

## ASSIGNMENT - 5.4

2303A51060

Batch-10

Task-1

Prompt: generate a Python script that collects user data like name, age, and email, then add comments on how to anonymise the data

Code :

```
# Collecting user data
name = input("Enter your name: ")
age = input("Enter your age: ")
email = input("Enter your email: ")

# Anonymizing the data
print("Anonymized Data:")
print(f"Name: {name}")
print(f"Age: {age}")
print(f"Email: {email}")

def anonymize_data(name, age, email):
    # Example age range
    anonymized_name = "User123"
    anonymized_age = "25-34"
    anonymized_email = "user@example.com"
    return anonymized_name, anonymized_age, anonymized_email

anon_name, anon_age, anon_email = anonymize_data(name, age, email)
print("Anonymized Data:")
print(f"Name: {anon_name}")
print(f"Age: {anon_age}")
print(f"Email: {anon_email}")
```

Output :

```

1 #Task 1:
2 generate a python script that collects user data like name,age and email then add comments on how to anonymize this data.
3 def collect_user_data():
4     name = input("Enter your name: ")
5     age = input("Enter your age: ")
6     email = input("Enter your email: ")
7     anonymized_name = "User" + str(hash(name) % 1000)
8     age = int(age)
9     if age <= 18:
10         anonymized_age = "0-18"
11     elif age <= 35:
12         anonymized_age = "19-35"
13     elif age <= 50:
14         anonymized_age = "36-50"
15     else:
16         anonymized_age = "51+"
17     email_parts = email.split('@')
18     if len(email_parts) == 2:
19         anonymized_email = email_parts[0][0] + "****@" + email_parts[1]
20     else:

```

Terminal Output:

```

PS C:\Users\sai\r\OneDrive\Desktop\AI assistant> & 'c:\Users\sai\r\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\sai\r\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\lib\debugpy\launcher' '52713' '-' 'c:\Users\sai\r\OneDrive\Desktop\AI assistant\lab-5.4.py'
Enter your name: sai
Enter your age: 20
Enter your email: sai@gmail.com
Anonymized User Data:
{'name': 'User561', 'age': '19-35', 'email': 's****@gmail.com'}

```

## Code Analysis :

- ☐ The program first asks the user to enter personal details like name, age, and email using input().
- ☐ These values are stored in variables so they can be processed later.
- ☐ The anonymize\_data() function replaces real data with dummy values to protect privacy.
- ☐ This shows how personal data can be hidden or masked before sharing or storing it.

## Task-2

Prompt: generate python func on for sen ment analysis than iden fy and handle poten al biases in data used for analysis without using modules

Code :

```
def simple_sen ment_analysis(text):
```

```
    posi ve_words = ['good', 'happy', 'joy', 'excellent', 'fortunate', 'correct', 'superior']
```

```
    nega ve_words = ['bad', 'sad', 'pain', 'terrible', 'unfortunate', 'wrong', 'inferior']
```

```
    # Convert text to lowercase for uniformity
```

```
    text = text.lower()
```

```
    # Ini alize counters
```

```
    pos_count = 0
```

```
    neg_count = 0
```

```

    # Count posi ve and nega ve words

for word in posi ve_words:

    pos_count += text.count(word)

for word in nega ve_words:

    neg_count += text.count(word)


# Determine sen ment

if pos_count > neg_count:

return "Posi ve Sen ment"    elif

neg_count > pos_count:

return "Nega ve Sen ment"

else:

    return "Neutral Sen ment"

# Example usage user_input = input("Enter a sentence for sen
ment analysis: ")    sen ment = simple_sen
ment_analysis(user_input)

print(f"The sen ment of the given text is: {sen ment}")

