Mini-Project 3

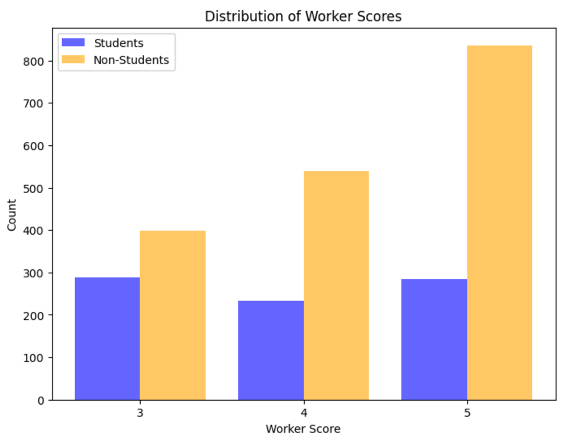
All tasks are also documented in the notebook, were you find the detailed code for the experiments. (here)

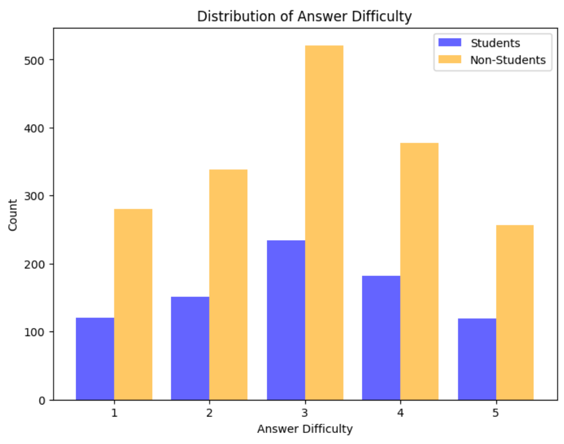
Task 1

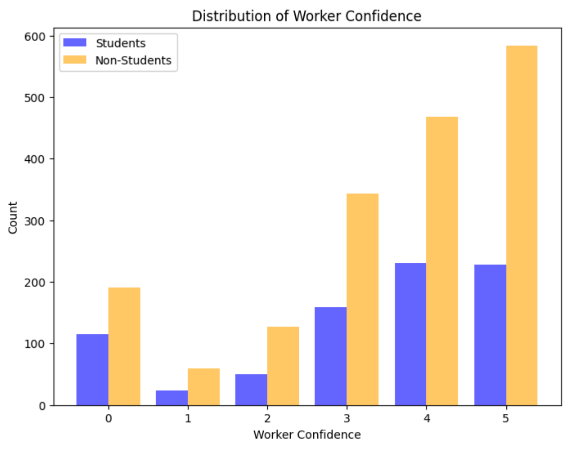
First we preprocessed our dataset like in the 2 exercises, which means that we applied some normalisation on Answer.Duration, halstead volume and explanation length and we also corrected the GroundTruth to Groundtruth = 1 if Answer.option=YES. After that we split the datasets into Students and Non-Students and investigated the different feature Distributions. For the categorical values we also conducted a chi-squared test to validate the visual differences in the distributions. And for the other features we conducted a Mann-Whitney Test.

Categorical Features

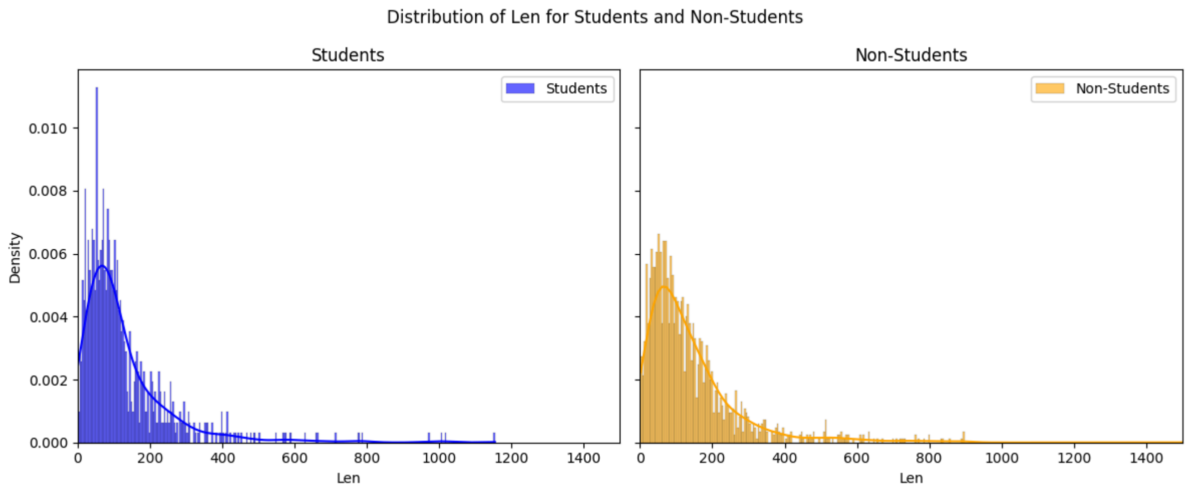
The p-values for the test were:
Worker Score: 1.256e-12
Answer Difficulty: 0.946
Worker Confidence: 0.042

