# I'm Beside You

## Documentation

### Topic:

# Introduction videos Evaluation

Prepared by: Adarsh Santoria
Institution: IIT Mandi

**Prompt1:** Passed the Question and Data Description and ordered to find the possible options to perform in the data?

## Response1:

<u>Transcript Data Analysis:</u>

**Sentiment Analysis:** Analyze sentiment scores (positive, negative) in transcripts to assess candidates' emotional tone.

**Communication Metrics:** Evaluate communication metrics (confidence, hesitance, conciseness, enthusiasm) to gauge communication quality.

**Speech Speed Analysis:** Examine speech speed to identify candidates who speak at an optimal pace.

**Textual Analysis:** Analyze transcript text to identify key topics or areas of expertise discussed by the candidates.

<u>Combined Analysis:</u>

**Correlation Analysis:** Explore correlations between emotion scores and transcript sentiment scores to understand the alignment of emotional expressiveness with content tone.

**Recommendation Generation:** Generate recommendations for each candidate's recruitment suitability, considering both emotional expressiveness and communication skills.

**Visualization:** Create visualizations (e.g., charts, plots) to effectively present insights.

## Rationale1:

These analysis options cover a wide range of factors that can influence recruitment decisions, including emotions, communication skills, and areas of expertise. By examining these aspects, we can gain a comprehensive understanding of the candidates and make informed recruitment choices. The integration of Emotion Data and Transcript Data allows for a holistic assessment, ensuring a balanced evaluation of candidates' qualifications and suitability for different roles. Visualizations enhance the presentation of findings, making them more accessible to stakeholders.

**Prompt2:** Passed all Transcripts and ordered to parse valuable data

## Response2:

Participant 1:

**Education:** Pursuing postgraduate and management from IIM Kozhikode, B.Tech in Biotechnology from Heritage Institute of Technology Kolkata, M.Tech from IIT Kharagpur.

**Work Experience:** Three years in regulatory affairs in the pharmaceutical industry, medical writer at Ciro Klein Farm, specialized in drug safety and risk management.

**Skills:** Regulatory affairs, medical writing, drug safety, risk management, biotechnology.

Participant 2:

**Education:** Completed BBA in 2022.

**Work Experience:** Interned in a boutique investment bank, interned with Kabadi Techno, focused on finance.

**Skills:** Investment banking, finance, financial modeling, startup growth.

Participant 3:

**Education:** School activity head, top 1.2% rank in undergraduate entrance exam, honours degree from Varanasi University.

**Work Experience:** Internship in a steel manufacturing firm, internship in an accounting firm.

**Skills:** Sales, accounting, event coordination.

Participant 4:

**Education:** Engineering graduate in electronics and communication.

**Work Experience:** Academic advisor in a school for 19 months.

**Skills:** Academic advising, electronics, communication.

Participant 5:

**Education:** Undergraduate in mass media with a specialization in advertising.

**Certifications:** Entrepreneurship course by Turnip, foundations of management by Google.

**Skills:** Advertising, entrepreneurship, content creation (drawing, painting, singing, writing movie reviews).

Participant 6:

**Education:** First-year MBA analytics student from IIM Kashipur, engineering graduate.

**Work Experience:** Three years at Deloitte, validation processes for pharmaceutical software.

**Skills:** Analytics, strategy, validation processes.

Participant 7:

**Education:** Undergraduation in earth science, worked at General Insurance Corporation of India.

**Skills:** Reinsurance, underwriting, verbal and written communication.

Participant 8:

**Education:** PGP finance student at IIM Co-Ecode, chartered accountant, CFA Level 1.

**Work Experience:** Over three years with PwC, internship with ITC Limited. Skills: Financial analysis, internal audit, edtech.

Participant 9:

**Education:** First-year MBA student at IIM Lucknow, B.Tech in Agriculture Engineering, M.Tech in Food Process Engineering.

**Entrepreneurship:** Co-founded an Agritech startup.

**Project Leadership:** Led a project on the application of remote sensing IoT and artificial intelligence in agriculture.

**Skills:** Agriculture, entrepreneurship, business development, strategy, remote sensing, IoT, AI.

Participant 10:

**Education:** Graduation in B.Com Honours. Work Experience: Interned as an Accounting Associate and Tax Associate.

**Leadership:** Captain of the students committee in school, class representative in college, part of Bad Scouts and Guide.
**Skills:** Accounting, taxation, leadership, social activities.

## Rationale2:

The response provides a structured summary of each participant's background, including education, work experience, skills, and any noteworthy achievements. This information is essential for evaluating candidates' qualifications and suitability for specific roles. The response is clear and concise, making it easy for stakeholders to review the key details of each participant.However, we didn't get comparable parameters to classify as the qualities of participants are different in all respects.

