

Prompt file for Data Analysis Assignment.

Ayush Kumar Mishra

September 18, 2024

Dashboard

For seeing all the code live interactively,

Visit **Dashboard**

<https://eda-analysis-iby-0.streamlit.app/>

Contents

1	Introduction	3
2	Data Preparation and Integration	4
2.1	Extracting Key Features from Emotion Data from each CSV file and making single DATAFRAME	4
2.2	Integrating transcript data with emotion data	6
2.3	Now, Integrating the gaze_df with the final_df_with_emotions	8
3	Data Preprocessing	9
4	Prompts for Analysis on the DataFrame made above	10
4.1	Basic Statistics	10
4.2	Correlation Analysis	11
4.3	Based on the final_df, I asked for analysis on the data.	12
5	Scoring of the Students based on the data.	16
6	Conclusion	18
7	All the chatgpt links for the responses	19

1 Introduction

So, In this document, I will try to elaborate the process of doing EDA and finding the insights from the data given, using the chatgpt. Though I have worked hard on this Assignment, Chatpgt has really made the tiring or repeating tasks very easy.

So, I used chatgpt for the following tasks:

- Like i am stuck somewhere, i asked chatgpt to give a framework or methodology to do that.
- I asked chatgpt to give me the code for the task i wanted to do.
- Or if i have already written some code, I have to do it for n times, so i asked chatgpt to do it for me.
- I take help from chatgpt to make Streamlit Dashboard for the project which is live at [here](#).
- Also in making this Pdf, I asked chatgpt to give me the prompts and responses in the tcolorbox, so that it looks good.

All in all, Now i will show you the whole process of doing EDA along with the responses i get from the chatgpt. And if they were wrong how i corrected them or did further prompting.

2 Data Preparation and Integration

In this section, I explain how the data was prepared and important features were extracted from the three provided CSV files.

2.1 Extracting Key Features from Emotion Data from each CSV file and making single DATAFRAME

Initial Prompt:

Prompt

```
movie_id image_seq  angry disgust  fear  happy sad  surprise neutral
dominant_emotion
0  93663f94-bf0a-4ce8-a29a-a5236cc7fe6a  0  4.31735  0.000594  2.879790
  1.65035  2.779980  0.600814  87.77110 neutral
1  93663f94-bf0a-4ce8-a29a-a5236cc7fe6a  1  53.22530  2.981640  12.736800
  1.52347  1.051320  27.216800  1.26462  angry

this is data of csv file of every emotion data and this is student 1.. there
are 10
such students, so 10 such df_emotion, so i want to make
combined df where two most frequent emotion data throughout
the video of the student reflects All df_emotion csv files are emotion_data
folder.
```

Chatgpt gave wrong response to the initial prompt. The response was giving wrong results.
So i further explained what i intended to do

Refined Prompt:

You are doing it incorrectly. I want to first calculate the most frequent of every single emotion CSV and then make one column out of it.
Then, finally make 10 columns out of 10 CSV files with column names dominant_top1, dominant_top2 emotions. From every emotion CSV, find 2 max frequency dominant features and then add that.

For this prompt, response from the chatgpt was:

```
import pandas as pd
import os

# Path to the folder containing emotion data
folder_path = 'emotion_data'

# Initialize an empty dictionary to store top 2 dominant emotions for each student
dominant_emotions = {'student_top1': [], 'student_top2': []}

# Iterate over all files in the folder (assuming 10 files: emotion-df1.csv to emotion-df10.csv)
for i in range(1, 11):
    file_name = f'{i}/emotion.csv'
    file_path = os.path.join(folder_path, file_name)

    # Read the CSV file into a DataFrame
    df = pd.read_csv(file_path)

    # Find the 2 most frequent dominant emotions
    top_emotions = df['dominant_emotion'].value_counts().nlargest(2).index.tolist()

    # Append the top 2 emotions for this student (add NaN if there's less than 2 emotions)
    dominant_emotions['student_top1'].append(top_emotions[0] if len(top_emotions) > 0 else None)
    dominant_emotions['student_top2'].append(top_emotions[1] if len(top_emotions) > 1 else None)

# Create a DataFrame for top emotions for every student
df_dominantEmotion_for_everystudent = pd.DataFrame(dominant_emotions)

# Rename columns to reflect student numbers correctly (top1 and top2)
df_dominantEmotion_for_everystudent.columns = [
    f'student_{i}_dominant_emotion_top1' if 'top1' in col else
    f'student_{i}_dominant_emotion_top2'
    for i, col in enumerate(df_dominantEmotion_for_everystudent.columns, start=1)
]

# Set index to range from 1 to 10
df_dominantEmotion_for_everystudent.index = range(1, 11)

# Print the final DataFrame
print(df_dominantEmotion_for_everystudent)
```

