

## **Assignment: 5.4**

Name: Ch.Adarsh

Hno: 2303A51044

## 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.

The screenshot shows a code editor with a Python script named `task1.py`. The script includes sections for collecting user name, age, and email, each with validation logic and privacy considerations. A sidebar on the right provides annotations and features.

**Annotations:**

- User Data Collection Script with Privacy and Anonymization Guidelines**: A comment at the top of the script.
- Create a Python script to collect name, age, and email from a user and add comments explaining how to anonymize or safeguard sensitive data.**: A callout box in the top right.
- Read Python script**: A link in the sidebar.
- Now I'll create a comprehensive Python script that collects user data and includes detailed comments on data protection and anonymization:**: A callout box below the sidebar.
- Created task1.py**: A message indicating the script was created.
- Done! I've created a comprehensive Python script that:**: A callout box below the creation message.
- Features:**
  - Collects name, age, and email with input validation
  - Includes detailed comments on privacy and data protection throughout
  - Demonstrates anonymization techniques:
    - Name hashing for irreversible anonymization
    - Age range conversion to reduce precision and privacy risk
    - Email hashing with salt for secure comparison
  - Provides best practices for data storage including encryption, database security, and compliance (GDPR/CCPA)
  - Shows secure data storage patterns with hash IDs for tracking without exposing PII
  - Displays both original and anonymized data for comparison
- The script is production-aware with explanations of encryption, access controls, audit logging, and data retention policies**: A callout box below the features section.
- task1.py**: A file icon in the bottom right.
- Describe what to build next**: A placeholder for user input.
- Agent Auto**: A dropdown menu in the bottom right.

```
=====  
USER DATA COLLECTION WITH PRIVACY SAFEGUARDS  
=====
```

```
Enter your name: Yashwanth  
Enter your age: 23  
Enter your email: pateruyashwanth6671@gmail.com
```

```
=====  
DATA COLLECTION SUMMARY  
=====
```

```
Original Data Collected:
```

```
Name: Yashwanth  
Age: 23  
Email: pateruyashwanth6671@gmail.com
```

```
Anonymized/Safeguarded Data:
```

```
Anonymous ID (name hash): f49876d80754ccfc  
Age Range (anonymized): 18-25  
Email Hash (hashed): 72503dcb95c46f52
```

```
Data prepared for secure storage:
```

```
Hash ID (for tracking): 8014ff3e137fc3c6  
Timestamp: 2026-01-22T13:39:24.414973
```

```
=====  
PRIVACY NOTES:  
=====
```

- ✓ Original data should be encrypted before storage
- ✓ Anonymized data can be used for analytics safely
- ✓ Implement role-based access controls
- ✓ Keep audit logs of all data access
- ✓ Regularly review data retention policies
- ✓ Comply with GDPR, CCPA, and local privacy laws

## 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.

```

1 """
2 Sentiment Analysis Function with Bias Detection and Mitigation
3 This module demonstrates sentiment analysis techniques while addressing
4 common sources of bias in NLP models and data.
5 """
6
7 import re
8 from collections import Counter
9 from typing import Dict, List, Tuple
10 from dataclasses import dataclass
11
12 # ===== SENTIMENT LEXICONS =====
13
14 # Basic sentiment word lists (in production, use comprehensive lexicons)
15 POSITIVE_WORDS = [
16     'good', 'great', 'excellent', 'amazing', 'wonderful', 'fantastic',
17     'love', 'awesome', 'beautiful', 'brilliant', 'perfect', 'brilliant',
18     'happy', 'joy', 'grateful', 'pleased', 'delighted', 'outstanding'
19 ]
20
21 NEGATIVE_WORDS = [
22     'bad', 'terrible', 'horrible', 'awful', 'poor', 'hate',
23     'disgusting', 'pathetic', 'disappointing', 'useless', 'waste',
24     'angry', 'sad', 'upset', 'furious', 'disgusted', 'awful'
25 ]
26
27 NEUTRAL_WORDS = [
28     'is', 'was', 'are', 'the', 'a', 'an', 'and', 'or', 'but', 'in', 'on'
29 ]
30
31
32 @dataclass
33 class SentimentResult:
34     """Data class for sentiment analysis results with bias metrics."""
35     text: str
36     sentiment: str
37     confidence: float
38     score: float
39     bias_flags: List[str]
40     demographic_language: List[str]
41     recommendation: str

