

Assignment-5.4

Task-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 the GitHub Copilot interface. On the left, the code editor displays Task1.py with comments and functions for collecting user data. On the right, the AI panel provides tips for data collection and anonymization, lists key features, and shows a preview of the generated code.

```
Task1.py > ...
1  """
2  User Data Collection Script with Data Protection and Anonymization Guidelines
3  This script demonstrates how to collect user data while implementing security
4  and privacy best practices.
5  """
6
7  import hashlib
8  import re
9  from datetime import datetime
10 import json
11
12 # =====
13 # DATA COLLECTION FUNCTIONS
14 # =====
15 #
16
17 def collect_user_data():
18     """
19         Collect user information with input validation.
20
21         Data Protection Note: Always validate and sanitize user inputs to prevent
22         injection attacks and ensure data quality.
23     """
24     print("\n==== User Data Collection Form ===\n")
25
26     # Collect Name
27     while True:
28         name = input("Enter your full name: ").strip()
29         # Validation: Check if name is not empty and contains only letters and spaces
30         if name and all(c.isalpha() or c.isspace() for c in name):
31             break
32         print("Invalid input. Please enter a valid name (letters and spaces only)")
33
34     # Collect Age
35     while True:
36         try:
37             age = int(input("Enter your age: "))
38
39     # Collect Email
40     while True:
41         try:
42             email = input("Enter your email address: ")
43             if "@" in email and ".com" in email:
44                 break
45             print("Invalid email address. Please enter a valid email address (e.g., example@gmail.com)")
46
47     # Hash Email
48     hashed_email = hashlib.sha256(email.encode()).hexdigest()
49
50     # Create Original User Data JSON
51     original_data = {
52         "Name": name,
53         "Age": age,
54         "Email": email,
55         "EmailHash": hashed_email
56     }
57
58     # Create Anonymized User Data JSON
59     anonymized_data = {
60         "User ID": "4954da2894a2e1d6...", # unique anonymous hash
61         "Age Range": "18-25",
62         "Timestamp": "2026-01-22T13:00:00" # rounded to hour
63     }
64
65     # Save Original Data
66     with open("user_data_original.json", "w") as f:
67         json.dump(original_data, f)
68
69     # Save Anonymized Data
70     with open("user_data_anonymized.json", "w") as f:
71         json.dump(anonymized_data, f)
72
73     # Print Summary
74     print("Original data saved to: user_data_original.json")
75     print("WARNING: This file contains PII and should be encrypted!")
76     print("Anonymized data saved to: user_data_anonymized.json")
77     print("This file is privacy-safe for analytics and sharing.")
78
79     # =====
80     # Data collection and anonymization complete!
81     # =====
```

USER DATA COLLECTION SCRIPT WITH ANONYMIZATION TIPS

generate a Python script that collects user data (e.g., name, age, email). Try to add comments on how to anonymize or protect the data

✓ Created Python script for user data collection

I'll create a Python script that collects user data with comprehensive comments on data protection and anonymization practices.

✓ Created Task1.py

I've created a comprehensive Python script that collects user data with extensive comments on data protection and anonymization. Here's what's included:

Key Features:

1. **Data Collection** - Validates name, age, and email inputs with proper error handling
2. **Data Protection Comments** covering:
 - o Input validation and sanitization
 - o Email verification best practices
 - o Password hashing strategies (bcrypt, argon2)
 - o Encryption recommendations (AES-256)
 - o Database security measures
 - o GDPR/HIPAA compliance notes
3. **Anonymization Techniques:**
 - o Email hashing (SHA-256) for creating unique, irreversible IDs
 - o Age binning to prevent re-identification

Task1.py

Describe what to build next

Agent Auto

OUTPUT:

The terminal window shows the command to run the script and the resulting output. The output details the user data collection process, including the original user data (with PII), the anonymized user data (using unique hashes for email and timestamp), and a warning about the original data containing PII.

```
PS C:\Users\tloke\OneDrive\Documents\AI_79_Assignments> & C:/Users/tloke/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users/tloke/OneDrive/Documents/AI_79_Assignments/Task1.py
```

```
=====
USER DATA COLLECTION AND PRIVACY DEMONSTRATION
=====

==== User Data Collection Form ====

Enter your full name: LOKESH NAIDU
Enter your age: 20
Enter your email address: lokesh@gmail.com

--- Original User Data (with PII) ---
Name: LOKESH NAIDU
Age: 20
Email: lokesh@gmail.com
Email (masked): l****h@gmail.com

--- Anonymized User Data (privacy-safe) ---
User ID: 4954da2894a2e1d6... (unique anonymous hash)
Age Range: 18-25
Timestamp: 2026-01-22T13:00:00 (rounded to hour)

✓ Original data saved to: user_data_original.json
WARNING: This file contains PII and should be encrypted!
✓ Anonymized data saved to: user_data_anonymized.json
This file is privacy-safe for analytics and sharing.

=====
Data collection and anonymization complete!
=====
```

```
PS C:\Users\tloke\OneDrive\Documents\AI_79_Assignments>
```

Task2: Ask Copilot to generate a Python function for sentiment analysis. Then prompt Copilot to identify and handle potential biases in the data.

