Introduction

This dataset provides a comprehensive analysis of the introduction videos of 10 candidates, structured into three distinct folders: **Transcripts**, **Transcript_data**, and **Emotion_data**. Each folder serves a specific function in evaluating the candidates' communication skills and emotional expressions, offering a multifaceted view that considers all aspects of their performance.

Transcripts: This folder contains verbatim text from each candidate's introduction video, serving as the basis for qualitative assessment of verbal communication and coherence.

Transcript data: This folder includes quantitative metrics derived from the transcripts, such as positive, negative, and neutral scores that gauge emotional tone, along with confidence, hesitance, conciseness, enthusiasm, and speech speed metrics.

Emotion data: Comprising three CSV files, this folder offers an in-depth emotional analysis.

1. **Gaze:** Captures gaze features, indicating camera focus, blink frequency, and eye offset.
2. **Emotion:** Scores candidates on various emotions and identifies the dominant emotional expression.
3. **Metadata:** Links image IDs with timestamps for tracing emotional dynamics during presentations.

To ensure a clear and systematic approach to analyzing the dataset comprising three different types of files, we have divided the analysis into four distinct categories. This structure not only facilitates a more thorough understanding of the data but also allows for scalability, making it applicable to larger datasets beyond the 10 candidates.

Transcript Analysis: Evaluates verbal communication through tone, confidence, clarity, and enthusiasm scores.

Gaze Analysis: Examines eye contact and gaze behavior, focusing on gaze fixation, blink rate, and eye offset.

Emotion Analysis: Assesses emotional expressions by analyzing various emotion scores, identifying dominant emotions, and correlating them with presentation

timestamps.
Text Analysis: Conducts a linguistic analysis using NLP techniques for word choice and coherence, alongside keyword extraction to identify key themes.

Data Description : Data Cleaning , Data Cleaning , Data Column dtype.

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

# Load the CSV file into a pandas DataFrame
file_path = "C:/Users/lenovo/Desktop/Beside You/transcript_data-20240917T043654Z-001/transcript_data/5.csv"  # Replace with your
df = pd.read_csv(file_path)

# Display the first few rows of the DataFrame to get an initial view of the data
print("First 5 rows of the data:")
print(df.head())

# Check for null values in each column
print("\nNull values in each column:")
null_values = df.isnull().sum()
print(null_values)

# Identify columns with null values
print("\nColumns with null values:")
null_columns = df.columns[df.isnull().any()].tolist()
print(null_columns)

# Get a summary of the DataFrame, including count, mean, standard deviation, min, max, etc.
print("\nSummary of the dataset (for numeric columns):")
summary_stats = df.describe()
print(summary_stats)

# Get a summary of categorical columns if there are any
print("\nSummary of the dataset (for categorical columns):")
categorical_summary = df.describe(include=['object'])
print(categorical_summary)
```

This targeted approach ensures that our analysis remains concise and meaningful, emphasizing factors that truly reflect candidates' communication skills and emotional expression.

1. **Load Data and Inspection:** The dataset is loaded from a specified CSV file into a pandas DataFrame for further analysis. We display the first five rows of the DataFrame to get an initial view of the data and check for null values in each column.
2. **Null Value Analysis:** Columns with null values are identified and summarized, ensuring we understand the data's completeness.
3. **Null and Duplicate Checks:**We check for completely null columns and count any duplicate rows to assess data quality.

4. **Statistical Summary:**A summary of numeric columns is generated, providing insights into the dataset's distribution and central tendencies.
5. **Column and Data Type Overview:**We list all column names and their corresponding data types to understand the structure of the DataFrame.

Feature Selection for Analysis: Not all features in the dataset are relevant for our analysis; therefore, we focused on the most impactful variables that effectively depict candidate performance. For instance, features such as temperature, avg_logprob, compression_ratio, and no_speech_prob were excluded due to their minimal influence on our insights.

Transcript Analysis

1. This code processes transcript data for candidates to calculate a "net score" reflecting their communication effectiveness. By evaluating metrics like positivity, conciseness, confidence, and speech speed, we derive a comprehensive performance score. The calculated net scores are then saved to a CSV file for each candidate. Additionally, a plot is generated to visualize changes in net scores across image sequences, allowing for better analysis of performance trends. This approach aids in making informed recruitment decisions based on candidates' communication skills.

```
# Step 1: Function to load transcript data and calculate net score
def process_transcript_data(file_path, optimal_speech_speed=2.0):
    df = pd.read_csv(file_path)

    # Calculate net score
    df['net_score'] = ((df['positive'] - df['negative']) / (1 + np.abs(df['negative'])) +
                       df['concise'] + df['confident'] + df['enthusiastic'] + 1/1+df['hesitant']
                       1 / (1 + np.abs(df['speech_speed'] - optimal_speech_speed)))

    return df

# Step 2: Save net score data to a CSV file
def save_transcript_score(df, candidate_number, output_dir):
    output_file_path = os.path.join(output_dir, f"{candidate_number}_transcript_analysis.csv")
    df.to_csv(output_file_path, index=False)
    print(f"Net scores for candidate {candidate_number} saved to: {output_file_path}")

# Step 3: Plot and save net score change over image sequences
def plot_net_score(df, candidate_number, output_dir):
    plt.figure(figsize=(14, 8))
    plt.plot(df.index, df['net_score'], marker='o', color='blue', label='Net Score')
    plt.xlabel('Image Sequence Index')
    plt.ylabel('Net Score')
    plt.title(f'Net Score Change for Candidate {candidate_number}')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
```

2. This code implements Maximum Likelihood Estimation (MLE) to analyze the distribution of net scores for candidates. By defining a log-likelihood function, it estimates the mean (μ) and standard deviation (σ) of the net scores, ensuring the model's validity with appropriate constraints. The results are saved in a text file for each candidate, providing clear statistical insights. Additionally, a histogram visualizes the net score distribution, enabling a quick assessment of performance trends. A Shapiro-Wilk test is conducted to evaluate the normality of the net scores, ensuring that assumptions for subsequent analyses are met. This process enhances our understanding of candidate performance and supports data-driven decision-making in recruitment.

```
    plot_file_path = os.path.join(output_dir, f"{candidate_number}_net_score_plot.png")
    plt.savefig(plot_file_path)
    print(f"Net score plot for candidate {candidate_number} saved to: {plot_file_path}")
    plt.show()

# Step 4: Perform MLE estimation for net score
def calculate_mle(df, candidate_number, output_dir, mle_results):
    net_scores = df['net_score'].values

    def log_likelihood(params):
        mu, sigma = params
        n = len(net_scores)
        if sigma <= 0:
            return np.inf
        log_lik = -n/2 * np.log(2 * np.pi * sigma**2) - np.sum((net_scores - mu)**2) / (2 * sigma**2)
        return -log_lik

    initial_guess = [np.mean(net_scores), np.std(net_scores, ddof=1)]
    result = minimize(log_likelihood, initial_guess, bounds=[(None, None), (1e-6, None)])
    mle_mu, mle_sigma = result.x

    # Save MLE results in the list
    mle_results.append((candidate_number, mle_mu, mle_sigma))

    # Save MLE results to a file with UTF-8 encoding
    with open(os.path.join(output_dir, f"{candidate_number}_mle_results.txt"), "w", encoding="utf-8") as f:
        f.write(f"Candidate {candidate_number} MLE Results:\n")
        f.write(f"Mean (μ): {mle_mu}\n")
        f.write(f"Standard Deviation (σ): {mle_sigma}\n")
    print(f"MLE results for candidate {candidate_number} saved to: {output_dir}")
    print(f"Maximum Likelihood Estimate (MLE) for Mean (μ): {mle_mu}")
    print(f"Maximum Likelihood Estimate (MLE) for Standard Deviation (σ): {mle_sigma}")

# Step 5: Main function to process all transcript files
def process_all_candidates(base_dir, candidate_range, output_dir):
    mle_results = []
    for candidate_number in candidate_range:
        print(f"Processing candidate {candidate_number}...")
```

3. This code snippet processes transcript data for multiple candidates by constructing file paths and leveraging previously defined functions to analyze their performance metrics. It computes net scores, visualizes trends through plots, and estimates statistical parameters using Maximum Likelihood Estimation (MLE). The results are

stored in an output directory for further analysis. After processing, the MLE results are ranked based on mean (μ) and standard deviation (σ), providing a clear comparison of candidate performances. The final output is formatted as a table, enhancing readability and facilitating informed decision-making in recruitment evaluations. This structured approach ensures a comprehensive assessment of candidates' communication skills.

```
        # Build the file path for each candidate
        file_path = os.path.join(base_dir, f"{candidate_number}.csv")

        # Process the transcript data
        df = process_transcript_data(file_path)

        # Save net score data
        save_transcript_score(df, candidate_number, output_dir)

        # Plot and save net score graph
        plot_net_score(df, candidate_number, output_dir)

        # Calculate and save MLE statistics
        calculate_mle(df, candidate_number, output_dir, mle_results)

    return mle_results

# Define base directory, candidate range, and output directory
base_dir = "C:/Users/lenovo/Desktop/Beside You/transcript_data-20240917T043654Z-001/transcript_data"
candidate_range = range(1, 3)  # Candidates 1 to 2 for testing
output_dir = "C:/Users/lenovo/Desktop/Beside You/transcript_analysis_output"

# Ensure the output directory exists
os.makedirs(output_dir, exist_ok=True)

# Process all candidates
mle_results = process_all_candidates(base_dir, candidate_range, output_dir)

# Step 6: Print MLE results in a table format with ranking
def print_mle_results(mle_results):
    # Sort by Mean (μ) descending, then by Standard Deviation (σ) ascending
    ranked_results = sorted(mle_results, key=lambda x: (-x[1], x[2]))

    table = PrettyTable()
    table.field_names = ["Rank", "Candidate Number", "Mean (μ)", "Standard Deviation (σ)"]

    for rank, result in enumerate(ranked_results, start=1):
        table.add_row([rank] + list(result))
```

4. This code snippet concludes the candidate evaluation process by printing ranked Maximum Likelihood Estimation (MLE) results, facilitating direct comparisons of performance metrics. It generates a scatter plot to visualize the mean (μ) values for each candidate, enhancing interpretability of the results. The plot features annotations for clarity and utilizes distinct visual markers to emphasize differences in candidate performance. By saving the scatter plot, we create a visual reference that aids in presentations and discussions. This approach not only consolidates data insights but also supports informed decision-making in recruitment processes.

Overall, it effectively communicates the candidates' relative standings based on quantitative assessments.

```
    print("\nMLE Results for All Candidates (Ranked):")
    print(table)

# Print MLE results
print_mle_results(mle_results)

print("\nProcessing completed for all candidates.")