**Prompt3:** Please generate Python code to concatenate transcript and emotion data for each participant from '1.csv' to '10.csv' files located in 'transcript_data' and 'emotion_data' folders respectively. After merging the data, calculate the dominant emotion for each participant and save the resulting DataFrame to a CSV file named 'unfiltered_data.csv'. Please include comments to explain the code.

## Response3:

```
# Create an empty DataFrame to store the concatenated data
final_data = pd.DataFrame()

# Loop through participant files (assuming they are numbered from 1 to 10)
for participant_id in range(1, 11):
    # Define file paths for transcript and emotion data
    transcript_file_path = os.path.join('transcript_data',
f'{participant_id}.csv')
    emotion_file_path = os.path.join('emotion_data',
f'{participant_id}.csv')

    # Check if both transcript and emotion files exist
    if os.path.exists(transcript_file_path) and
os.path.exists(emotion_file_path):
        # Read transcript and emotion data for the participant
        transcript_data = pd.read_csv(transcript_file_path)
        emotion_data = pd.read_csv(emotion_file_path)
```

```
        # Concatenate transcript and emotion data
        merged_data = pd.concat([transcript_data, emotion_data], axis=1)

        # Calculate dominant emotion (assuming an 'emotion' column exists)
        dominant_emotion = merged_data['emotion'].mode().values[0]

        # Add a 'dominant_emotion' column to the merged data
        merged_data['dominant_emotion'] = dominant_emotion

        # Append the participant's data to the final DataFrame
        final_data = final_data.append(merged_data, ignore_index=True)

# Save the resulting DataFrame to a CSV file
final_data.to_csv('unfiltered_data.csv', index=False)

print("Data concatenation and dominant emotion calculation completed.")
```

## Modification3:

Data looks a little bigger than expected. We can match time durations of both characteristics in the dataset.

## Rationale3:

The goal of this prompt is to generate Python code that concatenates transcript and emotion data for each of the 10 participants and then combines all the data into one dataframe.

**Prompt4:** Ensure For each row in 'transcript_data,' filter rows in 'emotion_data' where 'elapsed_time' falls within the range defined by 'start' and 'end' columns. If there are matching rows, calculate the mean for numeric columns in the filtered 'emotion_data.

## Response4:

```
for _, row in transcript_data.iterrows():
        # Filter rows based on 'elapsed_time' between 'start' and 'end'
        start_time = row['start']
        end_time = row['end']
```

```
        time_condition = (emotion_data['elapsed_time'] >= start_time) &
(emotion_data['elapsed_time'] <= end_time)
        # Check if filtered_emotion_data is empty, and continue to the
next iteration if it is
        if filtered_emotion_data.empty:
            continue

        # Separate numeric and text columns
        numeric_columns =
filtered_emotion_data.select_dtypes(include=['number']).columns
        text_columns =
filtered_emotion_data.select_dtypes(include=['object']).columns

        # Initialize dictionaries for numeric and text data
        numeric_data_dict = {}
        text_data_dict = {}

        if not numeric_columns.empty:
            # Calculate the mean for numeric columns
            numeric_mean_values =
filtered_emotion_data[numeric_columns].mean()
```

## Rationale4:

The shortened code is concise, efficient, and task-focused. It filters data based on 'transcript_data' time ranges and calculates means when necessary, without extra complexity.

**Prompt5:** Generate Python code to calculate the dominant emotion as it got affected in the last step for each row in a DataFrame named 'unfiltered_data' based on a list of emotion characteristics. The list of emotion characteristics is provided as 'emotion_characteristics,' which includes 'angry,' 'disgust,' 'fear,' 'happy,' 'sad,' 'surprise,' and 'neutral_y' columns. The code should determine the column name (emotion) with the maximum value for each row and store this information in a new 'dominant_emotion' column in the DataFrame. Please include comments to explain the code.

## Response5:

```
# List of emotion characteristics
emotion_characteristics = ["angry", "disgust", "fear", "happy", "sad", "surprise", "neutral_y"]

# Calculate the column name (emotion) with the maximum value for each row (candidate)
unfiltered_data['dominant_emotion'] = unfiltered_data[emotion_characteristics].idxmax(axis=1)
```

## Rationale5:

The provided response effectively calculates the dominant emotion for each candidate in the 'unfiltered_data' DataFrame by identifying the emotion column with the highest value for each row, storing this information in a new 'dominant_emotion' column. This step is crucial for determining the primary emotional expression of each candidate, aiding further analysis and decision-making in candidate assessment.

**Prompt6:** How can we do data reduction here

## Response6:

1. You can drop columns with redundant or faulty information.
2. You can choose a subset of the most relevant columns (features) and remove the less important ones. This can help reduce the dimensionality of your dataset while retaining important information.
3. Apply PCA

## Rationale6:

These processes will be effective in solving but first we have to go through the data or make various plots.