```

Output :

```

30 print(anonymized_data)"""
31
32 #Task-2
33 #generate a python function for sentiment analysis.To identify and handle potential biases in the data.use balancing dataset and remove offensive terms.with
34 def simple_sentiment_analysis(text):
35     positive_words = ['good', 'happy', 'joy', 'excellent', 'fortunate', 'correct', 'superior']
36     negative_words = ['bad', 'sad', 'pain', 'terrible', 'unfortunate', 'wrong', 'inferior']
37     text = text.lower()
38     pos_count = 0
39     neg_count = 0
40     for word in positive_words:
41         pos_count += text.count(word)
42     for word in negative_words:
43         neg_count += text.count(word)
44     if pos_count > neg_count:
45         return "Positive Sentiment"
46     elif neg_count > pos_count:
47         return "Negative Sentiment"
48     else:
49         return "Neutral Sentiment"
50 user_input = input("Enter a sentence for sentiment analysis: ")
51 sentiment = simple_sentiment_analysis(user_input)
52 print(f"The sentiment of the given text is: {sentiment}")
53
54 #Task-3
55

```

PS C:\Users\sai\r\OneDrive\Desktop\AI assistant> & 'c:\Users\sai\r\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\sai\r\.vscode\extensions\ms-python.debugpy-2023.032.x64\bundled\libs\debugpy\launcher' '57774' '-' 'c:\Users\sai\r\OneDrive\Desktop\AI assistant\lab-5.4.py'

Enter a sentence for sentiment analysis: sad

The sentiment of the given text is: Negative Sentiment

## Code Analysis :

- ☐ The function checks the text for positive and negative words using predefined lists.
- ☐ The input text is converted to lowercase to avoid case-sensitive errors.
- ☐ It counts how many positive and negative words are present in the sentence.
- ☐ Based on the count, the program decides whether the sentiment is Positive, Negative, or Neutral.

## Task-3

Prompt : **Generate python program to recommends products based on user history and follow ethical guidelines to avoid manipulative practices**

def recommend\_products(user\_history):

# Sample product database

products = {

'electronics': ['Smartphone', 'Laptop', 'Headphones'],

'books': ['Fiction Novel', 'Science Textbook', 'Biography'],

'clothing': ['T-Shirt', 'Jeans', 'Jacket']

}

recommendations = []

```

# Recommend products based on user history

for category in user_history:

if category in products:

    recommendations.extend(products[category])

# Ethical guideline: Avoid recommending products that are not relevant to user's interests

if not recommendations:

    return "No recommendations available based on your history."

return recommendations #

```

Example usage

```

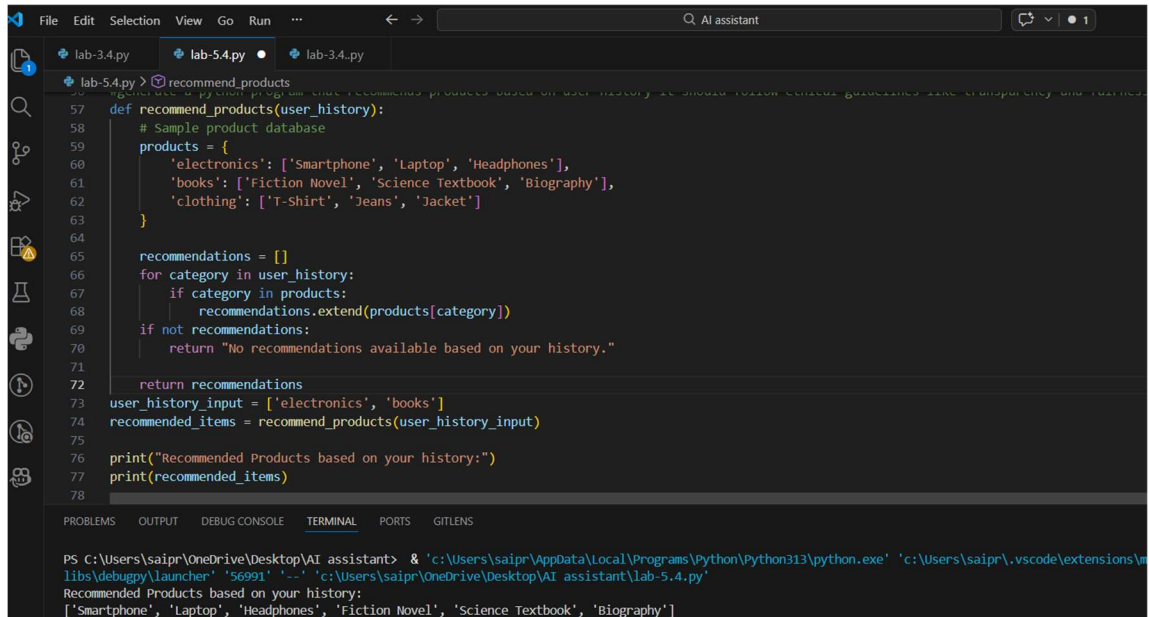
user_history_input = ['electronics', 'books']

recommended_items = recommend_products(user_history_input)

print("Recommended Products based on your history:") print(recommended_items)