The screenshot shows the Microsoft Copilot interface. On the left, a code editor displays a Python script named Task02.py. The script includes comments explaining various bias detection and mitigation techniques such as dataset balancing, masking offensive terms, and avoiding forced classifications. On the right, a sidebar titled "USER DATA COLLECTION SCRIPT WITH ANONYMIZATION TIPS" provides instructions for creating a comprehensive sentiment analysis function. Below the sidebar, a message from Copilot states: "I'll create a comprehensive sentiment analysis script for Task02.py with detailed comments on bias identification and mitigation strategies." A "Run" button is present, along with an "Allow" or "Skip" option for executing the script.

```

1 Task02.py > ...
2 """
3 Advanced Sentiment Analysis with Bias Detection and Mitigation
4 -----
5 This module demonstrates sentiment analysis with comprehensive strategies
6 for balancing and mitigating data biases. It includes techniques for:
7 - Balancing datasets to prevent majority class bias
8 - Masking/removing offensive terms that introduce demographic bias
9 - Avoiding forced sentiment classification (allowing neutral/uncertain classifications)
10 - Detecting and flagging potentially biased predictions
11 """
12
13 import re
14 from collections import Counter
15 from typing import Dict, List, Tuple, Optional
16 from dataclasses import dataclass
17 from random import shuffle, seed
18
19
20 # -----
21 # DATA STRUCTURES FOR SENTIMENT ANALYSIS
22 #
23
24 @dataclass
25 class SentimentResult:
26     """
27     Data structure for sentiment analysis results with bias metadata.
28
29     This structure ensures we track not just the prediction, but also:
30     - Confidence levels (to avoid forced classifications)
31     - Bias warnings and flags
32     - Processing information (cleaned text, masked terms)
33     """
34
35     text: str
36     original_text: str
37     sentiment_label: str # 'positive', 'negative', 'neutral', 'uncertain'
38     confidence_score: float # 0.0-1.0, None means insufficient confidence
39     word_scores: Dict[str, float] # Individual word sentiment contributions
40     offensive_terms_masked: List[str]
41     text_was_cleaned: bool
42     bias_warnings: List[str]
43     is_forced_classification: bool # True if forced despite low confidence
44
45     #

```

OUTPUT:

The screenshot shows the output of the generated Python script. It starts with a section on "DATASET BALANCING (Addressing Majority Class Bias)" which compares the distribution before and after a hybrid balancing method. The "BEFORE BALANCING (Imbalanced)" section shows 5 positive (62.5%), 2 negative (25.0%), and 0 neutral (0.0%) samples. The "AFTER BALANCING (Hybrid method)" section shows 3 positive (33.3%), 3 negative (33.3%), and 0 neutral (0.0%) samples. Following this is a "DATASET BIAS ANALYSIS" section that analyzes 8 texts. It reports a 50% positive distribution and identifies one occurrence of the offensive term 'stupid'. It also provides recommendations to remove or mask such terms. The final section is "BIAS MITIGATION BEST PRACTICES", which lists three main categories: CONFIDENCE-BASED CLASSIFICATION, DATASET BALANCING, and OFFENSIVE TERM HANDLING, each with specific guidelines.

2. DATASET BALANCING (Addressing Majority Class Bias)

BEFORE BALANCING (Imbalanced):

- Positive : 5 samples (62.5%)
- Negative : 2 samples (25.0%)
- Neutral : 0 samples (0.0%)

AFTER BALANCING (Hybrid method):

- Positive : 3 samples (33.3%)
- Negative : 3 samples (33.3%)
- Neutral : 0 samples (0.0%)

3. DATASET BIAS ANALYSIS

```
Total texts analyzed: 8

Sentiment distribution:
  Positive : 50.0%
  Uncertain : 25.0%
  Negative : 25.0%

⚠️ Offensive terms found:
  - 'stupid': 1 occurrence(s)

💡 Recommendations:
  - Remove or mask offensive terms from dataset
```