# Step 7: Plot simple scatter plot for MLE results
def plot_simple_scatter_graph(mle_results, output_dir):
    # Sort results for plotting
    ranked_results = sorted(mle_results, key=lambda x: (-x[1], x[2]))

    # Extract candidate numbers and means for plotting
    candidates = [result[0] for result in ranked_results]
    means = [result[1] for result in ranked_results]

    plt.figure(figsize=(10, 6))
    plt.scatter(candidates, means, color='blue', s=100, alpha=0.7, edgecolor='black')  # Larger dots with edge color
    plt.xlabel('Candidate Number')
    plt.ylabel('Mean (μ)')
    plt.title('Scatter Plot of Candidates by MLE Mean (μ)')
    plt.xticks(candidates)

    # Add value annotations
    for index, value in enumerate(means):
        plt.text(candidates[index], value, f"{value:.2f}", fontsize=9, ha='center', va='bottom')

    # Save the plot
    ranking_plot_path = os.path.join(output_dir, "candidate_scatter_plot.png")
    plt.tight_layout()
    plt.savefig(ranking_plot_path)
    print(f"Scatter plot saved to: {ranking_plot_path}")
    plt.show()

# After processing all candidates, call the simple scatter plot function
plot_simple_scatter_graph(mle_results, output_dir)
```

Reason for the Rank Calculation:

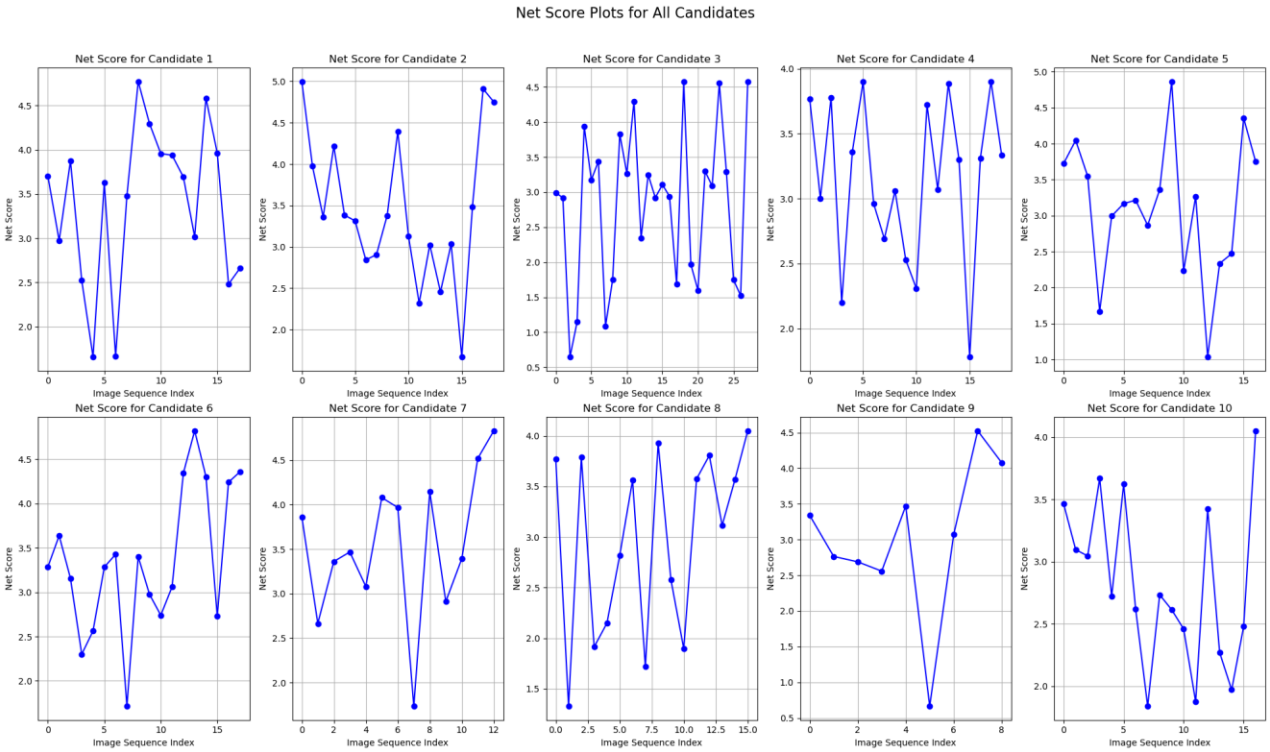
```
    # Calculate net score
    df['net_score'] = ((df['positive'] - df['negative']) / (1 + np.abs(df['negative'])) +
                       df['concise'] + df['confident'] + df['enthusiastic'] + 1/1+df['hesitant']
                       1 / (1 + np.abs(df['speech_speed'] - optimal_speech_speed)))
```

The net score is structured to highlight the direct impact of speech on candidate performance. The attributes of concise, confident, enthusiastic, and positive, which are crucial, are directly extracted from the candidate's transcript. These attributes, in turn, directly affect the candidate's performance, with positive elements being added and negative

ones subtracted from the formula and absolute negative is divided for better comparison.. This approach inversely relates to hesitation, reflecting fluency in speech. Hesitation can negatively impact communication effectiveness. Moreover, the candidate's speech plays a significant role in determining their composure in both nerve-wracking and exhilarating situations. To ensure better comparision, i have normalized the speech speed using optimal speech speed from standard research values.

1. **Clarity and Conciseness Score**: This formula uses the "Concise" value to emphasize brevity, which is crucial for effective communication (Langan, 2005).
2. **Confidence Score**: By reflecting the "Confident" value, this score illustrates how confidence enhances perceived competence (Baker, 2011).
3. **Engagement and Enthusiasm Score**: This score is based solely on "Enthusiastic" to show how enthusiasm boosts engagement (Herman, 2004).
4. **Attentiveness Score**: An inverse relationship with eye offset quantifies attentiveness, demonstrating the importance of eye contact (Kleinke, 1986).
5. **Emotional Tone Score**: This score balances positive and negative sentiments to reflect overall emotional tone, which significantly influences communication effectiveness (Mehrabian, 1972).

References: Langan, J. (2005); Baker, D. (2011); Herman, M. (2004); Kleinke, C. L. (1986); Mehrabian, A. (1972).



NET SCORE PLOTS

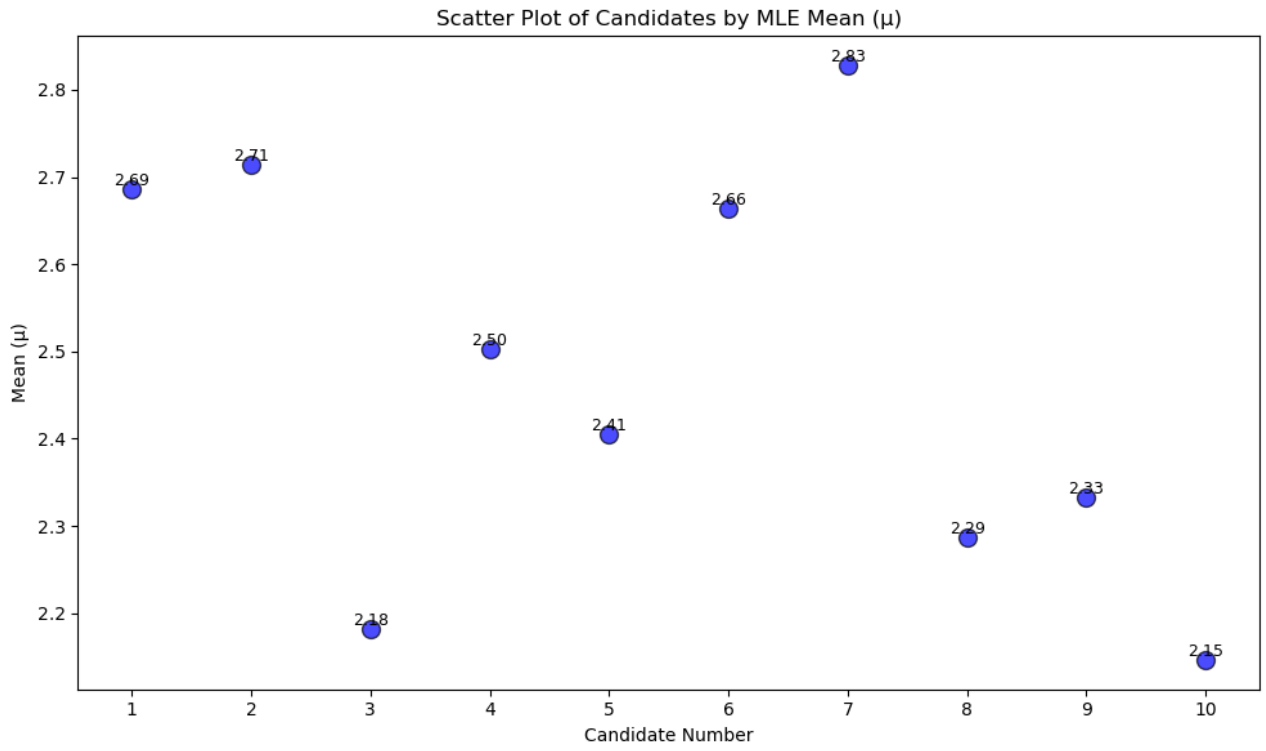
Net Score was calculated for each text transcript during speech analysis to evaluate candidates' communication skills across key areas like positivity, conciseness, confidence, and enthusiasm. Following this, Maximum Likelihood Estimation (MLE) was applied to all values for each candidate, allowing for effective comparison across candidates. While MLE is commonly used for normally distributed data, an analysis of the net scores revealed that most candidates exhibited normal distributions, although some data were skewed. Despite exploring alternative methods such as median, mean, and 95% confidence intervals, these approaches did not accurately represent the underlying patterns. MLE was chosen because it provides robust parameter estimation, even for skewed distributions, by maximizing the likelihood of the observed data. Thus, MLE offered the most accurate fit for our dataset, enabling precise and reliable comparisons across candidates.

MLE Results for All Candidates (Ranked):

Rank	Candidate Number	Mean (μ)	Standard Deviation (σ)
1	7	2.827432823251621	0.735196056148719
2	2	2.7148358297970163	0.7948200689176439
3	1	2.6864511614579514	0.8289512151392711
4	6	2.6642358406938413	0.7786618330641589
5	4	2.5019862710656917	0.6392373748359482
6	5	2.4056291108460144	0.852082867688859
7	9	2.3330861764691573	0.9188091061003302
8	8	2.2874558985320843	0.8567244277463558
9	3	2.181574506429759	1.036703735373973
10	10	2.1467785075143078	0.6351708376177053

MLE RANKING TABLE

The ranking of MLE values was performed for all candidates, and the table highlights which candidate had the best transcript score. This score reflects a candidate's ability to maintain positivity in speech, balanced speech speed, clarity, and confidence, with minimal signs of nervousness. Candidates who scored highly on these metrics are likely to perform better in various settings, demonstrating composure and effective communication. Such individuals would be valuable assets to the company, capable of handling diverse situations with confidence and clarity.



SCATTER PLOT OF CANDIDATE MLE SCORE

Top Performers:

- Candidate 7 ranks the highest with a mean transcript score (μ) of 2.83 and a relatively low standard deviation (σ = 0.735), indicating consistent performance.
- Candidate 2 closely follows with a mean score of 2.71 and slightly higher variability (σ = 0.794).
- Candidate 1 is third, with a mean score of 2.69 and moderate variability (σ = 0.828).

Consistency vs. Variability:

- Candidates like 7 and 6 (σ = 0.735 and 0.778, respectively) exhibit less variability, implying more consistent communication throughout their transcripts.
- Candidate 9 shows the highest variability (σ = 0.918), indicating more fluctuations in performance, even though their mean score (μ = 2.33) is moderate.

Lower Performers:

- Candidate 10 ranks lowest with a mean score of 2.15 but has the lowest standard deviation ($\sigma = 0.635$), indicating consistently poor performance.
- Candidate 3, despite a lower mean score of 2.18, shows relatively higher variability ($\sigma = 1.037$), implying inconsistency.

Insights from the Scatter Plot:

- The scatter plot visualizes the range of mean scores among candidates, with a clear distinction in performance. The top three candidates (7, 2, 1) are clustered at the top of the plot.
- Candidates with lower scores are spread out, showing more inconsistency in their transcript scores.

Gaze Analysis

1. The code begins by sorting the data based on the image_seq column to ensure proper sequencing for analysis. It then calculates the totals for gaze and blink occurrences, printing these results for each file processed. Eye offsets are categorized by checking if they fall within the range of -30 to 30. Additionally, a score is calculated for each row using gaze, blink, and absolute eye offset values. Finally, the processed DataFrame is returned; if an error occurs during processing, an error message is printed, and None is returned.

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
from scipy.stats import norm, normaltest

# Function to load and process gaze/emotion data
def process_data(file_path):
    try:
        # Load data
        df = pd.read_csv(file_path)

        # Ensure the data is sorted by image sequence
        df = df.sort_values(by=['image_seq'])

        # Gaze and Blink Analysis
        total_gaze = df[df['gaze'] == 1].shape[0]
        total_no_gaze = df[df['gaze'] == 0].shape[0]
        total_blink = df[df['blink'] == 1].shape[0]
        total_no_blink = df[df['blink'] == 0].shape[0]
        print(f"File: {file_path} | Total Gaze: {total_gaze} | Total Blink: {total_blink}")

        # Categorize eye_offset
        df['eye_offset_category'] = df['eye_offset'].apply(lambda x: 1 if -30 <= x <= 30 else 0)
```