With the help of this code, two most freq dominant emotions were extracted from each emotion CSV file and added to the final dataframe.

Now the final dataframe has 10 columns with features top_1_dominant, top_2_dominant emotion.

	dominant_emotion_top1	dominant_emotion_top2
1	neutral	fear
2	happy	neutral
3	neutral	fear
4	neutral	fear
5	neutral	neutral
6	neutral	neutral
7	fear	sad
8	neutral	fear
9	neutral	happy
10	sad	fear

2.2 Integrating transcript data with emotion data

Prompt:

'Prompt'

```
id      "neutral  confident  hesitant   concise  enthusiastic  speech_speed
0      0.267454   0.846701   0.845698   0.635805      0.647783      2.517986
1      0.260410   0.679283   0.733701   0.544145      0.417390      3.217822
```

this is data of csv file of every transcript data and this is student 1.. there are 10 such students, so 10 such df_transcript , so i want to make combined df where two most frequent text data throughout the video of the student reflects All df_transcript files are transcript_data folder.

The response from the chatgpt awesome this time.

```

# Path to the folder containing transcript data
folder_path = 'transcript_data'

# Initialize a list to store average DataFrames for each student
average_dfs = []

for f in os.listdir(folder_path):
    id = f.split('.')[0]
    file_path = os.path.join(folder_path, f)
    features = ['positive', 'negative', 'neutral', 'confident', 'hesitant',
                'concise', 'enthusiastic', 'speech_speed']
    df = pd.read_csv(file_path, usecols=features)

    # Calculate averages
    new_features = ['avg_positive', 'avg_negative', 'avg_neutral',
                    'avg_confident', 'avg_hesitant', 'avg_concise', 'avg_enthusiastic',
                    'avg_speech_speed']
    avg_values = df[features].mean().values

    # Create a new DataFrame with averages and id
    new_df = pd.DataFrame([avg_values], columns=new_features)
    new_df['id'] = id

    # Append to the list of average DataFrames
    average_dfs.append(new_df)

# Combine all average DataFrames into a single DataFrame
final_df = pd.concat(average_dfs, ignore_index=True)

print(final_df)

```

Using above code, It extracted the above features of transcript data of each student and added to the final dataframe.

	avg_positive	avg_negative	avg_neutral	avg_confident	avg_hesitant	avg_concise	avg_enthusiastic	avg_speech_speed
1	0.709199	0.141214	0.149586	0.733828	0.485172	0.429418	0.466497	3.113771
2	0.722006	0.107541	0.170453	0.684879	0.436158	0.484221	0.516685	3.269092
3	0.567257	0.264337	0.168406	0.573566	0.604004	0.394715	0.448050	3.385636
4	0.655748	0.169142	0.175110	0.621740	0.570452	0.403479	0.440626	2.775454
5	0.630573	0.187013	0.182414	0.590094	0.461488	0.413644	0.378110	2.817341
6	0.711182	0.138992	0.149826	0.679755	0.490252	0.367792	0.481433	2.583163
7	0.717354	0.140232	0.142414	0.703714	0.457070	0.398571	0.463940	2.284897
8	0.605402	0.192292	0.202306	0.555011	0.507622	0.352011	0.437399	2.902953
9	0.617353	0.223949	0.158699	0.591842	0.538732	0.381809	0.505152	3.329938
10	0.589267	0.220948	0.189785	0.619852	0.520637	0.385655	0.325507	3.248518

Note

Now we have to merge the transcript data with the emotion data.
For that we have to merge the dataframes on the basis of student id.