BIAS MITIGATION BEST PRACTICES

- 1. CONFIDENCE-BASED CLASSIFICATION (NOT Forced Classification)**
 - ✓ Use 'uncertain' for low-confidence predictions (default behavior)
 - ✓ Never force sentiment labels on ambiguous text
 - ✗ Avoid: Always forcing every text into positive/negative/neutral
- 2. DATASET BALANCING**
 - ✓ Ensure roughly equal samples per sentiment class
 - ✓ Use hybrid approach: balance between upsampling and downsampling
 - ✗ Avoid: Training on heavily imbalanced data (>70% one class)
- 3. OFFENSIVE TERM HANDLING**
 - ✓ Detect and mask offensive/derogatory language
 - ✓ Remove terms that correlate with discrimination
 - ✗ Avoid: Including derogatory terms in sentiment scoring

Task3: Use Copilot to write a Python program that recommends products based on user history. Ask it to follow ethical guidelines like transparency and fairness.

```
Task3.py > ...
1 """
2 Ethical AI Product Recommendation System
3 -----
4
5 This module implements a product recommendation engine with built-in fairness checks,
6 transparency, and ethical AI principles. It addresses key challenges in recommendation systems:
7
8 FAIRNESS CHALLENGES:
9 1. Brand Bias: Recommending expensive/popular brands disproportionately
10 2. Category Bias: Over-recommending certain product categories
11 3. Filter Bubbles: Only showing products similar to past purchases
12 4. Cold Start Problem: Difficulty recommending to new users
13 5. Diversity: Ensuring variety in recommendations, not just safe choices
14
15 ETHICAL PRINCIPLES IMPLEMENTED:
16 - TRANSPARENCY: Users can see why products are recommended
17 - FAIRNESS: Diversity metrics prevent brand/category favoritism
18 - USER CONTROL: Feedback mechanism gives users influence over recommendations
19 - EXPLAINABILITY: Clear explanations of recommendation reasoning
20 - PRIVACY: No actual user data tracking (simplified model)
21
22
23 from dataclasses import dataclass, field
24 from typing import Dict, List, Tuple, Optional, Set
25 from collections import Counter
26 import random
27 from enum import Enum
28
29
30 # -----
31 # DATA STRUCTURES
32 # -----
33
34 class ProductCategory(Enum):
35     """Product categories to prevent over-recommending same category."""
36     ELECTRONICS = "Electronics"
37     CLOTHING = "Clothing"
38     HOME = "Home & Garden"
39     BOOKS = "Books"
40     SPORTS = "Sports & Outdoors"
41     FOOD = "Food & Beverages"
42     HEALTH = "Health & Beauty"
43     TOYS = "Toys & Games"
44
45
```

The screenshot shows the Microsoft Copilot interface. On the left, the code for Task3.py is displayed. On the right, a sidebar provides instructions: "Write a Python program that recommends products based on user purchase history. Include clear comments explaining how recommendations work, add a fairness check to avoid favoritism toward one brand or category, and provide an option for user feedback (like/dislike) to improve future recommendations. Ensure the code follows ethical AI principles such as transparency and fairness." Below this, a message says "Created Python file". A command-line input field contains "python Task3.py" with "Allow" and "Skip" buttons. At the bottom, there's a "Describe what to build next" field and some agent settings.

OUTPUT:

```

1. USER PROFILE AND PURCHASE HISTORY
-----
User ID: user_123
Purchase history (5 items):
• Wireless Headphones (Electronics) - $79.99 - Rating: 4.5/5
• Running Shoes (Clothing) - $99.99 - Rating: 4.7/5
• Vitamin C Serum (Health & Beauty) - $39.99 - Rating: 4.6/5
• Python Programming (Books) - $49.99 - Rating: 4.8/5
• Yoga Mat (Sports & Outdoors) - $24.99 - Rating: 4.7/5

2. FAIR RECOMMENDATIONS (WITH FAIRNESS CONSTRAINTS)
-----
Top 5 Recommendations:

1. Bicycle Helmet
  Brand: SafeHelm | Category: Sports & Outdoors
  Price: $59.99 | Rating: 4.6/5
  Relevance Score: 0.73
  Diversity Score: 0.50
  Final Score: 0.73
  Confidence: 0.78
  Why? Popular category in your purchase history • Highly rated (4.6/5 stars) • Similar price to your typical purchases

2. Jeans
  Brand: DenimCo | Category: Clothing
  Price: $59.99 | Rating: 4.3/5
  Relevance Score: 0.71
  Diversity Score: 0.50
  Final Score: 0.71
  Confidence: 0.76
  Why? Popular category in your purchase history • Similar price to your typical purchases

3. Laptop Stand
  Brand: ErgoWorks | Category: Electronics
  Price: $39.99 | Rating: 4.6/5
  Relevance Score: 0.67
  Diversity Score: 0.50
  Final Score: 0.67
  Confidence: 0.78
  Why? Popular category in your purchase history • Highly rated (4.6/5 stars) • Similar price to your typical purchases

4. Face Moisturizer
  Brand: BeautyGlow | Category: Health & Beauty
  Price: $34.99 | Rating: 4.5/5
  Relevance Score: 0.64
  Diversity Score: 0.93
  Final Score: 0.73
  Confidence: 0.77
  Why? Popular category in your purchase history • Highly rated (4.5/5 stars) • Similar price to your typical purchases

5. Cookbook
  Brand: FoodBiz | Category: Books
  Price: $29.99 | Rating: 4.4/5
  Relevance Score: 0.62
  Diversity Score: 0.92
  Final Score: 0.71
  Confidence: 0.76