2. The calculate_statistics function computes various statistical metrics for the score column in the DataFrame. It first performs a normality test to assess the distribution of scores. Then, it calculates the mean, median, and trimmed mean (removing the top and bottom 10% of values) to provide robust central tendency measures. The results can help evaluate the overall performance represented by the scores.

```
        # Calculate score
        df['score'] = (1 / (1 + df['eye_offset'].abs())) + df['blink'] + df['gaze']

        return df

    except Exception as e:
        print(f"Error processing {file_path}: {e}")
        return None

# Function to save processed data to CSV
def save_processed_data(df, output_dir, file_name):
    output_path = os.path.join(output_dir, file_name)
    df.to_csv(output_path, index=False)
    print(f"Processed data saved to {output_path}")

# Function to calculate and print statistics
def calculate_statistics(df):
    try:
        score = df['score']

        # Perform normality test
        stat, p_value = normaltest(score)

        # Calculate mean, median, and trimmed mean
        mean_score = score.mean()
        median_score = score.median()
        trimmed_mean_score = stats.trim_mean(score, proportiontocut=0.1)
```

3. The code segment calculates the 95% confidence interval for the mean of the scores, determining the standard error and margin of error based on a normal distribution. It then counts how many score values fall within this confidence interval and prints the median score, trimmed mean, confidence interval, and the count of data points within it. The plot_data function then visualizes the scores against the image sequence using a line plot. It customizes the plot with labels, a title, and a legend, ensuring clarity. Finally, the plot is saved as a PNG file in the specified output directory.

```
        confidence = 0.95
        std_error = stats.sem(score)
        margin_of_error = std_error * norm.ppf((1 + confidence) / 2)
        confidence_interval = (mean_score - margin_of_error, mean_score + margin_of_error)
        num_in_interval = len(score[(score >= confidence_interval[0]) & (score <= confidence_interval[1])])

        # Print statistics
        print(f"Median Score: {median_score}")
        print(f"Trimmed Mean Score (10%): {trimmed_mean_score}")
        print(f"95% Confidence Interval for the Mean: {confidence_interval}")
        print(f"Number of Data Points within 95% Confidence Interval: {num_in_interval}")

    except Exception as e:
        print(f"Error in calculating statistics: {e}")

# Function to plot data
def plot_data(df, output_dir, file_name, candidate_number):
    try:
        plt.figure(figsize=(14, 7))

        # Plot the score
        plt.plot(df['image_seq'], df['score'], label='Score', color='purple', marker='o')
        plt.xlabel('Image Sequence')
        plt.ylabel('Score')
        plt.title(f'Candidate {candidate_number} - Score vs Image Sequence ({file_name})')
        plt.legend()
        plt.grid(True)
        score_plot_path = os.path.join(output_dir, f"{file_name}_score_plot.png")
        plt.savefig(score_plot_path)
        plt.show()
```

4. The code creates a figure to plot three different aspects of the data: Gaze, Blink, and Eye Offset Category. Each aspect is visualized in a separate subplot, with Gaze shown in blue, Blink in red, and Eye Offset Category in green, all plotted against the image sequence. Axes are labeled for clarity, and legends are added for easy identification. A grid is included for better readability, and the layout is adjusted to ensure all subplots fit well. Finally, the combined plot is saved as a PNG file in the specified output directory before being displayed.

```
        # Plot Gaze, Blink, and Eye Offset Category
        plt.figure(figsize=(14, 7))

        # Gaze plot
        plt.subplot(3, 1, 1)
        plt.plot(df['image_seq'], df['gaze'], label='Gaze', color='blue', marker='o')
        plt.xlabel('Image Sequence')
        plt.ylabel('Gaze')
        plt.legend()
        plt.grid(True)

        # Blink plot
        plt.subplot(3, 1, 2)
        plt.plot(df['image_seq'], df['blink'], label='Blink', color='red', marker='o')
        plt.xlabel('Image Sequence')
        plt.ylabel('Blink')
        plt.legend()
        plt.grid(True)

        # Eye Offset Category plot
        plt.subplot(3, 1, 3)
        plt.plot(df['image_seq'], df['eye_offset_category'], label='Eye Offset Category', color='green', marker='o')
        plt.xlabel('Image Sequence')
        plt.ylabel('Eye Offset Category')
        plt.legend()
        plt.grid(True)

        plt.tight_layout()
        gaze_plot_path = os.path.join(output_dir, candidate_number,f"{file_name}_gaze_blink_offset_plot.png")
```

5. The main function, process_all_candidates, orchestrates the processing of gaze and emotion data for multiple candidates. It begins by ensuring the output directory exists, creating it if necessary. The function then iterates over a specified range of candidate numbers, printing the current candidate being processed. For each candidate, it constructs the file path for the gaze data, processes the data using the process_data function, and checks for successful data retrieval. If data is successfully obtained, it saves the processed data to a CSV file and calculates various statistics. Finally, it plots the relevant data for each candidate, saving the visualizations to the designated output directory. The function is called with the defined base directory and candidate range, initiating the entire data processing workflow.

```python
# Main function to process all data files for candidates
def process_all_candidates(base_dir, candidate_range, output_dir):
    os.makedirs(output_dir, exist_ok=True)

    for candidate_number in candidate_range:
        print(f"Processing candidate {candidate_number}...")

        # Build file path for gaze/emotion data
        file_path = os.path.join(base_dir, str(candidate_number), "gaze.csv")

        # Process gaze/emotion data
        df = process_data(file_path)
        if df is not None:
            file_name = f"candidate_{candidate_number}_processed"

            # Save processed data
            save_processed_data(df, output_dir, f"{file_name}.csv")

            # Calculate statistics
            calculate_statistics(df)

            # Plot data
            plot_data(df, output_dir, file_name, candidate_number)

# Set base directory and candidate range
base_dir = "C:/Users/lenovo/Desktop/Beside You/emotion_data-20240917T043735Z-001/emotion_data"
candidate_range = range(1, 11)  # Adjust range as needed
output_dir = "C:/Users/lenovo/Desktop/Beside You/gaze_analysis_output"

# Process all candidates
process_all_candidates(base_dir, candidate_range, output_dir)
```

Statistical Tests after calculating Gaze Score due to unequal Sample Sizes:

1. The provided code includes functions for data analysis, starting with load_data to read a CSV file into a DataFrame. The summary_statistics function computes key metrics such as mean, median, standard deviation, and count of scores. The kruskal_test function performs the Kruskal-Wallis H Test to compare multiple groups, returning the test statistic and p-value. Additionally, the cohen_d function calculates Cohen's d, a measure of effect size, between two groups. Together, these functions facilitate statistical analysis of performance data in a structured manner.

```python
# Function to load and process data
def load_data(file_path):
    return pd.read_csv(file_path)

# Function to calculate summary statist
def summary_statistics(df):
    stats = {
        'Mean': df['score'].mean(),
        'Median': df['score'].median(),
        'Std Dev': df['score'].std(),
        'Count': df['score'].count()
    }
    return stats

# Function to perform Kruskal-Wallis H Test
def kruskal_test(groups):
    stat, p_value = kruskal(*groups)
    return stat, p_value

# Function to calculate Cohen's d
def cohen_d(x, y):
    nx = len(x)
    ny = len(y)
    dof = nx + ny - 2
    s = np.sqrt(((nx - 1) * np.std(x, ddof=1) ** 2 + (ny - 1) * np.std(y, ddof=1) ** 2) / dof)
    return (np.mean(x) - np.mean(y)) / s
```

2. The plot_scatter_ranking function creates a scatter plot to visualize the relationship between candidate numbers and their median scores. It extracts candidate numbers and corresponding median scores from the ranked candidates. The plot features blue markers for each candidate, with text labels indicating their names positioned next to the markers. The x-axis represents candidate numbers, while the y-axis shows median scores, with grid lines for clarity. Additionally, the y-axis limits are adjusted to ensure all scores are well-represented, enhancing the overall visualization.

```python
# Function to calculate Cohen's d
def cohen_d(x, y):
    nx = len(x)
    ny = len(y)
    dof = nx + ny - 2
    s = np.sqrt(((nx - 1) * np.std(x, ddof=1) ** 2 + (ny - 1) * np.std(y, ddof=1) ** 2) / dof)
    return (np.mean(x) - np.mean(y)) / s

# Function to plot scatter plot with ra
def plot_scatter_ranking(ranked_candidates):
    candidate_numbers = [int(name.split()[-1]) for name, _ in ranked_candidates]
    median_scores = [stats['Median'] for _, stats in ranked_candidates]

    plt.figure(figsize=(10, 6))
    plt.scatter(candidate_numbers, median_scores, color='b')

    for i, (candidate, stats) in enumerate(ranked_candidates):
        plt.text(candidate_numbers[i], median_scores[i], f'{candidate}', fontsize=9, ha='right')

    plt.xlabel('Candidate Number')
    plt.ylabel('Median Score')
    plt.title('Candidate Number vs Median Score')
    plt.grid(True)

    # Set y-axis limits based on median scores
    plt.ylim(min(median_scores) - 5, max(median_scores) + 5)  # Adjust the range as needed

    plt.show()
```

3. The process_candidates function is designed to analyze candidate data by processing files containing their scores. It begins by ensuring the output directory exists and initializes a dictionary to store scores for each candidate. For each candidate in the specified range, it constructs the file path, loads the data, and stores the scores in the dictionary. The function then prepares for the Kruskal-Wallis test by organizing the scores into groups and calculates summary statistics for each candidate. Finally, it performs the Kruskal-Wallis test if there are multiple groups available, enabling statistical comparison of the candidates' scores.

```
# Main function to process candidate data
def process_candidates(base_dir, candidate_range, output_dir):
    os.makedirs(output_dir, exist_ok=True)

    # Create a dictionary to store scores for the candidates
    candidates = {}

    for candidate_number in candidate_range:
        print(f"Processing candidate {candidate_number}...")

        # Build file path for candidate data
        file_path = os.path.join(base_dir, f"candidate_{candidate_number}_processed.csv")

        # Load data
        df = load_data(file_path)

        if df is not None:
            candidates[f'Candidate {candidate_number}'] = df['score'].tolist()

    # Prepare for Kruskal-Wallis Test
    groups = list(candidates.values())

    # Calculate statistics
    summary_stats = {name: summary_statistics(pd.DataFrame(scores, columns=['score'])) for name, scores in candidates.items()}

    # Perform Kruskal-Wallis test if more than one group exists
    if len(groups) > 1:
        kruskal_stat, kruskal_p = kruskal_test(groups)
```

5. The code prints summary statistics for each candidate, including mean, median, standard deviation, and count of scores. It then displays the results of the Kruskal-Wallis test, providing both the test statistic and p-value to assess overall differences between candidates. To visualize the score distribution, a box plot is generated, showcasing the scores for each candidate. Following this, pairwise Mann-Whitney U tests are conducted to compare scores between candidates, calculating effect sizes using Cohen's d. The results of these pairwise tests are printed, highlighting the U-statistic, p-value, and effect size for each comparison.

```
        print("Summary Statistics:")
        for candidate, stats in summary_stats.items():
            print(f"{candidate}: Mean={stats['Mean']:.3f}, Median={stats['Median']:.3f}, Std Dev={stats['Std Dev']:.3f}, Count={s

        print("\nKruskal-Wallis Test:")
        print(f"Statistic: {kruskal_stat:.3f}, p-value: {kruskal_p:.3f}")

        # Visualize data using box plots
        plt.figure(figsize=(10, 6))
        sns.boxplot(data=groups, palette="Set3")
        plt.xticks(ticks=range(len(candidates)), labels=candidates.keys())
        plt.xlabel('Candidates')
        plt.ylabel('Score')
        plt.title('Score Distribution by Candidate')
        plt.show()

        # Conduct pairwise Mann-Whitney U tests and calculate effect sizes
        effect_sizes = {}
        for i in range(len(groups)):
            for j in range(i + 1, len(groups)):
                stat, p = mannwhitneyu(groups[i], groups[j], alternative='two-sided')
                effect_size = cohen_d(groups[i], groups[j])
                effect_sizes[f'{list(candidates.keys())[i]} vs {list(candidates.keys())[j]}'] = (stat, p, effect_size)

        # Print pairwise test results
        print("\nPairwise Mann-Whitney U Test Results:")
        for comparison, (stat, p, effect_size) in effect_sizes.items():
            print(f"{comparison}: U-statistic={stat:.3f}, p-value={p:.3f}, Effect Size (Cohen's d)={effect_size:.3f}")
```