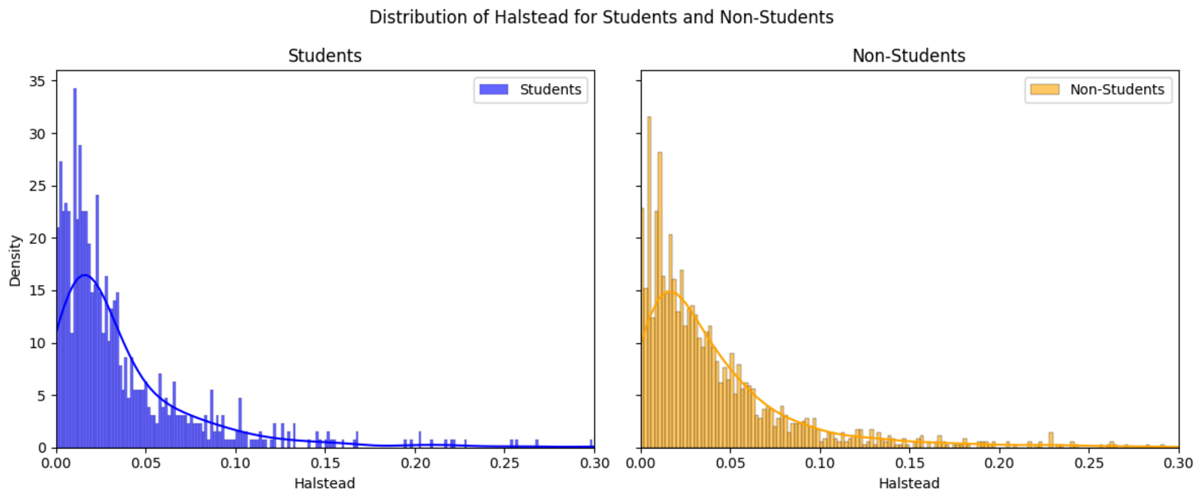


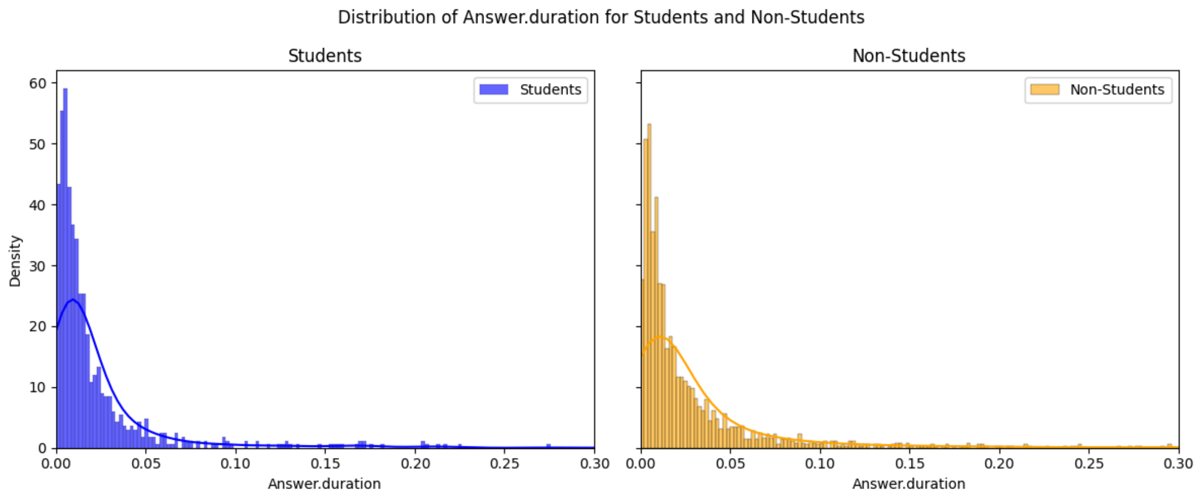




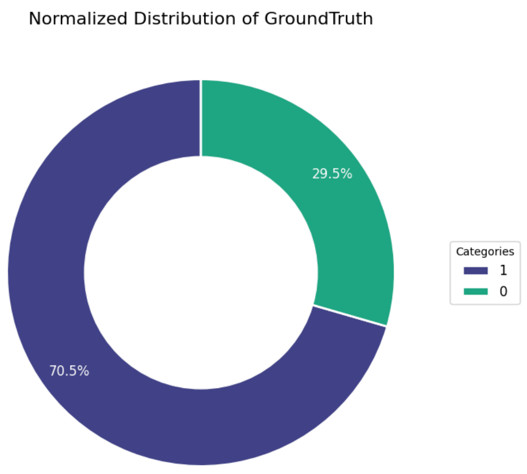
For the Mann-Whitney Tests the p-Values were:
Answer length: 0.0016
Halstead: 0.0034
Answer duration normalized: 4.456e-06





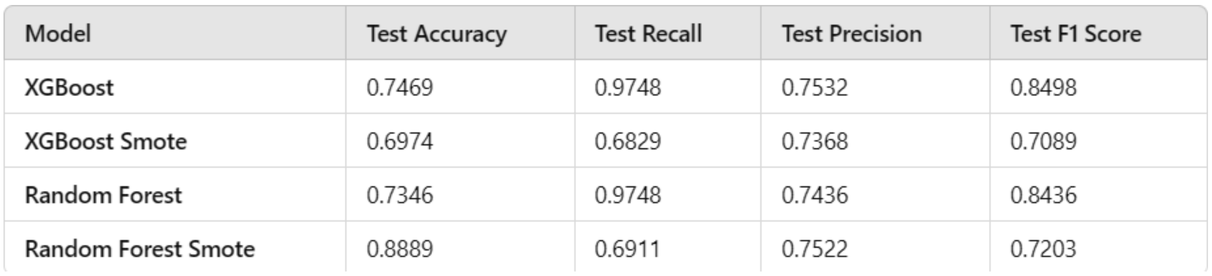


Summary:
We found out that only the distribution of the Answer.Difficulty is very similar for students and non-students. For all other features, the differences between the two groups seem reasonable. The visual differences for len, halstead metric and Answer.duration are not really visible but the statistical tests showed a significant difference.



