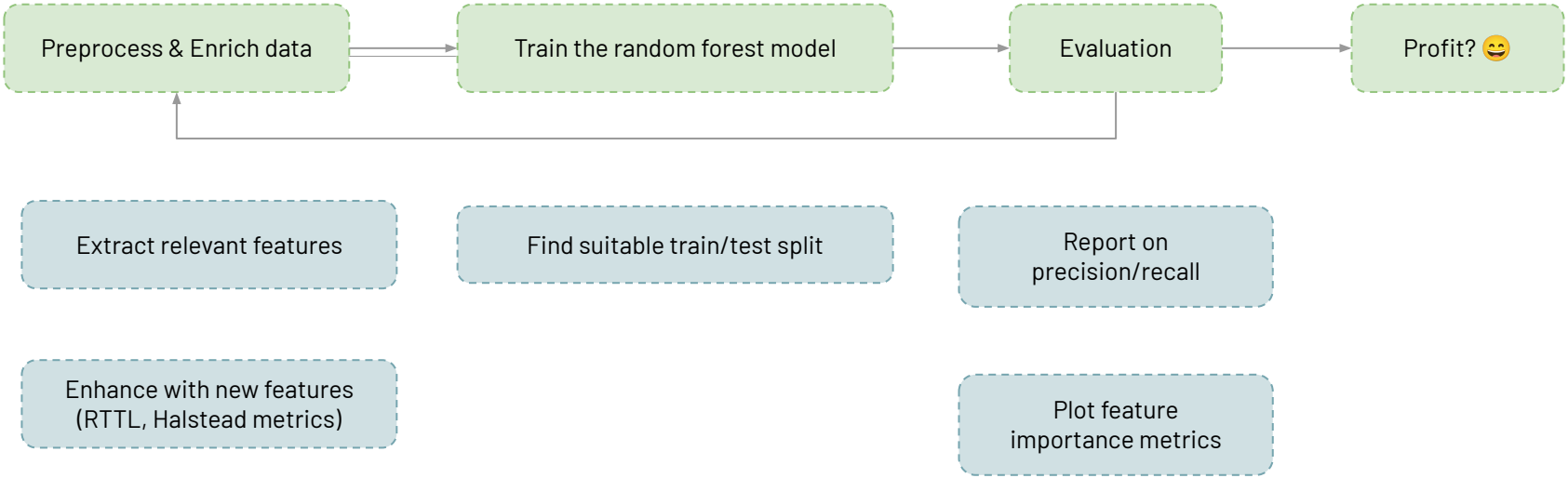
Mini Project 2

Cedric Lorenz, Leonard Dreessen, Oliver Hess, and Raphael Reimann

Task 1: Categorizing the answers

Goal: Train a classifier model in order to evaluate whether an answer correctly identifies the bug.