```

Task4: Prompt Copilot to generate logging functionality in a Python web application. Then, ask it to ensure the logs do not record sensitive information.

```

❶ Task4.py > ...
1   """
2   Ethical Web Application Logging System
3   -----
4
5   This module provides logging functionality designed with privacy and ethics in mind.
6   Key principles implemented:
7
8   ETHICAL LOGGING PRINCIPLES:
9   1. DATA MINIMIZATION: Only log what's necessary for operations/debugging
10  2. PRIVACY PROTECTION: Never log sensitive/personal information (PII)
11  3. TRANSPARENCY: Clear documentation of what is/isn't logged
12  4. COMPLIANCE: Follows GDPR, CCPA, and similar privacy regulations
13  5. USER TRUST: Users have right to know their data isn't exposed in logs
14  6. SECURITY: Logs are a security risk if they contain sensitive data
15
16  SENSITIVE DATA CATEGORIES EXCLUDED:
17  - Authentication: Passwords, API keys, tokens, session IDs
18  - Personal Information: Names, emails, phone numbers, addresses
19  - Identifiers: User IDs, account numbers, social security numbers
20  - Financial: Credit card numbers, bank account details
21  - Medical: Health information, medications
22  - Biometric: Fingerprints, facial recognition data
23  - Location: GPS coordinates, precise home addresses
24
25  WHY EXCLUDE SENSITIVE DATA:
26  1. SECURITY RISK: Logs can be accessed by unauthorized users, stolen, or exposed
27  2. COMPLIANCE RISK: GDPR and CCPA impose fines for exposing personal data
28  3. TRUST DAMAGE: Users lose trust if their data is found in logs
29  4. OPERATIONAL RISK: Sensitive data in logs complicates log retention/deletion
30  5. TECHNICAL DEBT: Creates complexity in log access control and archival
31
32  BEST PRACTICES IMPLEMENTED:
33  - Automatic redaction of known sensitive patterns
34  - Structured logging with separate sensitive/non-sensitive fields
35  - Hash user identifiers for correlation without exposure
36  - Log rotation and secure deletion
37  - Audit logging without PII
38  - Developer guidance on safe logging
39  """
40
41 import logging
42 import logging.handlers
43 import re
44 import hashlib
45 import json

```

← USER DATA COLLECTION SCRIPT WITH ANONYMIZATION TIPS

Write Python logging functionality for a web application. Ensure logs do not record sensitive or personal information such as passwords, emails, or user IDs. Add comments explaining ethical logging practices, data privacy, and why sensitive data is excluded from logs.

✓ Created Python file

Perfect! I've created a comprehensive ethical logging system for Task4.py. Let me verify it works:

Run `push` command?

`python Task4.py 2>&1 | head -100`

Allow Skip

Describe what to build next

Agent Auto

OUTPUT:

```

1. SENSITIVE DATA REDACTION

Before and After Redaction:

Original: User logged in with email: john.doe@example.com
Redacted: User logged in with email: [EMAIL_REDACTED]

Original: Password reset requested for user: 123-45-6789
Redacted: Password reset requested for user: [SSN_REDACTED]

Original: API call with key: sk-1234567890abcdef
Redacted: API call with key: sk-1234567890abcdef

Original: Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
Redacted: [AUTH_HEADER_REDACTED]

Original: Payment processed: 4532-1234-5678-9012
Redacted: Payment processed: [CREDIT_CARD_REDACTED]

Original: User phone: (555) 123-4567
Redacted: User phone: [PHONE_REDACTED]

2. DICTIONARY/FORM DATA REDACTION

Original form data:
{
    "username": "john_doe",
    "email": "john@example.com",
    "password": "SecurePass123!",
    "api_key": "sk_test_12345",
    "phone": "555-123-4567",
    "address": "123 Main St"
}

Redacted form data:
{
    "username": "john_doe",
    "email": "[EMAIL_REDACTED]",
    "password": "[REDACTED]",
    "api_key": "[REDACTED]",
    "phone": "[PHONE_REDACTED]",
    "address": "123 Main St"
}

3. USER ANONYMIZATION (Consistent Hashing)

```

Task5: 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).