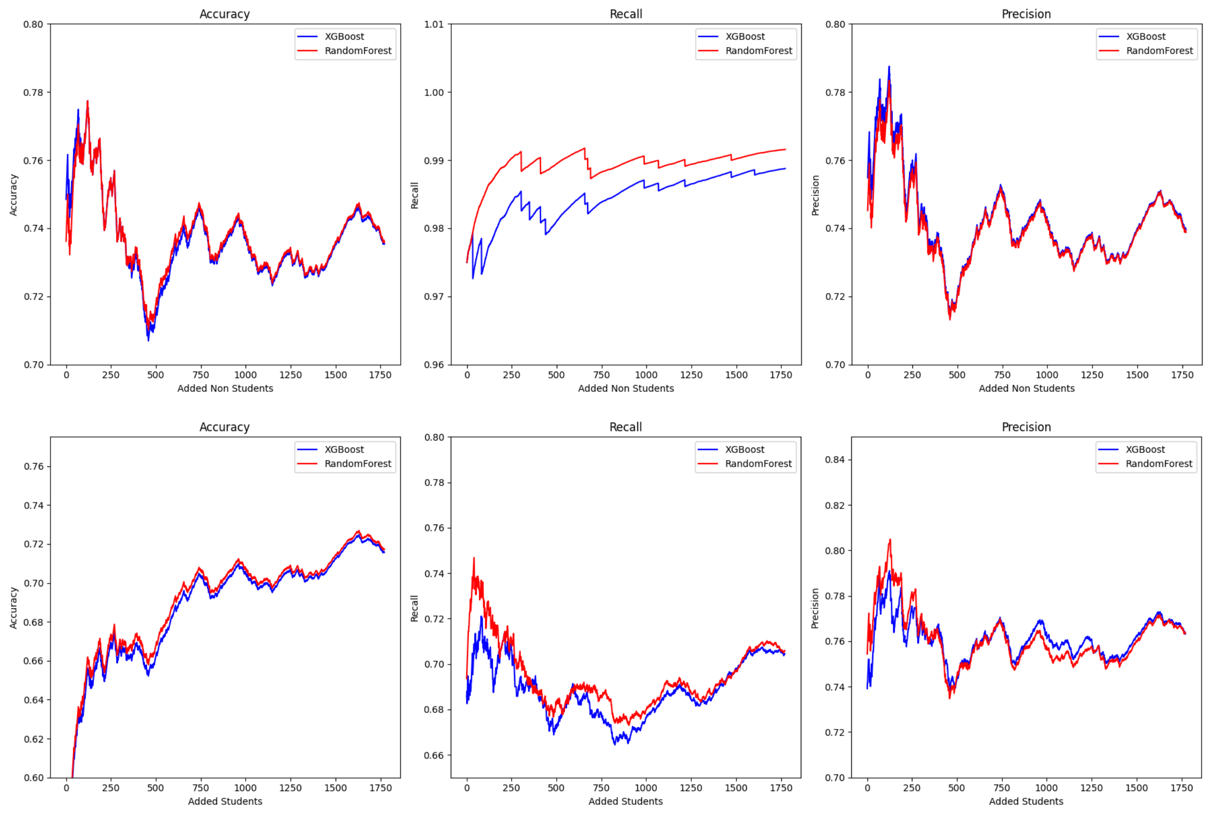
After this we performed the train test split on the student dataset and validated the number of True and False labels. And saw that we got far more True Labels than False ones, so we decided to train a model for the original data set and one where we used SMOTE and added new 331 False labels so that we achieved a 50:50 distribution. The sizes of the train test splits were:
Without resampling: Train: 645; Test: 162
With resampling: Train: 910; Test: 228

We continued with training 4 models in total, one RandomForestClassifier and one XGBoostModel for each dataset (see above). To get details on the training see jupyter notebook (here). The result of the final model on the test set were:

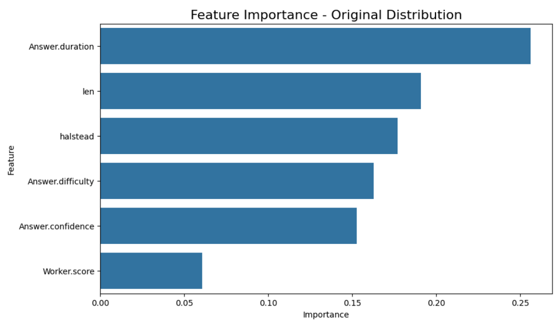


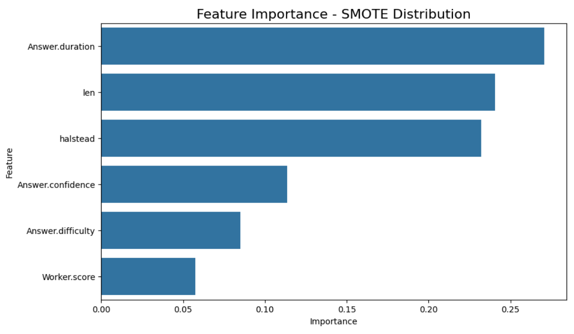
Model	Test Accuracy	Test Recall	Test Precision	Test F1 Score
XGBoost	0.7469	0.9748	0.7532	0.8498
XGBoost Smote	0.6974	0.6829	0.7368	0.7089
Random Forest	0.7346	0.9748	0.7436	0.8436
Random Forest Smote	0.8889	0.6911	0.7522	0.7203

After that we added non-students to the test-dataset and investigated the difference in the metrics and received the following plots (seed 42 for split: also tested with 42069 but achieved similar results).



Unexpectedly we don't see the expected decrease in the performance metrics. Strangely the accuracy and recall and also precision increased. Because of this trend, we looked at the feature importances of the models and plotted the importances of the RandomForest Classifier Model (also similar results for XGBoost - see notebook). We found that some features are more important than others. However, *Answer.difficulty*, where the distributions were nearly identical, is not the most important feature. Therefore, we cannot conclude that this behavior is due to similar distributions.





Based on these results we decided to skip Question 1.2, because the model did not get worse with more Non-students in the test set and it wouldn't make sense to retrain with non-students (performance already is equal).

Task 2

Question 2.1

For this task, we want to get a good consolidated explanation by the LLM while giving it the least possible explanations as a reference. We will accept the consolidated explanation if it lies above a certain threshold for readability and semantic similarity.
We defined the thresholds as the following: for BertScore (semantic similarity), we want to get an f1 score over 0.9. For readability, we utilize the Flesch Reading Ease Score and the Flesch Kincaid Grade Level as metrics. In general we aim for a general and fairly easy explanation, which means we aim for a Flesch Reading Ease Score of over 60. Additionally we want that the explanation is understandable and clear for programmers with intermediate programming skills expecting a Flesch-Kincaid Grade Level between 8 and 12.