Steps

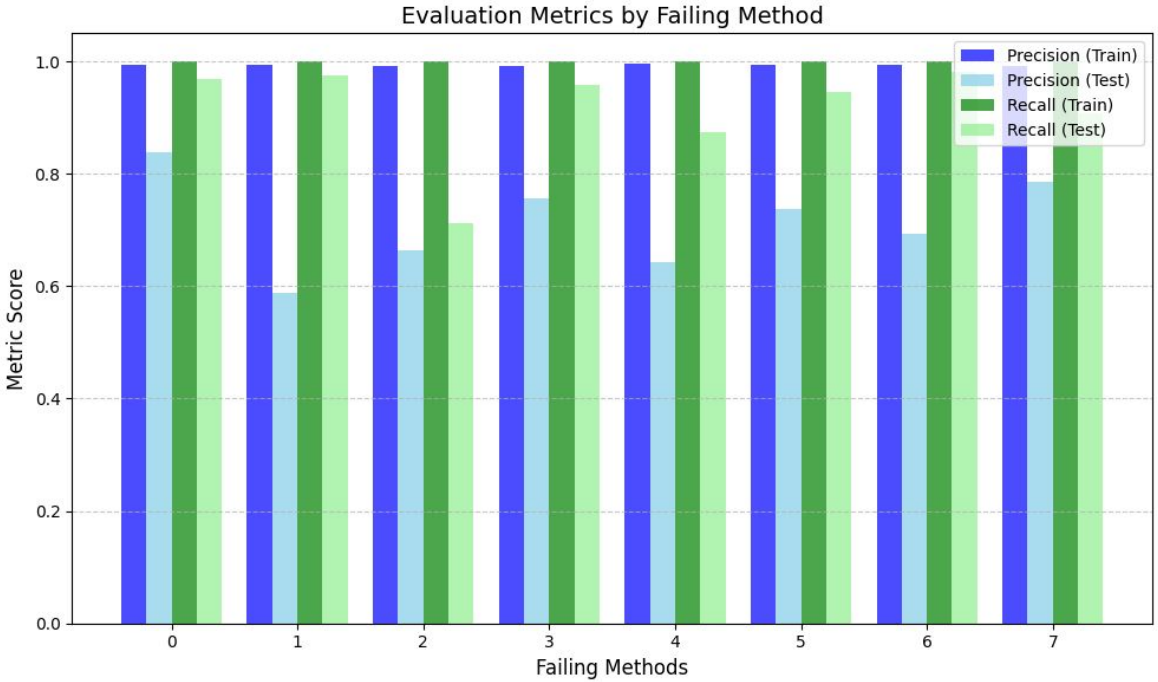


Task 1: Categorizing the answers

- ~0.73 test precision
- ~0.92 rest recall

- ~0.51 test precision
- ~0.23 test recall
- All kinds of weird behaviors because the model cannot learn properly

We wanted to check how different test sets affect performance

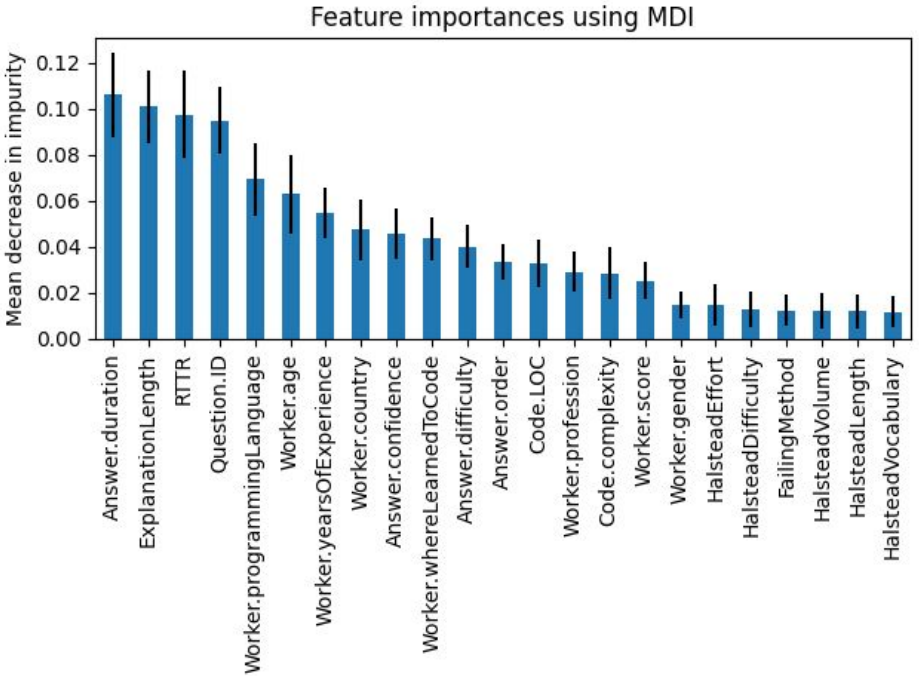


… since taking 2 fixed sets is not robust

Results:

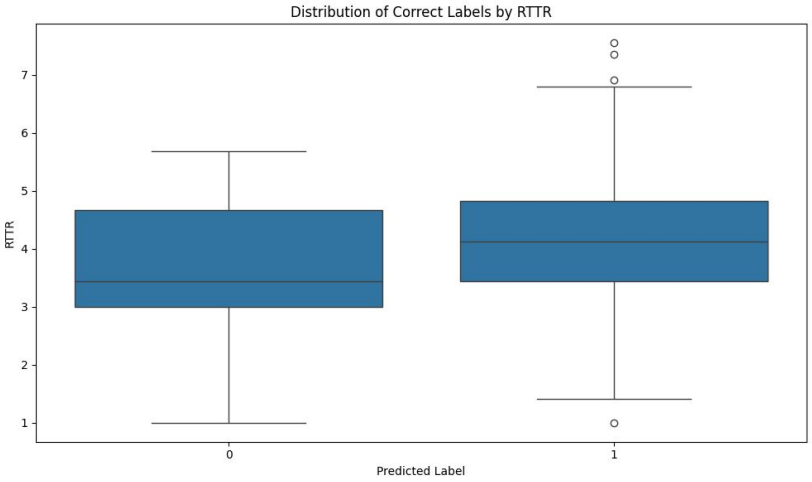
- Some FailingMethods actually are harder to predict
- Also: we do see overfitting. But less overfitting would also reduce performance

