

AI Assisted Coding LAB ASSIGNMENT-5.4

NAME: D.Nithin

Batch:13

Hall Ticket:2303A51845

Task Description #1:

- Prompt GitHub Copilot to generate a Python script that collects user data (e.g., name, age, email). Then, ask Copilot to add comments on how to anonymize or protect this data.

PROMPT

```
# Generate a Python script that collects user data such as  
name, age, and email.
```

```
# Add inline comments explaining how to protect or  
anonymize this data,
```

```
# such as hashing email addresses, avoiding plain-text  
storage,
```

```
# and following basic privacy best practices.
```

The screenshot shows a GitHub repository named "user-data-collection" with a file "user_data_gathering.py". The code implements various security measures:

- Hashing passwords using SHA-256.
- Validating user input for age and email.
- Using environment variables for sensitive data.
- Minimizing data collection by asking for only what's necessary.
- Ensuring secure user data collection.
- Collecting name and validating age.
- Collecting email and validating it.
- Collecting password and validating its length.
- Creating a user record with minimal sensitive data.

The repository also includes a "README.md" file with instructions and a "LICENSE" file.

The screenshot shows a developer's environment with two code editors and a terminal window.

Code Editors:

- Left Editor:** Displays Python code for a user data manager. It includes methods for adding users, displaying user data, and saving user data securely. The code uses SHA-256 hashing for password storage and includes annotations for security best practices like password complexity and secure file handling.
- Right Editor:** Displays a Python script for user data collection, annotated with security best practices such as using secure file formats and avoiding plain-text storage.

Terminal:

- Shows the command `python3 user_data_collector.py` being run.
- Output indicates the creation of a file named `user_data_collected.txt`.
- Annotations explain the use of `argparse` for command-line arguments and the use of `argparse` and `argon2` for password hashing.

The screenshot shows a dual-monitor setup. The left monitor displays a code editor with Python code for user data privacy, while the right monitor shows a terminal window.

Left Monitor (Code Editor):

```
EDITOR
> HTML_TUTORIALS
  |- PROJECTS
  |- DOCUMENTATION
  > JAVA_PROJECTS
    > HTML_Tutorials

--> file:///C:/Users/Sneha/Desktop/PyAss/  user_data_privacy.py <-- user_data_privacy.py | main
  001 import json
  002 import os
  003 import secrets
  004 import string
  005
  006 class UserDatabase:
  007     def __init__(self):
  008         self.users = {}
  009
  010     def add_user(self, user_id):
  011         if user_id in self.users:
  012             print("User ID already exists!")
  013         else:
  014             self.users[user_id] = {
  015                 "name": secrets.token_hex(8),
  016                 "age": secrets.choice(range(18, 65)),
  017                 "email": f"{secrets.token_hex(4)}@{secrets.choice(['gmail', 'outlook', 'yahoo'])}.com",
  018                 "password": secrets.token_hex(16)
  019             }
  020             print(f"User {user_id} added successfully!")
  021
  022     def get_user(self, user_id):
  023         if user_id in self.users:
  024             return self.users[user_id]
  025         else:
  026             print("User ID not found")
  027             return None
  028
  029     def update_user(self, user_id, new_name):
  030         if user_id in self.users:
  031             self.users[user_id]["name"] = new_name
  032             print(f"User {user_id} updated successfully!")
  033         else:
  034             print("User ID not found")
  035
  036     def delete_user(self, user_id):
  037         if user_id in self.users:
  038             del self.users[user_id]
  039             print(f"User {user_id} deleted successfully!")
  040         else:
  041             print("User ID not found")
  042
  043     def save_users(self):
  044         with open("users.json", "w") as f:
  045             json.dump(self.users, f)
  046
  047     def load_users(self):
  048         try:
  049             with open("users.json", "r") as f:
  050                 self.users = json.load(f)
  051         except FileNotFoundError:
  052             print("No users found in storage")
  053
  054     def __str__(self):
  055         return str(self.users)
  056
  057
  058 def main():
  059     """Main function demonstrating privacy-first user data handling"""
  060
  061     print("*" * 40)
  062     print("USER DATA COLLECTION WITH PRIVACY PROTECTION")
  063     print("*" * 40)
  064     print("This script demonstrates privacy best practices:")
  065     print("- Masking sensitive data (names, passwords)")
  066     print("- Generating unique user IDs")
  067     print("- Input validation and sanitization")
  068     print("- Secure file permissions")
  069     print("- Implementing data collection principles")
  070     print("*" * 40)
  071
  072     manager = UserDatabase()
  073
  074     # Add some initial users
  075     manager.add_user("user_01")
  076     manager.add_user("user_02")
  077
  078     print("\nAvailable Options:")
  079     print("1. Add new user")
  080     print("2. View anonymized data")
  081     print("3. Update user securely")
  082     print("4. Delete user")
  083
  084     choice = input("Select option (1-4): ")
  085
  086     if choice == "1":
  087         name = input("Enter name: ")
  088         age = int(input("Enter age: "))
  089         email = input("Enter email: ")
  090         password = input("Enter password: ")
  091
  092         manager.add_user("user_03")
  093         manager.save_users_securely()
  094
  095     elif choice == "2":
  096         manager.display_user_data_anonymized()
  097
  098     elif choice == "3":
  099         name = input("Enter name: ")
 100         age = int(input("Enter age: "))
 101         email = input("Enter email: ")
 102
 103         manager.update_user("user_01", name)
 104         manager.save_users_securely()
 105
 106     elif choice == "4":
 107         user_id = input("Enter user ID: ")
 108
 109         manager.delete_user(user_id)
 110         manager.save_users_securely()
 111
 112     else:
 113         print("Error: Invalid option. Please select 1-4.")
 114
 115
 116 if __name__ == "__main__":
 117     main()
```

Right Monitor (Terminal):

```
C:\> BLACKBOX <-- C:\>
CHW + - ×
← → ⌂ user_data_privacy.py <-- user_data_privacy.py | main
Detected when to build next
Ansys 2023 R1 Ansys 2023 R1
```

The terminal window shows the command to run the script and indicates that it has detected when to build the next file.

Expected Output #1:

- A script with inline Copilot-suggested code and comments explaining how to safeguard or anonymize user information (e.g., hashing emails, not storing data unencrypted).

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + - | X powers... Python

Select option (1-4): & C:/Users/Sreeshma/AppData/Local/Programs/Python/Python310/python.exe "c:/Users/Sreeshma/Documents/AI ASS/user_data_privacy.py"
ERROR: Invalid option. Please select 1-4.

--- OPTIONS ---
1. Add new user
2. View anonymized data
3. Save data securely
4. Exit

Select option (1-4): 1

== Secure User Data Collection ==
Enter your name (will be stored): Nitish
Enter your age: 20
Enter your email (will be hashed for privacy): nitishrajkond@gmail.com
Enter a password (hidden for security):
ERROR: Password must be at least 8 characters.

--- OPTIONS ---
1. Add new user
2. View anonymized data
3. Save data securely
4. Exit
```

Task Description #2:

- Ask Copilot to generate a Python function for sentiment analysis.

Then prompt Copilot to identify and handle potential biases in the data.

PROMPT: # Generate a Python function for sentiment analysis.

Add comments or code to identify and reduce potential biases in the data,

such as removing offensive terms, balancing positive and negative samples,

and avoiding biased language in predictions.

