

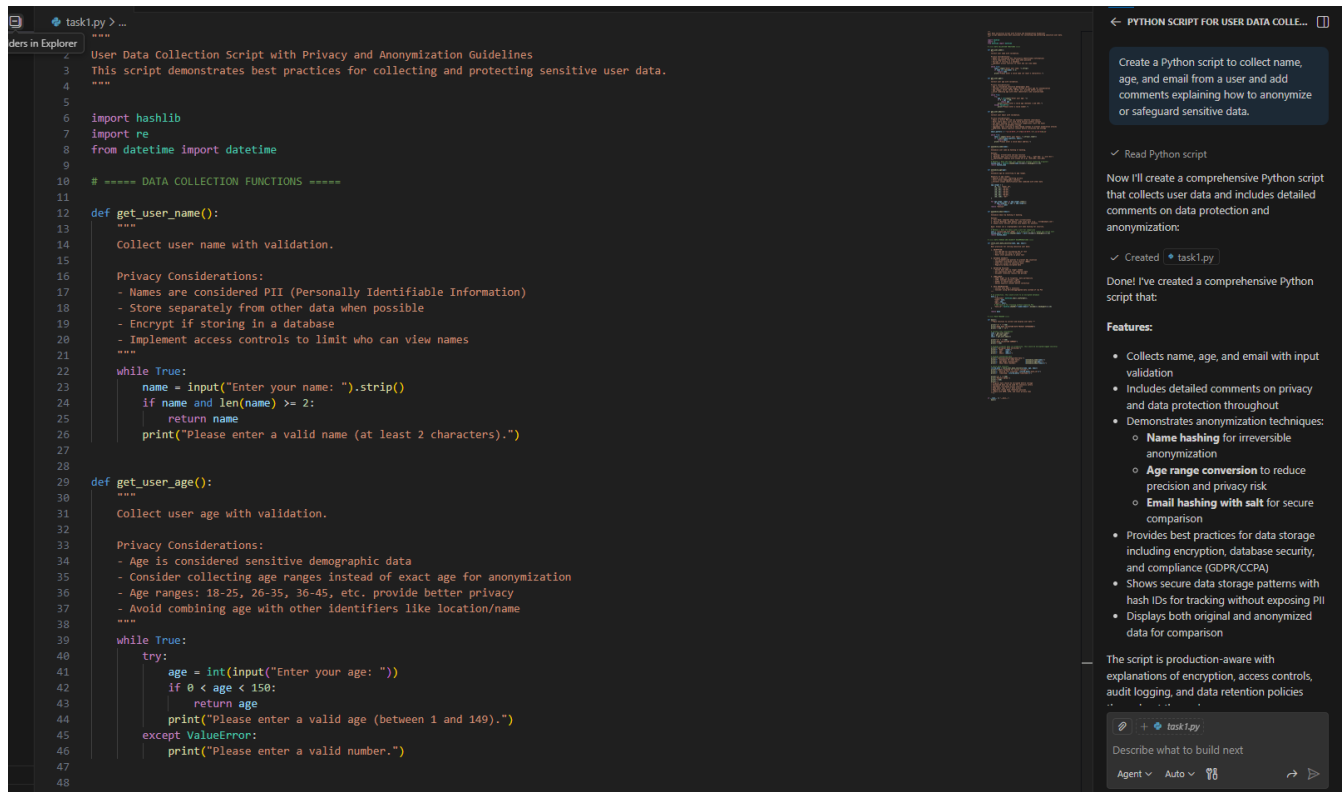
# Assignment: 5.4

Name: B. Sai Laxman

Hno: 2303A51045

## 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.



```
task1.py > ...
User Data Collection Script with Privacy and Anonymization Guidelines
This script demonstrates best practices for collecting and protecting sensitive user data.
'''
5
import hashlib
import re
from datetime import datetime
9
# ===== DATA COLLECTION FUNCTIONS =====
11
def get_user_name():
12     '''
13     Collect user name with validation.
14
15     Privacy Considerations:
16     - Names are considered PII (Personally Identifiable Information)
17     - Store separately from other data when possible
18     - Encrypt if storing in a database
19     - Implement access controls to limit who can view names
20     '''
21
22     while True:
23         name = input("Enter your name: ").strip()
24         if name and len(name) >= 2:
25             return name
26         print("Please enter a valid name (at least 2 characters).")
27
28
def get_user_age():
29     '''
30     Collect user age with validation.
31
32     Privacy Considerations:
33     - Age is considered sensitive demographic data
34     - Consider collecting age ranges instead of exact age for anonymization
35     - Age ranges: 18-25, 26-35, 36-45, etc. provide better privacy
36     - Avoid combining age with other identifiers like location/name
37     '''
38
39     while True:
40         try:
41             age = int(input("Enter your age: "))
42             if 0 < age < 150:
43                 return age
44             print("Please enter a valid age (between 1 and 149).")
45         except ValueError:
46             print("Please enter a valid number.")
47
48
def get_user_email():
49     '''
50     Collect user email with validation.
51
52     Privacy Considerations:
53     - Email addresses are PII
54     - Use hashed email addresses for tracking without exposing PII
55     - Implement access controls
56     '''
57
58     while True:
59         email = input("Enter your email: ").strip()
60         if email and re.match(r'[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}', email):
61             return email
62         print("Please enter a valid email address.")