We thought of two approaches to solve this problem:

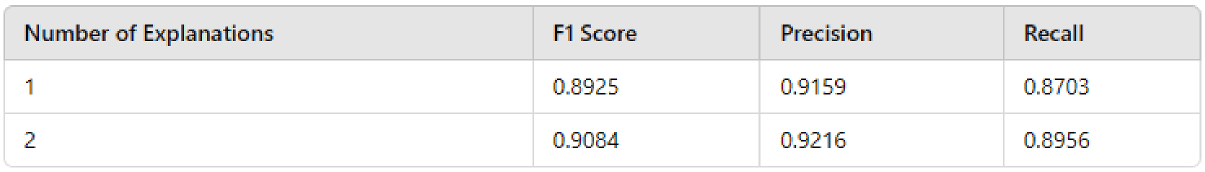
- randomly selecting explanations and adding them to the LLM prompt
- sorting explanations first and then adding them one by one to the LLM prompt

To “sort” the explanations, we used the calculated halstead volume and ordered them by descending halstead value.

Our prompt for the LLM was the following:
“””
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. Avoid headings or special formatting.

Here are the explanations, each separated by a newline:
[list of explanations]
“””

We tried out both approaches on one bug type, and found significant differences:
For the sorted one we only need 2 samples to pass the desired threshold (BERT-Score > 0.9):

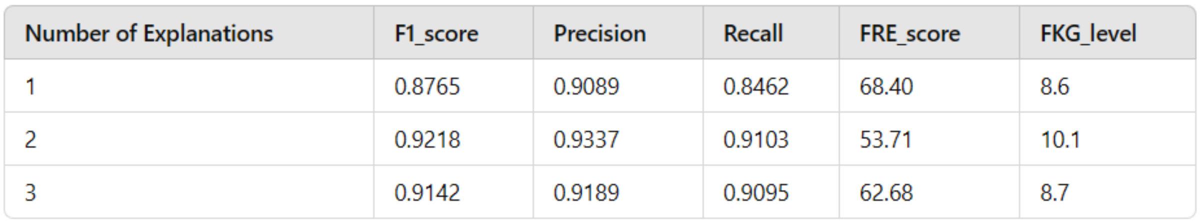


Number of Explanations	F1 Score	Precision	Recall
1	0.8925	0.9159	0.8703
2	0.9084	0.9216	0.8956

For the random one, we did three test runs and on average 12 samples (20, 11, 4) to pass the threshold. For the detailed results see Fig. 1 in the Appendix. The difference between our sorted approach and the random approach is very apparent, therefore we decided to always use the sorted approach from now on since it performs much better and it is easier to work with. Using our original prompt, we were not able to achieve our defined readability thresholds. That is why we adjusted the prompt and added the following line to the end of the instruction.

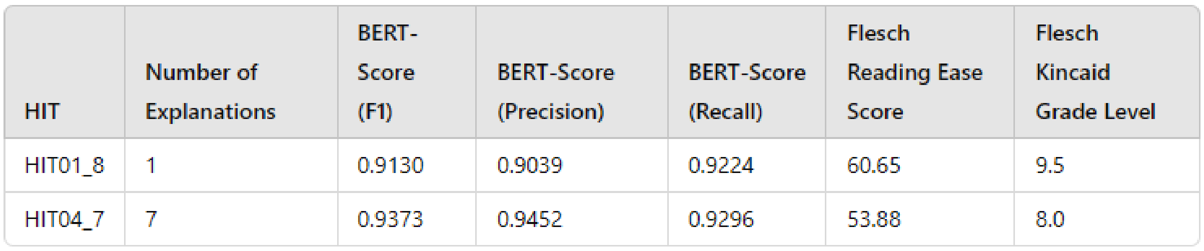
“”””
Additionally, aim for a Flesch Reading Ease Score between 60 and 80, and a Flesch-Kincaid Grade Level between 8 and 12.
“”””

We tried this for Bug type HIT07_33:
After adding only 3 explanations, we got a suitable result:



Number of Explanations	F1_score	Precision	Recall	FRE_score	FKG_level
1	0.8765	0.9089	0.8462	68.40	8.6
2	0.9218	0.9337	0.9103	53.71	10.1
3	0.9142	0.9189	0.9095	62.68	8.7

To verify the thresholds and the approach we also tested on other bug types and encountered much more problems than expected. For bug type HIT01_8 and HIT04_7 we didn’t get suitable results for 15 given explanations, mainly because the Readability scores were too low. It seems that a Flesch Reading Ease of 60 is too strict for these bug types, because the naming of the variables is complicated and the code examples did not help to make the text more understandable. The Reading Ease score may not be the most suitable metric when code examples and mathematical operations are included in the explanation. In this case, the Flesch-Kincaid score would likely be a more appropriate metric. Detailed results on the ran experiments can be found in the Appendix in Fig. 2 for HIT01_8 and Fig.3 for HIT04_7.
That is why we ignored the Flesch for now, and picked the number of explanations which achieved the best BertScore. Using this number of explanations (1 for HIT01_8, 7 for HIT04_7), we adjusted our prompt by modifying the readability metrics. To improve the LLM's performance on the reading ease score, we replaced the target score of 60 with 70 and reran the experiment with the same prompt till we get a suitable consolidation. With this new prompt and a couple of retries, we were able to achieve better results:

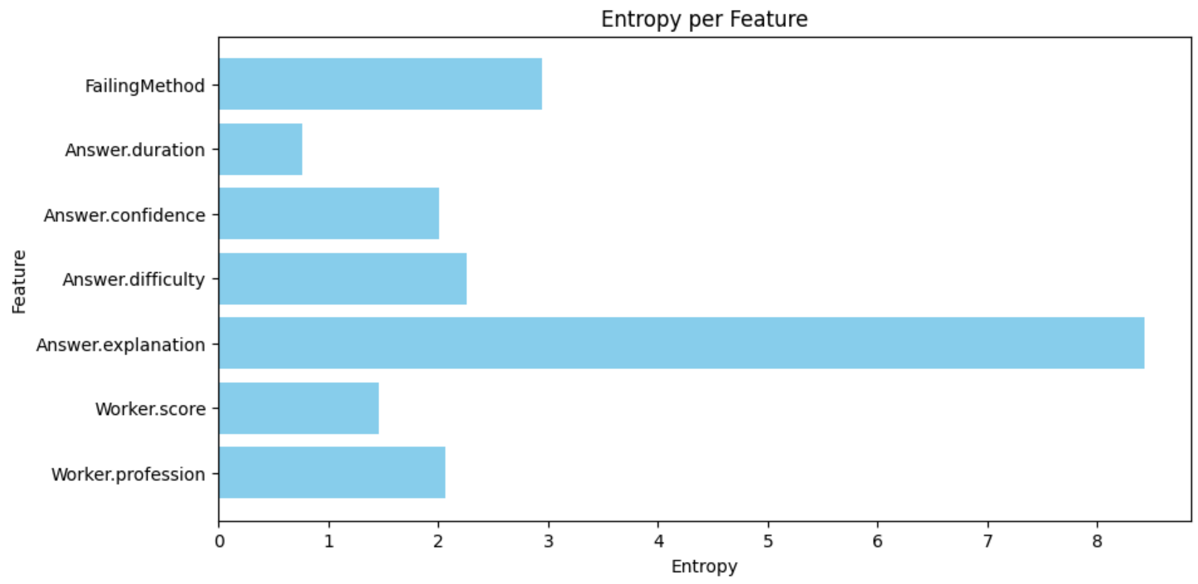


HIT	Number of Explanations	BERT-Score (F1)	BERT-Score (Precision)	BERT-Score (Recall)	Flesch Reading Ease Score	Flesch Kincaid Grade Level
HIT01_8	1	0.9130	0.9039	0.9224	60.65	9.5
HIT04_7	7	0.9373	0.9452	0.9296	53.88	8.0

The Flesch Reading Ease Score for HIT04_7 still doesn’t pass our threshold, however the Grade Level is fine. During our testing, we observed that the Flesch Reading Ease Score and the Flesch Kincaid Grade level are inverse to each other - meaning that if the Reading Ease gets higher, the Grade level drops and vice versa.

Task 3

Question 3.1



To measure diversity, we take a look at the entropy of each feature, or more specifically the shannon entropy.
Doing this, we notice that the Answer.explanation feature has the highest entropy with more than 8, therefore contributing most to a diverse dataset. We know that we look at different failing bug methods, so an entropy of around 3 is to be expected for the FailingMethod feature.
The other features don't seem to be too diverse, with Worker profession and Answer difficulty being over 2 - this indicates some diversity, however not nearly as much diversity as with the Answer explanation. Answer duration, answer confidence and worker score have an entropy of less than 2, meaning they don't contribute a lot to a diverse dataset.

Since it is apparent that the Explanation is the most diverse feature, we decided to focus solely on this. From now on, we only inspect the explanations, disregarding the other features such as worker profession because it does not play such a big role.

Question 3.2

Now, we take a look at semantic similarity and readability independently of the diversity. To us, the semantic similarity is more important than the readability, so we first gather all explanations that are similar enough to our ground truth.
To find these, we defined the threshold of 0.87 for the F1 value of the BertScore. First, we tried 0.9, however this yielded no results, or sometimes one result only. Since we wanted multiple explanations though, we decided to lower the threshold to 0.85. This yielded way too many explanations however. So we adjusted the threshold again, and with 0.87, we get between 5 and 10 explanations which are considered as similar enough. Using Fletcher Readability Ease score, we then sorted all of the previously chosen explanations by

readability, combining semantic similarity with good readability. Examples for Bug Type HIT01_8 are listed in the Appendix under Explanations 1.

Question 3.3

Using the explanations with the highest semantic similarity, we first compute the entropy for every single explanation on its own. This allows us to find the most diverse single explanation that lies over our semantic similarity threshold. Examples for Bug Type HIT01_8 are listed in the Appendix under Explanations 2.

Since we wanted to find a diverse set of explanations, we then created all possible subsets from the explanations that passed our semantic similarity threshold. Doing this, we were able to achieve an even higher entropy:

Most Diverse Set (Highest Entropy):
Entropy: 6.551794385202084

There is a logical check for if minuteOffset is less than 0 or greater than 59 causing it to throw an exception because the value is out of bounds (negative number)

YES. The issue is on line 279 (as I explained in my first question; of which I misunderstood that I was only being asked about the specific issue; not generalized issue). On line 279 the variable "minutesOffSet" is parameterized to throw an exception if it is < 0 or > 59. Line 279 should read "if (minutesOffset < -59 || minutesOffset > 59) {" because now the method can take in the number of minutes as a negative and will allow the method to properly progress to invoke/call further methods such as those asked about in the two previous questions.

The variable "minutesOffset" is checked incorrectly by the IF statement on line 279. Any negative value for "minutesOffset" will throw this exception; while the documentation states that "minutesOffset" can be negative in some cases.

The error is stemming from line 279 because the value of -15 for minutesOffset is < 0. The line should be   if (minutesOffset < -59 || minutesOffset > 59) {

You are passing it a negative offset value (-15) and the conditionals are set to reject any offset that is less than 0 or greater than 59 and throw a new exception.

-15 is less then 0; so it throws IllegalArgumentException

In this case, the most diverse subset is the one that includes all explanations, however for different bug types, the subset did not include all explanations.