```
INTERNAL TUTORIALS
PROJECTS
DOCUMENTATION
JAVA PROJECTS
JAVA TUTORIALS

D - kmeans

C:\Users\steveh\Documents\Java\kmeans\testSentimentAnalysis.java % SimpleSentimentAnalyzer % @_ad_
1 - "Simple sentiment analysis with k-means clustering"
2
3 run collections report Counter
4
5 class SimpleSentimentAnalyzer {
6     /*Basic sentiment analyzer with this stringifier*/
7
8     String[] positive_words = {"good", "great", "amazing", "excellent", "happy", "love"};
9     String[] negative_words = {"bad", "terrible", "awful", "hate", "over", "uwf"};
10
11     /* Bias mitigation: offensive/banned terms to remove
12     static String[] ban_terms = {"idiot", "douchebag", "retard"} */
13
14     class Sentence {
15         String text;
16         String label;
17         double[] features;
18     }
19
20     class SentenceList {
21         List<Sentence> sentences;
22         String label;
23     }
24
25     void calculate_mean(List<String> words) {
26         double total_pos = 0;
27         double total_neg = 0;
28         for (String word : words) {
29             if (positive_words.contains(word)) {
30                 total_pos += 1;
31             } else if (negative_words.contains(word)) {
32                 total_neg += 1;
33             }
34         }
35         double mean_pos = total_pos / words.size();
36         double mean_neg = total_neg / words.size();
37         System.out.println("Mean Positive: " + mean_pos);
38         System.out.println("Mean Negative: " + mean_neg);
39     }
40
41     void calculate_mean_sents(List<Sentence> sents) {
42         double total_pos = 0;
43         double total_neg = 0;
44         for (Sentence sent : sents) {
45             if (positive_words.contains(sent.text)) {
46                 total_pos += 1;
47             } else if (negative_words.contains(sent.text)) {
48                 total_neg += 1;
49             }
50         }
51         double mean_pos = total_pos / sents.size();
52         double mean_neg = total_neg / sents.size();
53         System.out.println("Mean Positive: " + mean_pos);
54         System.out.println("Mean Negative: " + mean_neg);
55     }
56
57     void calculate_mean_sents_label(List<SentenceList> sents) {
58         double total_pos = 0;
59         double total_neg = 0;
60         int pos_count = 0;
61         int neg_count = 0;
62         for (SentenceList sent : sents) {
63             if (sent.label.equals("POSITIVE")) {
64                 total_pos += 1;
65                 pos_count++;
66             } else if (sent.label.equals("NEGATIVE")) {
67                 total_neg += 1;
68                 neg_count++;
69             }
70         }
71         double mean_pos = total_pos / sents.size();
72         double mean_neg = total_neg / sents.size();
73         System.out.println("Mean Positive: " + mean_pos);
74         System.out.println("Mean Negative: " + mean_neg);
75         System.out.println("Pos Count: " + pos_count);
76         System.out.println("Neg Count: " + neg_count);
77     }
78
79     void calculate_mean_sents_label(List<SentenceList> sents, String label) {
80         double total_pos = 0;
81         double total_neg = 0;
82         int pos_count = 0;
83         int neg_count = 0;
84         for (SentenceList sent : sents) {
85             if (sent.label.equals(label)) {
86                 total_pos += 1;
87                 pos_count++;
88             } else if (sent.label.equals("NEGATIVE")) {
89                 total_neg += 1;
90                 neg_count++;
91             }
92         }
93         double mean_pos = total_pos / sents.size();
94         double mean_neg = total_neg / sents.size();
95         System.out.println("Mean Positive: " + mean_pos);
96         System.out.println("Mean Negative: " + mean_neg);
97         System.out.println("Pos Count: " + pos_count);
98         System.out.println("Neg Count: " + neg_count);
99     }
100
101     void calculate_mean_sents_label(List<SentenceList> sents, String label, int count) {
102         double total_pos = 0;
103         double total_neg = 0;
104         int pos_count = 0;
105         int neg_count = 0;
106         for (SentenceList sent : sents) {
107             if (sent.label.equals(label)) {
108                 total_pos += 1;
109                 pos_count++;
110             } else if (sent.label.equals("NEGATIVE") & count > 0) {
111                 total_neg += 1;
112                 neg_count++;
113             }
114         }
115         double mean_pos = total_pos / sents.size();
116         double mean_neg = total_neg / sents.size();
117         System.out.println("Mean Positive: " + mean_pos);
118         System.out.println("Mean Negative: " + mean_neg);
119         System.out.println("Pos Count: " + pos_count);
120         System.out.println("Neg Count: " + neg_count);
121     }
122
123     void calculate_mean_sents_label(List<SentenceList> sents, String label, int count, String[] words) {
124         double total_pos = 0;
125         double total_neg = 0;
126         int pos_count = 0;
127         int neg_count = 0;
128         for (SentenceList sent : sents) {
129             if (sent.label.equals(label)) {
130                 total_pos += 1;
131                 pos_count++;
132             } else if (sent.label.equals("NEGATIVE") & count > 0) {
133                 total_neg += 1;
134                 neg_count++;
135             }
136             for (String word : words) {
137                 if (sent.text.contains(word)) {
138                     if (sent.label.equals("POSITIVE")) {
139                         total_pos -= 1;
140                         pos_count--;
141                     } else if (sent.label.equals("NEGATIVE")) {
142                         total_neg -= 1;
143                         neg_count--;
144                     }
145                 }
146             }
147         }
148         double mean_pos = total_pos / sents.size();
149         double mean_neg = total_neg / sents.size();
150         System.out.println("Mean Positive: " + mean_pos);
151         System.out.println("Mean Negative: " + mean_neg);
152         System.out.println("Pos Count: " + pos_count);
153         System.out.println("Neg Count: " + neg_count);
154     }
155
156     void calculate_mean_sents_label(List<SentenceList> sents, String label, int count, String[] words, String[] ban_terms) {
157         double total_pos = 0;
158         double total_neg = 0;
159         int pos_count = 0;
160         int neg_count = 0;
161         for (SentenceList sent : sents) {
162             if (sent.label.equals(label)) {
163                 total_pos += 1;
164                 pos_count++;
165             } else if (sent.label.equals("NEGATIVE") & count > 0) {
166                 total_neg += 1;
167                 neg_count++;
168             }
169             for (String word : words) {
170                 if (sent.text.contains(word)) {
171                     if (sent.label.equals("POSITIVE")) {
172                         total_pos -= 1;
173                         pos_count--;
174                     } else if (sent.label.equals("NEGATIVE")) {
175                         total_neg -= 1;
176                         neg_count--;
177                     }
178                 }
179             }
180             for (String term : ban_terms) {
181                 if (sent.text.contains(term)) {
182                     if (sent.label.equals("POSITIVE")) {
183                         total_pos -= 1;
184                         pos_count--;
185                     } else if (sent.label.equals("NEGATIVE")) {
186                         total_neg -= 1;
187                         neg_count--;
188                     }
189                 }
190             }
191         }
192         double mean_pos = total_pos / sents.size();
193         double mean_neg = total_neg / sents.size();
194         System.out.println("Mean Positive: " + mean_pos);
195         System.out.println("Mean Negative: " + mean_neg);
196         System.out.println("Pos Count: " + pos_count);
197         System.out.println("Neg Count: " + neg_count);
198     }
199
200     void calculate_mean_sents_label(List<SentenceList> sents, String label, int count, String[] words, String[] ban_terms, String[] train_labels) {
201         double total_pos = 0;
202         double total_neg = 0;
203         int pos_count = 0;
204         int neg_count = 0;
205         for (SentenceList sent : sents) {
206             if (sent.label.equals(label)) {
207                 total_pos += 1;
208                 pos_count++;
209             } else if (sent.label.equals("NEGATIVE") & count > 0) {
210                 total_neg += 1;
211                 neg_count++;
212             }
213             for (String word : words) {
214                 if (sent.text.contains(word)) {
215                     if (sent.label.equals("POSITIVE")) {
216                         total_pos -= 1;
217                         pos_count--;
218                     } else if (sent.label.equals("NEGATIVE")) {
219                         total_neg -= 1;
220                         neg_count--;
221                     }
222                 }
223             }
224             for (String term : ban_terms) {
225                 if (sent.text.contains(term)) {
226                     if (sent.label.equals("POSITIVE")) {
227                         total_pos -= 1;
228                         pos_count--;
229                     } else if (sent.label.equals("NEGATIVE")) {
230                         total_neg -= 1;
231                         neg_count--;
232                     }
233                 }
234             }
235             if (train_labels != null) {
236                 if (train_labels.get(sent.id).equals("POSITIVE")) {
237                     total_pos -= 1;
238                     pos_count--;
239                 } else if (train_labels.get(sent.id).equals("NEGATIVE")) {
240                     total_neg -= 1;
241                     neg_count--;
242                 }
243             }
244         }
245         double mean_pos = total_pos / sents.size();
246         double mean_neg = total_neg / sents.size();
247         System.out.println("Mean Positive: " + mean_pos);
248         System.out.println("Mean Negative: " + mean_neg);
249         System.out.println("Pos Count: " + pos_count);
250         System.out.println("Neg Count: " + neg_count);
251     }
252
253     void calculate_mean_sents_label(List<SentenceList> sents, String label, int count, String[] words, String[] ban_terms, String[] train_labels, List<String> train_ids) {
254         double total_pos = 0;
255         double total_neg = 0;
256         int pos_count = 0;
257         int neg_count = 0;
258         for (SentenceList sent : sents) {
259             if (sent.label.equals(label)) {
260                 total_pos += 1;
261                 pos_count++;
262             } else if (sent.label.equals("NEGATIVE") & count > 0) {
263                 total_neg += 1;
264                 neg_count++;
265             }
266             for (String word : words) {
267                 if (sent.text.contains(word)) {
268                     if (sent.label.equals("POSITIVE")) {
269                         total_pos -= 1;
270                         pos_count--;
271                     } else if (sent.label.equals("NEGATIVE")) {
272                         total_neg -= 1;
273                         neg_count--;
274                     }
275                 }
276             }
277             for (String term : ban_terms) {
278                 if (sent.text.contains(term)) {
279                     if (sent.label.equals("POSITIVE")) {
280                         total_pos -= 1;
281                         pos_count--;
282                     } else if (sent.label.equals("NEGATIVE")) {
283                         total_neg -= 1;
284                         neg_count--;
285                     }
286                 }
287             }
288             if (train_ids != null) {
289                 if (train_ids.contains(sent.id)) {
290                     if (train_labels.get(sent.id).equals("POSITIVE")) {
