targets.

Candidate 3:

Michael Guzman:

Score: 6/10

Remarks: Michael Guzman shares his academic and extracurricular background but could improve by providing concrete examples of goal-setting and achievement. His response still lacks a strong connection to the job.

Candidate 4:

Monique McCormick:

Score: 8/10

Remarks: Monique effectively highlights her educational background and work experiences. While she mentions her passion for challenges and resilience, more specific examples of setting and exceeding targets would enhance her response.

Candidate 5:

Sakshi:

Score: 8/10

Remarks: Sakshi provides a good overview of her academic background and certifications. Her mention of winning an international art competition demonstrates a commitment to achieving goals. She could elaborate more on her goals related to using AI for neurodevelopmental disorders.

Candidate 6:

Nathan Lewis:

Score: 8/10

Remarks: Nathan effectively communicates his transition from consulting to pursuing an MBA in analytics. He mentions his desire for exploration and highlights his involvement in college committees. His enthusiasm for setting and achieving targets is evident.

Candidate 7:

Joseph Nichols:

Score: 7/10

Remarks: Joseph shares his educational and work background and expresses a desire for new experiences. While he mentions setting targets, his response could benefit from more specific examples of goal achievement.

Candidate 8:

Srivats Biyani:

Score: 8/10

Remarks: Srivats provides a clear overview of his educational and professional journey. He expresses a strong interest in education and setting bigger targets. He could elaborate further on his approach to achieving these targets.

Candidate 9:

Alexander Smith:

Score: 10/10

Remarks: Alexander effectively communicates his background and passion for entrepreneurship and AI. He links his experiences to the role's focus on AI and facial expression analysis. His commitment to setting and achieving bigger targets aligns with the company's mission.

Candidate 10:

Michael Ramos:

Score: 8/10

Remarks: Michael Ramos presents a well-rounded profile, highlighting his educational background in Commerce and his internship experience in accounting and taxation.

AREAS OF EXPERTISE

1. **Jeffrey Shepherd**:
   * Areas of Expertise: Regulatory affairs in the pharmaceutical industry, medical writing, drug safety, risk management, biotechnology, baking, research, attention to detail.
2. **Cameron Barajas**:
   * Areas of Expertise: Venture network development, startup growth, financial modeling, finance.
3. **Michael Guzman**:
   * Areas of Expertise: Academic excellence, event organizing, volunteering, sales, accounting, guitar playing, fingerstyle guitar, online content creation.
4. **Monique McCormick**:
   * Areas of Expertise: Electronics and communication engineering, academic advising, auditing, data analysis, machine learning, badminton, chess.
5. **Sakshi**:
   * Areas of Expertise: Mass media, advertising, entrepreneurship, art (drawing and painting), singing, writing, mental health awareness.
6. **Nathan Lewis**:
   * Areas of Expertise: Consulting, validation processes, softwares for pharmaceutical clients, media and public relations, social media management, content writing.
7. **Joseph Nichols**:
   * Areas of Expertise: Earth science, reinsurance, analytical skills, market understanding, adaptability to new experiences.
8. **Srivats Biyani**:
   * Areas of Expertise: Chartered accountancy, CFA, internal audit, financial analysis, edtech, finance, data analysis.
9. **Alexander Smith**:
   * Areas of Expertise: Agriculture engineering, entrepreneurship, agritech startups, remote sensing, IoT, artificial intelligence, business development, strategy.
10. **Michael Ramos**:
    * Areas of Expertise: Commerce, accounting, taxation, leadership (students committee, class representative), extracurricular activities (Bad Scouts and Guide), reading, goal setting.

# EDA ON GAZE DATA:

It is calculated for the analysis of a candidate's eye contact with the camera which would be helpful for screening candidates, if other attributes are almost similar.

We have taken the sum of the eye offset values for each of the given gaze.csv files for each candidate.

Here in case the gaze is one then we have not considered the corresponding eye offset value.

A candidate with lesser eye offset has a greater chance of selection.

import pandas as pd

# Define a list of file paths

file\_paths = [

    '/content/drive/MyDrive/I\_am\_beside\_u\_project/1/gaze.csv',

    '/content/drive/MyDrive/I\_am\_beside\_u\_project/2/gaze.csv',

    '/content/drive/MyDrive/I\_am\_beside\_u\_project/3/gaze (1).csv',

    '/content/drive/MyDrive/I\_am\_beside\_u\_project/4/gaze.csv',

    '/content/drive/MyDrive/I\_am\_beside\_u\_project/5/gaze.csv',

    '/content/drive/MyDrive/I\_am\_beside\_u\_project/6/gaze.csv',

    '/content/drive/MyDrive/I\_am\_beside\_u\_project/7/gaze.csv',

    '/content/drive/MyDrive/I\_am\_beside\_u\_project/8/gaze.csv',

    '/content/drive/MyDrive/I\_am\_beside\_u\_project/9/gaze.csv',

    '/content/drive/MyDrive/I\_am\_beside\_u\_project/10/gaze.csv'