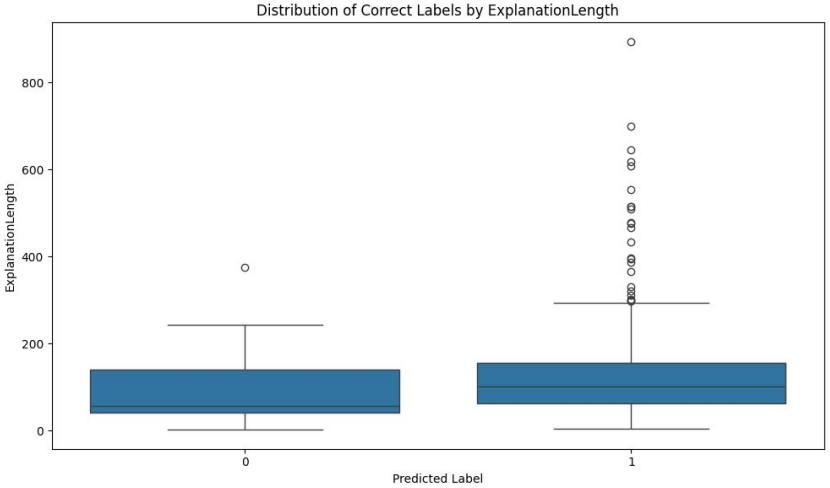
Explainability



- The model recognizes answer-related features to be most important (Duration, Length, RTTR)
- Code-related features are less important (Code.LOC, Complexity, Halstead Metrics)
- The RF cannot really relate the code to the answer but uses proxy variables
    - We thought of integrating code / text embeddings into the model →too complex and not helpful

Distribution of Correct Labels by RTTR / Explanation Len





Reflection on Classifier

- **Lots of preprocessing:** Lots of time investigating suitable features and enriching the data with new metrics to use.
  → New data income or updates? Again!

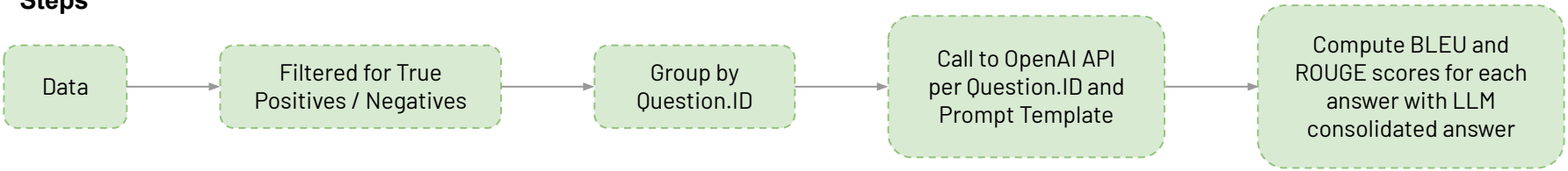
- **Detecting possible overfitting & biases**: Crucial for testing the classifier
  → Worker.gender - is this ethical?
  → New data income or updates? Needs monitoring!

- **Overall sanity of using a classifier?** How does the model actually relate the code bugs to the explanation of the user? More via correlation than causation 🤔

Task 2: LLM consolidated answers

Goal: Merge multiple human-written bug explanations into one concise, consistent explanation.

Steps



Prompt templates

I have a dataset of code bug report explanations. There were unit test failures that were given to programmers together with the code. The programmers then should explain the problem in the code. Your task is to consolidate the answers of the programmers into a single explanation by merging the programmers' explanations in a way that minimizes redundant information, while keeping the information that would be necessary for someone else to fix the bug.