291                         total_pos -= 1;
292                         pos_count--;
293                     } else if (train_labels.get(sent.id).equals("NEGATIVE")) {
294                         total_neg -= 1;
295                         neg_count--;
296                     }
297                 }
298             }
299         }
300         double mean_pos = total_pos / sents.size();
301         double mean_neg = total_neg / sents.size();
302         System.out.println("Mean Positive: " + mean_pos);
303         System.out.println("Mean Negative: " + mean_neg);
304         System.out.println("Pos Count: " + pos_count);
305         System.out.println("Neg Count: " + neg_count);
306     }
307
308     void calculate_mean_sents_label(List<SentenceList> sents, String label, int count, String[] words, String[] ban_terms, String[] train_labels, List<String> train_ids, List<String> test_ids) {
309         double total_pos = 0;
310         double total_neg = 0;
311         int pos_count = 0;
312         int neg_count = 0;
313         for (SentenceList sent : sents) {
314             if (sent.label.equals(label)) {
315                 total_pos += 1;
316                 pos_count++;
317             } else if (sent.label.equals("NEGATIVE") & count > 0) {
318                 total_neg += 1;
319                 neg_count++;
320             }
321             for (String word : words) {
322                 if (sent.text.contains(word)) {
323                     if (sent.label.equals("POSITIVE")) {
324                         total_pos -= 1;
325                         pos_count--;
326                     } else if (sent.label.equals("NEGATIVE")) {
327                         total_neg -= 1;
328                         neg_count--;
329                     }
330                 }
331             }
332             for (String term : ban_terms) {
333                 if (sent.text.contains(term)) {
334                     if (sent.label.equals("POSITIVE")) {
335                         total_pos -= 1;
336                         pos_count--;
337                     } else if (sent.label.equals("NEGATIVE")) {
338                         total_neg -= 1;
339                         neg_count--;
340                     }
341                 }
342             }
343             if (train_ids != null) {
344                 if (train_ids.contains(sent.id)) {
345                     if (train_labels.get(sent.id).equals("POSITIVE")) {
346                         total_pos -= 1;
347                         pos_count--;
348                     } else if (train_labels.get(sent.id).equals("NEGATIVE")) {
349                         total_neg -= 1;
350                         neg_count--;
351                     }
352                 }
353             }
354             if (test_ids != null) {
355                 if (test_ids.contains(sent.id)) {
356                     if (sent.label.equals("POSITIVE")) {
357                         total_pos += 1;
358                         pos_count++;
359                     } else if (sent.label.equals("NEGATIVE")) {
360                         total_neg += 1;
361                         neg_count++;
362                     }
363                 }
364             }
365         }
366         double mean_pos = total_pos / sents.size();
367         double mean_neg = total_neg / sents.size();
368         System.out.println("Mean Positive: " + mean_pos);
369         System.out.println("Mean Negative: " + mean_neg);
370         System.out.println("Pos Count: " + pos_count);
371         System.out.println("Neg Count: " + neg_count);
372     }
373
374     void calculate_mean_sents_label(List<SentenceList> sents, String label, int count, String[] words, String[] ban_terms, String[] train_labels, List<String> train_ids, List<String> test_ids, List<String> balanced_ids) {
375         double total_pos = 0;
376         double total_neg = 0;
377         int pos_count = 0;
378         int neg_count = 0;
379         for (SentenceList sent : sents) {
380             if (sent.label.equals(label)) {
381                 total_pos += 1;
382                 pos_count++;
383             } else if (sent.label.equals("NEGATIVE") & count > 0) {
384                 total_neg += 1;
385                 neg_count++;
386             }
387             for (String word : words) {
388                 if (sent.text.contains(word)) {
389                     if (sent.label.equals("POSITIVE")) {
390                         total_pos -= 1;
391                         pos_count--;
392                     } else if (sent.label.equals("NEGATIVE")) {
393                         total_neg -= 1;
394                         neg_count--;
395                     }
396                 }
397             }
398             for (String term : ban_terms) {
399                 if (sent.text.contains(term)) {
400                     if (sent.label.equals("POSITIVE")) {
401                         total_pos -= 1;
402                         pos_count--;
403                     } else if (sent.label.equals("NEGATIVE")) {
404                         total_neg -= 1;
405                         neg_count--;
406                     }
407                 }
408             }
409             if (train_ids != null) {
410                 if (train_ids.contains(sent.id)) {
411                     if (train_labels.get(sent.id).equals("POSITIVE")) {
412                         total_pos -= 1;
413                         pos_count--;
414                     } else if (train_labels.get(sent.id).equals("NEGATIVE")) {
415                         total_neg -= 1;
416                         neg_count--;
417                     }
418                 }
419             }
420             if (test_ids != null) {
421                 if (test_ids.contains(sent.id)) {
422                     if (sent.label.equals("POSITIVE")) {
423                         total_pos += 1;
424                         pos_count++;
425                     } else if (sent.label.equals("NEGATIVE")) {
426                         total_neg += 1;
427                         neg_count++;
428                     }
429                 }
430             }
431             if (balanced_ids != null) {
432                 if (balanced_ids.contains(sent.id)) {
433                     if (sent.label.equals("POSITIVE")) {
434                         total_pos += 1;
435                         pos_count++;
436                     } else if (sent.label.equals("NEGATIVE")) {
437                         total_neg += 1;
438                         neg_count++;
439                     }
440                 }
441             }
442         }
443         double mean_pos = total_pos / sents.size();
444         double mean_neg = total_neg / sents.size();
445         System.out.println("Mean Positive: " + mean_pos);
446         System.out.println("Mean Negative: " + mean_neg);
447         System.out.println("Pos Count: " + pos_count);
448         System.out.println("Neg Count: " + neg_count);
449     }
450
451     void calculate_mean_sents_label(List<SentenceList> sents, String label, int count, String[] words, String[] ban_terms, String[] train_labels, List<String> train_ids, List<String> test_ids, List<String> balanced_ids, List<String> balanced_labels) {
452         double total_pos = 0;
453         double total_neg = 0;
454         int pos_count = 0;
455         int neg_count = 0;
456         for (SentenceList sent : sents) {
457             if (sent.label.equals(label)) {
458                 total_pos += 1;
459                 pos_count++;
460             } else if (sent.label.equals("NEGATIVE") & count > 0) {
461                 total_neg += 1;
462                 neg_count++;
463             }
464             for (String word : words) {
465                 if (sent.text.contains(word)) {
466                     if (sent.label.equals("POSITIVE")) {
467                         total_pos -= 1;
468                         pos_count--;
469                     } else if (sent.label.equals("NEGATIVE")) {
470                         total_neg -= 1;
471                         neg_count--;
472                     }
473                 }
474             }
475             for (String term : ban_terms) {
476                 if (sent.text.contains(term)) {
477                     if (sent.label.equals("POSITIVE")) {
478                         total_pos -= 1;
479                         pos_count--;
480                     } else if (sent.label.equals("NEGATIVE")) {
481                         total_neg -= 1;
482                         neg_count--;
483                     }
484                 }
485             }
486             if (train_ids != null) {
487                 if (train_ids.contains(sent.id)) {
488                     if (train_labels.get(sent.id).equals("POSITIVE")) {
489                         total_pos -= 1;
490                         pos_count--;
491                     } else if (train_labels.get(sent.id).equals("NEGATIVE")) {
492                         total_neg -= 1;
493                         neg_count--;
494                     }
495                 }
496             }
497             if (test_ids != null) {
498                 if (test_ids.contains(sent.id)) {
499                     if (sent.label.equals("POSITIVE")) {
500                         total_pos += 1;
501                         pos_count++;
502                     } else if (sent.label.equals("NEGATIVE")) {
503                         total_neg += 1;
504                         neg_count++;
505                     }
506                 }
507             }
508             if (balanced_labels != null) {
509                 if (balanced_labels.get(sent.id).equals("POSITIVE")) {
510                     total_pos += 1;
511                     pos_count++;
512                 } else if (balanced_labels.get(sent.id).equals("NEGATIVE")) {
513                     total_neg += 1;
514                     neg_count++;
515                 }
516             }
517         }
518         double mean_pos = total_pos / sents.size();
519         double mean_neg = total_neg / sents.size();
520         System.out.println("Mean Positive: " + mean_pos);
521         System.out.println("Mean Negative: " + mean_neg);
522         System.out.println("Pos Count: " + pos_count);
523         System.out.println("Neg Count: " + neg_count);
524     }
525
526     void calculate_mean_sents_label(List<SentenceList> sents, String label, int count, String[] words, String[] ban_terms, String[] train_labels, List<String> train_ids, List<String> test_ids, List<String> balanced_ids, List<String> balanced_labels, List<String> balanced_labels_ids) {
527         double total_pos = 0;
528         double total_neg = 0;
529         int pos_count = 0;
530         int neg_count = 0;
531         for (SentenceList sent : sents) {
532             if (sent.label.equals(label)) {
533                 total_pos += 1;
534                 pos_count++;
535             } else if (sent.label.equals("NEGATIVE") & count > 0) {
536                 total_neg += 1;
537                 neg_count++;
538             }
539             for (String word : words) {
540                 if (sent.text.contains(word)) {
541                     if (sent.label.equals("POSITIVE")) {
542                         total_pos -= 1;
543                         pos_count--;
544                     } else if (sent.label.equals("NEGATIVE")) {
545                         total_neg -= 1;
546                         neg_count--;
547                     }
548                 }
549             }
550             for (String term : ban_terms) {
551                 if (sent.text.contains(term)) {
552                     if (sent.label.equals("POSITIVE")) {
553                         total_pos -= 1;
554                         pos_count--;
555                     } else if (sent.label.equals("NEGATIVE")) {
556                         total_neg -= 1;
557                         neg_count--;
558                     }
559                 }
560             }
561             if (train_ids != null) {
562                 if (train_ids.contains(sent.id)) {