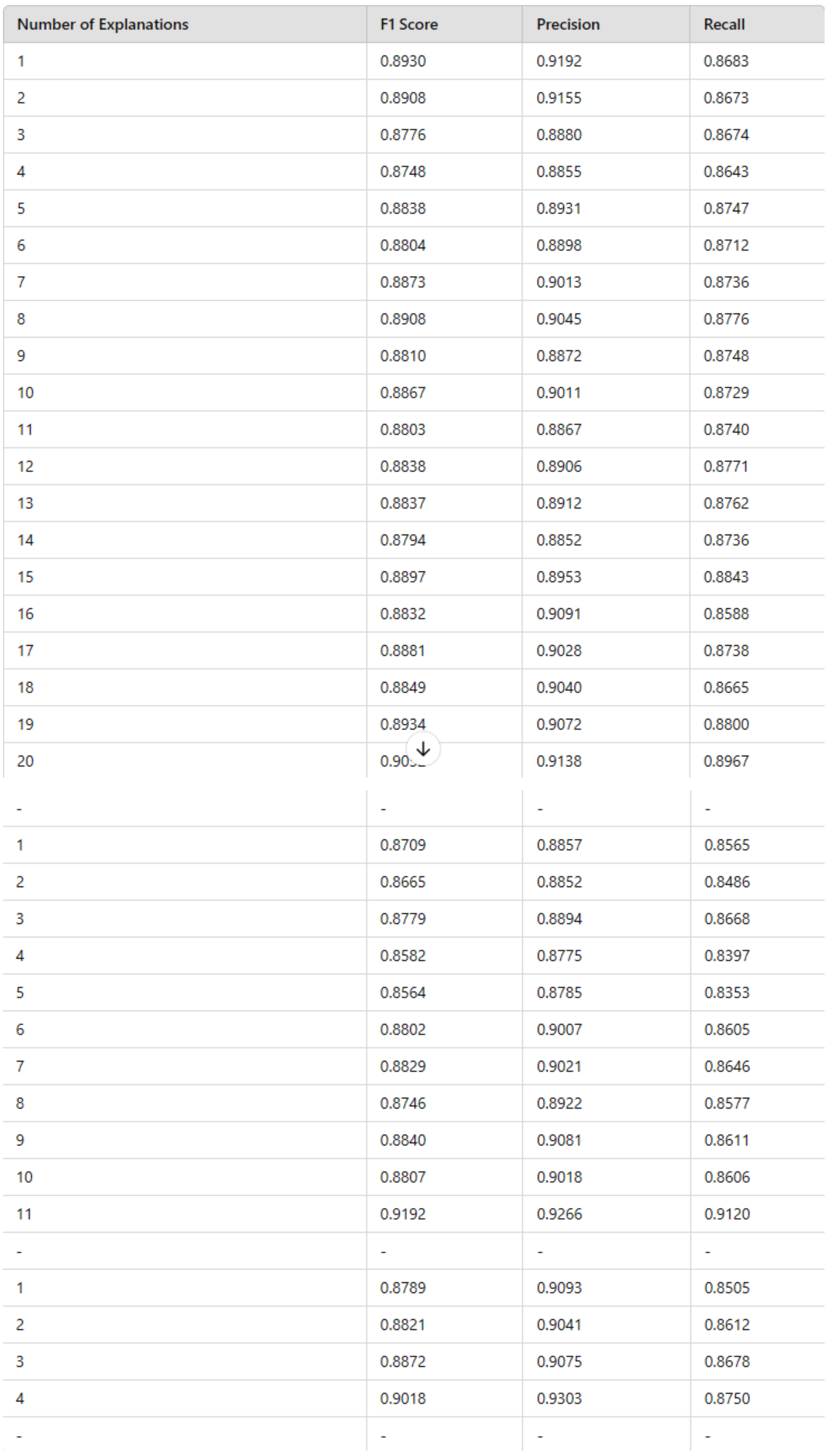
We were interested in how the LLM would perform with these diverse explanations. Using the following prompt:
“””
Programmers have provided bug explanations.
Your goal is to unify them into a single, clear explanation that keeps necessary technical details but avoids repeating the same info. Provide a short summary focusing on the steps needed to fix the bug. Avoid headings or special formatting.

Here are the explanations, each separated by a newline:
“””
(followed by all the explanations)

we generated a new explanation. When running our semantic similarity and readability checks, we were happy to see that the new explanation generated by the LLM performed very well:
We achieved a BertScore (semantic similarity) of 0.9268, a Fletcher Readability Ease score of 39.03 and a Fletcher Kincaid Grade Level of 11.6.
Since we are compromising the Readability, this is a very good result, giving us some of the highest semantic similarity we have seen so far while achieving maximum diversity by creating the set of explanations.

