

SR-UNIVERSITY

ASSIGNMENT-5.4

B. NO:10

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:

Write a Python script that collects user data: name, age, and email.

Hash the email using SHA-256 to anonymize it.

Store all user data in a JSON file.

The script should allow reading back the stored data.

Do not use any external libraries except standard Python modules.

Provide code that works without errors.

CODE:

```
import hashlib
import json
import os

FILENAME = "user_data.json"

def collect_user_data():
    name = input("Enter your name: ")
    age = input("Enter your age: ")
    email = input("Enter your email: ")
    hashed_email = hashlib.sha256(email.encode()).hexdigest()
    return {"name": name, "age": age, "email_hash": hashed_email}

def save_user_data(user_data):
    if os.path.exists(FILENAME):
        with open(FILENAME, "r") as f:
            all_data = json.load(f)
    else:
        all_data = []
```

```

    all_data.append(user_data)
    with open(FILENAME, "w") as f:
        json.dump(all_data, f)

def read_user_data():
    if not os.path.exists(FILENAME):
        return []
    with open(FILENAME, "r") as f:
        return json.load(f)

if __name__ == "__main__":
    user = collect_user_data()
    save_user_data(user)
    data = read_user_data()
    print("\nStored user data (email hashed):")
    for u in data:
        print(u)

```

The screenshot shows a VS Code editor with a Python script named `task1.py` open. The script defines functions for collecting user data, saving it to a JSON file, and reading it back. The terminal output shows the script being executed, with prompts for name, age, and email. The email is hashed using SHA256. The final output shows the stored user data as a list of dictionaries.

```

1 import hashlib
2 import json
3 import os
4
5 FILENAME = "user_data.json"
6
7 def collect_user_data():
8     name = input("Enter your name: ")
9     age = input("Enter your age: ")
10    email = input("Enter your email: ")
11    hashed_email = hashlib.sha256(email.encode()).hexdigest()
12    return {"name": name, "age": age, "email_hash": hashed_email}
13
14 def save_user_data(user_data):
15     if os.path.exists(FILENAME):
16         with open(FILENAME, "r") as f:
17             all_data = json.load(f)
18             all_data.append(user_data)
19             with open(FILENAME, "w") as f:
20                 json.dump(all_data, f)
21     else:
22         with open(FILENAME, "w") as f:
23             json.dump([user_data], f)
24
25 def read_user_data():
26     if not os.path.exists(FILENAME):
27         return []
28     with open(FILENAME, "r") as f:
29         return json.load(f)
30
31 if __name__ == "__main__":
32     user = collect_user_data()
33     save_user_data(user)
34     data = read_user_data()
35     print("\nStored user data (email hashed):")
36     for u in data:
37         print(u)

```

```

PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab> & 'c:\Users\DANDU RAHUL\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\DANDU RAHUL\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '52821' '-.' 'C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab\assignment 5.4\task1.py'
Enter your name: raju
Enter your age: 45
Enter your email: rajul@gmail.com

Stored user data (email hashed):
{'name': 'raju', 'age': '45', 'email_hash': 'd52dfad572aa59ca0f931d26f924d3afbe32a39c7bc1fd143878f9ed25ed3f09'}
PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>

```

Analysis:

The script collects user data, hashes emails for privacy, and stores it in a JSON file. It's simple and works without extra libraries but the file isn't fully encrypted.

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 t

Prompt

Generate a Python function to perform sentiment analysis on text data.
Then, add logic or comments to identify and handle potential biases in the dataset,
such as offensive language, underrepresented classes, or skewed distributions.

code:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

data = pd.DataFrame({
    "text": [
        "I love this product",
        "This is amazing",
        "Terrible experience",
        "I hate this item",
        "Absolutely fantastic",
        "Worst purchase ever"
    ],
    "sentiment": ["positive", "positive", "negative", "negative",
"positive", "negative"]
})

offensive_terms = ["hate", "worst"]
data['text'] = data['text'].apply(lambda x: ' '.join([w for w in
x.split() if w.lower() not in offensive_terms]))