```

Code editor interface showing the Python file task2.py. The code implements a sentiment analysis function with bias detection and mitigation. It includes lists of positive, negative, and neutral words, a dataclass for sentiment results, and logic for demographic language detection, sarcasm detection, cultural bias detection, and text length handling. A sidebar provides a summary of the project's purpose, key features, and comprehensive documentation.

```

183 def analyze_sentiment(text: str) -> SentimentResult:
C:\Users\HP\Downloads\ai_assistant_coding_68\task1.py
184
185     # Normalize text
186     normalized_text = text.lower().strip()
187
188     # ===== BIAS DETECTION CHECKS =====
189     bias_flags = []
190
191     # Check for demographic language
192     demographics = detect_demographic_language(text)
193     if demographics:
194         bias_flags.append(f"Demographic language detected: {', '.join(demographics)}")
195
196     # Check for sarcasm
197     if detect_sarcasm_and_context(text):
198         bias_flags.append("Potential sarcasm detected - may reverse sentiment")
199
200     # Check for cultural language
201     cultural_issues = detect_cultural_bias(text)
202     if cultural_issues:
203         bias_flags.append(f"Cultural/contextual markers: {', '.join(cultural_issues)}")
204
205     # Check text length (short texts are often misclassified)
206     if len(text.split()) < 3:
207         bias_flags.append("Very short text - classification may have low confidence")
208
209     # ===== SENTIMENT SCORING =====
210
211     words = normalized_text.split()
212     positive_count = sum(1 for word in words if word in POSITIVE_WORDS)
213     negative_count = sum(1 for word in words if word in NEGATIVE_WORDS)
214
215     # Handle negation (simple negation scope: 2 words before negator)
216     negation_words = {'not', 'no', 'never', 'neither', 'barely', 'hardly'}
217     negation_adjusted_pos = 0
218     negation_adjusted_neg = 0
219
220     for i, word in enumerate(words):
221         if word in negation_words:
222             # Check next 2 words
223             for j in range(i+1, min(i+3, len(words)-1)):
224                 if words[j] in negation_words:
225                     negation_adjusted_pos += 1
226                     negation_adjusted_neg -= 1
227
228             # Adjust sentiment based on negation
229             if negation_adjusted_pos > negation_adjusted_neg:
230                 sentiment = "negative"
231             else:
232                 sentiment = "positive"
233
234             confidence = 0.5 + abs(negation_adjusted_pos - negation_adjusted_neg) / 2
235
236             score = positive_count / len(words)
237             if sentiment == "negative":
238                 score *= confidence
239             else:
240                 score /= confidence
241
242             # Add bias flags
243             if bias_flags:
244                 bias_flags.append(f"Detected bias flags: {', '.join(bias_flags)}")
245
246             # Create result object
247             result = SentimentResult(
248                 text=text,
249                 sentiment=sentiment,
250                 confidence=confidence,
251                 score=score,
252                 bias_flags=bias_flags,
253                 demographic_language=demographics,
254                 recommendation=recommendation
255             )
256
257             return result
258
259
260             # Print output
261             print(f"Text: {text}")
262             print(f"Sentiment: {sentiment}")
263             print(f"Confidence: {confidence:.2f}")
264             print(f"Score: {score:.2f}")
265             print(f"Bias Flags: {', '.join(bias_flags)}")
266             print(f"Demographic Language: {', '.join(demographics)}")
267             print(f"Recommendation: {recommendation}")
268
269             # Save to file
270             with open("output.txt", "w") as f:
271                 f.write(result.json())
272
273             # Print message
274             print("Analysis completed successfully!")
275
276             # Exit loop
277             break
278
279         else:
280             # Create result object
281             result = SentimentResult(
282                 text=text,
283                 sentiment=sentiment,
284                 confidence=confidence,
285                 score=score,
286                 bias_flags=bias_flags,
287                 demographic_language=demographics,
288                 recommendation=recommendation
289             )
290
291             return result
292
293
294             # Print output
295             print(f"Text: {text}")
296             print(f"Sentiment: {sentiment}")
297             print(f"Confidence: {confidence:.2f}")
298             print(f"Score: {score:.2f}")
299             print(f"Bias Flags: {', '.join(bias_flags)}")
300             print(f"Demographic Language: {', '.join(demographics)}")
301             print(f"Recommendation: {recommendation}")
302
303             # Save to file
304             with open("output.txt", "w") as f:
305                 f.write(result.json())
306
307             # Print message
308             print("Analysis completed successfully!")
309
310             # Exit loop
311             break
312
313
314             # Print output
315             print(f"Text: {text}")
316             print(f"Sentiment: {sentiment}")
317             print(f"Confidence: {confidence:.2f}")
318             print(f"Score: {score:.2f}")
319             print(f"Bias Flags: {', '.join(bias_flags)}")
320             print(f"Demographic Language: {', '.join(demographics)}")
321             print(f"Recommendation: {recommendation}")
322
323             # Save to file
324             with open("output.txt", "w") as f:
325                 f.write(result.json())
326
327             # Print message
328             print("Analysis completed successfully!")
329
330             # Exit loop
331             break
332
333
334             # Print output
335             print(f"Text: {text}")
336             print(f"Sentiment: {sentiment}")
337             print(f"Confidence: {confidence:.2f}")
338             print(f"Score: {score:.2f}")
339             print(f"Bias Flags: {', '.join(bias_flags)}")
340             print(f"Demographic Language: {', '.join(demographics)}")
341             print(f"Recommendation: {recommendation}")
342
343             # Save to file
344             with open("output.txt", "w") as f:
345                 f.write(result.json())
346
347             # Print message
348             print("Analysis completed successfully!")
349
350             # Exit loop
351             break
352
353
354             # Print output
355             print(f"Text: {text}")
356             print(f"Sentiment: {sentiment}")
357             print(f"Confidence: {confidence:.2f}")
358             print(f"Score: {score:.2f}")
359             print(f"Bias Flags: {', '.join(bias_flags)}")
360             print(f"Demographic Language: {', '.join(demographics)}")
361             print(f"Recommendation: {recommendation}")
362
363             # Save to file
364             with open("output.txt", "w") as f:
365                 f.write(result.json())
366
367             # Print message
368             print("Analysis completed successfully!")
369
370             # Exit loop
371             break
372
373
374             # Print output
375             print(f"Text: {text}")
376             print(f"Sentiment: {sentiment}")
377             print(f"Confidence: {confidence:.2f}")
378             print(f"Score: {score:.2f}")
379             print(f"Bias Flags: {', '.join(bias_flags)}")