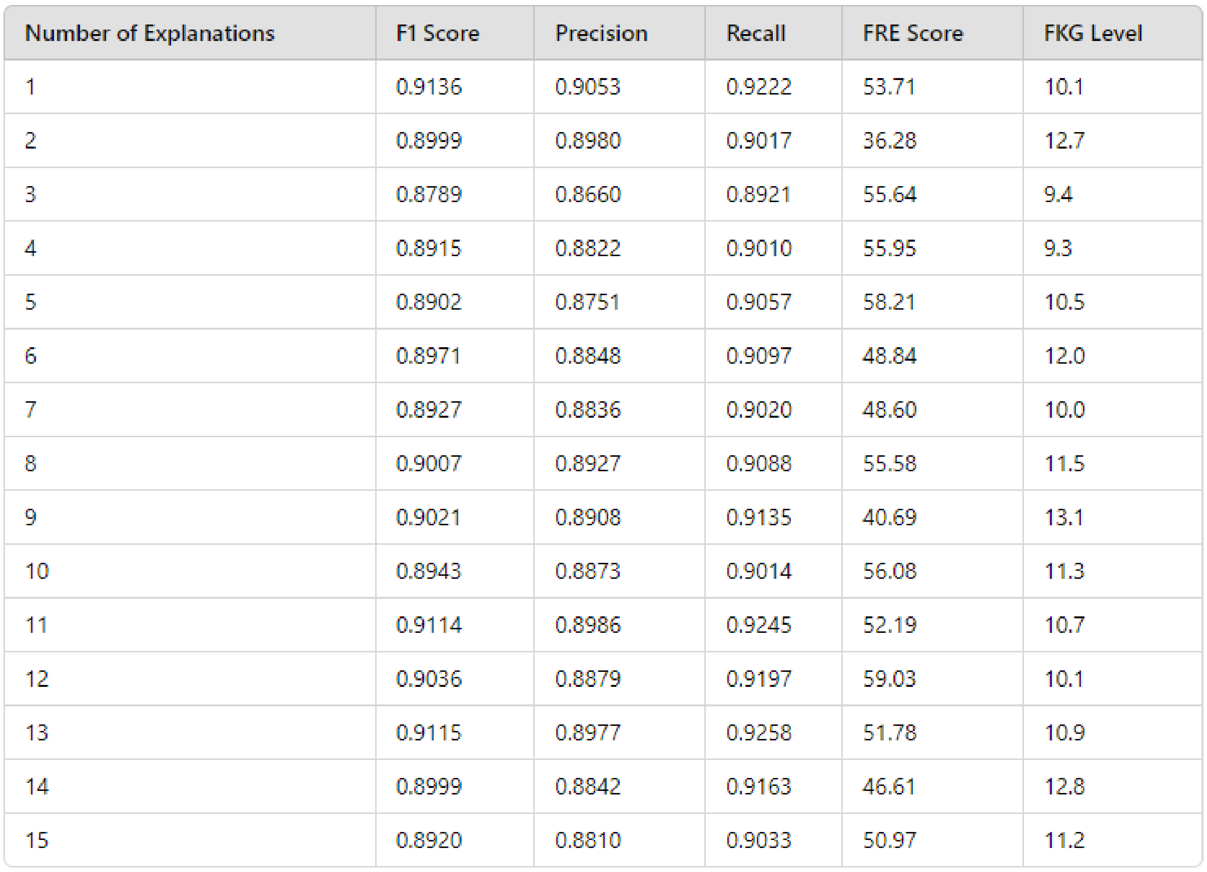
Appendix

Fig. 1. Random Samples for HIT07_33



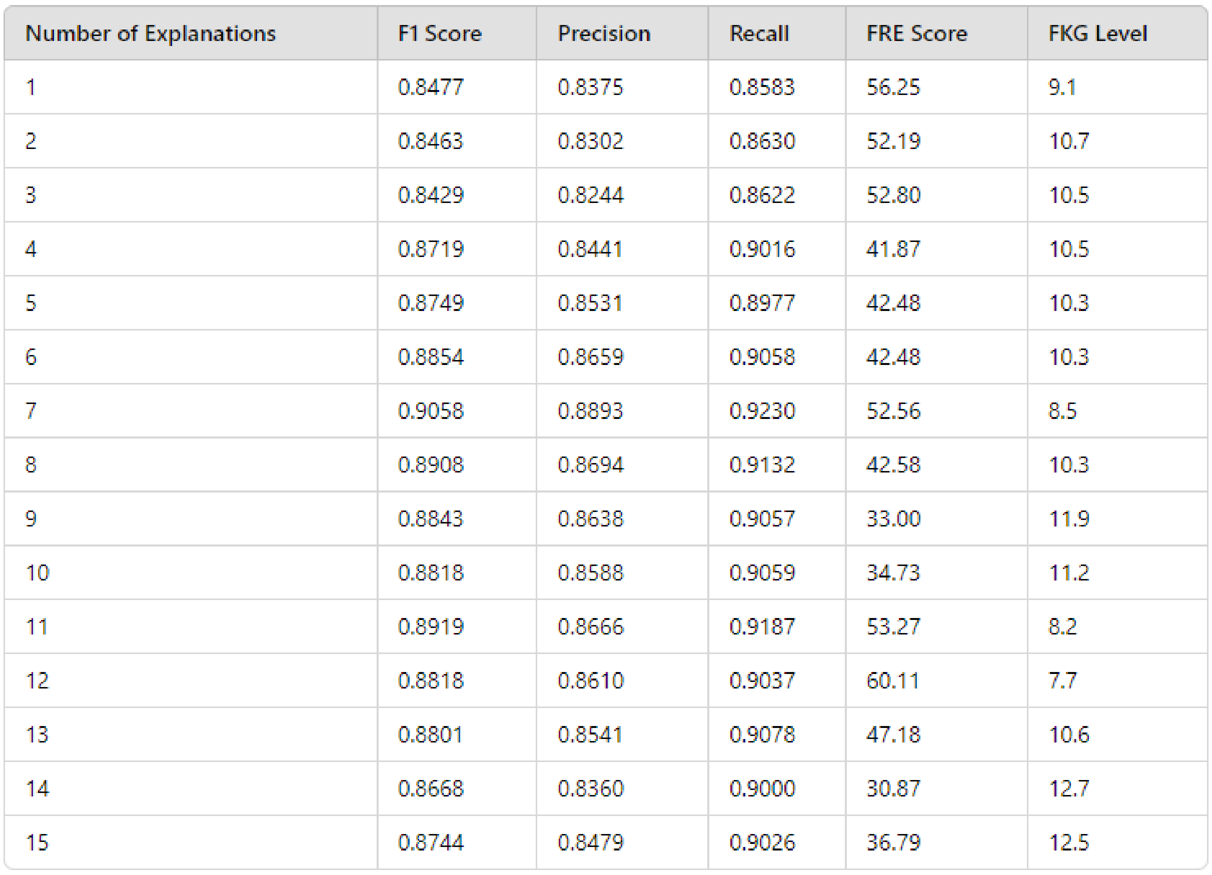
Number of Explanations	F1 Score	Precision	Recall
1	0.8930	0.9192	0.8683
2	0.8908	0.9155	0.8673
3	0.8776	0.8880	0.8674
4	0.8748	0.8855	0.8643
5	0.8838	0.8931	0.8747
6	0.8804	0.8898	0.8712
7	0.8873	0.9013	0.8736
8	0.8908	0.9045	0.8776
9	0.8810	0.8872	0.8748
10	0.8867	0.9011	0.8729
11	0.8803	0.8867	0.8740
12	0.8838	0.8906	0.8771
13	0.8837	0.8912	0.8762
14	0.8794	0.8852	0.8736
15	0.8897	0.8953	0.8843
16	0.8832	0.9091	0.8588
17	0.8881	0.9028	0.8738
18	0.8849	0.9040	0.8665
19	0.8934	0.9072	0.8800
20	0.9052	0.9138	0.8967
-	-	-	-
1	0.8709	0.8857	0.8565
2	0.8665	0.8852	0.8486
3	0.8779	0.8894	0.8668
4	0.8582	0.8775	0.8397
5	0.8564	0.8785	0.8353
6	0.8802	0.9007	0.8605
7	0.8829	0.9021	0.8646
8	0.8746	0.8922	0.8577
9	0.8840	0.9081	0.8611
10	0.8807	0.9018	0.8606
11	0.9192	0.9266	0.9120
-	-	-	-
1	0.8789	0.9093	0.8505
2	0.8821	0.9041	0.8612
3	0.8872	0.9075	0.8678
4	0.9018	0.9303	0.8750
-	-	-	-