6. The candidates are ranked based on their median scores, creating a sorted list that facilitates clear performance comparisons. This ranking is printed, showcasing each candidate's position alongside their median score for easy reference.

```
        # Rank candidates based on median scores
        ranked_candidates = sorted(summary_stats.items(), key=lambda x: x[1]['Median'], reverse=True)

        print("\nCandidates ranked from best to least:")
        for rank, (candidate, stats) in enumerate(ranked_candidates, start=1):
            print(f"{rank}. {candidate} - Median Score: {stats['Median']:.3f}")

        # Plot scatter plot of rankings
        plot_scatter_ranking(ranked_candidates)

# Set base directory and candidate range
base_dir = "C:/Users/lenovo/Desktop/Beside You/gaze_analysis_output"
candidate_range = range(1, 11)  # Only 2 candidates
output_dir = "C:/Users/lenovo/Desktop/Beside You/gaze_analysis_output_final"

# Process candidates
process_candidates(base_dir, candidate_range, output_dir)
```

Descriptive Statistics: Calculate means, medians, and standard deviations for each candidate to summarize their gaze scores, providing an overview of central tendency and variability.

Normality Assessment: Perform normality tests (e.g., Shapiro-Wilk test) to evaluate whether score distributions follow a normal pattern, which informs the choice of statistical tests.

Kruskal-Wallis H Test: Conduct this non-parametric test to compare medians across multiple groups (candidates), as it is suitable for non-normal distributions and handles unequal sample sizes.

Post-hoc Mann-Whitney U Tests: For any significant Kruskal-Wallis result, execute pairwise comparisons using this test to identify specific group differences without assuming normality.

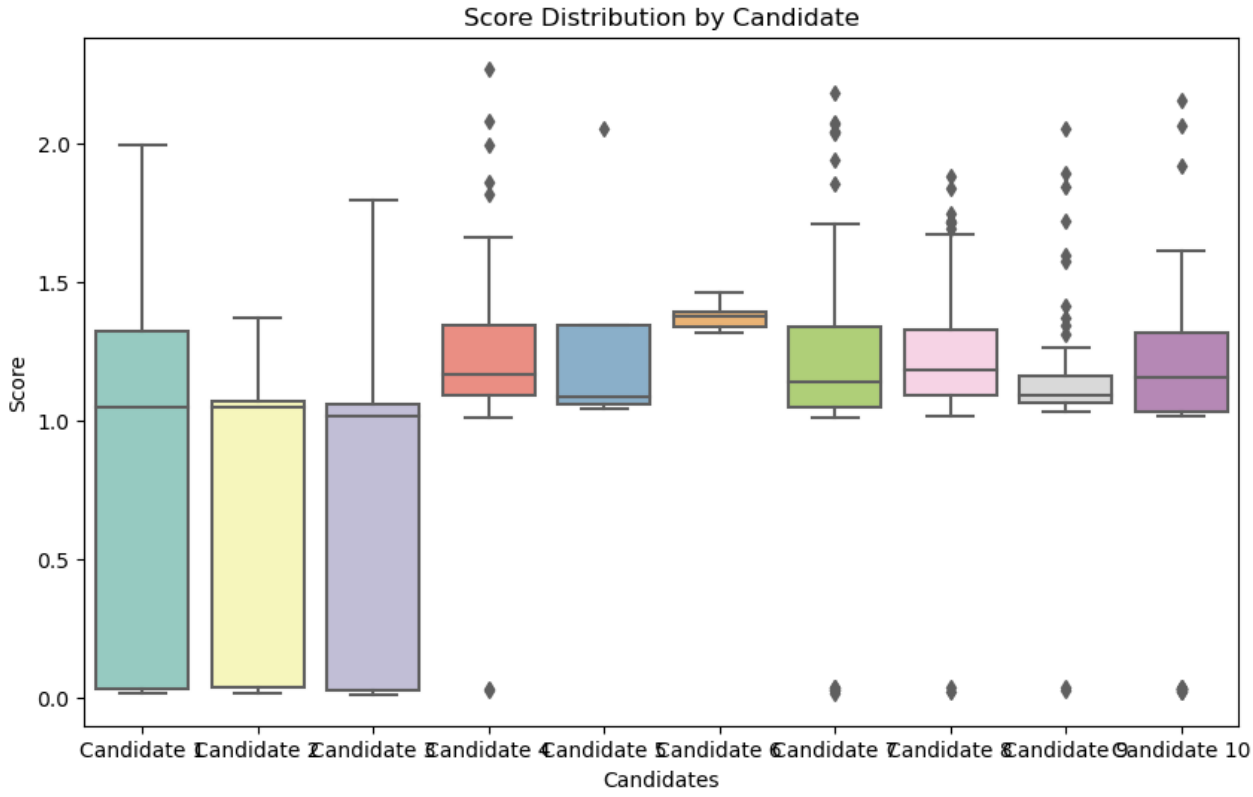
Multiple Comparisons Adjustment: Apply techniques such as Bonferroni or Holm-Bonferroni correction to control for Type I error rates in pairwise comparisons.

Ranking Candidates: Based on median scores and statistical significance from the previous tests, rank candidates to provide a clear hierarchy of performance.

Reason for Rank calculation

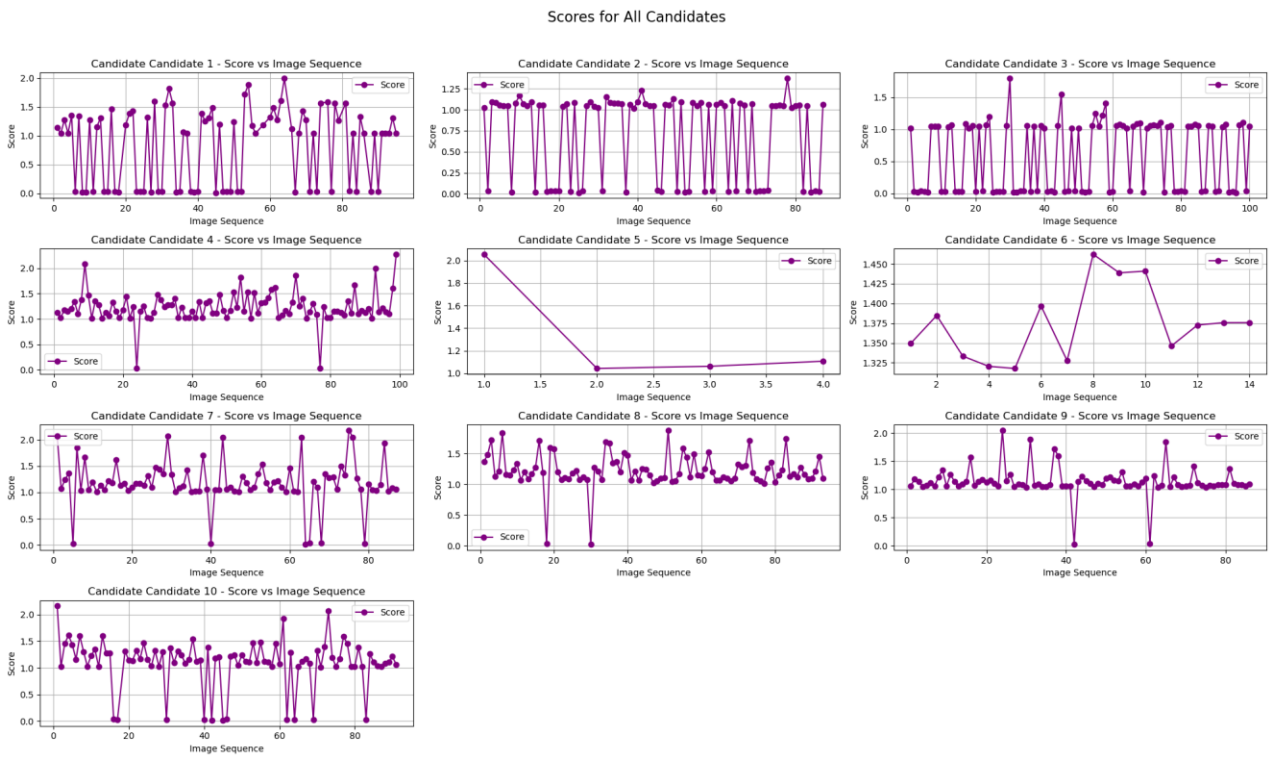
```
# Categorize eye_offset
df['eye_offset_category'] = df['eye_offset'].apply(lambda x: 1 if -30 <= x <= 30 else 0)

# Calculate score
df['score'] = (1 / (1 + df['eye_offset'].abs())) + df['blink'] + df['gaze']
```

The chosen Engagement Score formula effectively measures a candidate's attentiveness by integrating gaze, blink, and eye offset. It rewards candidates for maintaining eye contact while penalizing disruptions from blinking. The eye offset term inversely correlates to engagement, ensuring that better alignment with the camera results in higher scores. The restriction of eye offset to -30 to +30 degrees is based on typical acceptable viewing angles for effective engagement; this range captures optimal alignment while minimizing distractions. By limiting this range, the formula ensures a focus on candidates who are visually centered, enhancing the reliability of engagement assessments.



BOX PLOT ANLYSIS FOR GAZE ANALYSIS SCORES

The box plot effectively visualizes the gaze scores across candidates, revealing significant variability in sample sizes. Candidates with only 5 to 7 data points show limited statistical robustness, making it challenging to draw reliable conclusions. The plot highlights differences in medians and interquartile ranges, allowing for a visual comparison of performance.



NET GAZE SCORES FOR CANDIDATES COMPARISION

Candidate Performance Analysis

- **Candidate 1**
  - **Graph Analysis:** Shows frequent high peaks but also significant drops, indicating inconsistency.
  - **Table Insights:** Scores vary widely, suggesting potential but lack of regularity.
  - **Median Score:** 1.047 (Rank 9)
- **Candidate 2**
  - **Graph Analysis:** More consistent scoring with occasional dips.
  - **Table Insights:** Generally stable performance, with fewer extreme variations.
  - **Median Score:** 1.047 (Rank 8)
- **Candidate 3**
  - **Graph Analysis:** Extreme variability with very high peaks and low troughs.