]

candidates = 0

# Iterate through the file paths

for file\_path in file\_paths:

    candidates += 1

    # Initialize a variable to store the total eye\_offset for the current file

    total\_eye\_offset = 0

    # Read the CSV file into a pandas DataFrame

    df = pd.read\_csv(file\_path)

    # Iterate through the DataFrame rows

    for index, row in df.iterrows():

        if row['gaze'] == 0:

            # Take the absolute value of eye\_offset

            absolute\_eye\_offset = abs(row['eye\_offset'])

            # Add the absolute value to the total eye\_offset for the current file

            total\_eye\_offset += absolute\_eye\_offset

    # Print the total eye\_offset for the current file

    print(f"Total Eye Offset for candidate {candidates}: {total\_eye\_offset}")

Total Eye Offset for candidate 1: 1142.3617000000002

Total Eye Offset for candidate 2: 1249.0812

Total Eye Offset for candidate 3: 2371.4755999999998

Total Eye Offset for candidate 4: 1030.9413

Total Eye Offset for Candidate 5: 0

Total Eye Offset for Candidate 6: 0

Total Eye Offset for candidate 7: 961.5231

Total Eye Offset for candidate 8: 220.85079999999996

Total Eye Offset for candidate 9: 92.6077

Total Eye Offset for candidate 10: 923.9181000000001

Here we got 0 for candidates 5 and 6 due to insufficient data.  
If an interviewer is going to screen candidates based on this then we may assume candidates 5 and 6 to have median values i.e. 996.2322 (if all the other attributes of candidates are similar then this would be helpful to screen the candidate).

Code for plotting graph of candidate’s vs total eye offset

import matplotlib.pyplot as plt

import numpy as np

# Total eye offset values for candidates

candidates = ["Candidate 1", "Candidate 2", "Candidate 3", "Candidate 4", "Candidate 7", "Candidate 8", "Candidate 9", "Candidate 10"]

eye\_offsets = [1142.3617, 1249.0812, 2371.4756, 1030.9413, 961.5231, 220.8508, 92.6077, 923.9181]

# Create an array for the x-axis positions

x = np.arange(len(candidates))

# Create a figure and specify the figure size

plt.figure(figsize=(10, 6))

# Create a grouped bar chart

plt.bar(x, eye\_offsets, color='blue', alpha=0.7, label='Total Eye Offset')

# Customize the plot

plt.xlabel('Candidates')

plt.ylabel('Total Eye Offset')

plt.title('Total Eye Offset for Candidates (Excluding Candidate 5 and Candidate 6)')

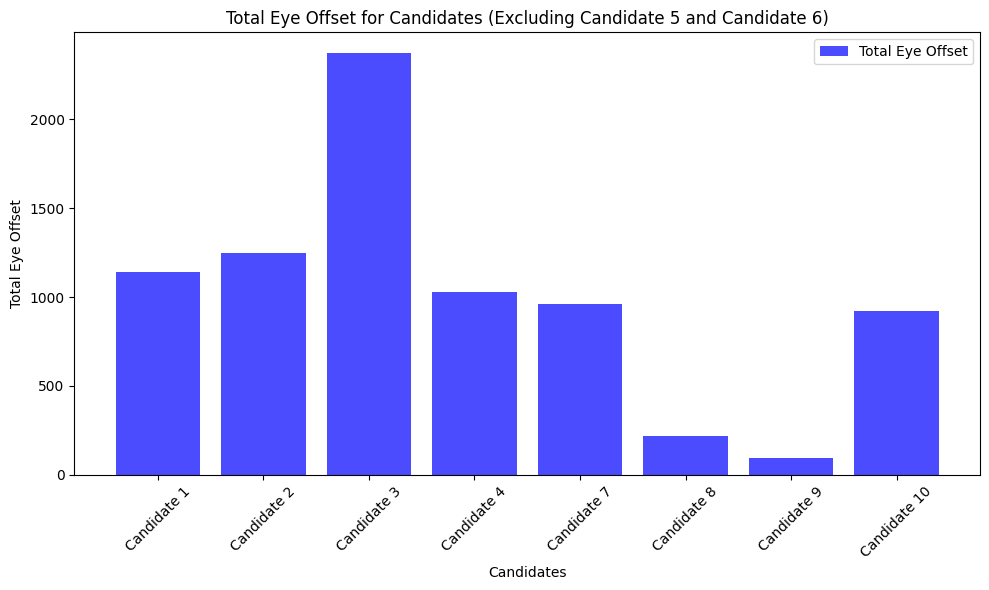
plt.xticks(x, candidates, rotation=45)

plt.legend()