63
64
def main():
65     '''
66     Main function to collect user data.
67
68     Privacy Considerations:
69     - Collect data in a secure, encrypted database
70     - Implement access controls
71     - Regularly audit data collection and storage
72     - Provide users with options to delete their data
73     - Anonymize data where possible
74     - Use secure communication channels (HTTPS)
75     - Limit data collection to what is necessary
76     - Store data securely (encrypted)
77     - Implement access controls
78     - Regularly audit data collection and storage
79     - Provide users with options to delete their data
80     - Anonymize data where possible
81     - Use secure communication channels (HTTPS)
82     - Limit data collection to what is necessary
83     '''
84
85     get_user_name()
86     get_user_age()
87     get_user_email()
88
89     # Anonymization and Protection
90     # Hash the email address
91     email_hash = hashlib.sha256(email.encode()).hexdigest()
92
93     # Store data in a secure database
94     # Example: Insert into database
95     # Example: Insert into database
96     # Example: Insert into database
97
98     # Log collection
99     print("User data collected successfully.")
100
101 if __name__ == '__main__':
102     main()
```

PYTHON SCRIPT FOR USER DATA COLLE...

Create a Python script to collect name, age, and email from a user and add comments explaining how to anonymize or safeguard sensitive data.

✓ Read Python script

Now I'll create a comprehensive Python script that collects user data and includes detailed comments on data protection and anonymization:

✓ Created task1.py

Done! I've created a comprehensive Python script that:

**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

+ task1.py

Describe what to build next

Agent Auto

```

=====
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:
=====

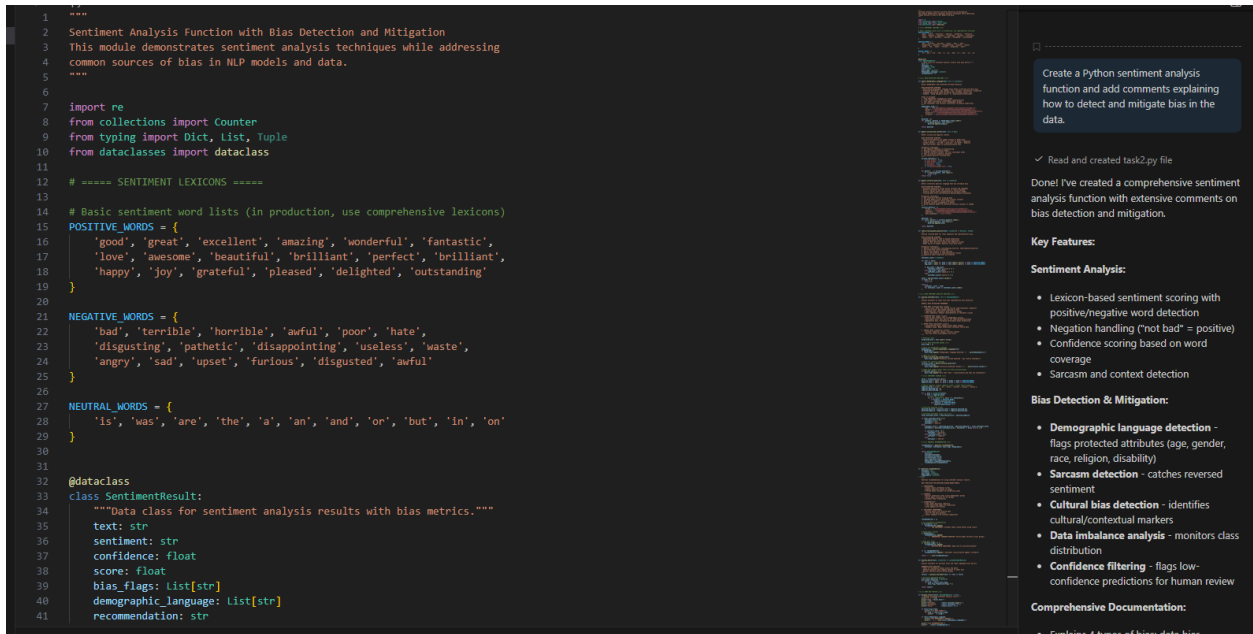
  ✓ Original data should be encrypted before storage
  ✓ Anonymized data can be used for analytics safely
  ✓ Implement role-based access controls
  ✓ Keep audit logs of all data access
  ✓ Regularly review data retention policies
  ✓ Comply with GDPR, CCPA, and local privacy laws