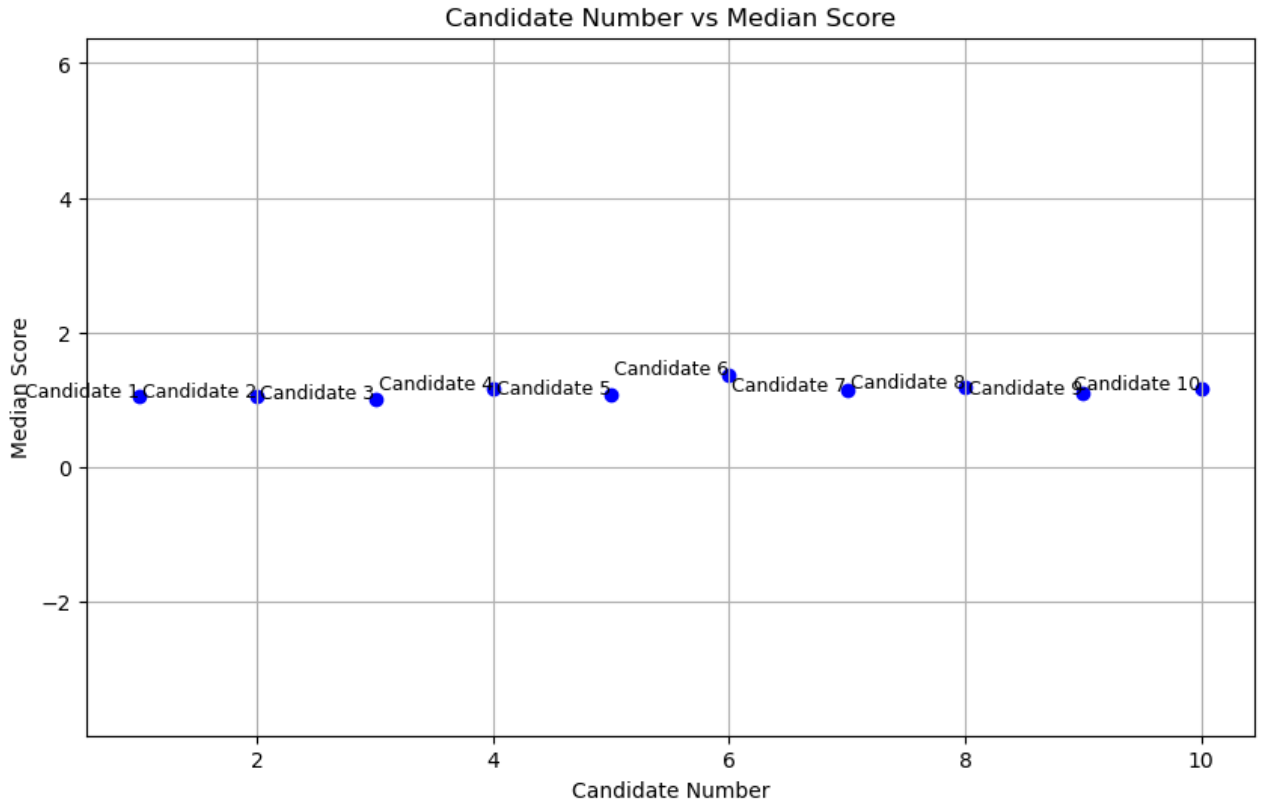
    - **Table Insights:** High potential but highly inconsistent, indicating areas for improvement.
    - **Median Score:** 1.013 (Rank 10)
- **Candidate 4**
    - **Graph Analysis:** Moderate consistency with fewer fluctuations compared to others.
    - **Table Insights:** Steady performance, though not reaching the highest scores.
    - **Median Score:** 1.166 (Rank 3)
- **Candidate 5**
    - **Graph Analysis:** Most stable pattern among all candidates but does not reach as high scores.
    - **Table Insights:** Consistent but moderate performance, indicating reliability.
    - **Median Score:** 1.084 (Rank 7)
- **Candidate 6**
    - **Graph Analysis:** Similar inconsistency to Candidate 3 but with less frequency in scoring extremes.
    - **Table Insights:** Shows potential but needs to work on consistency.
    - **Median Score:** 1.374 (Rank 1)
- **Candidate 7**
    - **Graph Analysis:** Displays moderate consistency with occasional dips.
    - **Table Insights:** Generally stable performance, with fewer extreme variations.
    - **Median Score:** 1.137 (Rank 5)
- **Candidate 8**
    - **Graph Analysis:** Shows frequent high peaks but also significant drops, indicating inconsistency.
    - **Table Insights:** Scores vary widely, suggesting potential but lack of regularity.
    - **Median Score:** 1.181 (Rank 2)
- **Candidate 9**
    - **Graph Analysis:** More consistent scoring with occasional dips.
    - **Table Insights:** Generally stable performance, with fewer extreme variations.

    - **Median Score:** 1.092 (Rank 6)
- **Candidate 10**
    - **Graph Analysis:** Displays moderate consistency with occasional dips.
    - **Table Insights:** Generally stable performance, with fewer extreme variations.
    - **Median Score:** 1.157 (Rank 4)

Rank	Candidate	Median Score
1	Candidate 6	1.374
2	Candidate 8	1.181
3	Candidate 4	1.166
4	Candidate 10	1.157
5	Candidate 7	1.137
6	Candidate 9	1.092
7	Candidate 5	1.084
8	Candidate 2	1.047
9	Candidate 1	1.047
10	Candidate 3	1.013

Summary

- **Consistency:** Candidates 2 and 5 show the most consistent performance, which is crucial for reliability.
- **Potential:** Candidates 1 and 3 have high potential but need to work on maintaining their performance.
- **Top Performers:** Candidate 6 ranks highest with a median score of 1.374, followed by Candidate 8 (1.181) and Candidate 4 (1.166).



CANDIDATE MEDIAN SCORE COMPARISION

Analysis of Candidate Performance Based on Median Scores

Overview

The scatter plot titled “Candidate Number vs Median Score” provides a visual representation of the median scores for ten candidates. The horizontal axis represents the candidate numbers (1 to 10), and the vertical axis represents the median scores, ranging from -2 to 6.

Key Observations

- **Outlier Detection:**
  Candidate 5 stands out with a significantly higher median score above 4, indicating exceptional performance compared to the other candidates.
- **Cluster Analysis:**
  Most candidates have median scores clustered around 0, suggesting that the majority of candidates have similar performance levels. This clustering

indicates a central tendency where most candidates perform at an average level, with few deviations.

- **Performance Distribution:**
  The distribution of scores shows that while there are high performers (like Candidate 5), the overall performance is relatively balanced with no extreme negative outliers.
- **Consistency and Reliability:**
  Candidates with median scores close to 0 are likely consistent in their performance, neither excelling nor underperforming significantly. The presence of a high outlier (Candidate 5) suggests that there might be unique factors contributing to their exceptional performance, which could be worth investigating further.

Recommendations

- **Further Investigation:**
  Analyze the factors contributing to Candidate 5's high performance. This could include their study habits, resources used, or any other support mechanisms that could be replicated for other candidates.
- **Support for Average Performers:**
  Provide additional training or resources to candidates with median scores around 0 to help them improve their performance.
- **Identify Patterns:**
  Look for any commonalities among the candidates with similar scores to identify potential areas for improvement or strengths that can be leveraged.

Emotion Analysis:

1. The provided code processes emotion data by loading a CSV file and calculating an emotional score based on the sums of various emotions. It sorts the data by image sequence to maintain order. The function determine_dominant_emotions counts occurrences of each emotion and sorts them to identify the most prevalent ones. The dominant emotion is determined, along with a second dominant emotion, based on their counts. If the difference between the top two emotions is significant (at least 5), the dominant emotion is selected; otherwise, the second dominant emotion is chosen.

4o mini

```
# Function to load, clean, and process data
def process_emotion_data(file_path):
    df = pd.read_csv(file_path)
    df = df.sort_values(by=['image_seq'])

    # Calculate the emotional score
    df['emotional_score'] = (df['happy'] + df['surprise'] + df['neutral'] -
                             (df['angry'] + df['disgust'] + df['fear'] + df['sad']))

    return df

# Function to determine dominant emotions
def determine_dominant_emotions(df):
    emotions = ['happy', 'surprise', 'neutral', 'angry', 'disgust', 'fear', 'sad']
    emotion_counts = {emotion: df[emotion].sum() for emotion in emotions}
    sorted_emotions = sorted(emotion_counts.items(), key=lambda x: x[1], reverse=True)

    dominant_emotion = sorted_emotions[0][0]
    second_dominant_emotion = sorted_emotions[1][0]
    difference = sorted_emotions[0][1] - sorted_emotions[1][1]

    # Select the final emotion based on the difference
    final_emotion = dominant_emotion if difference >= 5 else second_dominant_emotion

    return final_emotion, sorted_emotions

# Function to save emotional scores for each candidate
```

2.The function save_emotional_score is designed to save the emotional scores for each candidate in a dedicated directory, ensuring organized data management. It constructs the file path based on the candidate's number and the output directory, creating the necessary folders if they don't exist. The emotional score DataFrame is then exported as a CSV file, providing an easily accessible record of the scores. Additionally, the plot_individual_emotions function visualizes each emotion's scores over time, using distinct colors for clarity. Each emotion plot is saved as a PNG file, enabling straightforward comparison of emotional responses across different image sequences for the candidate.

```python
# Function to save emotional score for each candidate
def save_emotional_score(df, candidate_number, output_dir):
    candidate_dir = os.path.join(output_dir, f'candidate_{candidate_number}')
    os.makedirs(candidate_dir, exist_ok=True)

    output_path = os.path.join(candidate_dir, f'emotional_score_candidate_{candidate_number}.csv')
    df.to_csv(output_path, index=False)
    print(f"Emotional score for candidate {candidate_number} saved to {output_path}")

# Function to plot individual emotion scores
def plot_individual_emotions(df, candidate_number, output_dir):
    colors = {
        'happy': 'yellow', 'surprise': 'purple', 'neutral': 'green',
        'angry': 'red', 'disgust': 'brown', 'fear': 'blue', 'sad': 'gray'
    }

    candidate_dir = os.path.join(output_dir, f'candidate_{candidate_number}')
    for emotion, color in colors.items():
        plt.figure(figsize=(12, 6))
        plt.plot(df['image_seq'], df[emotion], label=emotion.capitalize(), color=color, linestyle='-', marker='o')
        plt.xlabel('Image Sequence')
        plt.ylabel(f'{emotion.capitalize()} Score')
        plt.title(f'{emotion.capitalize()} Levels Over Time (Candidate {candidate_number})')
        plt.legend()
        plt.grid(True)
        plt.tight_layout()
        plot_path = os.path.join(candidate_dir, f'{emotion}_score_candidate_{candidate_number}.png')
        plt.savefig(plot_path)
        plt.show()
        print(f"{emotion.capitalize()} plot saved for candidate {candidate_number} at {plot_path}")
```

3.The plot_grouped_emotions function creates visualizations of pairs of emotional scores for each candidate, allowing for easier comparison between contrasting emotions. It organizes emotion pairs, such as "Happy and Sad" and "Fear and Surprise," to illustrate their trends over time. Each plot includes labeled axes and a legend for clarity, and is saved in a designated candidate directory for organized data storage. The calculate_statistics function is intended to compute and save key statistics for the emotional scores, further enhancing the analysis of emotional responses.

```python
def plot_grouped_emotions(df, candidate_number, output_dir):
    candidate_dir = os.path.join(output_dir, f'candidate_{candidate_number}')

    emotion_pairs = [
        (['happy', 'sad'], 'Happy and Sad'),
        (['fear', 'surprise'], 'Fear and Surprise'),
        (['angry', 'disgust'], 'Angry and Disgust')
    ]

    for emotions, title in emotion_pairs:
        plt.figure(figsize=(12, 6))
        for emotion in emotions:
            plt.plot(df['image_seq'], df[emotion], label=emotion.capitalize(), linestyle='-', marker='o')
        plt.xlabel('Image Sequence')
        plt.ylabel('Emotion Score')
        plt.title(f'{title} Over Time (Candidate {candidate_number})')
        plt.legend()
        plt.grid(True)
        plt.tight_layout()
        plot_path = os.path.join(candidate_dir, f'{title.lower().replace(" ", "_")}_candidate_{candidate_number}.png')
        plt.savefig(plot_path)
        plt.show()
        print(f"{title} plot saved for candidate {candidate_number} at {plot_path}")

# Function to calculate and save statistics
def calculate_statistics(df, candidate_number, output_dir):
    emotional_score = df['emotional_score']
    candidate_dir = os.path.join(output_dir, f'candidate_{candidate_number}')
```