X_train, X_test, y_train, y_test = train_test_split(data['text'],
data['sentiment'], test_size=0.33, random_state=42)

vectorizer = CountVectorizer()
X_train_vect = vectorizer.fit_transform(X_train)
X_test_vect = vectorizer.transform(X_test)

model = MultinomialNB()
model.fit(X_train_vect, y_train)
```

```

y_pred = model.predict(X_test_vect)
print("Accuracy:", accuracy_score(y_test, y_pred))

def analyze_sentiment(text):
    clean_text = ' '.join([w for w in text.split() if w.lower() not in
offensive_terms])
    vect_text = vectorizer.transform([clean_text])
    return model.predict(vect_text)[0]

print(analyze_sentiment("I love this experience"))
print(analyze_sentiment("This was the worst product ever"))
print(analyze_sentiment("Absolutely fantastic"))
print(analyze_sentiment("Terrible experience"))

```

```

assignment 5.4 > task2.py > ...
1 import pandas as pd
2 from sklearn.model_selection import train_test_split
3 from sklearn.feature_extraction.text import CountVectorizer
4 from sklearn.naive_bayes import MultinomialNB
5 from sklearn.metrics import accuracy_score
6
7 data = pd.DataFrame({
8     "text": [
9         "I love this product",
10        "This is amazing",
11        "Terrible experience",
12        "I hate this item",
13        "Absolutely fantastic",
14        "Worst purchase ever"
15    ],
16    "sentiment": ["positive", "positive", "negative", "negative", "positive", "negative"]
17 })
18
19
20 offensive_terms = ["hate", "worst"]
21 data['text'] = data['text'].apply(lambda x: ' '.join([w for w in x.split() if w.lower() not in offens
22
23 X_train, X_test, y_train, y_test = train_test_split(data['text'], data['sentiment'], test_size=0.33,
24
25
26 vectorizer = CountVectorizer()
27 X_train_vect = vectorizer.fit_transform(X_train)
28 X_test_vect = vectorizer.transform(X_test)
29
30

```

VS Code interface details:

- VARIABLES:** Locals, special variables, Globals.
- WATCH:** (Empty)
- CALL STACK:** Python D... (NO MODULE NAMED 'PANDAS'), task3.py (1:1).
- BREAKPOINTS:** Raised Exceptions, Uncaught Exceptions.
- TERMINAL:** PS c:\Users\DANDU\RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab> & 'c:\Users\DANDU\RAHUL\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\DANDU\RAHUL\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' "56146" "--" "c:\Users\DANDU\RAHUL\OneDrive\Desktop\Documents\Desktop\ai Lab\assignment

Analysis:

The code trains a Naive Bayes model to classify text as positive or negative, removing offensive words to reduce bias. It can predict sentiment for new sentences using the `analyze_sentiment()` function.

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:

Write a Python program to recommend products based on user history, showing reasons for each suggestion and asking for user feedback. Ensure fairness and transparency.

Code:

```
products = [
    {"id": 1, "name": "Laptop", "category": "Electronics"},
    {"id": 2, "name": "Headphones", "category": "Electronics"},
    {"id": 3, "name": "T-shirt", "category": "Clothing"},
    {"id": 4, "name": "Sneakers", "category": "Footwear"},
    {"id": 5, "name": "Coffee Maker", "category": "Home Appliances"},
]

user_history = ["Electronics", "Footwear"]

def recommend_products(user_history, products):
    """
    Recommend products to a user based on their history.
    Ethical guidelines implemented:
    - Transparency: Shows why a product is recommended
    - Fairness: Avoids repeatedly favoring same products
    - Feedback: Allows user to indicate if recommendations are useful
    """
    recommended = []

    for product in products:
        if product["category"] in user_history:
            recommended.append({
                "product": product["name"],
                "reason": f"You have previously shown interest in {product['category']}."
            })
```