# Display the chart

plt.tight\_layout()

plt.show()



Code for plotting graph of candidates’ vs total eye offset and marking aa different color for least offset

import matplotlib.pyplot as plt

import numpy as np

# Total eye offset values for candidates

candidates = ["Candidate 1", "Candidate 2", "Candidate 3", "Candidate 4", "Candidate 7", "Candidate 8", "Candidate 9", "Candidate 10"]

eye\_offsets = [1142.3617, 1249.0812, 2371.4756, 1030.9413, 961.5231, 220.8508, 92.6077, 923.9181]

# Find the index of the candidate with the least eye offset

min\_offset\_index = np.argmin(eye\_offsets)

# Create an array for the x-axis positions

x = np.arange(len(candidates))

# Create a figure and specify the figure size

plt.figure(figsize=(10, 6))

# Create a bar chart and set a different color for the bar with the least eye offset

bars = plt.bar(x, eye\_offsets, color=['blue' if i != min\_offset\_index else 'green' for i in range(len(candidates))], alpha=0.7, label='Total Eye Offset')

# Customize the plot

plt.xlabel('Candidates')

plt.ylabel('Total Eye Offset')

plt.title('Total Eye Offset for Candidates (Excluding Candidate 5 and Candidate 6)')

plt.xticks(x, candidates, rotation=45)

plt.legend()

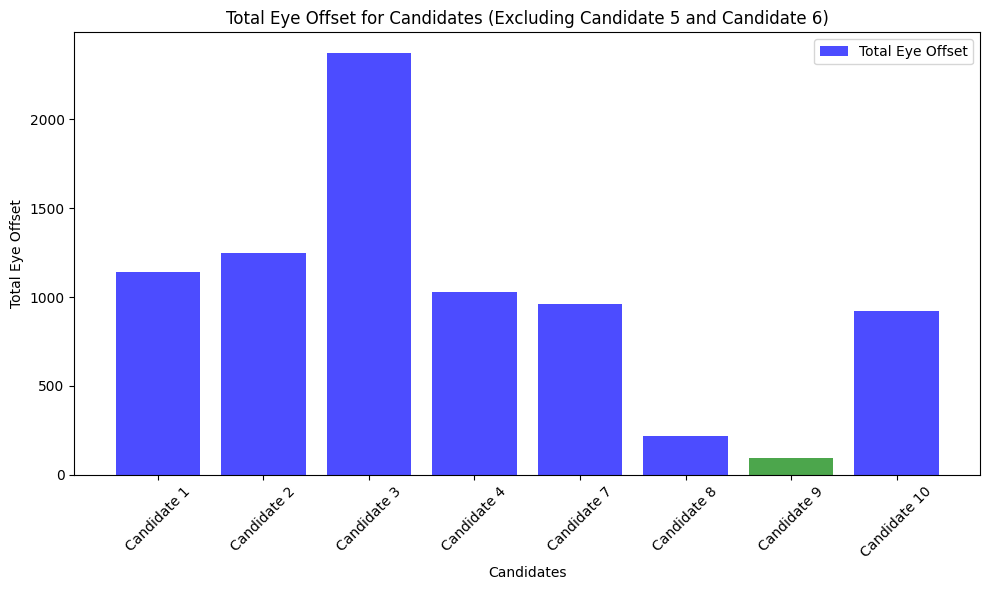
# Add a label to the bar with the least eye offset

bars[min\_offset\_index].set\_label('Lowest Eye Offset')

# Display the chart

plt.tight\_layout()

plt.show()



Marking a dotted line as a threshold value (we have taken it to be the average value i.e 799.27595 and then considered eye offset values below that threshold value to be selected.  
(Not that threshold value can be changed according to the selection process)

import matplotlib.pyplot as plt

import numpy as np

# Total eye offset values for candidates

candidates = ["Candidate 1", "Candidate 2", "Candidate 3", "Candidate 4", "Candidate 7", "Candidate 8", "Candidate 9", "Candidate 10"]

eye\_offsets = [1142.3617, 1249.0812, 2371.4756, 1030.9413, 961.5231, 220.8508, 92.6077, 923.9181]

# Eye offset threshold value

threshold\_value = 799.27595

# Find the index of the candidate with the least eye offset

min\_offset\_index = np.argmin(eye\_offsets)