Prompt for merging the emotion_df and transcript_df:

```

[title=Prompt]
df_dominantEmotion_for_everystudent

```


and final_df add these df on index

response from the chatgpt:

```
final_df_with_emotions = pd.concat([final_df,
    df_dominantEmotion_for_everystudent], axis=1)

# Print the final merged DataFrame
print(final_df_with_emotions.head(10))
```

Using the above code, the emotion data and transcript data were merged on the basis of student id. The final dataframe after merging the emotion and transcript data is shown below:

Final Dataframe

id	avg_positive	avg_negative	avg_neutral	avg_confident
1	0.709199	0.141214	0.149586	0.733828
2	0.722006	0.107541	0.170453	0.684879
avg_hesitant	avg_concise	avg_enthusiastic	avg_speech_speed	
0.485172	0.429418	0.466497	3.113771	
0.436158	0.484221	0.516685	3.269092	
dominant_emotion_top1		dominant_emotion_top2		
neutral		fear		
happy		neutral		

2.3 Now, Integrating the gaze_df with the final_df_with_emotions

But before that i have to extract the key features from all 10 gaze data and make a single gaze_df. Prompt for this:

```
i have gaze_df of 10 students
in each csv of student there are
  movie_id image_seq gaze blink eye_offset
0  93663f94-bf0a-4ce8-a29a-a5236cc7fe6a   1   1   0   6.2253
1  93663f94-bf0a-4ce8-a29a-a5236cc7fe6a   2   1   0  22.7274

data like this
now i want to make merged_gaze_df of all 10 students how can i do this?
```

Response:

I am just showing the code here. The whole response is [here](#)

```
# Read all 10 gaze CSV files and store them in a list
folder_path = 'emotion_data'
dfs= []
for i in range(1, 11):
    file_name = f'{i}/gaze.csv'
    file_path = os.path.join(folder_path, file_name)

    df = pd.read_csv(file_path)
    aggregated_df = df.groupby('movie_id').agg({
        'gaze': 'mean',          # Average gaze data over all frames
        'blink': 'sum',          # Total blinks across all frames
        'eye_offset': ['mean',   # Average eye_offset over the entire video
                      'max',     # Maximum eye_offset
                      'min',     # Minimum eye_offset
                      'std'],    # Standard deviation of eye_offset
        'image_seq': 'count'    # Count the number of frames
    }).reset_index()
    aggregated_df.columns = ['_'.join(col).strip() if isinstance(col, tuple)
                             else col for col in aggregated_df.columns]
    dfs.append(aggregated_df)
    aggregated_df.head(2)

# print(dfs)

final_gaze_df = pd.concat(dfs)
```

3 Data Preprocessing

Prompt: "I have to check whether there are any missing values in the final DataFrame or not and also check the basic statistics of the final DataFrame."

```
import os

folder_path = 'emotion_data'
# if there is any data missing print the missing data
def check_dataframe(df):
    missing_values = df.isnull().sum()
    print("Missing values in each column:\n", missing_values)

    basic_statistics = df.describe()
    print("\nBasic statistics of the DataFrame:\n", basic_statistics)

for i in range(1, 11):
    file_name = f'{i}/emotion.csv'
    file_path = os.path.join(folder_path, file_name)

    if os.path.exists(file_path):
        df = pd.read_csv(file_path)
        print(f"Data from {file_path}:")
        check_dataframe(df)
    else:
        print(f"File {file_path} does not exist.")
```

from the results, i got following observations from the running this code:-

- As the Count of all the features is 10, means there is not Null values.
- The standard deviation of all the features is also very low compared to the mean, indicating that the data is not spread out (low chances of outliers).
- As i have already done the encoding of the categorical features, so no need to do it again.

