Prompt Engineering Documentation

Saurabh Kumar Sah

21110188

IIT Gandhinagar

Task: To generate valuable and actionable insights from the dataset of 10 candidates containing emotion scores, transcript scores, and the corresponding transcripts extracted from their introduction videos.

27/09/2023

Finding a Roadmap for analysis:

Prompt Given to Chat GPT:

I have dataset of 10 candidates containing emotional score, transcript score, and the corresponding transcripts extracted from their introduction videos. how can I generate actionable insights from the data using EDA

Response received from Chat GPT:

Exploratory Data Analysis (EDA) is crucial in understanding your dataset and extracting actionable insights from it. To analyze your dataset of 10 candidates with emotional scores, transcript scores, and corresponding transcripts from their introduction videos, you can follow these steps for EDA:

1. **Data Cleaning**:
   - Start by loading and inspecting your dataset. Check for any missing values, duplicate records, or outliers. Clean the data as needed.

2. **Summary Statistics**:
   - Calculate basic summary statistics for your numerical variables (emotional scores and transcript scores). This includes mean, median, standard deviation, and quartiles. This will give you an initial sense of the data's distribution.

3. **Data Visualization**:
   - Create various visualizations to better understand the data distribution and relationships. Some useful plots include:
     - Histograms or density plots for the distribution of emotional scores and transcript scores.
     - Box plots to identify outliers in the scores.
     - Scatter plots to explore relationships between emotional scores and transcript scores.

4. **Correlation Analysis**:
   - Calculate the correlation between emotional scores and transcript scores. This will help you understand if there's a relationship between these two variables.

5. **Text Analysis**:
   - Perform text analysis on the transcripts. You can:
     - Calculate the average word count for each transcript.

- Perform sentiment analysis on the text to understand the overall sentiment in the introductions.
- Use natural language processing (NLP) techniques to extract keywords or topics from the transcripts.

6. **Segmentation**:
- Consider segmenting your candidates into groups based on their emotional and transcript scores. This can help you identify patterns or differences between high and low performers.

7. **Time Series Analysis** (if applicable):
- If timestamps are associated with the data, you can perform time series analysis to see if there are trends or patterns over time.

8. **Hypothesis Testing** (if applicable):
- If you have specific hypotheses about your data, you can conduct hypothesis tests to validate or reject these hypotheses. For example, you could test if there's a significant difference in emotional scores between candidates from different backgrounds.

9. **Machine Learning Models** (if needed):
- Depending on your goals, you can build predictive models to predict outcomes or identify important features contributing to success in this context. For example, you might build a model to predict candidate performance based on emotional and transcript scores.

10. **Interpretation and Actionable Insights**:
- Based on your EDA findings, interpret the results and extract actionable insights. For example, candidates with higher emotional scores perform better, or you might identify common themes in the transcripts associated with successful candidates. Use these insights to inform decision-making or further analysis.

Remember that EDA is an iterative process, and exploring your data thoroughly is essential to gain a deep understanding of it. The insights you generate can guide future actions, such as refining recruitment processes, providing targeted training, or making informed decisions about candidate selection.

Finding relation among different parameters

Prompt given to chat GPT

avg_logprob ,compression_ratio, no_speech_prob, positive, negative, neutral, confident, hesitant, concise, enthusiastic, speech_speed, i got above data from video analysis of self-introductory video of 10 candidates for the role. I need to analyse data to find the candidate best fits the role. can you help me with this?

Response recived

Certainly, you can analyse the provided data to identify the best-fit candidate for the role. To do this, you can follow these steps:

1. **Data Preparation**:
   - Ensure that your dataset is clean and well-organized. Check for any missing values, outliers, or data inconsistencies.

2. **Define Criteria**:
   - Determine the criteria that make a candidate the best fit for the data scientist role. These criteria could include a combination of factors such as high positive sentiment, confidence, concise speech, and a moderate speech speed.

3. **Weighted Scoring**:
   - Assign weights to each of the criteria based on their importance. For example, you might decide that positive sentiment and confidence are more critical, so they get higher weights.

4. **Score Calculation**:
   - Calculate a score for each candidate by multiplying their performance on each criterion by the corresponding weight and summing up the scores. For example, if "Candidate A" has a positive sentiment score of 0.8 and confidence score of 0.7, and you've assigned weights of 0.4 to sentiment and 0.6 to confidence, the total score for "Candidate A" would be (0.8 * 0.4) + (0.7 * 0.6).

5. **Ranking**:
   - Rank the candidates based on their total scores. The candidate with the highest score will be considered the best fit according to your defined criteria.