```

## 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.



Create a Python sentiment analysis function and add comments explaining how to detect and mitigate bias in the data.

✓ Read and created task2.py file

Don't I've created a comprehensive sentiment analysis function with extensive comments on bias detection and mitigation.

#### Key Features:

##### Sentiment Analysis:

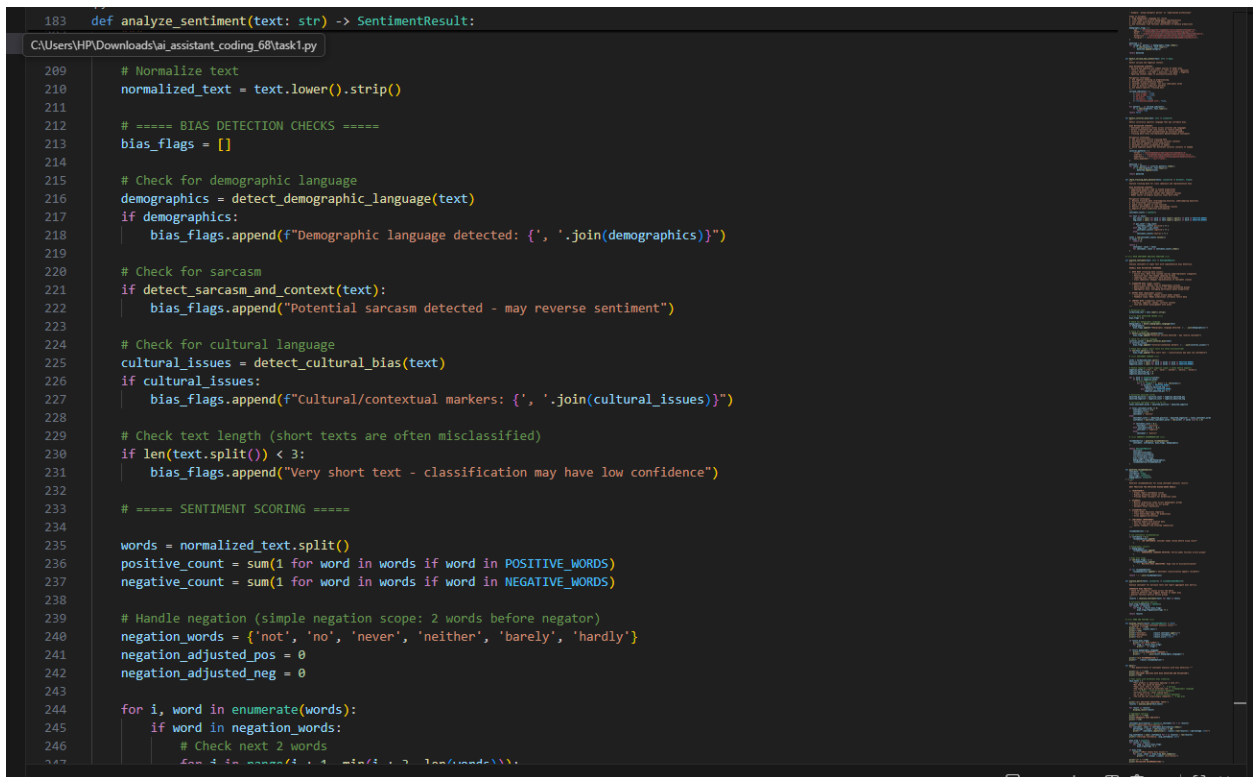
- Lexicon-based sentiment scoring with positive/negative word detection
- Negation handling ("not bad" = positive)
- Confidence scoring based on word coverage
- Sarcasm and context detection

