Mini project 2

Tony Arnold
Ole Becker

Task 1

Preprocessing:

- normalize explanation duration
- calculate and normalize halstead volume
- create explanation size column
- drop unnecessary columns

Training:

- Use 2 Problems as holdout set
- Train Random Forest Classifier Model
- using Bug types HIT07_33 and HIT01_8 as holdout set

Model

```
clf = RandomForestClassifier()

param_dist = {
    'n_estimators' : randint(50, 1000),
    'max_depth' : [15, 20, 25, 30, 35, 40, 45, 50, None],
    'min_samples_split' : randint(2, 6),
    'criterion' : ['gini', 'entropy']
}

random_search = RandomizedSearchCV(estimator = clf, param_distributions = param_dist, n_iter = 20, cv = 5, scoring= 'f1')
random_search.fit(X_train, y_train)
print("random search best params: ", random_search.best_params_)
best_model= random_search.best_estimator_
```

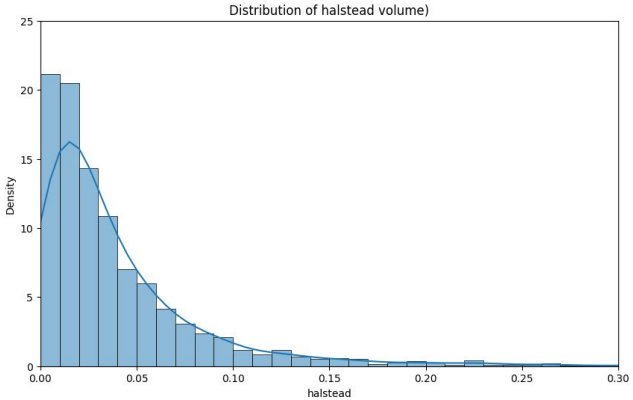
```
random search best params:  {'criterion': 'gini', 'max_depth': 20, 'min_samples_split': 5, 'n_estimators': 698}
```

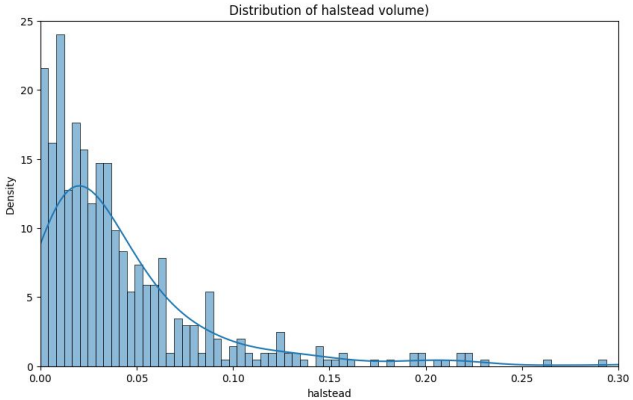
Problem

All data

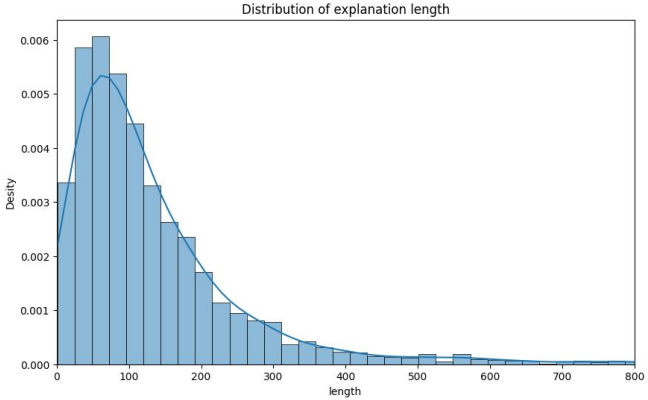
True Labels

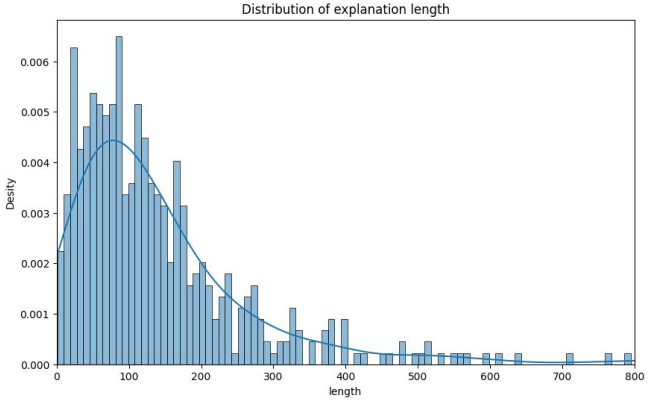
halstead volume





explanation length





Task 2

Prompt 1:

Generate a single explanation by merging the participants' explanations in a way that minimizes redundant information, while keeping the information that would be necessary for someone else to fix the bug. Types of information that, if present in the explanation, should be preserved - how the program works, how the failure is happening, what is problem in the code.