6. **Visualization**:
   - Create visualizations such as bar charts or radar plots to compare the

candidates' scores on different criteria. This can provide a clear overview of their strengths and weaknesses.

7. **Sensitivity Analysis**:
   - Perform sensitivity analysis by changing the weights assigned to criteria to see how it affects the rankings. This helps you understand how robust your selection process is to changes in criteria importance.

8. **Additional Insights**:
   - Consider other insights that might not be included in your scoring system, such as the content of their speech or specific domain knowledge relevant to the data scientist role.

9. **Final Decision**:
   - Based on your analysis, make a decision on the best-fit candidate for the data scientist role. Remember that your choice should align with the organization's needs and culture.

10. **Documentation**:
    - Document your analysis process and the rationale behind your choice to justify your decision-making.

Remember that this is a simplified approach, and the actual selection process for a data scientist role may involve more complex factors, such as technical skills, experience, and interviews. This analysis can serve as an initial screening or supplementary information to help inform your decision.

The same is applicable to other data.

More detail is needed for assigning weights:

Weighted mean

Prompts given:

Elaborate weighted Scoring

Response received:

Let's elaborate on the first candidate scoring method: the weighted scoring system.

Weighted Scoring System:

The weighted scoring system is a method for evaluating candidates based on predefined qualifications, attributes, or skills relevant to a specific role. This approach allows you to assign different levels of importance or priority (weights) to each qualification or attribute. Here's a step-by-step explanation of how the weighted scoring system works:

1. **Define Qualifications and Attributes:**
   - Begin by defining the qualifications, attributes, or skills that are important for success in the role. These can include both technical qualifications (e.g., programming languages, certifications) and soft skills (e.g., communication, teamwork).

2. **Assign Weights:**
   - Assign a weight or importance factor to each qualification or attribute. Weights are typically represented as percentages or on a scale of 1 to 10. The sum of all weights should equal 100% (or 1) to ensure a meaningful overall score.

3. **Assessment or Evaluation:**
   - Assess or evaluate each candidate based on each qualification or attribute. For example, if one of the qualifications is "Python programming skills," you might rate candidates on a scale of 1 to 5, with 1 being the lowest proficiency and 5 being the highest.

4. **Calculate Subscores:**
   - Calculate a subscore for each qualification or attribute by multiplying the candidate's rating by the assigned weight. For instance, if a candidate's Python programming skills are rated as 4 out of 5, and the weight for this qualification is 20%, the subscore would be 0.8 (4 * 20%).

5. **Aggregate Subscores:**
   - Sum up all the subscores to calculate a total subscore for each candidate. This total subscore represents the candidate's overall performance across all qualifications and attributes.

6. **Final Candidate Score:**
   - The total subscore becomes the candidate's final score. You can use this score to rank candidates in descending order, with the highest-scoring candidate considered the best-fit candidate for the role.

Here's an example to illustrate how the weighted scoring system works:

Let's say you have three qualifications for a data scientist role with the following weights:

- Qualification 1 (Python proficiency): Weight = 30%
- Qualification 2 (Machine learning knowledge): Weight = 40%
- Qualification 3 (Communication skills): Weight = 30%

Candidate A:
- Python proficiency: 4/5
- Machine learning knowledge: 3/5
- Communication skills: 5/5

Candidate B:
- Python proficiency: 5/5
- Machine learning knowledge: 4/5
- Communication skills: 3/5

Calculations:
- Candidate A's subscores: (4 * 0.30), (3 * 0.40), (5 * 0.30) = 1.2 + 1.2 + 1.5 = 3.9
- Candidate B's subscores: (5 * 0.30), (4 * 0.40), (3 * 0.30) = 1.5 + 1.6 + 0.9 = 4.0

Candidate B has a higher final score (4.0) compared to Candidate A (3.9), indicating that Candidate B is the preferred choice based on the weighted scoring system.

This method allows you to prioritize qualifications and attributes according to their significance for the role and make more informed hiring decisions.

Weight for emotion and transcript

Promt Given

If I have 2 data sets of student interviews, 1 data set is about their emotions, and the other is about their transcript, so what weight should I give each while giving rank to students

Response Received

Assigning exact weights without a specific context or understanding of your objectives is challenging. However, I can provide you with a general guideline, but please keep in mind that the choice of weights should depend on the relevance and importance of each dataset to your specific use case.

Let's assume that you want to rank students based on their overall performance in interviews, considering both their emotional state and the content of their transcripts. Here's a potential starting point for assigning weights:

1. **Emotions Dataset Weight**: 40%
2. **Transcript Dataset Weight**: 60%

In this example, you assign more weight to the transcript dataset, indicating that you consider the content of the interviews to be slightly more important in determining the students' rankings. The emotions dataset still plays a significant role, contributing 40% to the overall ranking.