4 Prompts for Analysis on the DataFrame made above

4.1 Basic Statistics

Prompt:

How to know whether there are **any** missing values **in** the final DataFrame?
or just give me description of the dataframe.

Response:

```
print(final_df_with_emotions.describe())
```

The basic statistics of the final DataFrame are as follows:

	avg_positive	avg_negative	avg_neutral	avg_confident	avg_hesitant	avg_concise	avg_enthusiastic	avg_speech_speed
count	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000
mean	0.652534	0.178566	0.168900	0.635428	0.507159	0.401131	0.446340	2.971076
std	0.058600	0.048346	0.019202	0.060950	0.052672	0.036592	0.057471	0.361271
min	0.567257	0.107541	0.142414	0.555011	0.436158	0.352011	0.325507	2.284897
25%	0.608390	0.140478	0.152044	0.590531	0.467409	0.382770	0.438206	2.785926
50%	0.643161	0.178077	0.169430	0.620796	0.498937	0.396643	0.455995	3.008362
75%	0.710686	0.213784	0.180588	0.683598	0.534208	0.411103	0.477699	3.263949
max	0.722006	0.264337	0.202306	0.733828	0.604004	0.484221	0.516685	3.385636

4.2 Correlation Analysis

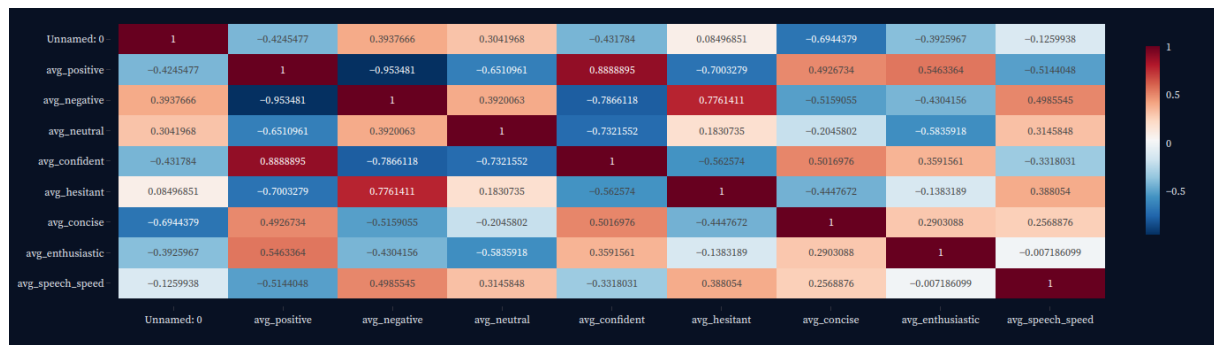
Now i did correlation analysis on the final DataFrame to understand the relationship between different features. **Prompt:**

Now, calculate the correlation matrix of the final DataFrame final_df.

Response:

```
correlation_matrix = final_df_with_emotions.corr()  
print(correlation_matrix)
```

Wrtitng and running the code, there was a error in chatgpt response, but I corrected it and ran the code. The correlation matrix is shown below:



4.3 Based on the final_df, I asked for analysis on the data.

Prompt:

now probelm statement **is**

In this task, you will work with a dataset of 10 candidates containing emotion scores, transcript scores, **and** the corresponding transcripts extracted **from** their introduction videos. Your goal **is** to use the ChatGPT with your prompt engineering skills along with performing Exploratory Data Analysis (EDA) on the data provided, to generate valuable **and** actionable insights **from** the data.

Actionable insights include:

1. Can we recruit the candidate **or not**? With suitable reasons **from** the data.
2. Analysis of communication skills **and** finding areas of expertise based on data.
3. Any other insights which **help** us make decision about the candidate are also welcomed.

This task **is** designed to assess your ability to preprocess data, create effective prompts, **and** perform EDA to extract meaningful **and** actionable information.

Main flow should be like this:

- give plots idea to perform some exploratory data analysis (EDA) on the datasets.
- how to analysis to generate insights about each candidate.
- recommendations based on those insights.
- Finally give a scoring mechanism to rank the students
- Any other insights **is** welcomed.