Fig. 2: HIT01_8



Number of Explanations	F1 Score	Precision	Recall	FRE Score	FKG Level
1	0.9136	0.9053	0.9222	53.71	10.1
2	0.8999	0.8980	0.9017	36.28	12.7
3	0.8789	0.8660	0.8921	55.64	9.4
4	0.8915	0.8822	0.9010	55.95	9.3
5	0.8902	0.8751	0.9057	58.21	10.5
6	0.8971	0.8848	0.9097	48.84	12.0
7	0.8927	0.8836	0.9020	48.60	10.0
8	0.9007	0.8927	0.9088	55.58	11.5
9	0.9021	0.8908	0.9135	40.69	13.1
10	0.8943	0.8873	0.9014	56.08	11.3
11	0.9114	0.8986	0.9245	52.19	10.7
12	0.9036	0.8879	0.9197	59.03	10.1
13	0.9115	0.8977	0.9258	51.78	10.9
14	0.8999	0.8842	0.9163	46.61	12.8
15	0.8920	0.8810	0.9033	50.97	11.2

Fig. 3: HIT04_7



Number of Explanations	F1 Score	Precision	Recall	FRE Score	FKG Level
1	0.8477	0.8375	0.8583	56.25	9.1
2	0.8463	0.8302	0.8630	52.19	10.7
3	0.8429	0.8244	0.8622	52.80	10.5
4	0.8719	0.8441	0.9016	41.87	10.5
5	0.8749	0.8531	0.8977	42.48	10.3
6	0.8854	0.8659	0.9058	42.48	10.3
7	0.9058	0.8893	0.9230	52.56	8.5
8	0.8908	0.8694	0.9132	42.58	10.3
9	0.8843	0.8638	0.9057	33.00	11.9
10	0.8818	0.8588	0.9059	34.73	11.2
11	0.8919	0.8666	0.9187	53.27	8.2
12	0.8818	0.8610	0.9037	60.11	7.7
13	0.8801	0.8541	0.9078	47.18	10.6
14	0.8668	0.8360	0.9000	30.87	12.7
15	0.8744	0.8479	0.9026	36.79	12.5

Explanations 1: The explanations that pass the semantic similarity threshold sorted by their readability scores
(Example for Bug Type HIT01_8):

Explanation: The error is stemming from line 279 because the value of -15 for minutesOffset is < 0. The line should be   if (minutesOffset < -59 || minutesOffset > 59) {
F1 Bert score: 0.8924573063850403
Readability: 67.25

Explanation: -15 is less then 0; so it throws IllegalArgumentException
F1 Bert score: 0.8712982535362244
Readability: 62.34

Explanation: You are passing it a negative offset value (-15) and the conditionals are set to reject any offset that is less than 0 or greater than 59 and throw a new exception.
F1 Bert score: 0.873517632484436
Readability: 55.92

Explanation: There is a logical check for if minuteOffset is less than 0 or greater than 59 causing it to throw an exception because the value is out of bounds (negative number)
F1 Bert score: 0.8766863942146301
Readability: 48.47

Explanation: YES. The issue is on line 279 (as I explained in my first question; of which I misunderstood that I was only being asked about the specific issue; not generalized issue). On line 279 the variable "minutesOffSet" is parameterized to throw an exception if it is < 0 or > 59. Line 279 should read "if (minutesOffset < -59 || minutesOffset > 59) {" because now the method can take in the number of minutes as a negative and will allow the method to properly progress to invoke/call further methods such as those asked about in the two previous questions.
F1 Bert score: 0.8827218413352966
Readability: 39.71

Explanation: The variable "minutesOffset" is checked incorrectly by the IF statement on line 279. Any negative value for "minutesOffset" will throw this exception; while the documentation states that "minutesOffset" can be negative in some cases.
F1 Bert score: 0.8805158138275146
Readability: 28.84

Explanations 2: The explanations that pass the semantic similarity threshold sorted by entropy

Explanation: YES. The issue is on line 279 (as I explained in my first question; of which I misunderstood that I was only being asked about the specific issue; not generalized issue). On line 279 the variable "minutesOffSet" is parameterized to throw an exception if it is < 0 or > 59. Line 279 should read "if (minutesOffset < -59 || minutesOffset > 59) {" because now the method can take in the number of minutes as a negative and will allow the method to properly progress to invoke/call further methods such as those asked about in the two previous questions.
Bert F1 score: 0.8827218413352966
Entropy: 6.0020480951008945

Explanation: The variable "minutesOffset" is checked incorrectly by the IF statement on line 279. Any negative value for "minutesOffset" will throw this exception; while the documentation states that "minutesOffset" can be negative in some cases.
Bert F1 score: 0.8805158138275146
Entropy: 4.748940047005428

Explanation: There is a logical check for if minuteOffset is less than 0 or greater than 59 causing it to throw an exception because the value is out of bounds (negative number)
Bert F1 score: 0.8766863942146301
Entropy: 4.7362967135428935

Explanation: You are passing it a negative offset value (-15) and the conditionals are set to reject any offset that is less than 0 or greater than 59 and throw a new exception.
Bert F1 score: 0.873517632484436
Entropy: 4.6875

Explanation: The error is stemming from line 279 because the value of -15 for minutesOffset is < 0. The line should be   if (minutesOffset < -59 || minutesOffset > 59) {
Bert F1 score: 0.8924573063850403
Entropy: 4.481727678869737

Explanation: -15 is less then 0; so it throws IllegalArgumentException
Bert F1 score: 0.8712982535362244
Entropy: 3.169925001442312