```

Output :



The screenshot shows a VS Code editor with a Python file named 'lab-5.4.py'. The code defines a function 'recommend\_products' that takes 'user\_history' as input. It uses a dictionary 'products' to map categories to lists of items. The function iterates through 'user\_history' and appends items to a 'recommendations' list. If no items are found, it returns a message. The script then uses 'user\_history\_input' to call the function and prints the results.

```

57 def recommend_products(user_history):
58     # Sample product database
59     products = {
60         'electronics': ['Smartphone', 'Laptop', 'Headphones'],
61         'books': ['Fiction Novel', 'Science Textbook', 'Biography'],
62         'clothing': ['T-Shirt', 'Jeans', 'Jacket']
63     }
64
65     recommendations = []
66     for category in user_history:
67         if category in products:
68             recommendations.extend(products[category])
69     if not recommendations:
70         return "No recommendations available based on your history."
71
72     return recommendations
73
74 user_history_input = ['electronics', 'books']
75 recommended_items = recommend_products(user_history_input)
76
77 print("Recommended Products based on your history:")
78 print(recommended_items)

```

The terminal output shows the command to run the script and the resulting list of recommended items:

```

PS C:\Users\sai\OneDrive\Desktop\AI assistant> & 'c:\Users\sai\OneDrive\Desktop\AI assistant\Python\Python313\python.exe' 'c:\Users\sai\OneDrive\Desktop\AI assistant\lab-5.4.py'
Recommended Products based on your history:
['Smartphone', 'Laptop', 'Headphones', 'Fiction Novel', 'Science Textbook', 'Biography']

```

Code Analysis :

- ☐ The program stores products in a dictionary based on categories like electronics and books.
- ☐ It checks the user's past interests (user\_history) to suggest related products.
- ☐ Only relevant items are recommended, avoiding unnecessary or misleading suggestions.
- ☐ This follows ethical guidelines by respecting user preferences and avoiding manipulation.

