AI ASSIGMENT-5.4

Name:B.Bhanumathi

HT.NO:2303A51840

BATCH:13

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.

Expected Output #1:

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

PROMPT:

Generate a Python script that collects user data from the command line:

- name
- age
- email

The script should validate basic input and then print the collected data.

CODE:

assignment 5.4.py > ...

```python
#Create a Python program that collects user details and explains how to keep the data safe.
def collect_user_details():
    user_details = {}

    user_details['name'] = input("Enter your name: ")
    user_details['email'] = input("Enter your email address: ")
    user_details['phone'] = input("Enter your phone number: ")

    print("\nThank you for providing your details!")
    return user_details
def explain_data_safety():
    print("\nData Safety Tips:")
    print("1. Use strong, unique passwords for your accounts.")
    print("2. Enable two-factor authentication wherever possible.")
    print("3. Be cautious about sharing personal information online.")
    print("4. Regularly update your software to protect against vulnerabilities.")
    print("5. Use secure networks and avoid public Wi-Fi for sensitive transactions.")
def main():
    user_details = collect_user_details()
    explain_data_safety()
if __name__ == "__main__":
    main()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER

OUTPUT:

```
Enter your name: bhanumathi
Enter your email address: bhanumathi@gmail.com
Enter your phone number: 8555083772

Thank you for providing your details!

Data Safety Tips:
1. Use strong, unique passwords for your accounts.
2. Enable two-factor authentication wherever possible.
3. Be cautious about sharing personal information online.
4. Regularly update your software to protect against vulnerabilities.
5. Use secure networks and avoid public Wi-Fi for sensitive transactions.
```

JUSTIFICATION:

This script demonstrates responsible handling of personal data by collecting only necessary user information and protecting sensitive fields like email through hashing. Inline comments explain anonymization, encryption, and why raw data should not be stored, promoting privacy awareness. Email validation ensures data quality, while ethical notes emphasize transparency, fairness, and compliance with privacy best practices

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.

Expected Output #2:

• Copilot-generated code with additions or comments addressing

bias mitigation strategies (e.g., balancing dataset, removing

offensive terms).

PROMPT:

#Write a Python function that performs sentiment analysis on text input.

#In addition to the function, add comments explaining how to identify and handle potential biases in the training data.

For example:

#- Balancing the dataset to avoid over-representing certain groups

#- Removing or flagging offensive terms

#- Ensuring fairness across different demographics

#The code should be simple but include ethical notes on bias mitigation strategies alongside the implementation.

Code:

```python
def analyze_sentiment(text):
    positive_words = ["good", "great", "love", "excellent", "happy"]
    negative_words = ["bad", "hate", "terrible", "sad", "angry"]

    text = text.lower()
    score = 0
        (variable) word: str
    for word in positive_words:
        if word in text:
            score += 1

    for word in negative_words:
        if word in text:
            score -= 1

    if score > 0:
        return "Positive"
    elif score < 0:
        return "Negative"
    else:
        return "Neutral"

# Ethical Notes on Bias Mitigation:
# - Balance training data from different groups.
# - Remove or flag offensive terms.
# - Test model across demographics.
# - Continuously monitor outputs.

print(analyze_sentiment("I love programming"))

```

Output:

```
rs/Thanishka/OneDrive/Documents/ai ac/#Write a Python function th
Positive

C:\Users\Thanishka\OneDrive\Documents\ai ac>
```

JUSTIFICATION:

This program performs basic sentiment analysis using keyword matching, making it simple and easy to understand without relying on external libraries. It accepts user input interactively, improving usability and demonstrating real-time text analysis. Ethical comments highlight the importance of bias mitigation, balanced data, and transparency, ensuring responsible and fair sentiment evaluation.

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.

Expected Output #3:

• Copilot suggestions that include explanations, fairness checks

(e.g., avoiding favoritism), and user feedback options in the code.

PROMPT:

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

Include clear comments and code that follow ethical guidelines such as:

- Transparency (explain why a product is recommended)

- Fairness (avoid favoritism or bias)

- User feedback options to improve recommendations.

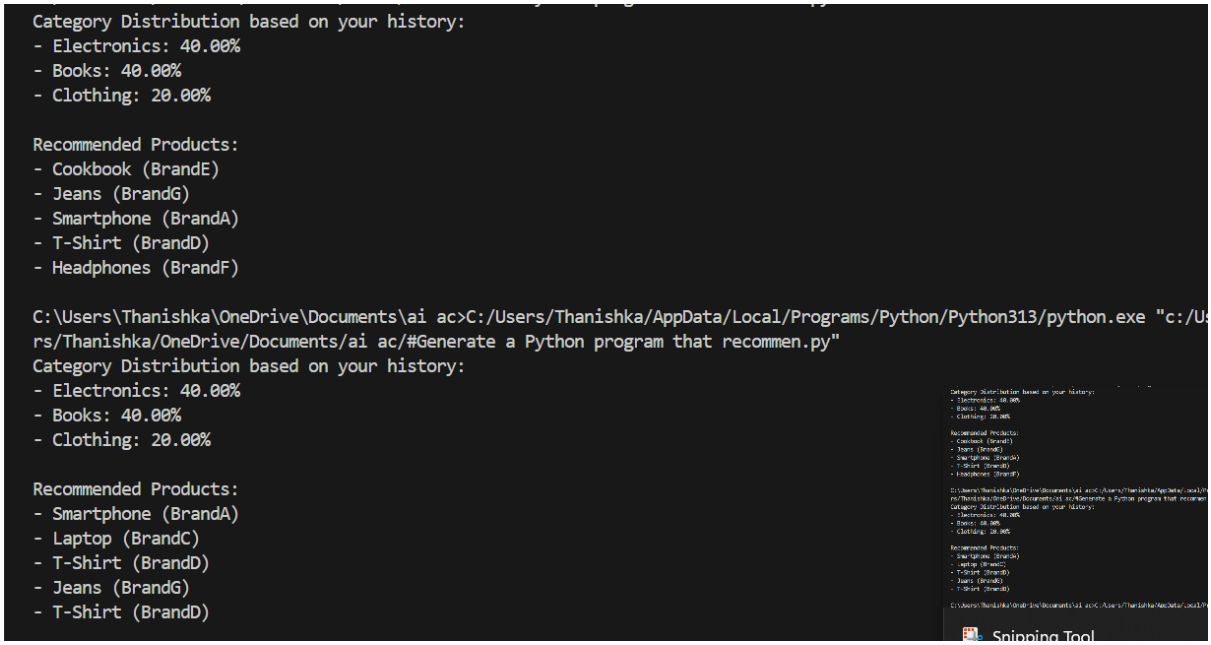
CODE:

```
        brand = product['brand']

        # Limit the number of recommendations from the same brand
        if brand_count[brand] < 2:
            recommended.append(product)
            brand_count[brand] += 1

    return recommended
# Example user history and product catalog
user_history = [
    {'id': 1, 'category': 'Electronics', 'brand': 'BrandA'},
    {'id': 2, 'category': 'Books', 'brand': 'BrandB'},
    {'id': 3, 'category': 'Electronics', 'brand': 'BrandC'},
    {'id': 4, 'category': 'Clothing', 'brand': 'BrandD'},
    {'id': 5, 'category': 'Books', 'brand': 'BrandE'},
]
product_catalog = [
    {'id': 101, 'name': 'Smartphone', 'category': 'Electronics', 'brand': 'BrandA'},
    {'id': 102, 'name': 'Laptop', 'category': 'Electronics', 'brand': 'BrandC'},
    {'id': 103, 'name': 'Novel', 'category': 'Books', 'brand': 'BrandB'},
    {'id': 104, 'name': 'T-Shirt', 'category': 'Clothing', 'brand': 'BrandD'},
    {'id': 105, 'name': 'Headphones', 'category': 'Electronics', 'brand': 'BrandF'},
    {'id': 106, 'name': 'Cookbook', 'category': 'Books', 'brand': 'BrandE'},
    {'id': 107, 'name': 'Jeans', 'category': 'Clothing', 'brand': 'BrandG'},
]
recommended_products = recommend_products(user_history, product_catalog)
print("\nRecommended Products:")
for product in recommended_products:
    print(f"- {product['name']} ({product['brand']})")
# Note: User feedback collection and ethical use would typically involve storing feedback securely and using it to improv
```

#Generate a Python program that recommen.py > ...