563                     if (train_labels.get(sent.id).equals("POSITIVE")) {
564                         total_pos -= 1;
565                         pos_count--;
566                     } else if (train_labels.get(sent.id).equals("NEGATIVE")) {
567                         total_neg -= 1;
568                         neg_count--;
569                     }
570                 }
571             }
572             if (test_ids != null) {
573                 if (test_ids.contains(sent.id)) {
574                     if (sent.label.equals("POSITIVE")) {
575                         total_pos += 1;
576                         pos_count++;
577                     } else if (sent.label.equals("NEGATIVE")) {
578                         total_neg += 1;
579                         neg_count++;
580                     }
581                 }
582             }
583             if (balanced_labels_ids != null) {
584                 if (balanced_labels_ids.contains(sent.id)) {
585                     if (balanced_labels.get(sent.id).equals("POSITIVE")) {
586                         total_pos += 1;
587                         pos_count++;
588                     } else if (balanced_labels.get(sent.id).equals("NEGATIVE")) {
589                         total_neg += 1;
590                         neg_count++;
591                     }
592                 }
593             }
594         }
595         double mean_pos = total_pos / sents.size();
596         double mean_neg = total_neg / sents.size();
597         System.out.println("Mean Positive: " + mean_pos);
598         System.out.println("Mean Negative: " + mean_neg);
599         System.out.println("Pos Count: " + pos_count);
600         System.out.println("Neg Count: " + neg_count);
601     }
602
603     void calculate_mean_sents_label(List<SentenceList> sents, String label, int count, String[] words, String[] ban_terms, String[] train_labels, List<String> train_ids, List<String> test_ids, List<String> balanced_ids, List<String> balanced_labels, List<String> balanced_labels_ids, List<String> balanced_labels_ids_ids) {
604         double total_pos = 0;
605         double total_neg = 0;
606         int pos_count = 0;
607         int neg_count = 0;
608         for (SentenceList sent : sents) {
609             if (sent.label.equals(label)) {
610                 total_pos += 1;
611                 pos_count++;
612             } else if (sent.label.equals("NEGATIVE") & count > 0) {
613                 total_neg += 1;
614                 neg_count++;
615             }
616             for (String word : words) {
617                 if (sent.text.contains(word)) {
618                     if (sent.label.equals("POSITIVE")) {
619                         total_pos -= 1;
620                         pos_count--;
621                     } else if (sent.label.equals("NEGATIVE")) {
622                         total_neg -= 1;
623                         neg_count--;
624                     }
625                 }
626             }
627             for (String term : ban_terms) {
628                 if (sent.text.contains(term)) {
629                     if (sent.label.equals("POSITIVE")) {
630                         total_pos -= 1;
631                         pos_count--;
632                     } else if (sent.label.equals("NEGATIVE")) {
633                         total_neg -= 1;
634                         neg_count--;
635                     }
636                 }
637             }
638             if (train_ids != null) {
639                 if (train_ids.contains(sent.id)) {
640                     if (train_labels.get(sent.id).equals("POSITIVE")) {
641                         total_pos -= 1;
642                         pos_count--;
643                     } else if (train_labels.get(sent.id).equals("NEGATIVE")) {
644                         total_neg -= 1;
645                         neg_count--;
646                     }
647                 }
648             }
649             if (test_ids != null) {
650                 if (test_ids.contains(sent.id)) {
651                     if (sent.label.equals("POSITIVE")) {
652                         total_pos += 1;
653                         pos_count++;
654                     } else if (sent.label.equals("NEGATIVE")) {
655                         total_neg += 1;
656                         neg_count++;
657                     }
658                 }
659             }
660             if (balanced_labels_ids_ids != null) {
661                 if (balanced_labels_ids_ids.contains(sent.id)) {
662                     if (balanced_labels.get(sent.id).equals("POSITIVE")) {
663                         total_pos += 1;
664                         pos_count++;
665                     } else if (balanced_labels.get(sent.id).equals("NEGATIVE")) {
666                         total_neg += 1;
667                         neg_count++;
668                     }
669                 }
670             }
671         }
672         double mean_pos = total_pos / sents.size();
673         double mean_neg = total_neg / sents.size();
674         System.out.println("Mean Positive: " + mean_pos);
675         System.out.println("Mean Negative: " + mean_neg);
676         System.out.println("Pos Count: " + pos_count);
677         System.out.println("Neg Count: " + neg_count);
678     }
679
680     void calculate_mean_sents_label(List<SentenceList> sents, String label, int count, String[] words, String[] ban_terms, String[] train_labels, List<String> train_ids, List<String> test_ids, List<String> balanced_ids, List<String> balanced_labels, List<String> balanced_labels_ids, List<String> balanced_labels_ids_ids) {
681         double total_pos = 0;
682         double total_neg = 0;
683         int pos_count = 0;
684         int neg_count = 0;
685         for (SentenceList sent : sents) {
686             if (sent.label.equals(label)) {
687                 total_pos += 1;
688                 pos_count++;
689             } else if (sent.label.equals("NEGATIVE") & count > 0) {
690                 total_neg += 1;
691                 neg_count++;
692             }
693             for (String word : words) {
694                 if (sent.text.contains(word)) {
695                     if (sent.label.equals("POSITIVE")) {
696                         total_pos -= 1;
697                         pos_count--;
698                     } else if (sent.label.equals("NEGATIVE")) {
699                         total_neg -= 1;
700                         neg_count--;
701                     }
702                 }
703             }
704             for (String term : ban_terms) {
705                 if (sent.text.contains(term)) {
706                     if (sent.label.equals("POSITIVE")) {
707                         total_pos -= 1;
708                         pos_count--;
709                     } else if (sent.label.equals("NEGATIVE")) {
710                         total_neg -= 1;
711                         neg_count--;
712                     }
713                 }
714             }
715             if (train_ids != null) {
716                 if (train_ids.contains(sent.id)) {
717                     if (train_labels.get(sent.id).equals("POSITIVE")) {
718                         total_pos -= 1;
719                         pos_count--;
720                     } else if (train_labels.get(sent.id).equals("NEGATIVE")) {
721                         total_neg -= 1;
722                         neg_count--;
723                     }
724                 }
725             }
726             if (test_ids != null) {
727                 if (test_ids.contains(sent.id)) {
728                     if (sent.label.equals("POSITIVE")) {
729                         total_pos += 1;
730                         pos_count++;
731                     } else if (sent.label.equals("NEGATIVE")) {
732                         total_neg += 1;
733                         neg_count++;
734                     }
735                 }
736             }
737             if (balanced_labels_ids_ids != null) {
738                 if (balanced_labels_ids_ids.contains(sent.id)) {
739                     if (balanced_labels.get(sent.id).equals("POSITIVE")) {
740                         total_pos += 1;
741                         pos_count++;
742                     } else if (balanced_labels.get(sent.id).equals("NEGATIVE")) {
743                         total_neg += 1;
744                         neg_count++;
745                     }
746                 }
747             }
748         }
749         double mean_pos = total_pos / sents.size();
750         double mean_neg = total_neg / sents.size();
751         System.out.println("Mean Positive: " + mean_pos);
752         System.out.println("Mean Negative: " + mean_neg);
753         System.out.println("Pos Count: " + pos_count);
754         System.out.println("Neg Count: " + neg_count);
755     }
756
757     void calculate_mean_sents_label(List<SentenceList> sents, String label, int count, String[] words, String[] ban_terms, String[] train_labels, List<String> train_ids, List<String> test_ids, List<String> balanced_ids, List<String> balanced_labels, List<String> balanced_labels_ids, List<String> balanced_labels_ids_ids, List<String> balanced_labels_ids_ids_ids) {
758         double total_pos = 0;
759         double total_neg = 0;
760         int pos_count = 0;
761         int neg_count = 0;
762         for (SentenceList sent : sents) {
763             if (sent.label.equals(label)) {
764                 total_pos += 1;
765                 pos_count++;
766             } else if (sent.label.equals("NEGATIVE") & count > 0) {
767                 total_neg += 1;
768                 neg_count++;
769             }
770             for (String word : words) {
771                 if (sent.text.contains(word)) {
772                     if (sent.label.equals("POSITIVE")) {
773                         total_pos -= 1;
774                         pos_count--;
775                     } else if (sent.label.equals("NEGATIVE")) {
776                         total_neg -= 1;
777                         neg_count--;
778                     }
779                 }
780             }
781             for (String term : ban_terms) {
782                 if (sent.text.contains(term)) {
783                     if (sent.label.equals("POSITIVE")) {
784                         total_pos -= 1;
785                         pos_count--;
786                     } else if (sent.label.equals("NEGATIVE")) {
787                         total_neg -= 1;
788                         neg_count--;
789                     }
790                 }
791             }
792             if (train_ids != null) {
793                 if (train_ids.contains(sent.id)) {
794                     if (train_labels.get(sent.id).equals("POSITIVE")) {
795                         total_pos -= 1;
796                         pos_count--;
797                     } else if (train_labels.get(sent.id).equals("NEGATIVE")) {
798                         total_neg -= 1;
799                         neg_count--;
800                     }
801                 }
802             }
803             if (test_ids != null) {
804                 if (test_ids.contains(sent.id)) {
805                     if (sent.label.equals("POSITIVE")) {
806                         total_pos += 1;
807                         pos_count++;
808                     } else if (sent.label.equals("NEGATIVE")) {
809                         total_neg += 1;
810                         neg_count++;
811                     }
812                 }
813             }
814             if (balanced_labels_ids_ids_ids != null) {
815                 if (balanced_labels_ids_ids_ids.contains(sent.id)) {
816                     if (balanced_labels.get(sent.id).equals("POSITIVE")) {
817                         total_pos += 1;
818                         pos_count++;
819                     } else if (balanced_labels.get(sent.id).equals("NEGATIVE")) {
820                         total_neg += 1;
821                         neg_count++;
822                     }
823                 }
824             }
825         }
826         double mean_pos = total_pos / sents.size();