**Prompt7:** Filter the 'unfiltered_data' dataframe by dropping columns that are consistently constant, unnecessary, or appear to be faulty. Specifically, remove the columns 'temperature' and 'distance' as they are always constant, and then drop columns 'avg_logprob,' 'compression_ratio,' 'seek,' 'upload_time,' 'id,' 'image_seq,' 'movie_id,' 'gaze,' and 'blink' due to their being deemed unnecessary or potentially faulty. Finally, print the resulting 'filtered_data' dataframe.

## Response7:

```
# Columns to drop
columns_to_drop = ['temperature', 'distance', 'avg_logprob',
'compression_ratio', 'seek', 'upload_time', 'id', 'image_seq', 'movie_id',
'gaze', 'blink']

# Remove the specified columns
filtered_data = unfiltered_data.drop(columns=columns_to_drop)
```

## Rationale7:

This prompt instructs the generation of code to filter and clean a dataframe named 'unfiltered_data.' The code drops specific columns that are consistently constant, deemed unnecessary, or potentially faulty. It then prints the resulting 'filtered_data' dataframe. The rationale is to efficiently preprocess the data by removing irrelevant or problematic columns, ensuring that only relevant and reliable data remains for analysis.

**Prompt8:** Generate Python code to perform exploratory data analysis on a DataFrame named 'filtered_data.' First, perform correlation analysis to visualize the correlation matrix as a heatmap with annotations to highlight relationships between columns. Use a suitable color map for the heatmap and provide a title. Next, calculate the covariance matrix row sum for numeric columns in the same DataFrame. Display the calculated covariance matrix row sum.

## Response8:

```
correlation_matrix = filtered_data.drop(columns=['participant_id']).corr()
plt.figure(figsize=(14, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
# Filter for Numeric Columns
numeric_data = filtered_data.select_dtypes(include='number')

# Calculate Covariance Matrix Row Sum
covariance_matrix = numeric_data.cov().sum()
```

**Rationale8:** Much correlated columns are dropped and this will increase the efficiency of future processes of analysis.

**Prompt9:** How can I perform sentiment analysis on a dataset using VADER in Python? Can you show me how to normalize a specific column in a pandas DataFrame?

**Response9:** To perform sentiment analysis using VADER, you can use the get_sentiment_vader function with the SentimentIntensityAnalyzer from the NLTK library. You can normalize a specific column in a pandas DataFrame by subtracting its minimum value and dividing by the range of values (maximum - minimum).

**Prompt10:** How can I perform sentiment analysis on a dataset using VADER in Python? Can you show me how to normalize a specific column in a pandas DataFrame? What is the purpose of the WeightedScorer class in this code? How do I load data from a CSV file into a pandas DataFrame?

**Response10:** To perform sentiment analysis using VADER in Python, use the SentimentIntensityAnalyzer from the NLTK library, as shown in the provided code's get_sentiment_vader function. To normalize a specific column in a pandas DataFrame, subtract its minimum value and divide by the range of values (maximum - minimum). The purpose of the WeightedScorer class in this code is to calculate a composite score by combining weighted attributes, allowing customization of the importance of different factors in the final score. You can load data from a CSV file into a pandas DataFrame using pd.read_csv('file.csv').

**Prompt11:** Explain the process of performing grid search for hyperparameter tuning. How do I create bar plots for multiple attributes in a single figure using Matplotlib?

**Response11:** Performing grid search for hyperparameter tuning involves trying different combinations of hyperparameters by specifying a grid of values. For each combination, a model is trained and evaluated using a specified scoring metric. The set of hyperparameters that yields the best score is chosen as the optimal configuration. To create bar plots for multiple attributes in a single figure using Matplotlib, iterate through the attributes and use plt.bar for each attribute, arranging the bars side by side or stacked depending on your preference. You can also customize the appearance and layout of the subplots as needed.

***A brief overview of other prompts used :***

- How can I create dummy variables for a categorical column in a DataFrame?
- What are the steps involved in grouping data by a specific column and calculating the mean of other columns in pandas?
- What libraries are used in this code, and how can I install them?
- How do I specify a custom scoring metric for grid search in scikit-learn?
- Can you explain the logic behind sorting candidates based on their final scores?
- What is the purpose of the transform method in the WeightedScorer class?
- How can I adjust the layout of subplots in a Matplotlib figure for better visualization?
- Explain the purpose of the perform_sentiment_analysis function in this code.
- What is the meaning of the agg method when used with groupby in pandas?
- How can I rename columns in a pandas DataFrame?
- What is the significance of the param_grid dictionary in the grid search process?