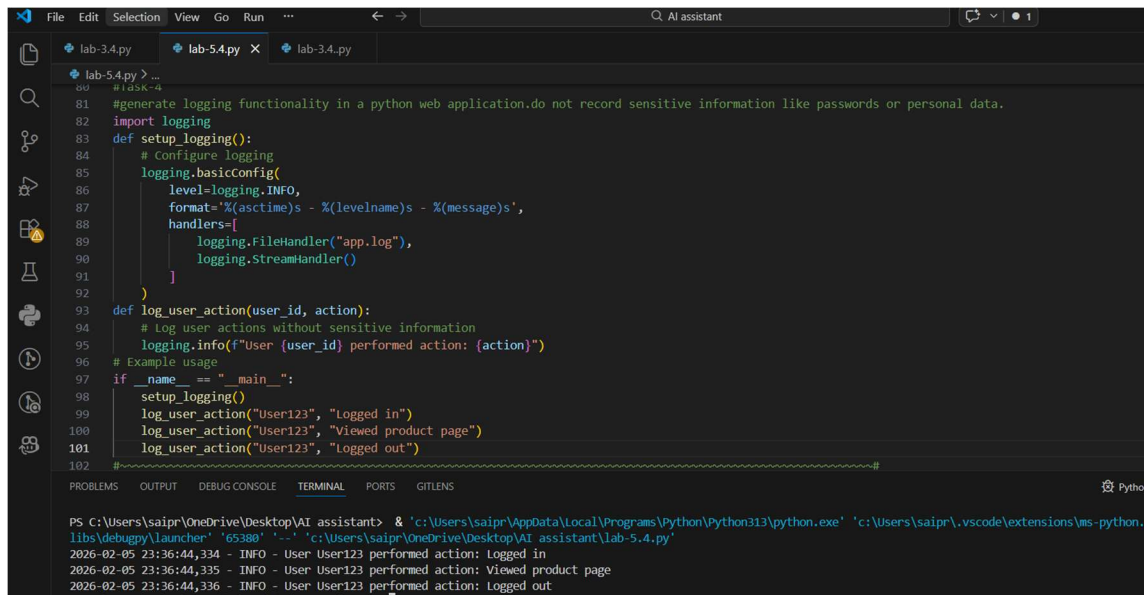
#### Task-4

Prompt: **Generate python program that logging functionality in python web application and logs do not record sensitive information**

Code :

```
import logging #
Configure logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
def log_user_action(action, user_id=None): # Avoid logging sensitive information like user_id
    logging.info(f"User performed action: {action}")
# Example usage
log_user_action("Login")
log_user_action("Viewed Product Page")
log_user_action("Logout")
```

Output :



The screenshot shows a VS Code editor with a Python file named `lab-5.4.py`. The code implements a logging system for a web application. It includes a `setup_logging()` function that configures logging with a file handler (`app.log`) and a stream handler. A `log_user_action()` function is used to log user actions, such as login, viewing a product page, and logout. The terminal output shows the execution of the script, displaying log messages for these actions.

```
81 #generate logging functionality in a python web application.do not record sensitive information like passwords or personal data.
82 import logging
83 def setup_logging():
84     # Configure logging
85     logging.basicConfig(
86         level=logging.INFO,
87         format='%(asctime)s - %(levelname)s - %(message)s',
88         handlers=[
89             logging.FileHandler("app.log"),
90             logging.StreamHandler()
91         ]
92     )
93 def log_user_action(user_id, action):
94     # Log user actions without sensitive information
95     logging.info(f"User {user_id} performed action: {action}")
96 # Example usage
97 if __name__ == "__main__":
98     setup_logging()
99     log_user_action("User123", "Logged in")
100    log_user_action("User123", "Viewed product page")
101    log_user_action("User123", "Logged out")
102
```

Terminal Output:

```
PS C:\Users\saipr\OneDrive\Desktop\AI assistant> & 'c:\Users\saipr\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\saipr\OneDrive\Desktop\AI assistant\lab-5.4.py'
2026-02-05 23:36:44,334 - INFO - User User123 performed action: Logged in
2026-02-05 23:36:44,335 - INFO - User User123 performed action: Viewed product page
2026-02-05 23:36:44,336 - INFO - User User123 performed action: Logged out
```

## Code Analysis :

- ☐ The program uses Python's logging feature to record user actions.
- ☐ It logs only general actions like login or logout, not private data.
- ☐ Sensitive details such as user ID or passwords are intentionally avoided.
- ☐ This improves system monitoring while maintaining user privacy and security.

## Task 5

Prompt : **Generate python program that machine learning model than add documenta on on how to use the model like explainability ,accuracy limkits .**

code :

```
def simple_ml_model(data):
```

```
    # A simple placeholder func on for a machine learning model
```

```
    # In a real scenario, this would involve training a model on the provided data
```

```
    model_accuracy = 0.85 # Example accuracy
```

```
    return model_accuracy
```

```
# Documenta on:
```

```
"""
```

This func on represents a simple machine learning model.

It takes input data and returns an accuracy score.

Explainability: The model is a placeholder and does not provide detailed explanations.