380             print(f"Demographic Language: {', '.join(demographics)}")
381             print(f"Recommendation: {recommendation}")
382
383             # Save to file
384             with open("output.txt", "w") as f:
385                 f.write(result.json())
386
387             # Print message
388             print("Analysis completed successfully!")
389
390             # Exit loop
391             break
392
393
394             # Print output
395             print(f"Text: {text}")
396             print(f"Sentiment: {sentiment}")
397             print(f"Confidence: {confidence:.2f}")
398             print(f"Score: {score:.2f}")
399             print(f"Bias Flags: {', '.join(bias_flags)}")
400             print(f"Demographic Language: {', '.join(demographics)}")
401             print(f"Recommendation: {recommendation}")
402
403             # Save to file
404             with open("output.txt", "w") as f:
405                 f.write(result.json())
406
407             # Print message
408             print("Analysis completed successfully!")
409
410             # Exit loop
411             break
412
413
414             # Print output
415             print(f"Text: {text}")
416             print(f"Sentiment: {sentiment}")
417             print(f"Confidence: {confidence:.2f}")
418             print(f"Score: {score:.2f}")
419             print(f"Bias Flags: {', '.join(bias_flags)}")
420             print(f"Demographic Language: {', '.join(demographics)}")
421             print(f"Recommendation: {recommendation}")
422
423             # Save to file
424             with open("output.txt", "w") as f:
425                 f.write(result.json())
426
427             # Print message
428             print("Analysis completed successfully!")
429
430             # Exit loop
431             break
432
433
434             # Print output
435             print(f"Text: {text}")
436             print(f"Sentiment: {sentiment}")
437             print(f"Confidence: {confidence:.2f}")
438             print(f"Score: {score:.2f}")
439             print(f"Bias Flags: {', '.join(bias_flags)}")
440             print(f"Demographic Language: {', '.join(demographics)}")
441             print(f"Recommendation: {recommendation}")
442
443             # Save to file
444             with open("output.txt", "w") as f:
445                 f.write(result.json())
446
447             # Print message
448             print("Analysis completed successfully!")
449
450             # Exit loop
451             break
452
453
454             # Print output
455             print(f"Text: {text}")
456             print(f"Sentiment: {sentiment}")
457             print(f"Confidence: {confidence:.2f}")
458             print(f"Score: {score:.2f}")
459             print(f"Bias Flags: {', '.join(bias_flags)}")
460             print(f"Demographic Language: {', '.join(demographics)}")
461             print(f"Recommendation: {recommendation}")
462
463             # Save to file
464             with open("output.txt", "w") as f:
465                 f.write(result.json())
466
467             # Print message
468             print("Analysis completed successfully!")
469
470             # Exit loop
471             break
472
473
474             # Print output
475             print(f"Text: {text}")
476             print(f"Sentiment: {sentiment}")
477             print(f"Confidence: {confidence:.2f}")
478             print(f"Score: {score:.2f}")
479             print(f"Bias Flags: {', '.join(bias_flags)}")
480             print(f"Demographic Language: {', '.join(demographics)}")
481             print(f"Recommendation: {recommendation}")
482
483             # Save to file
484             with open("output.txt", "w") as f:
485                 f.write(result.json())
486
487             # Print message
488             print("Analysis completed successfully!")
489
490             # Exit loop
491             break
492
493
494             # Print output
495             print(f"Text: {text}")
496             print(f"Sentiment: {sentiment}")
497             print(f"Confidence: {confidence:.2f}")
498             print(f"Score: {score:.2f}")
499             print(f"Bias Flags: {', '.join(bias_flags)}")
500             print(f"Demographic Language: {', '.join(demographics)}")
501             print(f"Recommendation: {recommendation}")
502
503             # Save to file
504             with open("output.txt", "w") as f:
505                 f.write(result.json())
506
507             # Print message
508             print("Analysis completed successfully!")
509
510             # Exit loop
511             break
512
513
514             # Print output
515             print(f"Text: {text}")
516             print(f"Sentiment: {sentiment}")
517             print(f"Confidence: {confidence:.2f}")
518             print(f"Score: {score:.2f}")
519             print(f"Bias Flags: {', '.join(bias_flags)}")
520             print(f"Demographic Language: {', '.join(demographics)}")
521             print(f"Recommendation: {recommendation}")
522
523             # Save to file
524             with open("output.txt", "w") as f:
525                 f.write(result.json())
526
527             # Print message
528             print("Analysis completed successfully!")
529
530             # Exit loop
531             break
532
533
534             # Print output
535             print(f"Text: {text}")
536             print(f"Sentiment: {sentiment}")
537             print(f"Confidence: {confidence:.2f}")
538             print(f"Score: {score:.2f}")
539             print(f"Bias Flags: {', '.join(bias_flags)}")
540             print(f"Demographic Language: {', '.join(demographics)}")
541             print(f"Recommendation: {recommendation}")
542
543             # Save to file
544             with open("output.txt", "w") as f:
545                 f.write(result.json())
546
547             # Print message
548             print("Analysis completed successfully!")
549
550             # Exit loop
551             break
552
553
554             # Print output
555             print(f"Text: {text}")
556             print(f"Sentiment: {sentiment}")
557             print(f"Confidence: {confidence:.2f}")
558             print(f"Score: {score:.2f}")
559             print(f"Bias Flags: {', '.join(bias_flags)}")
560             print(f"Demographic Language: {', '.join(demographics)}")
561             print(f"Recommendation: {recommendation}")
562
563             # Save to file
564             with open("output.txt", "w") as f:
565                 f.write(result.json())
566
567             # Print message
568             print("Analysis completed successfully!")
569
570             # Exit loop
571             break
572
573
574             # Print output
575             print(f"Text: {text}")
576             print(f"Sentiment: {sentiment}")
577             print(f"Confidence: {confidence:.2f}")
578             print(f"Score: {score:.2f}")
579             print(f"Bias Flags: {', '.join(bias_flags)}")
580             print(f"Demographic Language: {', '.join(demographics)}")
581             print(f"Recommendation: {recommendation}")
582
583             # Save to file
584             with open("output.txt", "w") as f:
585                 f.write(result.json())
586
587             # Print message
588             print("Analysis completed successfully!")
589
590             # Exit loop
591             break
592
593
594             # Print output
595             print(f"Text: {text}")
596             print(f"Sentiment: {sentiment}")
597             print(f"Confidence: {confidence:.2f}")
598             print(f"Score: {score:.2f}")
599             print(f"Bias Flags: {', '.join(bias_flags)}")