```
        if not recommended:
            recommended = [
                {"product": product["name"], "reason": "Popular choice, recommended fairly."}
                for product in products[:3]
            ]

        return recommended

recommendations = recommend_products(user_history, products)

print("Recommended Products:")
for rec in recommendations:
    print(f"- {rec['product']} (Reason: {rec['reason']})")

feedback = input("Did you find these recommendations useful? (yes/no): ")

if feedback.lower() == "yes":
    print("Thank you for your feedback! We will continue recommending similar products.")
else:
    print("Thanks! We'll try to improve future recommendations based on your preferences.")
```

```

1
2
3 products = [
4     {"id": 1, "name": "Laptop", "category": "Electronics"},
5     {"id": 2, "name": "Headphones", "category": "Electronics"},
6     {"id": 3, "name": "T-shirt", "category": "Clothing"},
7     {"id": 4, "name": "Sneakers", "category": "Footwear"},
8     {"id": 5, "name": "Coffee Maker", "category": "Home Appliances"},
9 ]
10
11 user_history = ["Electronics", "Footwear"]
12
13 def recommend_products(user_history, products):
14     """
15     Recommend products to a user based on their history.
16     Ethical guidelines implemented:
17     - Transparency: Shows why a product is recommended
18     - Fairness: Avoids repeatedly favoring same products
19     - Feedback: Allows user to indicate if recommendations are useful
20     """
21     recommended = []
22
23     for product in products:
24
25         if product["category"] in user_history:
26             recommended.append({
27                 "product": product["name"],
28                 "reason": f"You have previously shown interest in {product['category']}."
29             })
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python Debug Console

```

'...' 'C:\Users\DAVID\OneDrive\Desktop\Documents\Desktop\ai lab\assignment 5.4\assent 4.py'
Recommended Products:
- Laptop (Reason: You have previously shown interest in Electronics.)
- Headphones (Reason: You have previously shown interest in Electronics.)
- Sneakers (Reason: You have previously shown interest in Footwear.)
Did you find these recommendations useful? (yes/no): yes
Thank you for your feedback! We will continue recommending similar products.

```

Analysis:

Recommends products based on user history.

Explains suggestions (transparency) and avoids repetition (fairness).

Collects user feedback to improve future recommendations.

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 practices.

Prompt:

Write a Python program that logs user actions in a web application.

The logs should NOT store sensitive information like passwords or emails.

Mask any passwords or email addresses before logging.

Use user input for the log entries.

Code:

```

import logging
import re

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

def log_event(user_id, action, details=""):

```

```

        details = re.sub(r'\S+@\S+', '[email]', details)
        details = re.sub(r'password\s*=\s*\S+', 'password=****', details)
        logging.info(f"User {user_id} performed '{action}'. Details:
{details}")

# User input
user_id = input("Enter user ID: ")
action = input("Enter action performed: ")
details = input("Enter details (avoid sharing real passwords/emails):
")

# Log the event
log_event(user_id, action, details)
print("Event logged successfully! Check 'app.log' file.")

```

The screenshot shows a code editor with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a project named 'ai lab' with various files including 'assignment 5.4' and 'ass6.py'. The code editor displays the Python script from the previous block. The terminal shows the output of running the script, including user input and the log message.

```

1  import logging
2  import re
3
4  logging.basicConfig(filename='app.log', level=logging.INFO, format='% (asctime)s - %(message)s')
5
6  def log_event(user_id, action, details=""):
7      details = re.sub(r'\S+@\S+', '[email]', details)
8      details = re.sub(r'password\s*=\s*\S+', 'password=****', details)
9      logging.info(f"User {user_id} performed '{action}'. Details: {details}")
10
11 # User input
12 user_id = input("Enter user ID: ")
13 action = input("Enter action performed: ")
14 details = input("Enter details (avoid sharing real passwords/emails): ")
15
16 # Log the event
17 log_event(user_id, action, details)
18 print("Event logged successfully! Check 'app.log' file.")
19

```

```

Python Debug Console
C:\Users\DANDU\OneDrive\Desktop\Documents\Desktop\ai lab\assignment 5.4\ass6.py
Enter user ID: 121
Enter action performed: 2324
Enter details (avoid sharing real passwords/emails): 2345e
Event logged successfully! Check 'app.log' file.
PS C:\Users\DANDU\OneDrive\Desktop\Documents\Desktop\ai lab>

```

Analysis :

Logs user actions with timestamps in a file.