827         double mean_neg = total_neg / sents.size();
828         System.out.println("Mean Positive: " + mean_pos);
829         System.out.println("Mean Negative: " + mean_neg);
830         System.out.println("Pos Count: " + pos_count);
831         System.out.println("Neg Count: " + neg_count);
832     }
833
834     void calculate_mean_sents_label(List<SentenceList> sents, String label, int count, String[] words, String[] ban_terms, String[] train_labels, List<String> train_ids, List<String> test_ids, List<String> balanced_ids, List<String> balanced_labels, List<String> balanced_labels_ids, List<String> balanced_labels_ids_ids, List<String> balanced_labels_ids_ids_ids, List<String> balanced_labels_ids_ids_ids_ids) {
835         double total_pos = 0;
836         double total_neg = 0;
837         int pos_count = 0;
838         int neg_count = 0;
839         for (SentenceList sent : sents) {
840             if (sent.label.equals(label)) {
841                 total_pos += 1;
842                 pos_count++;
843             } else if (sent.label.equals("NEGATIVE") & count > 0) {
844                 total_neg += 1;
845                 neg_count++;
846             }
847             for (String word : words) {
848                 if (sent.text.contains(word)) {
849                     if (sent.label.equals("POSITIVE")) {
850                         total_pos -= 1;
851                         pos_count--;
852                     } else if (sent.label.equals("NEGATIVE")) {
853                         total_neg -= 1;
854                         neg_count--;
855                     }
856                 }
857             }
858             for (String term : ban_terms) {
859                 if (sent.text.contains(term)) {
860                     if (sent.label.equals("POSITIVE")) {
861                         total_pos -= 1;
862                         pos_count--;
863                     } else if (sent.label.equals("NEGATIVE")) {
864                         total_neg -= 1;
865                         neg_count--;
866                     }
867                 }
868             }
869             if (train_ids != null) {
870                 if (train_ids.contains(sent.id)) {
871                     if (train_labels.get(sent.id).equals("POSITIVE")) {
872                         total_pos -= 1;
873                         pos_count--;
874                     } else if (train_labels.get(sent.id).equals("NEGATIVE")) {
875                         total_neg -= 1;
876                         neg_count--;
877                     }
878                 }
879             }
880             if (test_ids != null) {
881                 if (test_ids.contains(sent.id)) {
882                     if (sent.label.equals("POSITIVE")) {
883                         total_pos += 1;
884                         pos_count++;
885                     } else if (sent.label.equals("NEGATIVE")) {
886                         total_neg += 1;
887                         neg_count++;
888                     }
889                 }
890             }
891             if (balanced_labels_ids_ids_ids_ids != null) {
892                 if (balanced_labels_ids_ids_ids_ids.contains(sent.id)) {
893                     if (balanced_labels.get(sent.id).equals("POSITIVE")) {
894                         total_pos += 1;
895                         pos_count++;
896                     } else if (balanced_labels.get(sent.id).equals("NEGATIVE")) {
897                         total_neg += 1;
898                         neg_count++;
899                     }
900                 }
901             }
902         }
903         double mean_pos = total_pos / sents.size();
904         double mean_neg = total_neg / sents.size();
905         System.out.println("Mean Positive: " + mean_pos);
906         System.out.println("Mean Negative: " + mean_neg);
907         System.out.println("Pos Count: " + pos_count);
908         System.out.println("Neg Count: " + neg_count);
909     }
910
911     void calculate_mean_sents_label(List<SentenceList> sents, String label, int count, String[] words, String[] ban_terms, String[] train_labels, List<String> train_ids, List<String> test_ids, List<String> balanced_ids, List<String> balanced_labels, List<String> balanced_labels_ids, List<String> balanced_labels_ids_ids, List<String> balanced_labels_ids_ids_ids, List<String> balanced_labels_ids_ids_ids_ids, List<String> balanced_labels_ids_ids_ids_ids_ids) {
912         double total_pos = 0;
913         double total_neg = 0;
914         int pos_count = 0;
915         int neg_count = 0;
916         for (SentenceList sent : sents) {
917             if (sent.label.equals(label)) {
918                 total_pos += 1;
919                 pos_count++;
920             } else if (sent.label.equals("NEGATIVE") & count > 0) {
921                 total_neg += 1;
922                 neg_count++;
923             }
924             for (String word : words) {
925                 if (sent.text.contains(word)) {
926                     if (sent.label.equals("POSITIVE")) {
927                         total_pos -= 1;
928                         pos_count--;
929                     } else if (sent.label.equals("NEGATIVE")) {
930                         total_neg -= 1;
931                         neg_count--;
932                     }
933                 }
934             }
935             for (String term : ban_terms) {
936                 if (sent.text.contains(term)) {
937                     if (sent.label.equals("POSITIVE")) {
938                         total_pos -= 1;
939                         pos_count--;
940                     } else if (sent.label.equals("NEGATIVE")) {
941                         total_neg -= 1;
942                         neg_count--;
943                     }
944                 }
945             }
946             if (train_ids != null) {
947                 if (train_ids.contains(sent.id)) {
948                     if (train_labels.get(sent.id).equals("POSITIVE")) {
949                         total_pos -= 1;
950                         pos_count--;
951                     } else if (train_labels.get(sent.id).equals("NEGATIVE")) {
952                         total_neg -= 1;
953                         neg_count--;
954                     }
955                 }
956             }
957             if (test_ids != null) {
958                 if (test_ids.contains(sent.id)) {
959                     if (sent.label.equals("POSITIVE")) {
960                         total_pos += 1;
961                         pos_count++;
962                     } else if (sent.label.equals("NEGATIVE")) {
963                         total_neg += 1;
964                         neg_count++;
965                     }
966                 }
967             }
968             if (balanced_labels_ids_ids_ids_ids_ids != null) {
969                 if (balanced_labels_ids_ids_ids_ids_ids.contains(sent.id)) {
970                     if (balanced_labels.get(sent.id).equals("POSITIVE")) {
971                         total_pos += 1;
972                         pos_count++;
973                     } else if (balanced_labels.get(sent.id).equals("NEGATIVE")) {
974                         total_neg += 1;
975                         neg_count++;
976                     }
977                 }
978             }
979         }
980         double mean_pos = total_pos / sents.size();
981         double mean_neg = total_neg / sents.size();
982         System.out.println("Mean Positive: " + mean_pos);
983         System.out.println("Mean Negative: " + mean_neg);
984         System.out.println("Pos Count: " + pos_count);
985         System.out.println("Neg Count: " + neg_count);
986     }
987
988     void calculate_mean_sents_label(List<SentenceList> sents, String label, int count, String[] words, String[] ban_terms, String[] train_labels, List<String> train_ids, List<String> test_ids, List<String> balanced_ids, List<String> balanced_labels, List<String> balanced_labels_ids, List<String> balanced_labels_ids_ids, List<String> balanced_labels_ids_ids_ids, List<String> balanced_labels_ids_ids_ids_ids, List<String> balanced_labels_ids_ids_ids_ids_ids) {
989         double total_pos = 0;
990         double total_neg = 0;
991         int pos_count = 0;
992         int neg_count = 0;
993         for (SentenceList sent : sents) {
994             if (sent.label.equals(label)) {
995                 total_pos += 1;
996                 pos_count++;
997             } else if (sent.label.equals("NEGATIVE") & count > 0) {
998                 total_neg += 1;
999                 neg_count++;
1000            }
1001        }
1002        double mean_pos = total_pos / sents.size();
1003        double mean_neg = total_neg / sents.size();
1004        System.out.println("Mean Positive: " + mean_pos);
1005        System.out.println("Mean Negative: " + mean_neg);
1006        System.out.println("Pos Count: " + pos_count);
1007        System.out.println("Neg Count: " + neg_count);
1008    }
1009
1010    void analyze() {
1011        SimpleSentimentAnalyzer analyzer = new SimpleSentimentAnalyzer();
1012        analyzer.analyze();
1013        System.out.println("Analyzed!");
1014    }
1015
1016    void analyze(String text) {
1017        SimpleSentimentAnalyzer analyzer = new SimpleSentimentAnalyzer();
1018        analyzer.analyze(text);
1019        System.out.println("Analyzed!");
1020    }
1021
1022    void analyze(String text, String label) {
1023        SimpleSentimentAnalyzer analyzer = new SimpleSentimentAnalyzer();
1024        analyzer.analyze(text, label);
1025        System.out.println("Analyzed!");
1026    }
1027
1028    void analyze(String text, String label, String[] words) {
1029        SimpleSentimentAnalyzer analyzer = new SimpleSentimentAnalyzer();
1030        analyzer.analyze(text, label, words);
1031        System.out.println("Analyzed!");
1032    }
1033
1034    void analyze(String text, String label, String[] words, String[] ban_terms) {
1035        SimpleSentimentAnalyzer analyzer = new SimpleSentimentAnalyzer();
1036        analyzer.analyze(text, label, words, ban_terms);
1037        System.out.println("Analyzed!");
1038    }
1039
1040    void analyze(String text, String label, String[] words, String[] ban_terms, String[] train_labels) {
1041        SimpleSentimentAnalyzer analyzer = new SimpleSentimentAnalyzer();
1042        analyzer.analyze(text, label, words, ban_terms, train_labels);
1043        System.out.println("Analyzed!");
1044    }
1045
1046    void analyze(String text, String label, String[] words, String[] ban_terms, String[] train_labels, List<String> train_ids) {
1047        SimpleSentimentAnalyzer analyzer = new SimpleSentimentAnalyzer();
1048        analyzer.analyze(text, label, words, ban_terms, train_labels, train_ids);
1049        System.out.println("Analyzed!");
1050    }
1051
1052    void analyze(String text, String label, String[] words, String[] ban_terms, String[] train_labels, List<String> train_ids, List<String> test_ids) {
1053        SimpleSentimentAnalyzer analyzer = new SimpleSentimentAnalyzer();
1054        analyzer.analyze(text, label, words, ban_terms, train_labels, train_ids, test_ids);
1055        System.out.println("Analyzed!");
1056    }
1057
1058    void analyze(String text, String label, String[] words, String[] ban_terms, String[] train_labels, List<String> train_ids, List<String> test_ids, List<String> balanced_ids) {
1059        SimpleSentimentAnalyzer analyzer = new SimpleSentimentAnalyzer();
1060        analyzer.analyze(text, label, words, ban_terms, train_labels, train_ids, test_ids, balanced_ids);
1061        System.out.println("Analyzed!");
1062    }
1063
1064    void analyze(String text, String label, String[] words, String[] ban_terms, String[] train_labels, List<String> train_ids, List<String> test_ids, List<String> balanced_ids, List<String> balanced_labels
```

Expected Output #2:

- Copilot-generated code with additions or comments addressing bias mitigation strategies (e.g., balancing dataset, removing offensive terms).

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  Python +    
```