600             print(f"Demographic Language: {', '.join(demographics)}")
601             print(f"Recommendation: {recommendation}")
602
603             # Save to file
604             with open("output.txt", "w") as f:
605                 f.write(result.json())
606
607             # Print message
608             print("Analysis completed successfully!")
609
610             # Exit loop
611             break
612
613
614             # Print output
615             print(f"Text: {text}")
616             print(f"Sentiment: {sentiment}")
617             print(f"Confidence: {confidence:.2f}")
618             print(f"Score: {score:.2f}")
619             print(f"Bias Flags: {', '.join(bias_flags)}")
620             print(f"Demographic Language: {', '.join(demographics)}")
621             print(f"Recommendation: {recommendation}")
622
623             # Save to file
624             with open("output.txt", "w") as f:
625                 f.write(result.json())
626
627             # Print message
628             print("Analysis completed successfully!")
629
630             # Exit loop
631             break
632
633
634             # Print output
635             print(f"Text: {text}")
636             print(f"Sentiment: {sentiment}")
637             print(f"Confidence: {confidence:.2f}")
638             print(f"Score: {score:.2f}")
639             print(f"Bias Flags: {', '.join(bias_flags)}")
640             print(f"Demographic Language: {', '.join(demographics)}")
641             print(f"Recommendation: {recommendation}")
642
643             # Save to file
644             with open("output.txt", "w") as f:
645                 f.write(result.json())
646
647             # Print message
648             print("Analysis completed successfully!")
649
650             # Exit loop
651             break
652
653
654             # Print output
655             print(f"Text: {text}")
656             print(f"Sentiment: {sentiment}")
657             print(f"Confidence: {confidence:.2f}")
658             print(f"Score: {score:.2f}")
659             print(f"Bias Flags: {', '.join(bias_flags)}")
660             print(f"Demographic Language: {', '.join(demographics)}")
661             print(f"Recommendation: {recommendation}")
662
663             # Save to file
664             with open("output.txt", "w") as f:
665                 f.write(result.json())
666
667             # Print message
668             print("Analysis completed successfully!")
669
670             # Exit loop
671             break
672
673
674             # Print output
675             print(f"Text: {text}")
676             print(f"Sentiment: {sentiment}")
677             print(f"Confidence: {confidence:.2f}")
678             print(f"Score: {score:.2f}")
679             print(f"Bias Flags: {', '.join(bias_flags)}")
680             print(f"Demographic Language: {', '.join(demographics)}")
681             print(f"Recommendation: {recommendation}")
682
683             # Save to file
684             with open("output.txt", "w") as f:
685                 f.write(result.json())
686
687             # Print message
688             print("Analysis completed successfully!")
689
690             # Exit loop
691             break
692
693
694             # Print output
695             print(f"Text: {text}")
696             print(f"Sentiment: {sentiment}")
697             print(f"Confidence: {confidence:.2f}")
698             print(f"Score: {score:.2f}")
699             print(f"Bias Flags: {', '.join(bias_flags)}")
700             print(f"Demographic Language: {', '.join(demographics)}")
701             print(f"Recommendation: {recommendation}")
702
703             # Save to file
704             with open("output.txt", "w") as f:
705                 f.write(result.json())
706
707             # Print message
708             print("Analysis completed successfully!")
709
709             # Exit loop
710             break
711
712
713             # Print output
714             print(f"Text: {text}")
715             print(f"Sentiment: {sentiment}")
716             print(f"Confidence: {confidence:.2f}")
717             print(f"Score: {score:.2f}")
718             print(f"Bias Flags: {', '.join(bias_flags)}")
719             print(f"Demographic Language: {', '.join(demographics)}")
720             print(f"Recommendation: {recommendation}")
721
722             # Save to file
723             with open("output.txt", "w") as f:
724                 f.write(result.json())
725
726             # Print message
727             print("Analysis completed successfully!")
728
729             # Exit loop
730             break
731
732
733             # Print output
734             print(f"Text: {text}")
735             print(f"Sentiment: {sentiment}")
736             print(f"Confidence: {confidence:.2f}")
737             print(f"Score: {score:.2f}")
738             print(f"Bias Flags: {', '.join(bias_flags)}")
739             print(f"Demographic Language: {', '.join(demographics)}")
740             print(f"Recommendation: {recommendation}")
741
742             # Save to file
743             with open("output.txt", "w") as f:
744                 f.write(result.json())
745
746             # Print message
747             print("Analysis completed successfully!")
748
749             # Exit loop
750             break
751
752
753             # Print output
754             print(f"Text: {text}")
755             print(f"Sentiment: {sentiment}")
756             print(f"Confidence: {confidence:.2f}")
757             print(f"Score: {score:.2f}")
758             print(f"Bias Flags: {', '.join(bias_flags)}")
759             print(f"Demographic Language: {', '.join(demographics)}")
760             print(f"Recommendation: {recommendation}")
761
762             # Save to file
763             with open("output.txt", "w") as f:
764                 f.write(result.json())
765
766             # Print message
767             print("Analysis completed successfully!")
768
769             # Exit loop
770             break
771
772
773             # Print output
774             print(f"Text: {text}")
775             print(f"Sentiment: {sentiment}")
776             print(f"Confidence: {confidence:.2f}")
777             print(f"Score: {score:.2f}")
778             print(f"Bias Flags: {', '.join(bias_flags)}")
779             print(f"Demographic Language: {', '.join(demographics)}")
780             print(f"Recommendation: {recommendation}")
781
782             # Save to file
783             with open("output.txt", "w") as f:
784                 f.write(result.json())
785
786             # Print message
787             print("Analysis completed successfully!")
788
789             # Exit loop
790             break
791
792
793             # Print output
794             print(f"Text: {text}")
795             print(f"Sentiment: {sentiment}")
796             print(f"Confidence: {confidence:.2f}")
797             print(f"Score: {score:.2f}")
798             print(f"Bias Flags: {', '.join(bias_flags)}")
799             print(f"Demographic Language: {', '.join(demographics)}")
800             print(f"Recommendation: {recommendation}")
801
802             # Save to file
803             with open("output.txt", "w") as f:
804                 f.write(result.json())
805
806             # Print message
807             print("Analysis completed successfully!")
808
809             # Exit loop
810             break
811
812
813             # Print output
814             print(f"Text: {text}")
815             print(f"Sentiment: {sentiment}")
816             print(f"Confidence: {confidence:.2f}")
817             print(f"Score: {score:.2f}")