+ followed by all the correct bug type reports

Task 2

Prompt 2:

Prompt 1 + The explanation should be maximum 100 words.

Prompt 3:

Prompt 1 + If you do a really good job at it, I will get a job promotion.

Task 2 - example HIT08_54 prompt 1 result

The bug is caused by an incorrect validation of the locale string format, specifically at line 115. The string is expected to follow a strict format where the third character (ch3) should be a capital letter (A-Z) but is instead an underscore ("_"). This causes the condition `if (ch3 > 'A' || ch3 < 'Z' || ch4 > 'A' || ch4 < 'Z')` to evaluate to true and throw an `IllegalArgumentException`, since the underscore is greater than 'Z'. The validation logic checks that the first two characters should be lowercase letters, followed by an underscore, two uppercase letters, another underscore, and three lowercase letters. In this case, the third character being an underscore causes the exception at line 115.

Additionally, the failure is compounded by a second check around line 121, where the string is expected to have an underscore as the sixth character, but if the input has a character other than an underscore, it will also throw an exception. The `fr__POSIX` input, for example, fails at both line 115 (due to ch3 being an underscore instead of a capital letter) and line 121 (where the fifth character is not an underscore as required).

The solution is to adjust the logic to handle cases where underscores are valid characters in the country code positions, or to ensure that the locale string strictly follows the expected format.

Task 2 - example HIT08_54 prompt 2 result

The bug occurs when the program checks the third character of the locale string in line 115. The locale format expects two lowercase characters, followed by an underscore, two uppercase letters, another underscore, and three lowercase letters. However, the third character (ch3) is an underscore, which is outside the 'A'-'Z' range, causing the condition ch3 > 'A' || ch3 < 'Z' to evaluate as true and throw an IllegalArgumentException. Additionally, line 121 checks for an underscore as the sixth character, but in the case of the input 'fr__POSIX', this also causes an exception due to the incorrect format.

Task 2 - example HIT08_54 prompt 3 result

The bug arises due to the input format validation in the program. The string format expected by the program is a locale string in the format cc__CCCCC, where cc are two lowercase letters, followed by an underscore, then two uppercase letters, another underscore, and three lowercase letters. The issue occurs with the input fr__POSIX, which is considered invalid.

The primary problem is in the check on line 115: if (ch3 > 'A' || ch3 < 'Z' || ch4 > 'A' || ch4 < 'Z'). This condition is meant to validate that the characters ch3 and ch4 (the third and fourth characters of the input string) are uppercase letters. However, when the input string has an underscore as the third character (ch3 = '_'), this condition fails because the ASCII value of '_' is greater than 'Z'. The check incorrectly considers this as a valid condition, resulting in an exception being thrown.

Additionally, line 121 checks if the sixth character is an underscore, and in the input fr__POSIX, the sixth character is not an underscore, leading to another exception. The program's logic for validating input format expects strict adherence to the format, and any deviation, such as an underscore where a capital letter is expected, causes the failure.

To fix this issue, the validation on line 115 needs to account for the possibility of underscores in the locale string. Specifically, the program should handle cases where the third or fourth characters are underscores (or potentially other characters that deviate from the strict format), instead of incorrectly throwing an exception for valid input cases like this one. Additionally, the validation on line 121 should ensure that it properly checks for the correct placement of underscores based on the length and structure of the string.

The failure occurs because the third character (an underscore) falls outside the acceptable range of uppercase letters, triggering the exception in the condition. It would be helpful to revise the error messages to provide more specific

Task 2: comparison of ChatGPT and correct bug reports (using fmeasure for rouge)

Column 1	Column 2	HIT01_8	HIT02_24	HIT03_6	HIT04_7	HIT05_35	HIT06_51	# HIT07_33	HIT08_54
Prompt 1	BLEU	0.16	0.12	0.08	0.15	0.14	0.00	0.18	0.20
	rouge1	0.30	0.49	0.37	0.25	0.34	0.46	0.42	0.33
	rouge2	0.13	0.20	0.16	0.10	0.17	0.15	0.20	0.18
	rougeL	0.17	0.23	0.16	0.13	0.17	0.20	0.21	0.17
Prompt 2	BLEU	0.18	0.11	0.07	0.14	0.23	0.00	0.10	0.24
	rouge1	0.15	0.25	0.14	0.12	0.14	0.24	0.24	0.17
	rouge2	0.07	0.11	0.06	0.04	0.08	0.09	0.12	0.10
	rougeL	0.10	0.15	0.09	0.07	0.09	0.18	0.16	0.11
Prompt 3	BLEU	0.14	0.06	0.07	0.10	0.19	0.00	0.13	0.13
	rouge1	0.28	0.52	0.34	0.34	0.42	0.44	0.39	0.41
	rouge2	0.11	0.20	0.13	0.14	0.22	0.13	0.16	0.17
	rougeL	0.16	0.24	0.15	0.16	0.20	0.21	0.21	0.18

Reflection for automation

- The model should prioritize high precision and acceptable recall

- prompts with a consistent output template for debugging descriptions to ensure uniform and expected results