Text: It's okay, nothing special.
Result: 0

```
PS C:\Users\Sreeshma\Downloads\HTML Tutorials & C:\Users\Sreeshma\AppData\Local\Programs\Python\Python310\python.exe "c:/Users/Sreeshma/Documents/AI ASS/sentiment_analysis_bias.py"  
--- Sentiment Analysis ---  
Text: This product is amazing and excellent!  
Result: {'text': 'This product is amazing and excellent!', 'score': 1.0, 'label': 'POSITIVE'}
```

Text: I hate this, it's terrible.
Result: {'text': "I hate this, it's terrible.", 'score': -1.0, 'label': 'NEGATIVE'}

```
PS C:\Users\Sreeshma\Downloads\HTML Tutorials & C:\Users\Sreeshma\AppData\Local\Programs\Python\Python310\python.exe "c:/Users/Sreeshma/Documents/AI ASS/sentiment_analysis_bias.py"  
--- Sentiment Analysis ---  
Text: This product is amazing and excellent!  
Result: {'text': 'This product is amazing and excellent!', 'score': 1.0, 'label': 'POSITIVE'}
```

Text: I hate this, it's terrible.
Result: {'text': "I hate this, it's terrible.", 'score': -1.0, 'label': 'NEGATIVE'}

```
PS C:\Users\Sreeshma\Downloads\HTML Tutorials & C:\Users\Sreeshma\AppData\Local\Programs\Python\Python310\python.exe "c:/Users/Sreeshma/Documents/AI ASS/sentiment_analysis_bias.py"  
--- Sentiment Analysis ---  
Text: This product is amazing and excellent!  
Result: {'text': 'This product is amazing and excellent!', 'score': 1.0, 'label': 'POSITIVE'}
```

Text: I hate this, it's terrible.
Text: This product is amazing and excellent!
Result: {'text': 'This product is amazing and excellent!', 'score': 1.0, 'label': 'POSITIVE'}

Text: I hate this, it's terrible.

```
PROBLEMS 0 OUTPUT DEBUG CONSOLE TERMINAL POINTS Python + ×
```

```
==> Dataset Balancing ==>
Before: { "POSITIVE": 8, "NEGATIVE": 2 }
Before: { "POSITIVE": 8, "NEGATIVE": 2 }
After: POSITIVE=2, NEGATIVE=2
After: POSITIVE=2, NEGATIVE=2
PS C:\Users\Sreeshma\Downloads\HTML Tutorials>
```



```
After: POSITIVE=2, NEGATIVE=2
PS C:\Users\Sreeshma\Downloads\HTML Tutorials>
```



```
After: POSITIVE=2, NEGATIVE=2
PS C:\Users\Sreeshma\Downloads\HTML Tutorials>
```



```
After: POSITIVE=2, NEGATIVE=2
PS C:\Users\Sreeshma\Downloads\HTML Tutorials>
```

Task Description #3:

- Use Copilot to write a Python program that recommends products based on user history. Ask it to follow ethical guidelines like transparency and fairness