818             print(f"Bias Flags: {', '.join(bias_flags)}")
819             print(f"Demographic Language: {', '.join(demographics)}")
820             print(f"Recommendation: {recommendation}")
821
822             # Save to file
823             with open("output.txt", "w") as f:
824                 f.write(result.json())
825
826             # Print message
827             print("Analysis completed successfully!")
828
829             # Exit loop
830             break
831
832
833             # Print output
834             print(f"Text: {text}")
835             print(f"Sentiment: {sentiment}")
836             print(f"Confidence: {confidence:.2f}")
837             print(f"Score: {score:.2f}")
838             print(f"Bias Flags: {', '.join(bias_flags)}")
839             print(f"Demographic Language: {', '.join(demographics)}")
840             print(f"Recommendation: {recommendation}")
841
842             # Save to file
843             with open("output.txt", "w") as f:
844                 f.write(result.json())
845
846             # Print message
847             print("Analysis completed successfully!")
848
849             # Exit loop
850             break
851
852
853             # Print output
854             print(f"Text: {text}")
855             print(f"Sentiment: {sentiment}")
856             print(f"Confidence: {confidence:.2f}")
857             print(f"Score: {score:.2f}")
858             print(f"Bias Flags: {', '.join(bias_flags)}")
859             print(f"Demographic Language: {', '.join(demographics)}")
860             print(f"Recommendation: {recommendation}")
861
862             # Save to file
863             with open("output.txt", "w") as f:
864                 f.write(result.json())
865
866             # Print message
867             print("Analysis completed successfully!")
868
869             # Exit loop
870             break
871
872
873             # Print output
874             print(f"Text: {text}")
875             print(f"Sentiment: {sentiment}")
876             print(f"Confidence: {confidence:.2f}")
877             print(f"Score: {score:.2f}")
878             print(f"Bias Flags: {', '.join(bias_flags)}")
879             print(f"Demographic Language: {', '.join(demographics)}")
880             print(f"Recommendation: {recommendation}")
881
882             # Save to file
883             with open("output.txt", "w") as f:
884                 f.write(result.json())
885
886             # Print message
887             print("Analysis completed successfully!")
888
889             # Exit loop
890             break
891
892
893             # Print output
894             print(f"Text: {text}")
895             print(f"Sentiment: {sentiment}")
896             print(f"Confidence: {confidence:.2f}")
897             print(f"Score: {score:.2f}")
898             print(f"Bias Flags: {', '.join(bias_flags)}")
899             print(f"Demographic Language: {', '.join(demographics)}")
900             print(f"Recommendation: {recommendation}")
901
902             # Save to file
903             with open("output.txt", "w") as f:
904                 f.write(result.json())
905
906             # Print message
907             print("Analysis completed successfully!")
908
909             # Exit loop
910             break
911
912
913             # Print output
914             print(f"Text: {text}")
915             print(f"Sentiment: {sentiment}")
916             print(f"Confidence: {confidence:.2f}")
917             print(f"Score: {score:.2f}")
918             print(f"Bias Flags: {', '.join(bias_flags)}")
919             print(f"Demographic Language: {', '.join(demographics)}")
920             print(f"Recommendation: {recommendation}")
921
922             # Save to file
923             with open("output.txt", "w") as f:
924                 f.write(result.json())
925
926             # Print message
927             print("Analysis completed successfully!")
928
929             # Exit loop
930             break
931
932
933             # Print output
934             print(f"Text: {text}")
935             print(f"Sentiment: {sentiment}")
936             print(f"Confidence: {confidence:.2f}")
937             print(f"Score: {score:.2f}")
938             print(f"Bias Flags: {', '.join(bias_flags)}")
939             print(f"Demographic Language: {', '.join(demographics)}")
940             print(f"Recommendation: {recommendation}")
941
942             # Save to file
943             with open("output.txt", "w") as f:
944                 f.write(result.json())
945
946             # Print message
947             print("Analysis completed successfully!")
948
949             # Exit loop
950             break
951
952
953             # Print output
954             print(f"Text: {text}")
955             print(f"Sentiment: {sentiment}")
956             print(f"Confidence: {confidence:.2f}")
957             print(f"Score: {score:.2f}")
958             print(f"Bias Flags: {', '.join(bias_flags)}")
959             print(f"Demographic Language: {', '.join(demographics)}")
960             print(f"Recommendation: {recommendation}")
961
962             # Save to file
963             with open("output.txt", "w") as f:
964                 f.write(result.json())
965
966             # Print message
967             print("Analysis completed successfully!")
968
969             # Exit loop
970             break
971
972
973             # Print output
974             print(f"Text: {text}")
975             print(f"Sentiment: {sentiment}")
976             print(f"Confidence: {confidence:.2f}")
977             print(f"Score: {score:.2f}")
978             print(f"Bias Flags: {', '.join(bias_flags)}")
979             print(f"Demographic Language: {', '.join(demographics)}")
980             print(f"Recommendation: {recommendation}")
981
982             # Save to file
983             with open("output.txt", "w") as f:
984                 f.write(result.json())
985
986             # Print message
987             print("Analysis completed successfully!")
988
989             # Exit loop
990             break
991
992
993             # Print output
994             print(f"Text: {text}")
995             print(f"Sentiment: {sentiment}")
996             print(f"Confidence: {confidence:.2f}")
997             print(f"Score: {score:.2f}")
998             print(f"Bias Flags: {', '.join(bias_flags)}")
999             print(f"Demographic Language: {', '.join(demographics)}")
1000            print(f"Recommendation: {recommendation}")
1001
1002            # Save to file
1003            with open("output.txt", "w") as f:
1004                f.write(result.json())
1005
1006            # Print message
1007            print("Analysis completed successfully!")
1008
1009            # Exit loop
1010            break
1011
1012
1013            # Print output
1014            print(f"Text: {text}")
1015            print(f"Sentiment: {sentiment}")
1016            print(f"Confidence: {confidence:.2f}")
1017            print(f"Score: {score:.2f}")
1018            print(f"Bias Flags: {', '.join(bias_flags)}")
1019            print(f"Demographic Language: {', '.join(demographics)}")
1020            print(f"Recommendation: {recommendation}")
1021
1022            # Save to file
1023            with open("output.txt", "w") as f:
1024                f.write(result.json())
1025
1026            # Print message
1027            print("Analysis completed successfully!")
1028
1029            # Exit loop
1030            break
1031
1032
1033            # Print output
1034            print(f"Text: {text}")
1035            print(f"Sentiment: {sentiment}")
1036            print(f"Confidence: {confidence:.2f}")
1037            print(f"Score: {score:.2f}")