However, please note that these weights are arbitrary and should be adjusted based on your specific research objectives, domain knowledge, and the characteristics of your datasets. It's also important to mention that when dealing with data analysis or machine learning, you can use techniques like feature importance analysis to determine the optimal weights based on your data's actual predictive power.

Consider this initial weight distribution as a starting point, and be prepared to adjust it as you gain more insights into how each dataset contributes to the rankings and meets your project's goals.

Eye offset

Prompt Given:

I want to take the average of eye_offset column from a CSV file and put the average in another Excel file. how can I do it. I need to take an average from 10 different cSv files and put it in a single Excel file

Response Received:

```
import pandas as pd
import os

# Create a list of file paths
csv_file_paths = [
    "/path/to/file1.csv",
    "/path/to/file2.csv",
    "/path/to/file3.csv",
    # Add more file paths as needed
]

# Create a list to store the average values
average_values = []

# Loop through the file paths
for file_path in csv_file_paths:
    if os.path.isfile(file_path):  # Check if the file exists
        # Extract the file name from the path
        filename = os.path.basename(file_path)
        df = pd.read_csv(file_path)
        average_eye_offset = df['eye_offset'].mean()
        average_values.append({'File': filename, 'Average Eye Offset':
average_eye_offset})

# Create a DataFrame from the list of average values
average_df = pd.DataFrame(average_values)

# Create a writer object for Excel
excel_writer = pd.ExcelWriter('average_eye_offset.xlsx', engine='openpyxl')

# Write the DataFrame to the Excel file
average_df.to_excel(excel_writer, sheet_name='Averages', index=False)

# Save the Excel file
excel_writer.save()
```

Plotting Bar Graphs

Prompt Given

I have 7 parameters in 7 columns in an Excel file. the 8th column contains the name of the candidate. I want to plot a bar graph with parameters as filters, and the name of the candidate at the axis

Response received

```
pip install pandas matplotlib

import pandas as pd

# Load data from the Excel file
df = pd.read_excel("data.xlsx")

import matplotlib.pyplot as plt

# Define the parameters you want to use as filters (adjust as needed)
parameters = ["Parameter1", "Parameter2", "Parameter3", "Parameter4",
"Parameter5", "Parameter6", "Parameter7"]

# Loop through each parameter and create a bar graph
for parameter in parameters:
    plt.figure(figsize=(10, 6))  # Adjust the figure size as needed
    plt.bar(df["CandidateName"], df[parameter])
    plt.xlabel("Candidate Name")
    plt.ylabel(parameter)
    plt.title(f"Bar Graph for {parameter}")
    plt.xticks(rotation=45)  # Rotate x-axis labels for readability
    plt.tight_layout()

    # Save or display the graph
    plt.savefig(f"{parameter}_bar_graph.png")  # Save as an image file
    # plt.show()  # Uncomment to display the graph interactively

# If you want to show or save all the parameter graphs together, you can do that
outside the loop.
# plt.show()
```

Plotting the graphs

I have an Excel file, and the Excel file has 11 worksheets. I want to draw the bar graph of column Candidate and weighted_mean in the 11th worksheet. I can i do it using pandas in jupyter notebook

Response Recived

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the Excel file into a DataFrame
excel_file_path = "your_excel_file.xlsx"
sheet_index = 10  # Index of the 11th worksheet
df = pd.read_excel(excel_file_path, sheet_name=sheet_index)

# Select the 'Candidate' and 'weighted_mean' columns
candidate_column = df['Candidate']
weighted_mean_column = df['weighted_mean']