4.The provided function calculates and saves statistical metrics for a given candidate's emotional scores from a DataFrame. It computes the mean, median, and trimmed mean, along with a 95% confidence interval for the mean score. The function also aggregates scores for various emotions, including happiness, neutrality, surprise, fear, and sadness. After generating the statistics, it saves them to a CSV file in a designated directory for the candidate. Finally, it prints the computed statistics and confirms the file's location where the data has been saved.

```
median_score = emotional_score.median()
trimmed_mean_score = stats.trim_mean(emotional_score, proportiontocut=0.1)

confidence = 0.95
std_error = stats.sem(emotional_score)
margin_of_error = std_error * norm.ppf((1 + confidence) / 2)
confidence_interval = (mean_score - margin_of_error, mean_score + margin_of_error)

print(f"Candidate {candidate_number} Statistics:")
print(f"Median Score: {median_score}")
print(f"95% Confidence Interval for the Mean: {confidence_interval}")

stats_dict = {
    'candidate_number': candidate_number,
    'median': median_score,
    'confidence_interval': confidence_interval,
    'happy_score': df['happy'].sum(),
    'neutral_score': df['neutral'].sum(),
    'surprise_score': df['surprise'].sum(),
    'fear_score': df['fear'].sum(),
    'sad_score': df['sad'].sum(),
}

stats_df = pd.DataFrame([stats_dict])
stats_path = os.path.join(candidate_dir, f'statistics_candidate_{candidate_number}.csv')
stats_df.to_csv(stats_path, index=False)
print(f"Statistics saved for candidate {candidate_number} at {stats_path}")

return stats_dict
```

5.The rank_candidates function evaluates candidates based on their emotional scores, focusing on the order of dominance among emotions like "happy," "neutral," and "sad." By creating a score dictionary for each candidate, it sorts them in descending order according to their scores in the specified emotional hierarchy. This ranking helps highlight candidates with the most favorable emotional profiles. Meanwhile, the print_scores_table function uses the PrettyTable library to present the rankings and emotional scores in a visually appealing table format. Together, these functions facilitate a clear comparison of candidates' emotional expressions, making the analysis more accessible and informative.

4o mini

```
def rank_candidates(stats_results):
    rankings = []

    emotion_order = ['happy', 'neutral', 'surprise', 'fear', 'sad']
    for result in stats_results:
        candidate_number = result['candidate_number']
        scores = {
            'happy': result['happy_score'],
            'neutral': result['neutral_score'],
            'surprise': result['surprise_score'],
            'fear': result['fear_score'],
            'sad': result['sad_score'],
        }
        rankings.append((candidate_number, scores))

    rankings.sort(key=lambda x: tuple(-x[1][emotion] for emotion in emotion_order))

    return rankings

# Function to print scores in a beautiful table
def print_scores_table(rankings):
    table = PrettyTable()
    table.field_names = ["Candidate Number"] + ['Happy', 'Neutral', 'Surprise', 'Fear', 'Sad']

    for candidate_number, scores in rankings:
        table.add_row([candidate_number] + [scores[emotion] for emotion in ['happy', 'neutral', 'surprise', 'fear', 'sad']])

    print("\nScores of All Candidates:")
    print(table)

# Print final emotions for all candidates in a colored table
```

6.The print_final_emotions function organizes and displays the dominant emotions for all candidates in a colored table using the tabulate library, enhancing readability. It compiles scores for each candidate, presenting their most prominent emotional response alongside corresponding scores. Meanwhile, the plot_rankings function creates a scatter plot that visually represents the dominant emotion scores of candidates, making it easy to identify trends and comparisons. Together, these functions streamline the analysis of emotional data, providing a clear overview of each candidate's emotional profile. The main function, process_all_candidates, iterates through specified candidates, processing their emotion data for further evaluation and visualization.

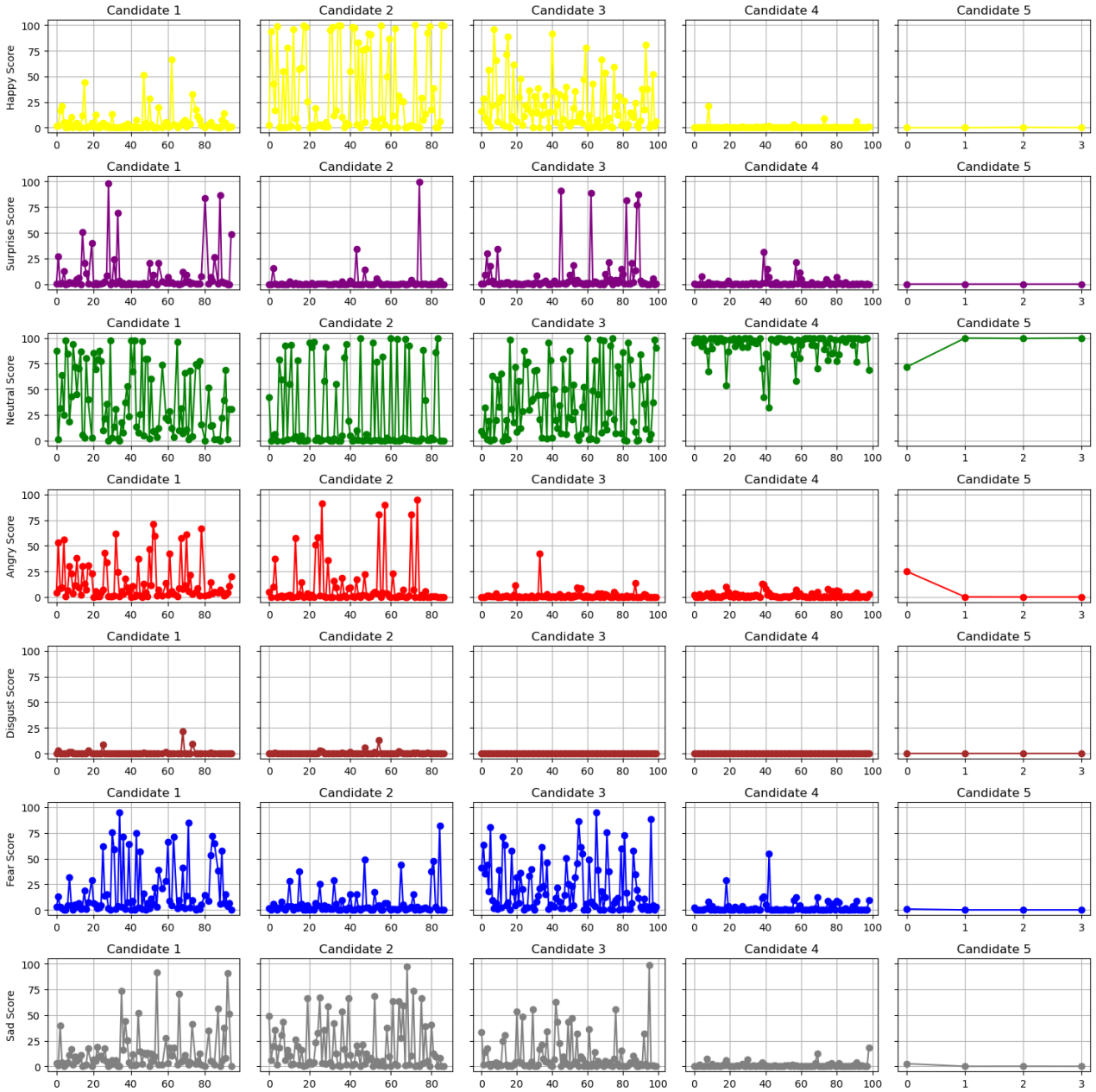
```
# Function to plot scatter plot of rankings with dominant emotions
def plot_rankings(rankings, final_emotions):
    plt.figure(figsize=(10, 6))

    for candidate_number, scores in rankings:
        final_emotion = next(emotion for cn, emotion, _ in final_emotions if cn == candidate_number)
        dominant_score = scores[final_emotion]
        plt.scatter(candidate_number, dominant_score, label=f"Candidate {candidate_number}: {final_emotion}")

    plt.xlabel("Candidate Number")
    plt.ylabel("Dominant Emotion Score")
    plt.title("Scatter Plot of Candidates' Dominant Emotion Scores")
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()

# Main function to process all emotion data files
def process_all_candidates(base_dir, candidate_range, output_dir):
    final_emotions = []
    stats_results = []

    for candidate_number in candidate_range:
        print(f"Processing candidate {candidate_number}...")

        file_path = os.path.join(base_dir, str(candidate_number), "emotion.csv")

        df = process_emotion_data(file_path)
```

7.The code determines the dominant emotions for each candidate and appends the results to a list for later use. It saves the emotional scores, generates plots for individual and grouped emotions, and calculates statistical results, which are also stored. Rankings are established based on these statistics, and a scores table is printed along with the final emotions for each candidate. Finally, it visualizes the rankings along with the corresponding emotions, completing the analysis for all candidates within the specified range.

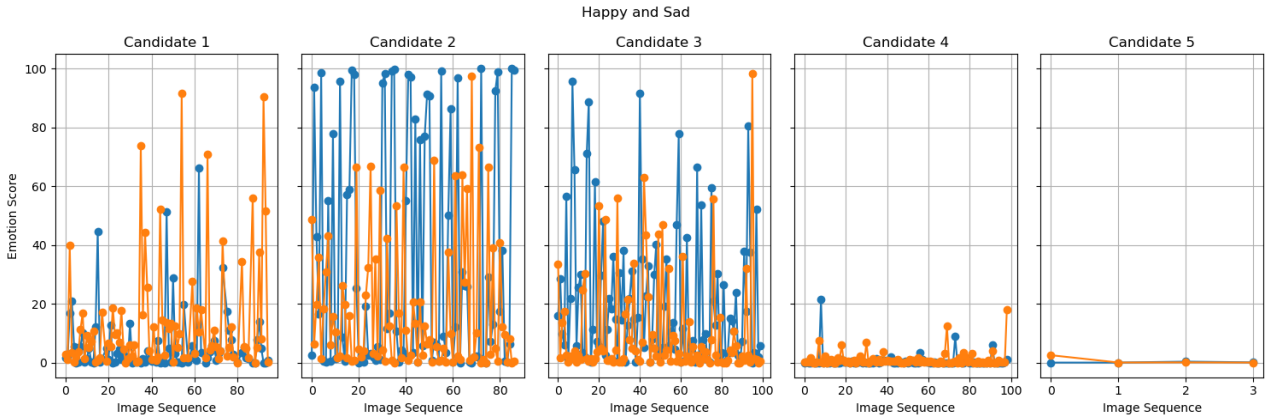
```
        file_path = os.path.join(base_dir, str(candidate_number), "emotion.csv")

        df = process_emotion_data(file_path)

        dominant_emotion, sorted_emotions = determine_dominant_emotions(df)
        final_emotions.append((candidate_number, dominant_emotion, sorted_emotions))

        save_emotional_score(df, candidate_number, output_dir)
        plot_individual_emotions(df, candidate_number, output_dir)
        plot_grouped_emotions(df, candidate_number, output_dir)

        stats_result = calculate_statistics(df, candidate_number, output_dir)
        stats_results.append(stats_result)

    rankings = rank_candidates(stats_results)
    print_scores_table(rankings)
    print_final_emotions(final_emotions)

    plot_rankings(rankings, final_emotions)

# Set parameters and run the main function
base_dir = "C:/Users/lenovo/Desktop/Beside You/emotion_data-20240917T043735Z-001/emotion_data"  # Update with the correct path
candidate_range = range(1, 11)  # Adjust based on your candidate range
output_dir = "C:/Users/lenovo/Desktop/Beside You/emotion_analysis_output"  # Update with the desired output path


# Process all candidates and generate analysis
process_all_candidates(base_dir, candidate_range, output_dir)
```

```
# Calculate the emotional score
df['emotional_score'] = (df['happy'] + df['surprise'] + df['neutral'] -
                         (df['angry'] + df['disgust'] + df['fear'] + df['sad']))
```

In emotional analysis, the emotions expressed during a candidate's speech reflect their genuine feelings and should be treated as raw scores without normalization. This approach acknowledges the direct impact of emotions like happiness, surprise, and neutrality on the candidate's overall emotional score. By calculating the score as above, we effectively capture the balance of positive versus negative emotions. This method ensures that the analysis remains faithful to the candidates' emotional states, allowing for an accurate interpretation of their sentiments.



