Discussion Notes

Train model

- We first split our data into training and holdout set where holdout set holding only the rows of the FailingMethods of 'HIT01_8', 'HIT02_24'.
- We first found a tree based classifier which was RandomForestClassifier fit our model on the training set
- We used TTR (which is used to calculate the lexical diversity on a piece of text) as an extra feature on our dataset as complexity column. To calculate it we have used the following formula:
    - (count of the unique words in the **Answer.explanation**) / count of all words in the (**Answer.explanation**)
- We have also added **Answer.size** as our feature as well which involved the word count of **Answer.explanation**

Consolidate Explanations:

First of all, we only extracted FailingMethods and Answer.explanation columns from the original dataset to feed it to LLM (in our case ChatGPT 4o). Then we have given our exported csv file to ChatGPT with 3 different prompting methods.

1. **Our own crafted prompt**
   - We have crafted a specific prompt for the which contained directly what is asked from the assignment sheet task as follows:

> ```
> I have attached a file which involves an inspection task for bug reports with explanations of the answers that are given by programmers. I want you to generate a single answer explanation for each bug report grouped in a way that minimizes redundant information while keeping the information that would be necessary for someone else to fix the bug. You should preserve the types of information that explains: - how the program works - how the failure is happening - and what is the problem in the code Give the answer in a tabular format.
> ```

- After the prompting, we got our data but it was looking like some kind of combination of all explanations without any cleansing redundant data or anything so we further instructed it to summarize the data and do sanitizing for proper csv exporting. After few

iterations we got our aimed output.
Result:

FailingMethod	TTR	Answer.size
HIT01_8	0.374593	614
HIT02_24	0.375224	557
HIT03_6	0.360515	1165
HIT04_7	0.350877	1254
HIT05_35	0.365721	916
HIT06_51	0.397756	802
HIT07_33	0.346756	894
HIT08_54	0.389503	724

2. **LLM Crafted Prompt**
- After successful result that was received from the first session, we asked LLM that it understood the task now and wanted from it to craft a prompt to carry out the exactly the same task. The generated prompt was as follows:

> I have a dataset containing bug reports and programmers' explanations for fixing those bugs. I need concise, formal summaries for each bug report grouped by the failing method. Each summary should focus on the following aspects:
>
> How the program works
> How the failure is happening
> What is the problem in the code
> The task involves:
>
> Removing redundant or informal information: Avoid phrases like "This is fine" or irrelevant text such as "ffffffff."
> Formalizing the language: Use clear and professional phrasing without slang or placeholders.
> Sanitizing text: Ensure the output is suitable for CSV export by removing problematic characters like semicolons, unnecessary newlines, and ensuring proper punctuation.
> Minimizing repetition: Consolidate similar explanations while retaining all necessary technical details for understanding and fixing the bug.
> The output should be in tabular format with two columns:
>
> Failing Method: The identifier for the bug report.
> Refined Explanation: A single, concise, and formal explanation summarizing the fix for the bug.

> Please ensure the summaries are actionable and easy for a developer to understand and implement.

- After giving the prompt it provided cut sentences which further prompting needed but lesser in amount compared with the first approach.

Result:

Failing Method	TTR	Answer.size
HIT01_8	0.335931	1155
HIT02_24	0.376749	929
HIT03_6	0.320336	2026
HIT04_7	0.314664	2339
HIT05_35	0.318946	1784
HIT06_51	0.337104	1768
HIT07_33	0.310073	1648
HIT08_54	0.298420	1709

3. **LLM Crafted Prompt with Roleplay**

- Not very much have changed in this prompt but we wanted to make a little roleplay to see if LLM going to perform better. The prompt was as follows:

> **Your are a senior data scientist / software engineer working in a company and you got a list of csv data that has explanations about bug reports written by several programmers that inspected the bug.**
>
> I have a dataset containing bug reports and programmers' explanations for fixing those bugs. I need concise, formal summaries for each bug report grouped by the failing method. Each summary should focus on the following aspects:
>
> How the program works
> How the failure is happening
> What is the problem in the code
> The task involves:
>
> Removing redundant or informal information: Avoid phrases like "This is fine" or irrelevant text such as "ffffffff."
> Formalizing the language: Use clear and professional phrasing without slang or placeholders.
> Sanitizing text: Ensure the output is suitable for CSV export by removing problematic characters like semicolons, unnecessary newlines, and ensuring proper

> punctuation.
> Minimizing repetition: Consolidate similar explanations while retaining all necessary technical details for understanding and fixing the bug.
> The output should be in tabular format with two columns:
>
> Failing Method: The identifier for the bug report.
> Refined Explanation: A single, concise, and formal explanation summarizing the fix for the bug.
> Please ensure the summaries are actionable and easy for a developer to understand and implement.

Result:

FailingMethod	TTR	Answer.size
HIT01_8	0.341530	1098
HIT02_24	0.406607	787
HIT03_6	0.326636	1941
HIT04_7	0.306500	2323
HIT05_35	0.325458	1693
HIT06_51	0.342700	1637
HIT07_33	0.320565	1488
HIT08_54	0.300371	1618

Compare Explanations

We picked first approach to compare the explanations generated from it with the original data. Firstly we have grouped our original data into FailingMethod and merged all of the answer explanations into one row per bug report to calculate the ROUGE and BLEU scores. Our results were as follows:

FailingMethod	ROUGE1	ROUGE2	ROUGEL	ROUGELsum	BLEU
HIT01_8	0.197467	0.188434	0.115622	0.115622	0.000187
HIT02_24	0.299141	0.283718	0.163802	0.163802	0.010667
HIT03_6	0.223827	0.213549	0.112195	0.112195	0.000646
HIT04_7	0.143587	0.135971	0.078544	0.078544	0.000005
HIT05_35	0.301356	0.287069	0.150678	0.150678	0.009524

BLEU

BLEU scores range from 0 to 1, with higher scores indicating greater similarity between LLM generated and reference texts.

In our findings, BLEU score were always very low which meant that text generated by the ChatGPT was not aligning with the original answer explanations. This suggests that the LLM generated content may lack informativeness in explaining the bug report.

ROUGE

In our findings, Rouge1 and Rouge2 scores actually performed higher compared with the other metrics meaning that there were more words that were in common with the original text in LLM content in terms of singular words and two words combined patterns.

Overall, lower scores doesnt mean that the resulting answers are not serving its purpose. It just means that sentences formed by the LLM is significantly different than the original explanations. his could result from ChatGPT using vocabulary or phrasing that differs from typical English usage