# Create an array for the x-axis positions

x = np.arange(len(candidates))

# Create a figure and specify the figure size

plt.figure(figsize=(10, 6))

# Create a bar chart and set colors based on the threshold

colors = ['red' if offset < threshold\_value else 'blue' for offset in eye\_offsets]

bars = plt.bar(x, eye\_offsets, color=colors, alpha=0.7, label='Total Eye Offset')

# Add a dotted line at the threshold value

plt.axhline(y=threshold\_value, linestyle='--', color='gray', label='Threshold')

# Customize the plot

plt.xlabel('Candidates')

plt.ylabel('Total Eye Offset')

plt.title('Total Eye Offset for Candidates (Excluding Candidate 5 and Candidate 6)')

plt.xticks(x, candidates, rotation=45)

plt.legend()

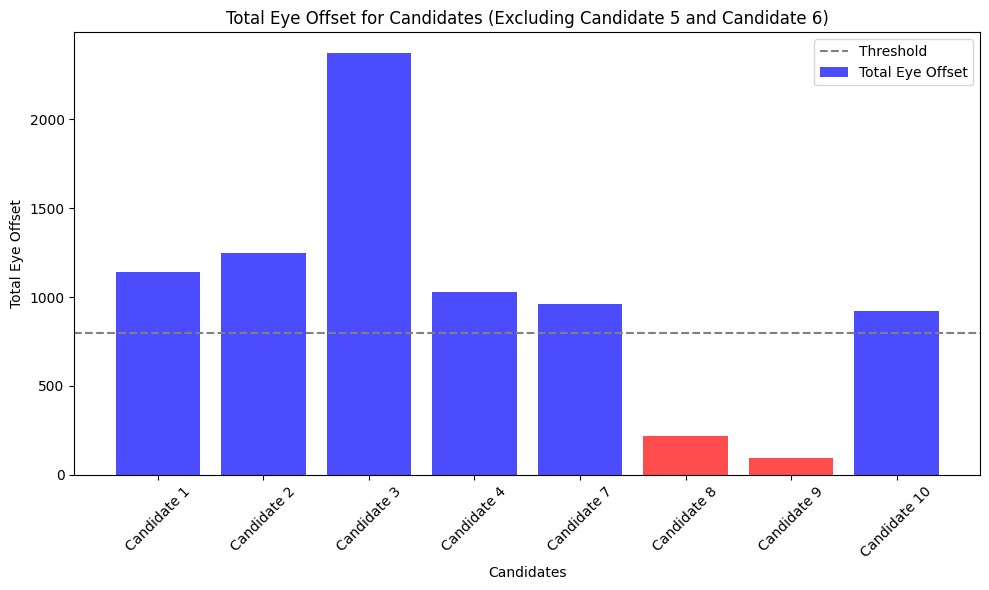
# Add a label to the bar with the least eye offset

bars[min\_offset\_index].set\_label('Lowest Eye Offset')

# Display the chart

plt.tight\_layout()

plt.show()



# EDA/CALCULATIONS ON OVERALL CANDIDATES’ PERFORMANCE:

Transcript text scores:

* candidate1=9/10
* candidate2=7/10
* candidate3=6/10
* candidate4= 8/10
* candidate5=8/10
* candidate6= 8/10
* candidate7=7/10
* candidate8=8/10
* candidate9=10/10
* candidate10=8/10

emotion scores:

* c1=210.77823699836864
* c2=244.10549888402204
* c3=236.10291955049618
* c4=288.8648272225419
* c5=286.4087061038691
* c6=299.4796434348859
* c7=177.9238883899423
* c8=256.011384222267
* c9=229.78589032343052
* c10=182.91470031850952

scaled emotion score: (scaled to 10)

* c1 = 3.172
* c2 = 4.689
* c3 = 4.054
* c4 = 7.813
* c5 = 7.537
* c6 = 9.325
* c7 = 0.100
* c8 = 5.647
* c9 = 3.065
* c10 =1.255

Here c1 is the score of candidate 1, c2 is the score of candidate 2, and so on.

trancript scores

* t1=9.151547941227388
* t2=8.979378582417203
* t3=8.474014169792463
* t4=9.73212821861804
* t5=9.331647825299882
* t6=9.513114706303854
* t7=8.067367322933022
* t8=9.456855186928024
* t10=8.344604034190048
* t9=(t1+t2+t3+t4+t5+t6+t7+t8+t10)/9

here we have taken t9 to be the average of all the scores

Here t1 is the score of Candidate 1, t2 is the score of Candidate 2, and so on.