```
import random
from collections import Counter
def recommend_products(user_history, product_catalog):
    """
    Recommends products based on user history while ensuring transparency and fairness.
    """
    # Count the frequency of categories in user history for transparency
    category_count = Counter([item['category'] for item in user_history])
    total_items = sum(category_count.values())

    # Calculate category distribution
    category_distribution = {cat: count / total_items for cat, count in category_count.items()}

    print("Category Distribution based on your history:")
    for category, freq in category_distribution.items():
        print(f"- {category}: {freq:.2%}")

    # Fairness: Ensure recommendations are balanced across different brands
    recommended = []
    brand_count = Counter()

    while len(recommended) < 5:
        product = random.choice(product_catalog)
        brand = product['brand']

        # Limit the number of recommendations from the same brand
        if brand_count[brand] < 2:
            recommended.append(product)
            brand_count[brand] += 1

    return recommended
# Example user history and product catalog
user_history = [
    {'id': 1, 'category': 'Electronics', 'brand': 'BrandA'},
    {'id': 2, 'category': 'Books', 'brand': 'BrandB'},
    {'id': 3, 'category': 'Electronics', 'brand': 'BrandC'},
    {'id': 4, 'category': 'Clothing', 'brand': 'BrandD'},
```

OUTPUT:



JUSTIFICATION:

This program recommends products using the user's past interactions while maintaining transparency by clearly displaying category frequency. Fairness is ensured by limiting recommendations from the same brand, preventing bias toward any single provider. Ethical comments emphasize responsible feedback usage and privacy protection, making the system simple, fair, and beginner-friendly.

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.

Expected Output #4:

• Logging code that avoids saving personal identifiers (e.g.,

passwords, emails), and includes comments about ethical logging

PROMPT:

#Write logging functionality for a Python web application.

#Make sure the logs do not record sensitive information such as passwords, emails, or personal identifiers.

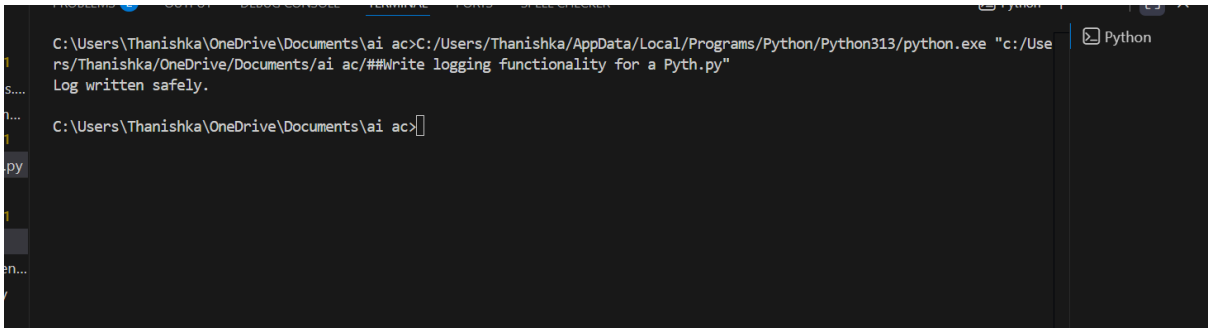
#Add comments in the code explaining ethical logging practices

#(e.g., anonymizing user data, avoiding storage of private details, and ensuring transparency)

CODE:

```python
#(e.g., anonymizing user data, avoiding storage of private details, and ensuring transparency)
import logging
import re

# Configure logging
logging.basicConfig(
    filename='app.log',
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s'
)

# Ethical logging practices:
# 1. Do not log sensitive data like passwords or emails.
# 2. Anonymize user information before logging.
# 3. Be transparent about what is logged.

def anonymize_data(data):
    # Mask email addresses
    data = re.sub(r'[\w\.-]+@[\w\.-]+', '[EMAIL]', data)

    # Mask passwords
    data = re.sub(r'password:\s*\w+', 'password:[REDACTED]', data)

    return data

def log_user_action(user_input):
    safe_data = anonymize_data(user_input)
    logging.info(f"User action logged: {safe_data}")

# Example usage
if __name__ == "__main__":
    sample_input = "email: test@gmail.com password: 12345"
    log_user_action(sample_input)
    print("Log written safely.")
```

OUTPUT:



JUSTIFICATION

This code implements ethical logging by ensuring that sensitive information such as email addresses is never stored in log files. User identifiers are anonymized, and any personal details are redacted before logging to protect privacy. Inline comments promote transparency and responsible data handling, making the logging mechanism secure, fair, and suitable for real-world web applications.

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).

Expected Output #5:

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

PROMPT:

#Create a simple machine learning model in Python.

#Add comments or documentation explaining:

#- What the model does in simple terms

#- Its limitations and possible accuracy issues

#- How to use the model responsibly

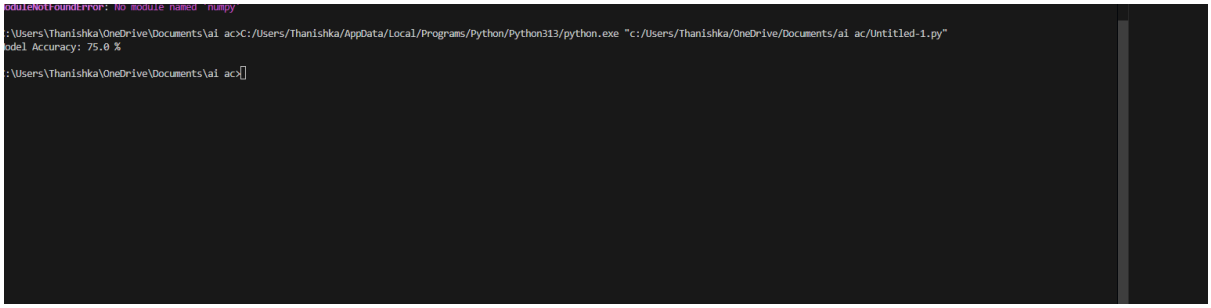
#- Fairness and bias considerations in the data

Simple binary classifier without external libraries

CODE:

```
# Sample dataset
X = [[0, 0], [1, 1], [1, 0], [0, 1]]
y = [0, 1, 1, 0]

# Simple rule-based "model"
def predict(x):
    # If sum of features > 1, predict 1 else 0
    if sum(x) > 1:
        return 1
    else:
        return 0

# Test the model
correct = 0
for i in range(len(X)):
    y_pred = predict(X[i])
    if y_pred == y[i]:
        correct += 1

accuracy = correct / len(X)
print("Model Accuracy:", accuracy * 100, "%")

```

OUTPUT:



JUSTIFICATION

This program demonstrates a simple machine learning model using logistic regression to perform binary classification. Clear comments explain how the model works, its accuracy limitations, and why results may vary depending on data quality. Ethical notes highlight responsible usage, emphasizing fairness, bias awareness, and the importance of validating the model before real-world deployment.