# Plot a bar graph
plt.figure(figsize=(10, 6))
plt.bar(candidate_column, weighted_mean_column)
plt.xlabel('Candidate')
plt.ylabel('Weighted Mean')
plt.title('Bar Graph of Candidate vs. Weighted Mean')
plt.xticks(rotation=45)  # Rotate x-axis labels if needed
plt.tight_layout()
plt.show()
```

Error encountered

unhashable type: 'dict' got this error

Respopnse Recives

The error "unhashable type: 'dict'" typically occurs when you try to use a dictionary as a key in another dictionary or a set. Dictionaries in Python are mutable, which means they cannot be used as keys in dictionaries or elements in sets because they are not hashable.

If you are encountering this error, it's likely that you are trying to use a dictionary as a key or an element somewhere in your code. To resolve this issue, you should avoid using dictionaries as keys in other dictionaries or elements in sets.

If you can provide more context or specific code where you're encountering this error, I can give you more targeted advice on how to resolve it.

df = pd.read_excel("C:/Users/Saurabh/OneDrive - iitgn.ac.in/Desktop/Imbesideu/EDA/emo_avg.xlsx",sheet_name=weighted_means)
here is the code

Thank you for providing the code. It seems like you are trying to read an Excel file using the sheet_name parameter with a variable weighted_means as the sheet name. However, if weighted_means is a dictionary and you are encountering the "unhashable type: 'dict'" error, you need to use a string as the sheet name, not a dictionary.

You should replace sheet_name=weighted_means with the actual name of the sheet you want to read. For example, if the sheet you want to read is named "Sheet1", you would do:

df = pd.read_excel("C:/Users/Saurabh/OneDrive - iitgn.ac.in/Desktop/Imbesideu/EDA/emo_avg.xlsx", sheet_name="Sheet1")

Replace "Sheet1" with the name of the sheet containing the data you want to work with. Make sure that the sheet name you provide is a string representing the exact name of the sheet in your Excel file.

Resolving error

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Create arrays for the data
Candidate = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

positive = np.array([
    0.684101781, 0.695265024, 0.557790242, 0.634166351, 0.60772926,
    0.685571328, 0.653518048, 0.584838561, 0.586453162, 0.569694695
])

negative = np.array([
    0.141900392, 0.106407051, 0.263896749, 0.167150718, 0.186369559,
    0.137888774, 0.120508928, 0.189059843, 0.224078848, 0.219818821
])
```

```
neutral = np.array([
    0.146155848, 0.167180087, 0.166521718, 0.171291272, 0.177025141,
    0.147613373, 0.124911856, 0.198051704, 0.152902784, 0.184322724
])

confident = np.array([
    0.707553093, 0.663680024, 0.565170524, 0.603627645, 0.572825236,
    0.656166194, 0.64698879, 0.538926368, 0.561752932, 0.600519032
])

hesitant = np.array([
    0.473953098, 0.43086725, 0.591700605, 0.553138397, 0.450532858,
    0.477232031, 0.407969638, 0.494785229, 0.522020789, 0.505677353
])

concise = np.array([
    0.421579242, 0.472924489, 0.389321928, 0.392478462, 0.404166613,
    0.359994479, 0.356440373, 0.344108591, 0.370964813, 0.377949231
])

enthusiastic = np.array([
    0.457486681, 0.506702429, 0.443517718, 0.431983846, 0.371447664,
    0.473094, 0.412926229, 0.430877007, 0.495289536, 0.316485744
])

speech_speed = np.array([
    2.988080205, 3.134381367, 3.301715066, 2.661664803, 2.706196778,
    2.483091674, 2.10128826, 2.777144196, 3.084927579, 3.105925746
])

# Create a DataFrame
data = {
    'Candidate': Candidate,
    'positive': positive,
    'negative': negative,
    'neutral': neutral,
    'confident': confident,
    'hesitant': hesitant,
    'concise': concise,
    'enthusiastic': enthusiastic,
    'speech_speed': speech_speed
}

df = pd.DataFrame(data)
```

```
# Set the 'Candidate' column as the index
df.set_index('Candidate', inplace=True)

# Create line graphs for selected columns
selected_columns = ['positive', 'negative', 'neutral', 'confident', 'hesitant', 'concise',
'enthusiastic', 'speech_speed']

for column_name in selected_columns:
    plt.figure(figsize=(8, 6))
    plt.plot(df.index, df[column_name].values, marker='o', label=column_name)  # Use
.values to access NumPy array
    plt.xlabel('Candidate')
    plt.ylabel(column_name)
    plt.legend()
    plt.title(f'Line Graph for {column_name}')
    plt.show()
```

i am getting this error in the above code
Multi-dimensional indexing (e.g. `obj[:, None]`) is no longer supported. Convert to a numpy array before indexing instead.

Solution given:

The error message you are seeing is due to the fact that multi-dimensional indexing is no longer supported in the version of `pandas` and `matplotlib` you are using. You can resolve this issue by converting the object to a NumPy array via `pandas.DataFrame.to_numpy()` before indexing instead. Here's the modified code:

```
# Create line graphs for selected columns
selected_columns = ['disgust', 'fear', 'happy', 'sad', 'surprise', 'neutral']

for column_name in selected_columns:
    plt.figure(figsize=(8, 6))
    plt.plot(df.index.to_numpy(), df[column_name].to_numpy(), marker='o',
label=column_name)
    plt.xlabel('Candidate')
    plt.ylabel(column_name)
    plt.legend()
    plt.title(f'Line Graph for {column_name}')
    plt.show()
```

This should resolve the error you are seeing and allow you to plot the line graphs successfully.