Response:

As the result of this response is very large, i am not pasting it here. But the link for the response is [here](#) But the crux of the response chatgpt is :

1. EDA Plots: Box plots, scatter plots, word clouds, heatmaps, and bar charts to visualize key metrics.
2. Analysis: Emotional stability, communication skills, and expertise areas via correlation analysis, sentiment analysis, and text-based NLP.
3. Recommendations: Tailored based on emotional and communication performance.
4. Scoring Mechanism: Combine emotional stability, transcript quality, and communication clarity for ranking.
5. Any Additional insights

So based on the response of the chatgpt, i started implementing each section one by one.

Note

In the responses, I am just showing the text response of the chatgpt. The code is not pasted here. I will explicitly include the code in different file or if the link for the chatgpt is mentioned, then no issues. If the response will be too long , i will just give what i understood from the response. and will paste the link for the response.

- Communication Scoring: **Prompts for this section:**

I want to plot **and** find the relationship between following features, give with code **and** logic :-

- a) relationship between conciseness **and** enthusiasm of the students.
 - b) relationship between avg_speech_speed **and** avg_hesitant **and** between avg_speech_speed **and** avg_confident.
 - c) relationship between **all** three text scores **and** confidence of the students.
 - d) Create a word cloud **from** the transcript to identify frequently used words **and** phrases.
-

In the response of this prompt, chatpgt gave codes for relations i wanted to find. I am not pasting the code here. Link for the chatgpt response is [here](#)

- Emotional State and Body Language Analysis: **Prompts for this section:**

Prompt

```
so in the data frame , i have eye_offset data of each frame of video of
each student(10 total)
which can be postive or negative .
Answer this:
How to use eye_offset for whole video for every student to find how much
his eyes deviates??
To determine the emotional Stability --how to achieve this

To determine how the emotions are changing--means what is the variation
over the video

To Body Language Analysis on the basis of columns of gaze_df which are
blink_sum, eye_offset, gaze_score.

give a Framework to do all the above tasks
```

Response:

Response

To analyze the eye offset data and determine emotional stability, changes in emotions, and body language, you can follow this framework:

1. Emotional Stability Analysis

Objective: Determine how much a student's eyes deviate throughout the video to gauge emotional stability.

Calculate Mean and Standard Deviation:

Compute the mean and standard deviation of the eye_offset for each student over the entire video.

Stability Metric: Lower standard deviation indicates more stability; higher indicates more fluctuation.

Cumulative Deviation:

Calculate the cumulative deviation over time to assess how the deviation accumulates.

Visualization:

Plot the eye_offset values over time to visualize stability.

Use rolling windows to smooth the data and highlight trends.

3. Emotional Variation Analysis

Objective: Analyze how emotions change over time within the video.

Calculate Moving Average:

Compute the moving average of the eye_offset to observe trends and variations.

Variation Metrics:

Measure the variance or standard deviation over time windows to capture emotional changes.

Visualization:

Create time series plots of eye_offset and its moving average.

Use heatmaps to show variation intensity over time.

4. Body Language Analysis

Objective: Analyze body language based on blink_sum, eye_offset, and gaze_score.

Correlate Metrics:

Compute correlations between eye_offset, blink_sum, and gaze_score.

Analyze how these metrics interact to provide insights into body language.

Blink Frequency:

Analyze the blink_sum to determine blinking patterns and their relation to emotional states.

Gaze Score Analysis:

Evaluate gaze_score to understand gaze direction and how it correlates with eye_offset and blinking.

Body Language Patterns:

Combine all metrics to identify patterns such as increased blinking with large eye_offsets or shifts in gaze.

Visualization:

Plot eye_offset, blink_sum, and gaze_score together to observe how they change over time.

Use scatter plots to explore correlations between metrics.

Further Prompts for this section:

prompt

what is eye-offset can u explain??

Response:

response

Eye offset **is** the distance between the center of the screen **and** the point where the user **is** looking.