The screenshot shows a code editor with Python code for a machine learning model and a Copilot interface.

```

53 import numpy as np
54 import json
55 from dataclasses import dataclass
56 from typing import Dict, List, Tuple, Optional
57 from collections import Counter
58 from abc import ABC, abstractmethod
59
60
61 # -----
62 # DATA STRUCTURES
63 #
64
65 @dataclass
66 class Applicant:
67     """
68     Loan applicant data.
69
70     IMPORTANT FAIRNESS NOTE: Do NOT include explicit demographic data
71     (race, gender, national origin) as features, as this would directly
72     violate fair lending laws. Instead:
73     - Use ONLY credit-relevant factors
74     - Monitor for PROXY DISCRIMINATION (e.g., zip code correlating to race)
75     - Audit model predictions across demographic groups separately
76     """
77     applicant_id: str
78     age: int # Age is allowed under ECOA (Equal Credit Opportunity Act)
79     annual_income: float
80     credit_score: int
81     years_employed: int
82     existing_debt: float
83     employment_industry: str
84
85     # Demographic data (NOT used as features, only for bias auditing)
86     # FAIRNESS: Collected separately for fairness testing
87     demographic_group: Optional[str] = None # e.g., "group_a", "group_b"
88
89
90 @dataclass
91 class ModelPrediction:
92     """Prediction result with explainability information."""
93     applicant_id: str
94     prediction: str # "low_risk" or "high_risk"
95     confidence: float # 0.0-1.0, probability of predicted class
96     risk_score: float # 0.0-1.0, higher = more risky

```

The Copilot interface on the right shows the following steps:

- Generate a simple machine learning model in Python. Include clear documentation (inline comments or a README) explaining how the model works, its accuracy limitations, and how to use it responsibly.
- Add notes on explainability, potential bias, and fairness considerations.
- Read Python file
- I'll create a comprehensive machine learning model with extensive documentation on responsible AI.
- Created Task5.py
- Perfect! I've created a comprehensive, responsible machine learning model. Let me verify it runs correctly.
- Run `push` command?
- python Task5.py 2>&1 | head -150
- Allow Skip

At the bottom, there are status indicators: Ln 818, Col 1, Spaces 4, UTF-8, CRLF, { } Python, 3.142, Go Live.

OUTPUT:

```
-----  
RESPONSIBLE MACHINE LEARNING MODEL - CREDIT RISK CLASSIFICATION  
-----  
  
1. DATASET CREATION  
  
Total applicants: 9  
Sample applicants:  
- app_001: Age 35, Income $80000k, Credit Score 750  
- app_002: Age 42, Income $95000k, Credit Score 760  
- app_003: Age 28, Income $60000k, Credit Score 720  
  
2. MODEL TRAINING  
  
✓ Model trained successfully  
✓ Training on 9 applicants  
✓ Features used: Age, Income, Credit Score, Years Employed, Existing Debt  
✓ No sensitive features (gender, race, etc.) used in model  
  
3. PREDICTIONS WITH EXPLAINABILITY  
  
Applicant: app_001  
Prediction: LOW_RISK  
Risk Score: 12.15%  
Confidence: 88%  
Uncertainty: 76%  
Explanation: Model predicts low risk (confidence: 76%) based on: Credit Score decreases risk, Income decreases risk, Ag  
e decreases risk  
Feature Contributions:  
- Credit Score: -1.00 ↓ risk  
- Income: -0.48 ↓ risk  
- Age: -0.28 ↓ risk  
- Years Employed: -0.23 ↓ risk  
- Existing Debt: +0.04 ↑ risk  
  
Applicant: app_002  
Prediction: LOW_RISK  
Risk Score: 8.69%  
Confidence: 91%  
Uncertainty: 83%  
Explanation: Model predicts low risk (confidence: 83%) based on: Credit Score decreases risk, Income decreases risk, Ag  
e decreases risk  
Feature Contributions:
```