1038            print(f"Bias Flags: {', '.join(bias_flags)}")
1039            print(f"Demographic Language: {', '.join(demographics)}")
1040            print(f"Recommendation: {recommendation}")
1041
1042            # Save to file
1043            with open("output.txt", "w") as f:
1044                f.write(result.json())
1045
1046            # Print message
1047            print("Analysis completed successfully!")
1048
1049            # Exit loop
1050            break
1051
1052
1053            # Print output
1054            print(f"Text: {text}")
1055            print(f"Sentiment: {sentiment}")
1056            print(f"Confidence: {confidence:.2f}")
1057            print(f"Score: {score:.2f}")
1058            print(f"Bias Flags: {', '.join(bias_flags)}")
1059            print(f"Demographic Language: {', '.join(demographics)}")
1060            print(f"Recommendation: {recommendation}")
1061
1062            # Save to file
1063            with open("output.txt", "w") as f:
1064                f.write(result.json())
1065
1066            # Print message
1067            print("Analysis completed successfully!")
1068
1069            # Exit loop
1070            break
1071
1072
1073            # Print output
1074            print(f"Text: {text}")
1075            print(f"Sentiment: {sentiment}")
1076            print(f"Confidence: {confidence:.2f}")
1077            print(f"Score: {score:.2f}")
1078            print(f"Bias Flags: {', '.join(bias_flags)}")
1079            print(f"Demographic Language: {', '.join(demographics)}")
1080            print(f"Recommendation: {recommendation}")
1081
1082            # Save to file
1083            with open("output.txt", "w") as f:
1084                f.write(result.json())
1085
1086            # Print message
1087            print("Analysis completed successfully!")
1088
1089            # Exit loop
1090            break
1091
1092
1093            # Print output
1094            print(f"Text: {text}")
1095            print(f"Sentiment: {sentiment}")
1096            print(f"Confidence: {confidence:.2f}")
1097            print(f"Score: {score:.2f}")
1098            print(f"Bias Flags: {', '.join(bias_flags)}")
1099            print(f"Demographic Language: {', '.join(demographics)}")
1100            print(f"Recommendation: {recommendation}")
1101
1102            # Save to file
1103            with open("output.txt", "w") as f:
1104                f.write(result.json())
1105
1106            # Print message
1107            print("Analysis completed successfully!")
1108
1109            # Exit loop
1110            break
1111
1112
1113            # Print output
1114            print(f"Text: {text}")
1115            print(f"Sentiment: {sentiment}")
1116            print(f"Confidence: {confidence:.2f}")
1117            print(f"Score: {score:.2f}")
1118            print(f"Bias Flags: {', '.join(bias_flags)}")
1119            print(f"Demographic Language: {', '.join(demographics)}")
1120            print(f"Recommendation: {recommendation}")
1121
1122            # Save to file
1123            with open("output.txt", "w") as f:
1124                f.write(result.json())
1125
1126            # Print message
1127            print("Analysis completed successfully!")
1128
1129            # Exit loop
1130            break
1131
1132
1133            # Print output
1134            print(f"Text: {text}")
1135            print(f"Sentiment: {sentiment}")
1136            print(f"Confidence: {confidence:.2f}")
1137            print(f"Score: {score:.2f}")
1138            print(f"Bias Flags: {', '.join(bias_flags)}")
1139            print(f"Demographic Language: {', '.join(demographics)}")
1140            print(f"Recommendation: {recommendation}")
1141
1142            # Save to file
1143            with open("output.txt", "w") as f:
1144                f.write(result.json())
1145
1146            # Print message
1147            print("Analysis completed successfully!")
1148
1149            # Exit loop
1150            break
1151
1152
1153            # Print output
1154            print(f"Text: {text}")
1155            print(f"Sentiment: {sentiment}")
1156            print(f"Confidence: {confidence:.2f}")
1157            print(f"Score: {score:.2f}")
1158            print(f"Bias Flags: {', '.join(bias_flags)}")
1159            print(f"Demographic Language: {', '.join(demographics)}")
1160            print(f"Recommendation: {recommendation}")
1161
1162            # Save to file
1163            with open("output.txt", "w") as f:
1164                f.write(result.json())
1165
1166            # Print message
1167            print("Analysis completed successfully!")
1168
1169            # Exit loop
1170            break
1171
1172
1173            # Print output
1174            print(f"Text: {text}")
1175            print(f"Sentiment: {sentiment}")
1176            print(f"Confidence: {confidence:.2f}")
1177            print(f"Score: {score:.2f}")
1178            print(f"Bias Flags: {', '.join(bias_flags)}")
1179            print(f"Demographic Language: {', '.join(demographics)}")
1180            print(f"Recommendation: {recommendation}")
1181
1182            # Save to file
1183            with open("output.txt", "w") as f:
1184                f.write(result.json())
1185
1186            # Print message
1187            print("Analysis completed successfully!")
1188
1189            # Exit loop
1190            break
1191
1192
1193            # Print output
1194            print(f"Text: {text}")
1195            print(f"Sentiment: {sentiment}")
1196            print(f"Confidence: {confidence:.2f}")
1197            print(f"Score: {score:.2f}")
1198            print(f"Bias Flags: {', '.join(bias_flags)}")
1199            print(f"Demographic Language: {', '.join(demographics)}")
1200            print(f"Recommendation: {recommendation}")
1201
1202            # Save to file
1203            with open("output.txt", "w") as f:
1204                f.write(result.json())
1205
1206            # Print message
1207            print("Analysis completed successfully!")
1208
1209            # Exit loop
1210            break
1211
1212
1213            # Print output
1214            print(f"Text: {text}")
1215            print(f"Sentiment: {sentiment}")
1216            print(f"Confidence: {confidence:.2f}")
1217            print(f"Score: {score:.2f}")
121
```

```
■ RECOMMENDATION:  
    ▲ LOW CONFIDENCE: Consider human review before using result | ▲ DEMOGRAPHIC LANGUAGE DETECTED: Verify model fairness across groups  
=====  
AGGREGATE BIAS ANALYSIS  
=====  
  
Sentiment Distribution:  
Positive: 2/8 (25.0%)  
Neutral: 5/8 (62.5%)  
Negative: 1/8 (12.5%)  
  
Average Confidence: 5.85%  
  
Most Common Bias Issues:  
• Demographic language detected: age, gender (2 occurrences)  
• Cultural/contextual markers: hyperbolic (1 occurrences)  
• Cultural/contextual markers: emoji_dependent (1 occurrences)  
=====  
MITIGATION RECOMMENDATIONS:  
=====  


1. COLLECT DIVERSE DATA:
  - Include multiple languages, cultures, demographics
  - Balance sentiment classes
  - Ensure representation of all user groups
2. IMPROVE PREPROCESSING:
  - Better sarcasm and negation detection
  - Handle emojis and modern language
  - Normalize cultural variations
3. ROBUST EVALUATION:
  - Test across demographic groups
  - Use fairness metrics (group calibration, equalized odds)
  - Conduct user studies with diverse participants
4. ONGOING MONITORING:
  - Track prediction distribution over time
  - Detect feedback loops
  - Audit decisions regularly
5. TRANSPARENCY:
  - Report confidence scores
  - Explain model limitations
  - Allow human review for important decisions