##### Bias Detection & Mitigation:

- **Demographic language detection** - flags protected attributes (age, gender, race, religion, disability)
- **Sarcasm detection** - catches reversed sentiment
- **Cultural bias detection** - identifies cultural/contextual markers
- **Data imbalance analysis** - monitors class distribution
- **Confidence filtering** - flags low-confidence predictions for human review

##### Comprehensive Documentation:

- Explains 4 types of bias data bias



```

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, asdict, field
12 from collections import defaultdict
13
14
15 # ===== DATA STRUCTURES =====
16
17 @dataclass
18 class Product:
19     """Represents a product in the recommendation system."""
20     id: str
21     name: str
22     category: str
23     price: float
24     rating: float
25     vendor: str
26     features: Dict[str, str]
27
28
29 @dataclass
30 class UserProfile:
31     """
32     User profile for recommendations.
33
34     ETHICAL PRINCIPLE - USER AUTONOMY:
35     - Store only necessary user data
36     - Allow users to opt-out of tracking
37     - Enable data deletion and portability
38     - Be transparent about what data we collect
39     """
40     user_id: str
41     name: str
42     preferences: Dict[str, float] # category -> preference score (0-1)
43     budget_range: Tuple[float, float]
44     past_purchases: List[str] = field(default_factory=list)
45     opt_in_recommendations: bool = True
46     opt_in_tracking: bool = True
47     feedback_history: List['RecommendationFeedback'] = field(default_factory=list)
48
49
50 """
51 Create a Python product recommendation program and include ethical AI guidelines such as transparency, fairness, and user feedback in the code.
52
53 ✓ Created Python file
54
55 Done! I've created a comprehensive ethical AI product recommendation system with extensive documentation on transparency, fairness, and user feedback.
56
57 Key Ethical Principles Implemented:
58
59 1. Transparency:
60
61 • Every recommendation includes detailed explanations for why it was suggested
62 • Confidence scores disclosed for each recommendation
63 • Potential biases clearly identified and explained
64 • Users understand the reasoning behind each suggestion
65
66 2. Fairness:
67
68 • Diversity constraints prevent filter bubbles (recommending only familiar categories)
69 • Diverse recommendations across categories and vendors
70 • Vendor monopoly prevention
71 • Price fairness (balance across price points, not just expensive items)
72
73 3. User Autonomy:
74
75 • Opt-in/opt-out controls for recommendations and tracking
76
77 task3.py
78 Describe what to build next
79 Agent Auto ↩ ↘
```

```
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]?)(?:[0-9]{3})\)?[-.\s]?[0-9]{3}[-.\s]?[0-9]{4}\b',
29
30     # Social Security Numbers (US)
31     'ssn': r'\b(?:[0-9]{3})-[0-9]{2}-[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     'ipv4': r'\b(?:?:25[0-5]|2[0-4][0-9]|01[0-9]?[0-9]?[0-9])\.(?:?:25[0-5]|2[0-4][0-9]|01[0-9]?[0-9]?[0-9])\b',
47 }
```

Create logging functionality in a Python web app and ensure logs do not capture sensitive user information.

✓ Created Python file

Done! I've created a comprehensive secure logging system for Python web applications that automatically protects sensitive user information.

**Key Features:**

- Pattern matching for credit cards, emails, phone numbers, SSNs, API keys, passwords, tokens, etc.
- Dictionary field detection (redacts values of fields named "password", "email", etc.)
- Automatic hashing of PII for tracking without exposing identity
- Consistent [REDACTED] placeholders

**Logging Configuration:**

- Console handler for development/monitoring
- Rotating file handler (10MB auto-rotation, keeps 10 backups)
- Optional syslog handler for centralized logging
- Secure formatter that automatically redacts all logs
- Configurable log levels and retention

**Privacy-Aware Logging Functions:**

- `log_user_login()` - Track logins without

task4.py

Describe what to build next.

```

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
=====

📁 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).

taskSpy > ...

C:\Users\HPA\Downloads\ai\_assistant\_coding\_68\secure\_app.log

```
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
```

PYTHON SCRIPT FOR USER DATA COLLE...

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,

taskSpy

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 CONCERNS:

- Model may have disparate impact on protected groups