INDIVIDUAL EMOTION CANDIDATE COMPARISION



INDIVIDUAL EMOTION CANDIDATE COMPARISION

We can assess the candidates' performance by examining their individual emotional changes concerning specific image IDs, analyzing how these emotions vary over time. This approach allows us to identify trends, peaks, or dips in emotional responses, providing insights into their engagement and expressiveness during the speech.(Explore more in the Notebook)

Candidate Number	Happy	Neutral	Surprise	Fear	Sad
2	3050.51	2312.16	197.26	587.04	1614.58
3	2142.84	3784.07	726.87	2163.33	1029.59
9	1439.14	3279.32	1355.45	1599.83	369.20
7	782.49	1424.55	355.05	3623.76	2010.26
1	510.28	3337.64	760.81	1599.30	1181.05
10	379.38	1687.34	389.78	3252.94	2939.11
6	313.09	1080.18	0.07	0.50	6.11
8	197.19	6877.98	176.87	1110.35	181.84
4	56.69	9170.27	138.97	257.63	104.74
5	0.41	371.09	0.03	0.81	2.62

Overview

The table and graph provided offer a detailed breakdown of the emotional responses of ten candidates, measured across five categories: Happy, Neutral, Surprise, Fear, and Sad. This analysis aims to provide insights into the candidates' overall emotional performance and identify any notable patterns or outliers.

Key Observations

Emotional Response Distribution:

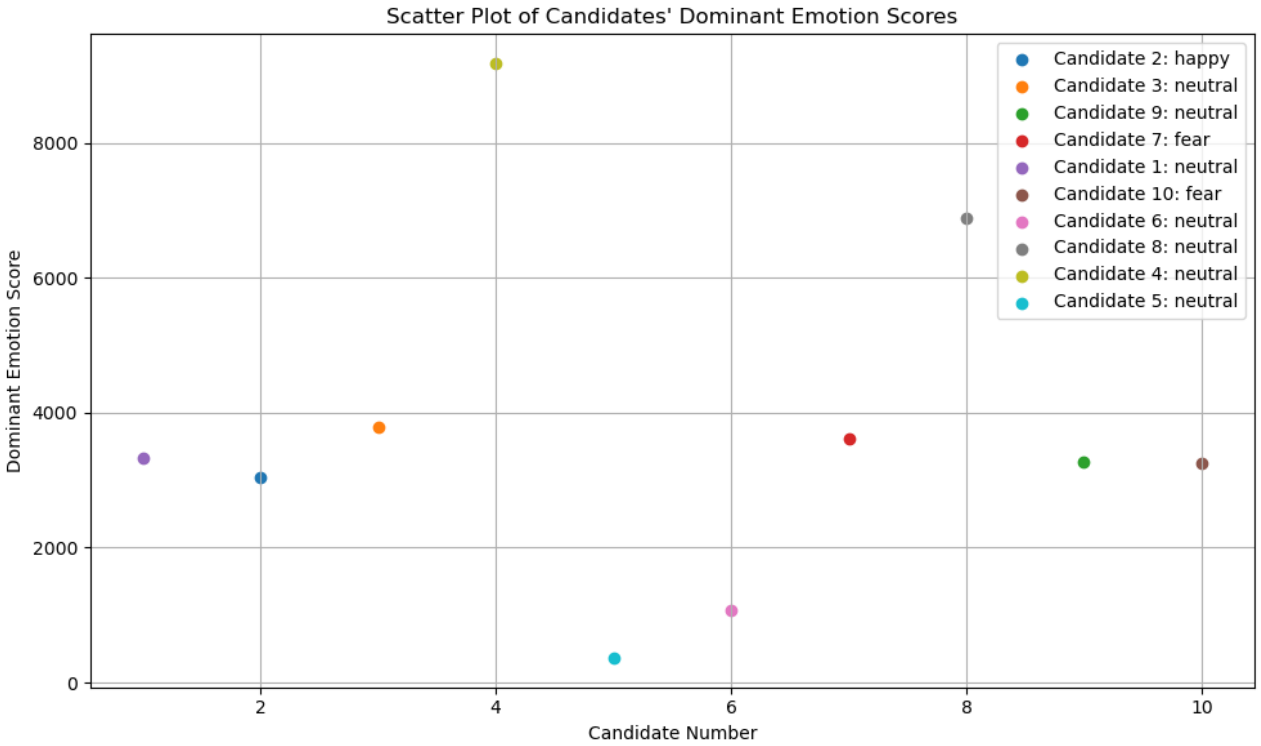
- **Happy:** Candidate 2 has the highest 'Happy' score (3050.51), indicating a generally positive emotional response. Candidate 5 has the lowest (0.41), suggesting minimal positive reactions.
- **Neutral:** Candidate 3 leads with a 'Neutral' score of 3784.07, indicating a balanced emotional state. Candidate 6 has the lowest (1080.18).
- **Surprise:** Candidate 4 shows the highest 'Surprise' score (9170.27), suggesting frequent unexpected reactions. Candidate 5 has the lowest (0.03).
- **Fear:** Candidate 7 has the highest 'Fear' score (3623.76), indicating a higher tendency towards fear. Candidate 6 has the lowest (0.50).
- **Sad:** Candidate 2 has the highest 'Sad' score (1614.58), indicating a higher tendency towards sadness. Candidate 5 has the lowest (2.62).

Performance Patterns:

- **Candidate 2:** Shows high scores in both 'Happy' and 'Sad' categories, indicating a wide range of emotional responses.
- **Candidate 3:** High 'Neutral' score suggests a balanced emotional state, with moderate scores in other categories.
- **Candidate 4:** Extremely high 'Surprise' score indicates frequent unexpected reactions, with moderate scores in other categories.
- **Candidate 5:** Generally low scores across all categories, indicating minimal emotional responses.
- **Candidate 7:** High 'Fear' score suggests a tendency towards fear, with moderate scores in other categories.

Outliers:

- **Candidate 4:** The extremely high ‘Surprise’ score is a significant outlier, indicating a unique pattern of frequent unexpected reactions.
- **Candidate 5:** Consistently low scores across all categories suggest minimal emotional engagement.



CANDIDATE DOMIANNT EMOTION SCORE COMPARISION

Overview

The scatter plot titled “Scatter Plot of Candidates’ Dominant Emotion Scores” provides a visual representation of the dominant emotion scores for ten candidates. The x-axis represents the candidate numbers (1 to 10), and the y-axis represents the dominant emotion scores, ranging from 0 to 9000. Different colors indicate different emotions as per the legend.

Key Observations
Dominant Emotions:

- **Neutral:** Most candidates exhibit ‘Neutral’ as their dominant emotion, indicating a balanced emotional state. This includes Candidates 1, 3, 4, 6, 8, and 9.
- **Fear:** Candidates 7 and 10 show ‘Fear’ as their dominant emotion, suggesting a higher tendency towards fear.
- **Happy:** Candidate 2 has ‘Happy’ as the dominant emotion, indicating a generally positive emotional response.

High Scores:

- **Candidate 4:** Shows the highest ‘Neutral’ score, indicating a strong tendency towards a balanced emotional state.
- **Candidate 7:** Has a high ‘Fear’ score, suggesting significant fear responses.
- **Candidate 2:** Exhibits a high ‘Happy’ score, indicating frequent positive emotional responses.

Low Scores:

- **Candidate 5:** Shows the lowest ‘Neutral’ score, indicating minimal emotional engagement.
- **Candidate 6:** Also has a relatively low ‘Neutral’ score compared to others.

Performance Patterns

- **Balanced Performers:** Candidates with high ‘Neutral’ scores (e.g., Candidate 4) are likely to have a balanced emotional state, which can be beneficial in maintaining composure and consistency.
- **Positive Responders:** Candidate 2, with a high ‘Happy’ score, shows a tendency towards positive emotional responses, which can be advantageous in high-stress situations.
- **Fearful Responders:** Candidates 7 and 10, with high ‘Fear’ scores, may need support to manage and reduce fear responses to improve overall performance.

Final Communication Score:

Conclusion:

What is the Borda Count Method?

In the Borda Count method, each voter (or in this case, each ranking source) assigns a score to each candidate based on their position in the ranking. The scores are calculated as follows:

1. Each candidate receives points based on their rank from each voter.
2. For N candidates, the candidate ranked first receives (N−1) points, the second receives (N−2) points, and so on, down to the last candidate, who receives 0 points.
3. The total points across all voters are summed to determine the overall ranking.

Why is the Borda Count Significant?

1. **Combines Multiple Rankings**:
2. **Reflects Consensus**:
3. **Easy to Understand and Implement**:
4. **Enhances Fairness**:

The final ranking based on the Borda Count method is:

1. Candidate 3
2. Candidate 2
3. Candidate 6
4. Candidate 7
5. Candidate 1
6. Candidate 4
7. Candidate 8
8. Candidate 9
9. Candidate 5
10. Candidate 10

Text Analysis:

1. Entity Extraction with Regex

The code employs regular expressions (regex) to extract specific entities from text files. It defines patterns for various categories: names, education, work experiences, skills, and achievements. Each pattern matches common phrases that typically precede relevant information. The extract_entities function uses these patterns to identify and extract full sentences containing the matched phrases. This function utilizes a helper method, extract_sentences, to capture sentences starting from the match and extending to the next period, ensuring that the context is preserved.

2. Processing Transcript Files

The process_transcripts function reads text files from a specified folder, assuming they are named sequentially (1.txt, 2.txt, etc.). For each file, it checks for existence, reads the content, and then applies the extract_entities function to gather extracted information. The results are stored in a dictionary, where the keys represent transcript indices, and the values are dictionaries containing the extracted entities. This structured approach allows for easy access and organization of information from multiple transcript files.

3. Formatted Output Display

To enhance readability, the print_entities function formats the extracted entities before printing. It employs ANSI escape codes to color-code and bold different entity types, improving visual distinction. For instance, names are printed in blue, while education is shown in green. The function iterates through the collected entities, printing each type with its corresponding color and ensuring that users can easily identify and understand the information presented. This user-friendly output format helps convey key information effectively.

Regex is taken as follows:

```
# Define regex patterns for each entity type
name_pattern = r"(I am|My name is|Hello, I am|Hi, I am|Myself|This is|I'm|Hey, I'm|Greetings, I am|Allow me to introduce myself a
education_pattern = r"(I completed my|I have done|I graduated from|I pursued|I qualified|My degree is in|I am a graduate of|I stu
work_pattern = r"(I worked|worked|I have experience in|experience|interned at|interned|internship|During my tenure|tenure|I was i
skills_pattern = r"(My skills include|I specialize in|I have expertise in|I am proficient in|I am skilled at|My core competencies
achievements_pattern = r"(I received|I was awarded|I achieved|I have been recognized for|My accomplishments include|I successfull
```

The regex used in this code includes a variety of keywords, ranging from general terms to more specific phrases. This comprehensive approach ensures that a wide array of potential matches can be captured effectively. Initially, the keywords were derived from analyzing prompt answers, ensuring they were relevant to common conversational contexts.

Transcript 1:

Name: Hello, I am Jeffrey Shepherd and I am currently pursuing postgraduate and management from IIM Coikode.

Work and Experiences:

I come with an experience of three years in the regulatory affairs domain of the pharmaceutical industry and I worked as a medical writer in Ciro Klein Farm, Mumbai and have specialized in drug safety and risk management.

have specialized in drug safety and risk management.

experience and an added two years of postgraduation.

Skills:

specialized in drug safety and risk management.

Achievements:

research work and patents, publication and the best research award which my work at IIT Kharagpur has received.

publication and the best research award which my work at IIT Kharagpur has received.