```

## 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.

```

1  """
2  Ethical AI Product Recommendation System
3  This module demonstrates best practices for building fair, transparent, and
4  user-respecting recommendation systems with ethical AI guidelines.
5  """
6
7  import json
8  import math
9  from datetime import datetime
10 from typing import List, Dict, Tuple, Optional
11 from dataclasses import dataclass, field
12 from collections import defaultdict
13
14 # ===== DATA STRUCTURES =====
15
16 @dataclass
17 class Product:
18     """Represents a product in the recommendation system."""
19     id: str
20     name: str
21     category: str
22     price: float
23     rating: float
24     vendor: str
25     features: Dict[str, str]
26
27
28 @dataclass
29 class UserProfile:
30     """
31     User profile for recommendations.
32
33     ETHICAL PRINCIPLE - USER AUTONOMY:
34     - Store only necessary user data
35     - Allow users to opt-out of tracking
36     - Enable data deletion and portability
37     - Be transparent about what data we collect
38     """
39     user_id: str
40     name: str
41     preferences: Dict[str, float] # category -> preference score (0-1)
42     budget_range: Tuple[float, float]
43     past_purchases: List[str] = field(default_factory=list)
44     opt_in_recommendations: bool = True
45     opt_in_tracking: bool = True
46     feedback_history: List[RecommendationFeedback] = field(default_factory=list)
47
48

```

Done! I've created a comprehensive ethical AI product recommendation system with extensive documentation on transparency, fairness, and user feedback.

**Key Ethical Principles Implemented:**

- 1. Transparency:**
  - Every recommendation includes detailed explanations for why it was suggested
  - Confidence scores disclosed for each recommendation
  - Potential biases clearly identified and explained
  - Users understand the reasoning behind each suggestion
- 2. Fairness:**
  - Diversity constraints prevent filter bubbles (recommending only familiar categories)
  - Diverse recommendations across categories and vendors
  - Vendor monopoly prevention
  - Price fairness (balance across price points, not just expensive items)
- 3. User Autonomy:**
  - Opt-in/opt-out controls for recommendations and tracking

```

PS C:\Users\HP\Downloads\ai_assistant_coding_68 & C:\Users\HP\AppData\Local\Programs\Python\Python311\python.exe c:/Users/HP/Downloads/ai_assistant_coding_68/task3.py
=====
ETHICAL AI PRODUCT RECOMMENDATION SYSTEM
=====

RECOMMENDATIONS FOR ALICE
=====

#1: Premium Laptop
Price:      $1299.99
Rating:     4.8/5.0
Confidence: 95.0%
Diversity:  50.0%

✓ WHY THIS RECOMMENDATION:
• You have strong interest in Electronics products
• Highly rated (4.8/5 stars)
• Includes popular features in this category

⚠ POTENTIAL BIASES:
• Limited user history - recommendation may not be accurate
• High-rated items may be over-recommended (popular item bias)

▢ ALTERNATIVES:
• Budget Laptop ($399.99)

RECOMMENDATIONS FOR BOB
=====

#1: Running Shoes
Price:      $149.99
Rating:     4.6/5.0
Confidence: 82.5%
Diversity:  33.3%

✓ WHY THIS RECOMMENDATION:
• You have strong interest in Sports products
• Price ($149.99) fits your budget range
• Highly rated (4.6/5 stars)
• Includes popular features in this category

⚠ POTENTIAL BIASES:
• Limited user history - recommendation may not be accurate
• High-rated items may be over-recommended (popular item bias)

▢ ALTERNATIVES:
• Coffee Maker ($59.99)
• Wireless Headphones ($89.99)

```

## **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.

```
1 """
2 Secure Logging Functionality for Python Web Applications
3 This module demonstrates best practices for logging in web applications while
4 protecting sensitive user information and maintaining privacy compliance.
5 """
6
7 import logging
8 import logging.handlers
9 import re
10 import json
11 from typing import Dict, Any, Optional
12 from datetime import datetime
13 from functools import wraps
14 import hashlib
15
16
17 # ===== SENSITIVE DATA PATTERNS =====
18
19 # Patterns to detect and redact sensitive information
20 SENSITIVE_PATTERNS = {
21     # Credit card patterns (simplified)
22     'credit_card': r'\b\d{4}[\s-?\d]{4}[\s-?\d]{4}[\s-?\d]{4}\b',
23
24     # Email addresses
25     'email': r'\b[A-Za-z0-9_.+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b',
26
27     # Phone numbers
28     'phone': r'\b(?:\+1[-.\s]?){2}(?:[0-9]{3}){3}[-.\s]?[0-9]{3}[-.\s]?[0-9]{4}\b',
29
30     # Social Security Numbers (US)
31     'ssn': r'\b(?:[0-9]{3}[6]{3})[0-9]{3}-?(?:[0-9]{2})[0-9]{2}-?(?:[0-9]{4})[0-9]{4}\b',
32
33     # API keys and tokens
34     'api_key': r'[Aa]pi[_-]?[Kk]ey["\']?\s*[:=]\s*["\']?[A-Za-z0-9][20_]\b',
35
36     # Passwords in common formats
37     'password': r'^(?i)(password|passwd|pwd)["\']?\s*[:=]\s*["\']?[^\s"\']]+',
38
39     # Bearer tokens
40     'bearer_token': r'[Bb]earer\s+[A-Za-z0-9_-]+',
41
42     # Database connection strings
43     'db_connection': r'^(?i)(user|password|host)=(["\s"]+)',
44
45     # IPv4 addresses (less sensitive but can be PII)
46     'ip4v': r'\b(?:25[0-5]|2[0-4][0-9]|0[1-9][0-9]-[0-9])\.(?:(?:25[0-5]|2[0-4][0-9]|0[1-9][0-9]-[0-9])\b|
```

```
PS C:\Users\HP\Downloads\ai_assistant_coding_68> & C:\Users\HP\AppData\Local\Programs\Python\Python311\python.exe c:/Users/HP/Downloads/ai_assistant_coding_68/task4.py
=====
SECURE LOGGING FOR PYTHON WEB APPLICATIONS
=====

1 LOGGING SCENARIOS:
=====

1 USER LOGIN LOGGING:
[2026-01-22 14:00:08,862] INFO - web_app - User HASH:f9e8e37d2e825eb0 logged in successfully
[2026-01-22 14:00:08,864] WARNING - web_app - Failed login attempt for user HASH:f9e8e37d2e825eb0
    ✓ Logged (sensitive email hashed)

2 API REQUEST LOGGING:
[2026-01-22 14:00:08,865] INFO - web_app - API GET /api/users/profile by HASH:f9e8e37d2e825eb0
    ✓ Logged (user ID hashed)

3 DATA ACCESS LOGGING:
[2026-01-22 14:00:08,866] INFO - web_app - User HASH:f9e8e37d2e825eb0 performed READ on payment_records
    ✓ Logged (sensitive access tracked)

4 ERROR LOGGING WITH CONTEXT:
[2026-01-22 14:00:08,867] ERROR - web_app - Error for user HASH:4e920dc577a96695: Payment processing failed
    ✓ Logged (sensitive fields automatically redacted)

5 SECURITY EVENT LOGGING:
[2026-01-22 14:00:08,868] ERROR - web_app - SECURITY EVENT [BRUTE_FORCE_ATTEMPT]: Multiple failed login attempts from IP [REDACTED]
    ✓ Logged (security incident tracked)

6 SENSITIVE DATA REDACTION EXAMPLES:
=====

Original: User payment card 4532-1234-5678-9010 was processed
Redacted: User payment card [REDACTED] was processed

Original: API key: sk_live_51234567890abcdef was used
Redacted: API key: sk_live_51234567890abcdef was used

Original: Password reset for user@example.com successful
Redacted: Password reset for [REDACTED] successful

Original: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9 authenticated
Redacted: [REDACTED] authenticated

Original: SSN 123-45-6789 verified
Redacted: SSN [REDACTED] verified

Original: Contact: +1-800-555-0123 or john.doe@company.com
Redacted: Contact: +[REDACTED] or [REDACTED]
```