Let us consider for a technical role so the weights are distributed as 0.4, 0.3, 0.3 for transcript text scores, emotion scores and transcript scores respectively as

(Transcript Text Scores (Weight: 0.4)) Reason: The transcript text scores likely represent the candidate's communication skills, technical knowledge, and ability to articulate their thoughts and ideas. In a technical role, effective communication is often crucial, as engineers and technical professionals need to explain complex concepts, collaborate with team members, and potentially interact with non-technical stakeholders.

Emotion scores/transcript scores (weight: 0.3) Reason: They provide insights into a candidate's emotional intelligence, interpersonal skills, and ability to work well in a team. These qualities are valuable in a technical role because engineers often collaborate on projects and need to understand and respond to the emotions and needs of team members.

Code:

cd1=0.4\*candidate1+0.3\*c1+0.3\*t1

print(cd1)

cd2=0.4\*candidate2+0.3\*c2+0.3\*t2

print(cd2)

cd3=0.4\*candidate3+0.3\*c3+0.3\*t3

print(cd3)

cd4=0.4\*candidate4+0.3\*c4+0.3\*t4

print(cd4)

cd5=0.4\*candidate5+0.3\*c5+0.3\*t5

print(cd5)

cd6=0.4\*candidate6+0.3\*c6+0.3\*t6

print(cd6)

cd7=0.4\*candidate7+0.3\*c7+0.3\*t7

print(cd7)

cd8=0.4\*candidate8+0.3\*c8+0.3\*t8

print(cd8)

cd9=0.4\*candidate9+0.3\*c9+0.3\*t9

print(cd9)

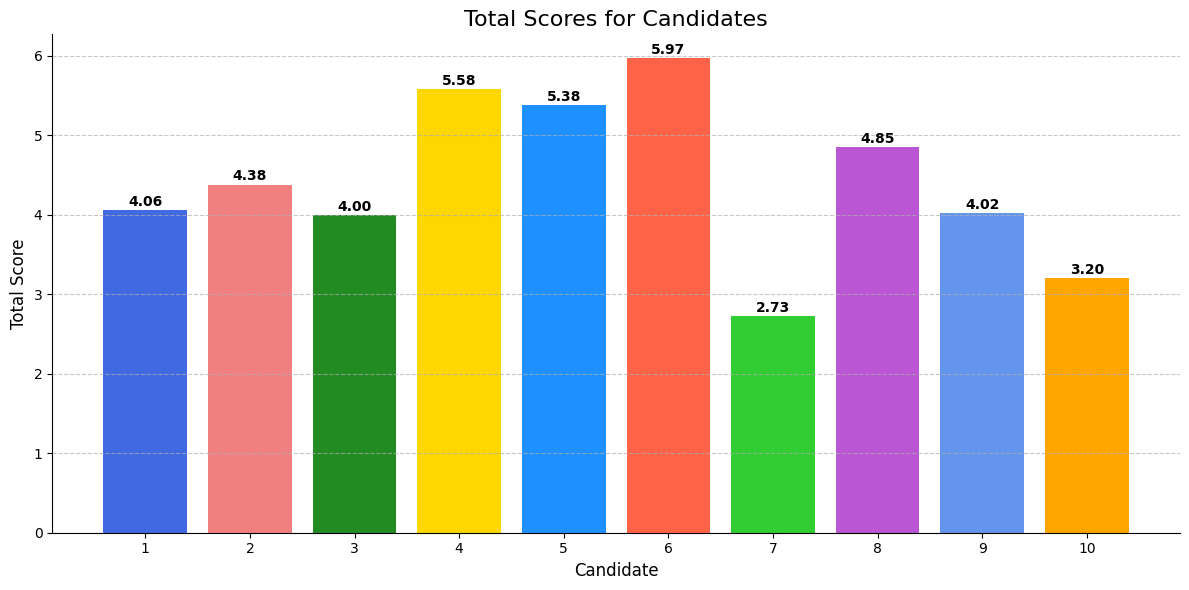
cd10=0.4\*candidate10+0.3\*c10+0.3\*t10

print(cd10)

Total scores of the candidates:

1. 4.057064382368217
2. 4.3805135747251605
3. 3.998404250937739
4. 5.583538465585412
5. 5.380594347589964
6. 5.971434411891156
7. 2.7302101968799066
8. 4.851156556078408
9. 4.02118859959033
10. 3.1998812102570144

(Here 1. Is for candidate 1, 2. For candidate 2 and so on)





Therefore, we can say that Candidate 6 is selected 1st and then Candidate 4, and so on, as seen in the plot of preference order based on total score vs. candidates.