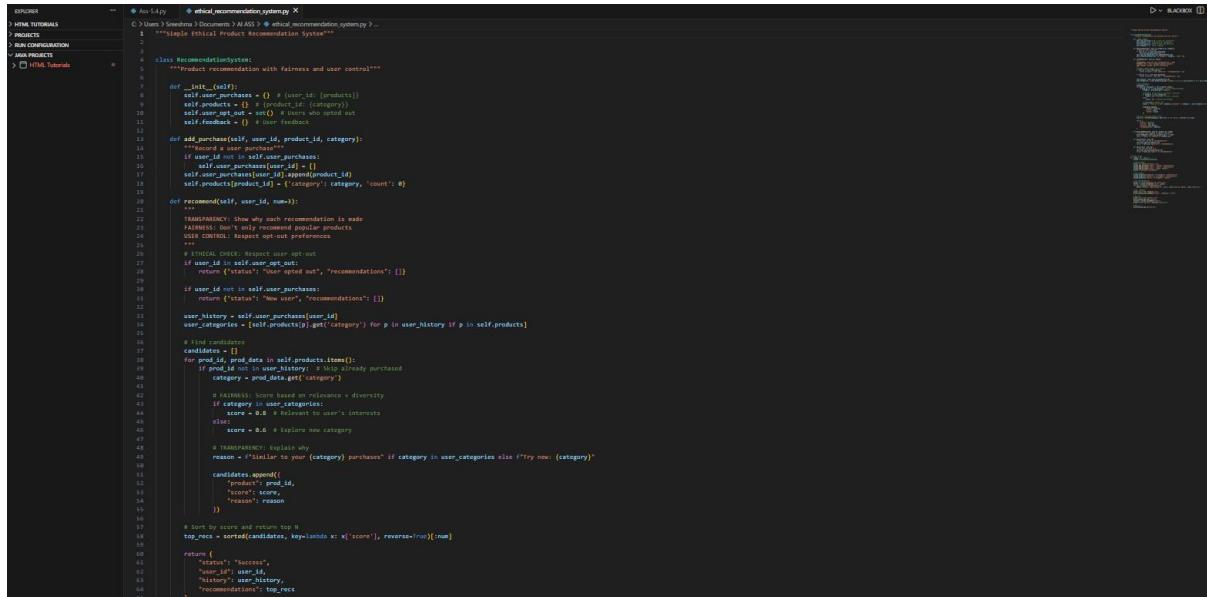
PROMPT: # Generate a Python program that recommends products based on user purchase history.

Follow ethical AI guidelines such as transparency, fairness, and user control.

Add comments explaining how recommendations are generated,

avoid favoritism toward only popular products,

and allow users to give feedback or opt out of recommendations.



The screenshot shows a code editor with a Python file named `ethical_recommendation_system.py`. The code implements a recommendation system with ethical considerations. It includes functions for adding purchases, recommending products, and calculating scores based on popularity, relevance, and diversity. The code is annotated with comments explaining its logic and ethical principles like transparency and fairness.

```
#!/usr/bin/env python3
# ethical_recommendation_system.py

class RecommendationSystem:
    """Implements ethical product recommendation System"""

    def __init__(self):
        self.user_purchases = {} # (user_id: [products])
        self.products = {} # product_id: (category)
        self.user_categories = {} # user_id: [categories]
        self.feedback = {} # user_feedback

    def add_purchase(self, user_id, product_id, category):
        """Record a user purchase"""
        if user_id not in self.user_purchases:
            self.user_purchases[user_id] = []
        self.user_purchases[user_id].append(product_id)
        self.user_categories[user_id].append(category)
        self.products[product_id] = {'category': category, 'count': 0}

    def recommend(self, user_id, num=10):
        """TRANSPARENCY: show why each recommendation is made
        Fairness: Don't only recommend popular products
        User control: Respect user's opt-out
        """
        if user_id not in self.user_purchases:
            return {"status": "User opted out", "recommendations": []}
        if user_id not in self.user_categories:
            return {"status": "New user", "recommendations": []}

        user_history = self.user_purchases[user_id]
        user_categories = self.products.get('category') for p in user_history if p in self.products

        # Find candidates
        candidates = []
        for prod_id, prod_data in self.products.items():
            if prod_id not in user_history: # Skip already purchased
                category = prod_data.get('category')
                if category in user_categories: # relevance + diversity
                    if category in user_categories:
                        score = 0.8 # Relevant to user's interests
                    else:
                        score = 0.6 # Explore new category
                else:
                    reason = "Similar to your [category] purchases" if category in user_categories else "[try now] (category)"
                    candidates.append({
                        "product": prod_id,
                        "score": score,
                        "reason": reason
                    })
        top_rec = sorted(candidates, key=lambda x: x['score'], reverse=True)[num]

        return {
            "user_id": user_id,
            "user_ip": user_ip,
            "history": user_history,
            "recommendations": top_rec
        }
```

```

# Example usage
if __name__ == "__main__":
    system = RecommendationSystem()

    # Add purchases
    print("--- Adding Purchases ---")
    system.add_purchase("user1", "laptop", "electronics")
    system.add_purchase("user1", "monitor", "electronics")
    system.add_purchase("user1", "book", "books")
    print("Purchases recorded")

    # Add products
    system.products["keyboard"] = {"category": "Electronics"}
    system.products["monitor"] = {"category": "Electronics"}
    system.products["book"] = {"category": "Books"}

    # Get recommendations
    print("--- Recommendations for user1 ---")
    result = system.recommend("user1", num=2)
    for rec in result["recommendations"]:
        print(f"Product: {rec['product']}, Score: {rec['score']}, Reason: {rec['reason']}")

    # Give feedback
    print("--- User Feedback ---")
    print(system.give_feedback("user1", "keyboard", True))

    # Opt out
    print("--- User Control ---")
    print(system.opt_out("user1"))
    result2 = system.recommend("user1")
    print("After opt-out: ", result2["status"])

    # Opt in
    print(system.opt_in("user1"))

```

Expected Output #3:

- Copilot suggestions that include explanations, fairness checks (e.g., avoiding favoritism), and user feedback options in the code.

```

--- Adding Purchases ---
Purchases recorded
PS C:\Users\greenba\Downloads\HTML Tutorials> & C:/Users/greenba/AppData/Local/Programs/Python/Python310/python.exe "C:/Users/greenba/Documents/AI AI/ethical_recommendation_system.py"
PS C:\Users\greenba\Downloads\HTML Tutorials> & C:/Users/greenba/AppData/Local/Programs/Python/Python310/python.exe "C:/Users/greenba/Documents/AI AI/ethical_recommendation_system.py"

--- Adding Purchases ---
Purchases recorded

--- Recommendations for user1 ---
Product: keyboard, Score: 8.8, Reason: Stellar to your Electronics purchases
Product: monitor, Score: 8.8, Reason: Stellar to your Electronics purchases

--- User Feedback ---
Thanks for feedback on keyboard

--- User Control ---
user1 opted out of recommendations
After opt-out: user1 opted out
user1 opted out of recommendations
PS C:\Users\greenba\Downloads\HTML Tutorials>

```

Task Description #4:

- Prompt Copilot to generate logging functionality in a Python web application. Then, ask it to ensure the logs do not record sensitive information.

PROMPT: # Generate logging functionality for a Python web application.

Ensure logs do NOT store sensitive information such as passwords,

emails, or personal identifiers.