## 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).

```

task5.py > ...
225
226
227
228 @dataclass
229 class ApplicantProfile:
230     """Loan applicant profile for demonstration."""
231     applicant_id: str
232     age: float
233     income: float
234     credit_score: float
235     employment_years: float
236     protected_group: str # For fairness analysis
237
238
239 @dataclass
240 class PredictionResult:
241     """
242     ML prediction result with full transparency.
243
244     TRANSPARENCY PRINCIPLE:
245     - Every prediction includes explanation
246     - Show confidence/probability
247     - Disclose factors contributing to decision
248     - Include fairness assessment
249     - Document model limitations
250
251     applicant_id: str
252     prediction: int # 0: Denied, 1: Approved
253     probability: float # Confidence score 0-1
254     explanation: List[str] # Why this decision
255     fairness_flag: bool # Potential fairness concern
256     fairness_reason: Optional[str]
257     appeal_available: bool # Can applicant appeal
258     model_version: str
259
260
261 @dataclass
262 class FairnessMetrics:
263     """Comprehensive fairness metrics."""
264     overall_accuracy: float
265     group_accuracies: Dict[str, float]
266     approval_rates: Dict[str, float] # Demographic parity
267     false_negative_rates: Dict[str, float] # Equalized odds
268     false_positive_rates: Dict[str, float] # Equalized odds
269     demographic_parity_difference: float # Max difference in approval rates
270     equalized_odds_difference: float # Max difference in error rates

```

Create a machine learning model in Python and document how to use it responsibly, including limitations and fairness considerations.

✓ Created Python file  
Done! I've created a comprehensive responsible ML model with extensive documentation on fairness and limitations.

**Key Features:**

**Model Card (Comprehensive Documentation):**

- Clear intended use and limitations
- Critical warnings about fairness concerns
- Known biases and mitigation strategies
- Regulatory considerations (Fair Housing Act, GDPR, CCPA, etc)
- Deployment requirements and monitoring

**Fairness Metrics:**

- Demographic Parity (approval rates across groups)
- Equalized Odds (false positive/negative rates)
- Disparate Impact analysis
- Group-level accuracy tracking
- Automatic fairness alert flags

**Transparency Features:**

- Human-readable explanations for every prediction
- Confidence scores disclosed
- Fairness concerns flagged for human review
- Rights information (appeals, transparency)

task5.py

Describe what to build next

```
PS C:\Users\HP\Downloads\ai_assistant_coding_68> ^C
PS C:\Users\HP\Downloads\ai_assistant_coding_68> C:/Users/HP/Downloads/ai_assistant_coding_68/.venv/Scripts/python.exe C:/Users/HP/Downloads/ai_assistant_coding_68/task5.py

=====
RESPONSIBLE MACHINE LEARNING MODEL
=====
```

LOAN ELIGIBILITY MODEL CARD

**MODEL OVERVIEW:**

---

Name: Loan Eligibility Classifier v1.0  
Type: Binary Classification (RandomForestClassifier)  
Training Date: 2026-01-22  
Purpose: Predict loan eligibility for demonstration purposes  
Intended Use: DEMONSTRATION ONLY - Not for production lending decisions

**INTENDED USE:**

---

✓ DO USE FOR:

- Educational demonstrations
- Understanding ML fairness concepts
- Testing and validation workflows
- Fairness auditing techniques

✗ DO NOT USE FOR:

- Actual lending decisions
- Production financial services
- High-stakes decisions affecting individuals
- Autonomous decision-making without human review

**CRITICAL LIMITATIONS:**

---

1. BIASED DATA:
  - Training data contains historical lending patterns
  - Reflects past discrimination and biases
  - May perpetuate unfair decisions
2. INCOMPLETE INFORMATION:
  - Only uses demographic and income features
  - Missing important factors (credit history, employment stability)
  - Cannot account for life circumstances
3. MODEL LIMITATIONS:
  - Assumes historical patterns predict future outcomes
  - Cannot capture economic changes or individual circumstances
  - Oversimplifies complex financial decisions
4. FAIRNESS CONCERN:
  - Model may have disparate impact on protected groups