Prompt A

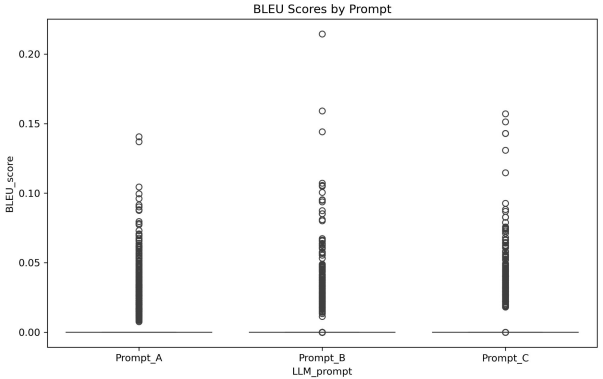
Programmers have provided bug explanations. Your goal is to unify them into a single, clear explanation that keeps necessary technical details but avoids repeating the same info. Provide a short summary focusing on the steps needed to fix the bug.

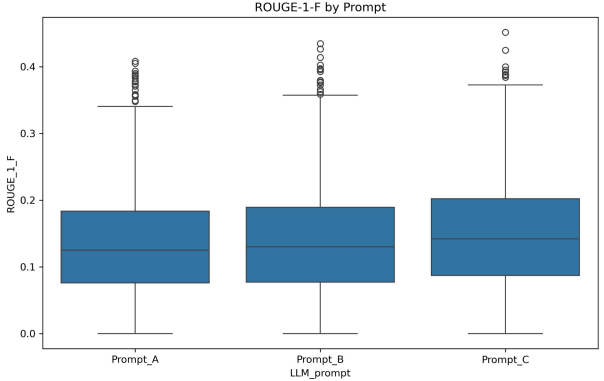
Prompt B

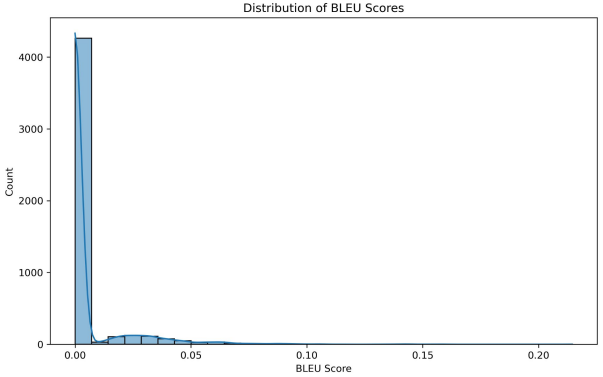
Combine the following bug explanations into one cohesive report. Minimize repetition, but ensure all crucial details needed for another developer to fix the bug are included. The final explanation should remain concise (ideally under 150 words).

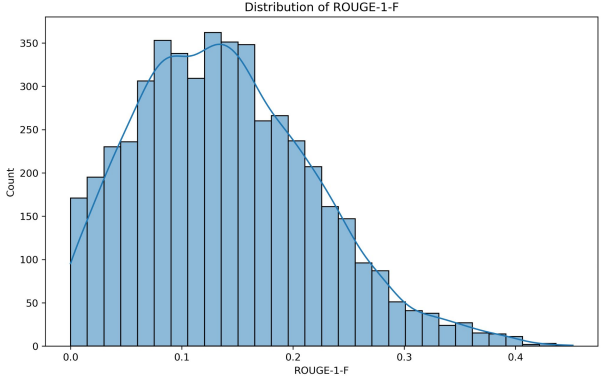
Prompt C

BLEU/ROUGE Scores









BLEU: Typically very low (often near zero) due to rewording.

ROUGE: Ranges more widely (0.0 to ~0.4 in F-score). Partial overlap with original text.

Low scores show rephrasing or partial coverage of the LLM answer, they don't mean the content is poor

Reflection on LLM consolidation

- **Low Overlap ≠ Low Quality**: The LLM's rephrasing can reduce textual similarity scores (BLEU, ROUGE), yet still produce a valid or even improved explanation
- **Prompt Engineering**: The LLM's focus can be steered with different prompt templates(e.g., "minimize redundancy, keep essential info")
- **Risk of Hallucination**: Hallucination is still a risk, with automation comes need for conscious checking of results

LLM-based consolidation can be useful but needs prompt engineering and domain checking to achieve correctness and completeness

Further Reflections

- Data quality
- Keeping the classifier up-to-date in the case of changes in the demographic of programmers or types of bugs
- Testing the output of the classifier and the LLM
- Estimating the quality of the consolidated explanations
- Debugging the integration between the classifier and the LLM