Add comments explaining ethical logging practices and privacy protection.

```
BUK -- ethical_logging.py ethical recommendation system.py ethical_logging.py
C:\Users>Somedude>Documents>AI\ADS>ethical_logging>...
1  #!/usr/bin/python
2  """
3      Simple Ethical Logging for web Applications
4  """
5  import logging
6  import re
7
8
9  class PrivacyFilter(logging.Filter):
10     """
11         Remove sensitive data from logs!!!
12     """
13     def filter(self, record):
14         """
15             Mask password, emails, tokens, cards before logging!!!
16         """
17         msg = self.filter(record)
18
19         # PRIVATE: Mask password
20         msg = re.sub(r'password=[a-zA-Z0-9]*', 'password=*****REDACTED***', msg, flags=re.IGNORECASE)
21
22         # PRIVATE: Mask emails (show domain only)
23         msg = re.sub(r'[\w.-]+@[a-zA-Z.-]+\.[\w.-]+', '[email]***', msg)
24
25         # PRIVATE: Mask phone numbers (show last 4 digits)
26         msg = re.sub(r'(\d{3})\d{4}(\d{3})(\d{4})', '(\d{3})CCCV\d{3}\1\d{4}', msg)
27
28
29         record.msg = msg
30
31     return True
32
33
34
35     def setup_logger(name, log_file='app.log'):
36         """
37             Setup logger with file protection!!!
38         """
39         logger = logging.getLogger(name)
40         logger.setLevel(logging.INFO)
41
42         # Add privacy filter
43         privacy_filter = PrivacyFilter()
44
45         # Console Handler
46         console_handler = logging.StreamHandler()
47         console_handler.addFilter(privacy_filter)
48         formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
49         console_handler.setFormatter(formatter)
50         logger.addHandler(console_handler)
51
52         # File Handler
53         if log_file:
54             file_handler = logging.FileHandler(log_file)
55             file_handler.addFilter(privacy_filter)
56             file_handler.setLevel(logging.DEBUG)
57             file_handler.setFormatter(formatter)
58             logger.addHandler(file_handler)
59
60             # PRIVATE: Restrict file permissions (owner read/write only)
61             import os
62             os.chmod(log_file, 0600)
63
64         return logger
65
66
67     def log_user_action(logger, action, user_id, *safe_details):
68         """
69             Log user action with only safe fields!!!
70         """
71         msg = "ACTION: %s | user: (%s)" % (action, user_id)
72         if safe_details:
73             msg += " | (%s)" % (safe_details)
74             logger.info(msg)
75
76
77     # Example Usage
78     if __name__ == "__main__":
79         print("---- simple Ethical Logging Done ----")
80
81         logger = setup_logger('app', 'app.log')
82
83         print("Test 1: Password Masking")
84         logger.info("login with password=SecurePass123")
85
86         print("Test 2: Email Masking")
87         logger.info("Send email to user@example.com")
88
```

```
BUK -- ethical_logging.py ethical recommendation system.py ethical_logging.py
C:\Users>Somedude>Documents>AI\ADS>ethical_logging>...
1  #!/usr/bin/python
2  """
3      Simple Ethical Logging for web Applications
4  """
5  def setup_logger(name, log_file='app.log'):
6
7      logger = logging.getLogger(name)
8      logger.setLevel(logging.INFO)
9
10     return logger
11
12
13     def log_user_action(logger, action, user_id, *safe_details):
14
15         """
16             Log user action with only safe fields!!!
17         """
18         msg = "ACTION: %s | user: (%s)" % (action, user_id)
19         if safe_details:
20             msg += " | (%s)" % (safe_details)
21             logger.info(msg)
22
23
24     # Example Usage
25     if __name__ == "__main__":
26         print("---- simple Ethical Logging Done ----")
27
28         logger = setup_logger('app', 'app.log')
29
30         print("Test 1: Password Masking")
31         logger.info("login with password=SecurePass123")
32
33         print("Test 2: Email Masking")
34         logger.info("Send email to user@example.com")
35
36         print("Test 3: API Key Masking")
37         logger.info("API key: sk_1live_1234abcde")
38
39         print("Test 4: Credit Card Masking")
40         logger.info("Payment with card 4321-1234-5678-9999")
41
42         print("Test 5: User Action Logging")
43         log_user_action(logger, "purchase", "user_123", status="success", amount=99.99)
44
45         print("User ID: %s" % (user_id))
46         print("---- LOGGING PRACTICES ----")
47         print("1. NO FILTER: Mask passwords, emails, tokens, cards")
48         print("2. MINIMAL DATA: Only log necessary information")
49         print("3. SECURE FILES: Set permissions to 600 (owner only)")
50         print("4. LOGGING LOG: Log for audit/tracing purposes")
51         print("5. NO SENSITIVE: Never store sensitive data in logs")
52
53
54
```

Expected Output #4:

- Logging code that avoids saving personal identifiers (e.g., passwords, emails), and includes comments about ethical logging practices.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PAGES
Test 5: User Action Logging
2024-01-29 18:20:55,566 - app - INFO - ACTION: purchase | user: user_123 | {'status': 'success', 'amount': 99.99}

ETHICAL LOGGING PRACTICES:
1. PRIVACY FILTER: Mask passwords, emails, tokens, cards
2. MINIMAL DATA: Only log necessary information
3. SECURE FILES: Set permissions to 600 (owner only)
4. AUDIT ACTIONS: Log for auditing and debugging
5. NO SECRETS: Never store sensitive data in logs
2024-01-29 18:20:55,566 - app - INFO - ACTION: purchase | user: user_123 | {'status': 'success', 'amount': 99.99}

ETHICAL LOGGING PRACTICES:
1. PRIVACY FILTER: Mask passwords, emails, tokens, cards
2. MINIMAL DATA: Only log necessary information
3. SECURE FILES: Set permissions to 600 (owner only)
4. AUDIT ACTIONS: Log for auditing and debugging
5. NO SECRETS: Never store sensitive data in logs
2024-01-29 18:20:55,566 - app - INFO - ACTION: purchase | user: user_123 | {'status': 'success', 'amount': 99.99}
5. NO SECRETS: Never store sensitive data in logs
5. NO SECRETS: Never store sensitive data in logs
5. NO SECRETS: Never store sensitive data in logs

```

Task Description #5:

- Ask Copilot to generate a machine learning model. Then, prompt it to add documentation on how to use the model responsibly (e.g., explainability, accuracy limits).

PROMPT: Generate a Python machine learning model (including data loading, training, and prediction steps).

Add inline documentation or a README-style comment section explaining how to use the model responsibly, including accuracy limitations, explainability considerations, fairness concerns, and appropriate use cases and restrictions.

```
EXPLORER
> HTML TUTORIALS
> PROJECTS
> RECOMMENDATION
> JAVA PROJECTS
> HTML Tutorials

-- Ass-5-4.py 伦理推荐系统.py ethical_logging.py responsible_ml_model.py

C:\Users\Seethima>Documents\AI\Ass-5>python responsible_ml_model.py

67     recs, reasons = recommend_products(user_id, user_history, product_catalog)
68     for prod, reason in zip(recs, reasons):
69         print(f"(prod['name']) {category}: (prod['category'])) -> (reason)")
70
71     # User feedback and opt-out
72     print("Would you like to provide Feedback or opt out of recommendations?")
73     feedback = input("Type 'feedback' to provide feedback or 'opt out' to stop recommendations: ")
74     if feedback.lower() == "opt out":
75         print("You have opted out of recommendations. Your preferences will be respected.")
76     else:
77         print(f"Thank you for your Feedback: {feedback}")
78
79 # --- Ethical AI Notes ---
80 # - Transparency: Each recommendation includes an explanation.
81 # - Fairness: The system ensures diversity and avoids recommending only from the most frequent category.
82 # - User Control: Users can provide feedback or opt out at any time.
83 # - Regularly audit recommendation logic for bias and update as needed.
84 # A few required packages are installed
85 import sys
86 import subprocess
87
88 def install_if_missing(package):
89     try:
90         __import__(package)
91     except ImportError:
92         print(f"Installing missing package: {package}")
93         subprocess.check_call([sys.executable, "-m", "pip", "install", package])
94
95 # Install 'textblob' if not present
96 install_if_missing('textblob')
97
98 # Sentiment analysis function with bias awareness and mitigation strategies
99 from textblob import TextBlob
100
101 def analyze_sentiment(text):
102     """
103     Analyze the sentiment of the input text.
104     Returns polarity (-1 to 1) and subjectivity (0 to 1).
105
106     Potential sources of bias in training data:
107     - Imbalanced datasets (e.g., more positive than negative samples)
108     - Presence of offensive, discriminatory, or culturally specific terms
109     - Overrepresentation or underrepresentation of certain topics or groups
110
111     Strategies to mitigate bias:
112     - Balance the dataset across sentiment classes and demographic groups
113     - Remove or flag offensive/discriminatory terms during preprocessing
114     - Use diverse and representative data sources
115     - Document keep logs and test for bias regularly
116     - Involve domain experts in dataset curation
117     ...
118
119     # Example using Textblob for simple sentiment analysis
120     blob = TextBlob(text)
121     polarity = blob.sentiment.polarity
122     subjectivity = blob.sentiment.subjectivity
123     return polarity, subjectivity
124
125 # Example usage
126 if __name__ == "__main__":
127     user_text = input("Enter text for sentiment analysis: ")
128     polarity, subjectivity = analyze_sentiment(user_text)
129     print(f"Polarity: {polarity}, Subjectivity: {subjectivity}")
130
131 # Note: For production, train your own model on a carefully curated dataset and regularly audit for bias.
132 # The above function uses Textblob, which is trained on general-purpose data and may inherit its biases.
```

Expected Output #5:

- Copilot-generated model code with a README or inline documentation suggesting responsible usage, limitations, and fairness considerations.

The screenshot shows a dual-monitor setup. The left monitor displays a Windows desktop with several open windows, including a PyCharm IDE showing code for a 'CP LAB ASS' project, a Task Manager, and a File Explorer. The right monitor displays a GitHub pull request titled 'Generate a Python machine learning model (including the loading, testing, and prediction logic)'. The pull request details include a description, code changes, and a review by 'mikesheld'. The GitHub interface also shows the repository's README, issues, and pull requests.