Masks sensitive info like emails and passwords.

Takes user input and keeps logs safe and useful

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:

Write a Python program to predict iris species using only basic Python (no sklearn or external libraries).

Use a small dataset and calculate the nearest average for each species.

Allow the user to input features and print the predicted species.

Code:

```
data = [
    [5.1, 3.5, 1.4, 0.2, "setosa"],
    [4.9, 3.0, 1.4, 0.2, "setosa"],
    [6.2, 3.4, 5.4, 2.3, "virginica"],
    [5.9, 3.0, 5.1, 1.8, "virginica"],
    [5.6, 2.9, 3.6, 1.3, "versicolor"],
    [6.1, 2.8, 4.0, 1.3, "versicolor"]
]

species_avg = {}
for entry in data:
    features, species = entry[:4], entry[4]
    if species not in species_avg:
        species_avg[species] = [0, 0, 0, 0, 0]
    for i in range(4):
        species_avg[species][i] += features[i]
    species_avg[species][4] += 1

for s in species_avg:
    count = species_avg[s][4]
    species_avg[s] = [x/count for x in species_avg[s][:4]]

def predict_species(sample):
    min_dist = float('inf')
    predicted = None
    for species, avg in species_avg.items():
```

```

        dist = sum((a-b)**2 for a,b in zip(sample, avg))**0.5
        if dist < min_dist:
            min_dist = dist
            predicted = species
    return predicted

print("Enter iris features (sepal_length sepal_width petal_length
petal_width):")
sample = [float(x) for x in input().split()]
prediction = predict_species(sample)
print("Predicted species:", prediction)

```

```

2  data = [
3      [5.1, 3.5, 1.4, 0.2, "setosa"],
4      [4.9, 3.0, 1.4, 0.2, "setosa"],
5      [6.2, 3.4, 5.4, 2.3, "virginica"],
6      [5.9, 3.0, 5.1, 1.8, "virginica"],
7      [5.6, 2.9, 3.6, 1.3, "versicolor"],
8      [6.1, 2.8, 4.0, 1.3, "versicolor"]
9  ]
10
11
12  species_avg = {}
13  for entry in data:
14      features, species = entry[:4], entry[4]
15      if species not in species_avg:
16          species_avg[species] = [0, 0, 0, 0]
17      for i in range(4):
18          species_avg[species][i] += features[i]
19      species_avg[species][4] += 1
20
21  for s in species_avg:
22      count = species_avg[s][4]
23      species_avg[s] = [x/count for x in species_avg[s][:4]]
24
25
26  def predict_species(sample):
27      min_dist = float('inf')
28      predicted = None
29
30      for species, avg in species_avg.items():
31          dist = sum((a-b)**2 for a,b in zip(sample, avg))**0.5
32          if dist < min_dist:
33              min_dist = dist
34              predicted = species
35      return predicted
36
37  print("Enter iris features (sepal_length sepal_width petal_length
38  petal_width):")
39  sample = [float(x) for x in input().split()]
40  prediction = predict_species(sample)
41  print("Predicted species:", prediction)

```

Terminal Output:

```

PS C:\Users\DANDU RAHUL\OneDrive\Desktop\Documents\Desktop\ai lab>
Enter iris features (sepal_length sepal_width petal_length petal_width):
23
Predicted species: virginica

```

Analysis :

The program predicts iris species using a small dataset and basic math (no external libraries). It calculates the average features for each species and compares user input using distance formula.

The model is simple, not highly accurate, and should only be used for learning purposes.