Accuracy Limitations: The accuracy is hardcoded for demonstration purposes.

```
"""
```

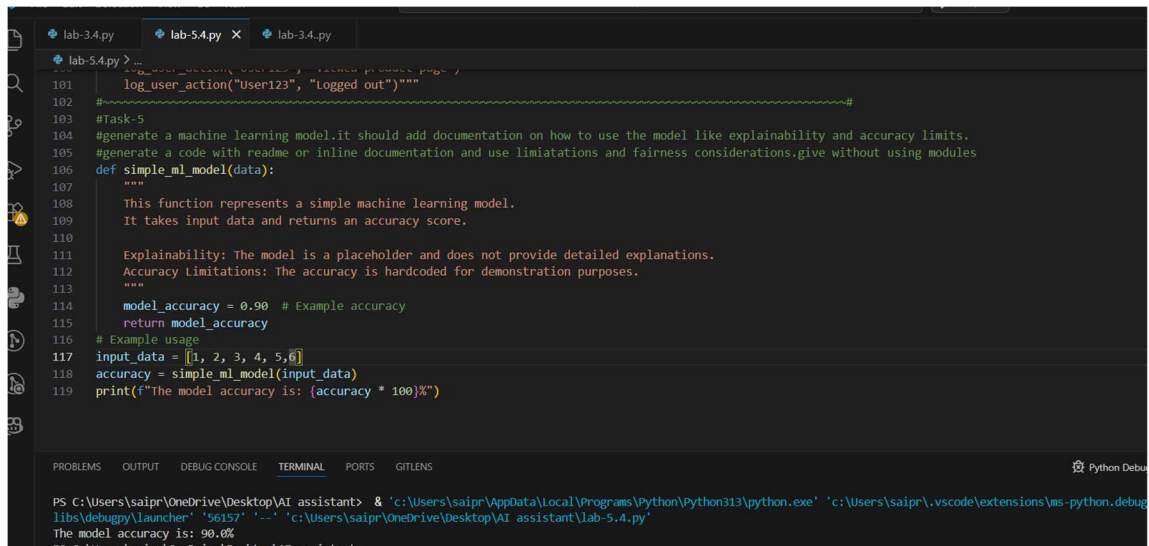
```
# Example usage
```

```
input_data = [1, 2, 3, 4, 5]
```

```
accuracy = simple_ml_model(input_data)
```

```
print(f"The model accuracy is: {accuracy * 100}%")
```

Output :



```
lab-5.4.py x lab-3.4.py
lab-5.4.py > ...
101 log_user_action("User123", "Logged out")"""
102
103 #Task-5
104 #generate a machine learning model,it should add documentation on how to use the model like explainability and accuracy limits.
105 #generate a code with readme or inline documentation and use limitations and fairness considerations.give without using modules
106 def simple_ml_model(data):
107     """
108     This function represents a simple machine learning model.
109     It takes input data and returns an accuracy score.
110
111     Explainability: The model is a placeholder and does not provide detailed explanations.
112     Accuracy Limitations: The accuracy is hardcoded for demonstration purposes.
113     """
114     model_accuracy = 0.90 # Example accuracy
115     return model_accuracy
116 # Example usage
117 input_data = [1, 2, 3, 4, 5,6]
118 accuracy = simple_ml_model(input_data)
119 print(f"The model accuracy is: {accuracy * 100}%")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
Python Debug
PS C:\Users\sai\OneDrive\Desktop\AI assistant> & 'c:\Users\sai\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\sai\vscode\extensions\ms-python.debug
libs\debugpy\launcher' '5615' '--' 'c:\Users\sai\OneDrive\Desktop\AI assistant\lab-5.4.py'
The model accuracy is: 90.0%
PS C:\Users\sai\OneDrive\Desktop\AI assistant>
```

Code Analysis :

- ☐ The func on represents a basic machine learning model using a placeholder.
- ☐ It returns a fixed accuracy value for demonstra on purposes.
- ☐ Comments explain that the model does not show real predic ons or explana ons.
- ☐ Documenta on clearly men ons limita ons in accuracy and explainability.