It **is** used to measure the stability of the user's gaze.

A higher eye offset indicates that the user's gaze is less stable.

Prompts for analysing gaze_df:

No, i want U to give Eye Offset Standard Deviation analysis **for** every student on the basis of

1. Too much Eye Movements:
2. Moderate Eye Movements:
3. Stable Eye Movements:

5 Scoring of the Students based on the data.

Prompt:

So i had already some latex code, on that i asked chatgpt to give me the scoring mechanism for the students based on the data. So my prompt was like this:

response

based on the data, do the changes `in` the latex also provide insights acc to the data
`and` comparison with other students (ranking `in` every feature), `and` use `tcolorbox` `for` heading only.

Response: The response of the chatgpt was NOTvery good. It gave me the scoring mechanism just on the basis of few features.

Refined Prompt:

Refined Prompt

Analysis the data deeply `and` on the basis of `all` the features of each student, give a scoring mechanism such that

- 1) It `is` robust
- 2) divides the student into highly fit `for` the role
- 3) Okk `for` the role
- 4) Not `for` the role
- 5) Main focus should be on transcript data(postive,negative,neutral) , emotion thorughtout the video, confidence, hesitance, speech,speed, body language, enthusiam

Response: This time, response was amazing, it gave a very nice framework to calculate and rank the students based on the data.

Though values used for the scoring were random, but the framework was very good.

Response

I will assign a weighted score to each feature, where each feature's contribution to the final score is proportional to its importance for the role. The focus will be primarily on the transcript data, confidence, enthusiasm, and body language. Heres how it can be structured:

a. Transcript Data (40% of total score):

Positive Transcript: +10 points for values 0.7, scaled down for lower values.

Negative Transcript: -10 points for values 0.25, scaled down for lower values.

Neutral Transcript: No direct impact, but consider it when evaluating other factors like enthusiasm.

b. Confidence (20% of total score):

+10 points for values 0.7, scaled down for lower values.

c. Hesitance (10% of total score):

-10 points for values 0.5, scaled down for lower values.

d. Enthusiasm (15% of total score):

+10 points for values 0.5, scaled down for lower values.

e. Speech Speed (5% of total score):

For values around 3.0, full score (e.g., 0.5 around the mean). Deviations from the optimal speed reduce the score.

f. Body Language (10% of total score):

Gaze Score: +5 points for scores 0.8, scaled down for lower scores.

Blink Sum: -5 points for blinks 20, scaled up for fewer blinks.

Eye Offset: Penalties for extreme offsets (high deviation in gaze).

3. Score Calculation

Using this weighted framework, we will calculate the score for each student. The total score will then classify them into:

Highly Fit for the Role: Score 75

Okay for the Role: Score between 55 and 74

Not Fit for the Role: Score < 55

This was the original response, here values used for the scoring and weightage were not good. So i changed them.

To see complete prompts and responses(including wrong one), click [here](#)

6 Conclusion

This section summarizes the key findings and insights from the data analysis project.

Prompt:

Prompt

Based on the analysis of the data, provide a conclusion **for** the project.

Include key insights, recommendations, **and any** additional observations.

Response:

Response

Based on the analysis of the data, the following conclusions can be drawn:

1. Students with high positive sentiment, confidence, **and** enthusiasm are more likely to be successful **in** the role.
2. Communication skills, including concise speech **and** moderate speech speed, are important **for** effective communication.
3. Emotional stability **and** body language play a significant role **in** conveying confidence **and** enthusiasm.
4. Students with stable eye movements **and** low eye offset are more likely to maintain focus **and** attention.
5. Recommendations: Provide training **in** emotional intelligence, communication skills, **and** body language to improve student performance.
6. Additional observations: The scoring mechanism can be further refined based on additional data **and** feedback **from** experts.

7 All the chatgpt links for the responses

- [prompt for gaze data](#)
- [prompt for analysis of the data](#)
- [prompt for communication scoring](#)
- [prompt for scoring mechanism](#)

ありがとうございました!

Thank You!