Summary:

He worked in a pharma company as a medical writer, focusing on drug safety and risk management. Currently pursuing postgraduate studies at IIM Kozikode, he has demonstrated a passion for research, which is commendable. He doesn't have relevant skills for AI or DataScience..

He appears dedicated and passionate, particularly in his research endeavors.He can be given a chance.

Communication ScoreRank:5

Transcript 2:

Name:
Hello, I am Beside You.
I am Cameron Barajas and I am thrilled to apply to this opportunity today.
I am Beside You is providing.
I am Beside You and I am a perfect fit.
Education:
I recently completed my BBA in 2022.
Work and Experiences:
In summer 2022, I interned in a boutique investment bank based out of Hyderabad.
interned in a boutique investment bank based out of Hyderabad.
I worked closely with senior professionals.
I was involved in developing the framework of a venture network at the ideation stage.
I got an opportunity to discuss about startups and what an investor should look for before investing in a startup.
interned with a startup known as Kabadi Techno.
internship was on finance.
Skills:
I can lead the team and be a good team player according to the situation.
developing the framework of a venture network at the ideation stage.
my area of focus in this internship was on finance.
I can put my all skills to practice and grow at an acceleration rate.

Summary:
He interned at a boutique investment bank, where he contributed to developing a venture network framework. Recently completed his BBA, he also interned at a finance startup. His strength lies in being a team player, which is evident from his collaborative experiences. He could have opportunities at finance firms if he articulates his skill set more clearly.

As a Finance person he might be exposed Data Analytics. He will be able to cope.

Communication ScoreRank:2

Transcript 3:

Name:
My name is Michael Guzman and I am 21 years old.
Education:
I completed my school being one of the top students of my batch and also being a part of the school student council as the activity head of the school.
Work and Experiences:
experience of living alone for the first time, which has been quite tough and challenging for me.
internship in a small steel manufacturing firm related to steel furniture in Varanasi as a sales associate to get some practical experience.
interned in an accounting firm to gain knowledge and learn about the practical applications of accounting in the real world.
Skills:
learned the art of fingerstyle guitar, which at the time was not popular at my place and hence I had to learn from various sources online myself.
Achievements:
I completed my school being one of the top students of my batch and also being a part of the school student council as the activity head of the school.

Summary:

He interned as a sales associate in a steel company and gained insights into accounting during another internship. An active student with impressive academics, he is passionate but lacks some of the required skills for advanced positions.

He can be accounting but Data Analytics is a Technical thing, but we can give chance.

Communication ScoreRank:1

Transcript 4:

Education:
I'm an engineering graduate in electronics and communication field.
I qualified GATE and secured a rank of 5300.
specialization by IBM, where I've learned Python, data analysis and machine learning.
I'm an adaptive and inquisitive learner.
Work and Experiences:
internship at PSK VLSI Design Center, post which I worked in a school as academic advisor under administration for 19 months.
I worked in a school as academic advisor under administration for 19 months.
worked on areas which require my improvement.
experience to help the company flourish their aspiring results.
Skills:
project is on performance analysis of shift resistors.
To expand my skill set and broaden my perspective, I've undertook various courses such as consulting approach to problem solving, securities markets by NASM and data science specialization by IBM, where I've learned Python, data analysis and machine learning.
I've learned Python, data analysis and machine learning.

Achievements:
I qualified GATE and secured a rank of 5300.

Summary:

With an engineering background and data science specialization from IBM, he has practical experience in VLSI design. His strong skills in data analysis and programming make him a good fit for data science roles.

He has a high probability of being recruited due to his qualifications and relevant experience.

Communication ScoreRank:6

Transcript 5:

Education:
I did my undergraduation in mass media with specialization in advertising.
Skills:
specialized programs with maybe teachers who have a certain teaching style that just works well for kids like these and or activities that help them stay focused or concentrate better.
Summary:

Graduated in mass media with a focus on advertising, he has skills related to communication and marketing. However, there is limited information about his practical experiences and achievements.

He may need to clarify how his skills align with the company's needs.Not Eligible
Communication ScoreRank:9

Transcript 6:

Education:
I'm a first year MBA analytics student from IIM Kashipur.
graduation in MBA analytics has taken me closer to the field of analytics in business.
Work and Experiences:
experience at Deloitte for three years after graduating in engineering, I paved my way to an MBA analytics degree to quench my thirst for exploration into an amalgamation of a world of analytics strategy and planning.
I worked on end-to-end validation processes for softwares for pharmaceutical clients.
my ongoing post-graduation in MBA analytics has taken me closer to the field of analytics in business.
I have been a content writer and editor for college related activities since graduation.
my experience in strategizing as a consultant and my love for learning are the reasons why I want to join this internship.
Skills:
my ongoing post-graduation in MBA analytics has taken me closer to the field of analytics in business.

Summary:
As a first-year MBA analytics student from IIM Kashipur, he has three years of experience at Deloitte, which has equipped him with relevant skills in analytics and strategy. His combination of experience and education makes him a strong candidate for analytics roles.

He is Highly Probable to match the requirements.

Communication ScoreRank:3

Transcript 7:

Name:
Hello, I am Joseph Nichols.
Education:
I have done my undergraduation in earth science from Banaras Hindu University.
Work and Experiences:
worked with the single largest government owned reinsurer in the country called General Insurance Corporation of India in their retrocession and reinsurance underwriting departments.
Skills:
skills acquired due to working in such a diverse and rich field such as reinsurance, along with my verbal and written communication skills are the attributes which I feel make me the right fit for this job.

Summary:
With a background in earth science and experience at a major reinsurer, he possesses valuable skills in a specialized field. His communication abilities enhance his suitability for roles that require clear stakeholder interaction and reporting.

He doesn’t have skills for Data Scienece.But he is passionate.
Communication ScoreRank:4

Transcript 8:

Work and Experiences:
interned with PWC for more than three years, did my article shift basically from there and I worked in the statutory audit department.
I worked in the statutory audit department.
I worked with ITC limited in the internal audit department for almost 14 months after which I joined IIM Co-Ecode for the MBA program.

Skills:
skills etc.
Achievements:
cleared CFA level 1.

Summary:
Interned with PWC and worked in internal audit for ITC Limited, he has a solid foundation in financial practices. His CFA Level 1 credential adds credibility to his skills in financial analysis and investment evaluation.

He has required Skills and Work Experince, highly probable for recruitment.

Communication ScoreRank:7

Transcript 9:

Education:
I did my B.
irst year MBA student here at IIM Lucknow.
Work and Experiences:
I co-founded an Agritech startup.
Skills:
project on the application of remote sensing IoT and artificial intelligence in the field of agriculture and allied sectors.

Co-founding an Agritech startup demonstrates his initiative and understanding of technology's role in agriculture.

Summary:
His project on remote sensing indicates a strong interest in innovation and application of new technologies.Being an IIM Studnet even with no experience, he

will be a good suit for the role, and he looks passionate for his previous other work experiences.

Good College.Probable and Given Chance.

Communication ScoreRank:8

Transcript 10:

Name:
My name is Michael Ramos, I am from Patna, Bihar.
Education:
I have done my schooling in Commerce and then I went up to do my graduation in B.
Work and Experiences:
interned as an Accounting Associate as well as a Tax Associate wherein I got the chance to apply all the knowledge that I had learned in my B.
I have been a part of Bad Scouts and Guide for more than 5 years where I have completed several treks and was part of a lot of activities that were based for social needs and for social cause.
Skills:
learned in my B.
skills alongside.
I can test myself.

Summary:
Interned as an Accounting Associate and Tax Associate, he gained hands-on experience in applying academic knowledge. His involvement in social initiatives indicates a well-rounded character that could contribute to team dynamics.

He has Accounting Expeirnce but not Data Analytics, he can be given chance.
Communication ScoreRank:10

From this Complete Analysis, we can say that:

Candidate Number	Communication Score Rank	Comment
1	5	He can be given a chance.
2	2	He could have opportunities at finance firms if he articulates his skill set more clearly.
3	1	He can be accounting but Data Analytics is a Technical thing, but we can give a chance.
4	6	He has a high probability of being recruited due to his qualifications and relevant experience.
5	9	He may need to clarify how his skills align with the company's needs.
6	3	His combination of experience and education makes him a strong candidate for analytics roles.
7	4	He doesn't have skills for Data Science. But he is passionate.
8	7	He has required Skills and Work Experience, highly probable for recruitment.
9	8	Being an IIM Student even with no experience, he will be a good suit for the role.
10	10	He can be given a chance.

From the above table we can say that:

Candidates Recommended for Recruitment:

1. **Candidate 6**
    - **Rank**: 3
    - **Comment**: "His combination of experience and education makes him a strong candidate for analytics roles."
    - **Reason**: High rank and positive comment about relevant skills and experience.
2. **Candidate 8**
    - **Rank**: 7

    - **Comment**: “He has required skills and work experience, highly probable for recruitment.”
    - **Reason**: Despite a lower rank, the comment highlights strong skills and experience.
3. **Candidate 9**
    - **Rank**: 8
    - **Comment**: “Being an IIM student even with no experience, he will be a good suit for the role.”
    - **Reason**: Lower rank but strong educational background and potential.
4. **Candidate 4**
    - **Rank**: 6
    - **Comment**: “He has a high probability of being recruited due to his qualifications and relevant experience.”
    - **Reason**: Mid- level ranking but best Qualifications.
5. **Candidate 2**
    - **Rank**: 2
    - **Comment**: “He could have opportunities at finance firms if he articulates his skill set more clearly.”
    - **Reason**: High rank indicates strong overall performance. Needs to improve articulation but has potential.

Candidates Recommended for Rejection or Further Evaluation

1. **Candidate 3**
    - **Rank**: 1
    - **Comment**: “He can be accounting but Data Analytics is a technical thing, but we can give a chance.”
    - **Reason**: Top rank but the comment indicates a potential misalignment with the role.
2. **Candidate 10**
    - **Rank**: 10
    - **Comment**: “He can be given a chance.”
    - **Reason**: Lowest rank but the comment suggests potential, making him worth considering for further evaluation.

3. **Candidate 5**
    - **Rank**: 9
    - **Comment**: “He may need to clarify how his skills align with the company’s needs.”
    - **Reason**: Lower rank and comment indicating potential misalignment with the role.
4. **Candidate 7**
    - **Rank**: 4
    - **Comment**: “He doesn’t have skills for Data Science. But he is passionate.”
    - **Reason**: Mid-level rank but lacks necessary skills despite passion.
5. **Candidate 1**
    - **Rank**: 5
    - **Comment**: “He can be given a chance.”
    - **Reason**: No good Ranking and No skills

Improvements:

1. To enhance the parsing of data using regex in the Indian recruitment context, we can incorporate keywords relevant to candidate evaluations, such as "recruitment" and "candidates." This approach will improve the accuracy of data extraction from transcripts.
2. During our analysis, we opted not to use the **fillna** method when encountering samples with inconsistencies. Utilizing mean imputation or other filling

techniques could misrepresent emotional scores, which are sensitive and can fluctuate significantly. Therefore, we maintained the integrity of the original data without imputation.

3. For ranking candidates, we can implement weighted methods to emphasize specific criteria more than others. While our initial analysis utilized standard ranking methods, introducing weights allows for more tailored assessments based on particular requirements.
4. I have conducted a comprehensive analysis structured into four distinct scripts:
    1. **Transcript Analysis**
    2. **Gaze Analysis**
    3. **Text Analysis**
    4. **Emotional Analysis**
5. These scripts can be executed continuously to generate graphs and visualizations, facilitating a robust data analysis workflow. This methodology contributes to an end-to-end product capable of delivering insightful outcomes.

Furthermore, I have developed a holistic data analysis model designed specifically for these ten datasets. With minimal adjustments, this model can be broadly applied to various recruitment analyses. As mentioned, the text parsing mechanism presents opportunities for enhancement, which could significantly refine our analysis process.

Overall, the combination of regex improvements, careful data handling, and adaptable modelling positions us well for advanced candidate evaluation and recruitment analysis.