

LAB ASSIGNMENT-5.4

2303A51337

BATCH-10

➤ TASK-1:

PROMPT:

Generate a python script that collects user data (name,age,email)then how to anonymise or protect the data:

CODE:

```
import hashlib

name = input("Enter your name: ")

age = input("Enter your age: ")

email = input("Enter your email: ")

# Hash email to protect sensitive information

hashed_email = hashlib.sha256(email.encode()).hexdigest()

# Store only required and protected data

user_data = {

    "name": name,

    "age": age,

    "email_hash": hashed_email

}

print("\nSecure User Data:")

print(user_data)

print("\nAnonymised Data:")

print("protected_data", user_data)
```

OUTPUT:

```
lab54.py > ...
1  #generate a python script that collects user data (name,age,email)then how to anonymise or protect the data:
2  import hashlib
3  name = input("Enter your name: ")
4  age = input("Enter your age: ")
5  email = input("Enter your email: ")
6  # Hash email to protect sensitive information
7  hashed_email = hashlib.sha256(email.encode()).hexdigest()
8  # Store only required and protected data
9  user_data = {
10     "name": name,
11     "age": age,
12     "email_hash": hashed_email
13 }
14 print("\nSecure User Data:")
15 print(user_data)
16 print("\nAnonymised Data:")
17 print("protected_data", user_data)
18
19
20
21
22
23
24
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

Enter your name: varsha
Enter your age: 20
Enter your email: srivarshapabbu@gmail.com

Secure User Data:
{'name': 'varsha', 'age': '20', 'email_hash': '1fe2a6b684bc17fe11c0ea070168c73b62543f7f9bdbb17bdad24f3e92f6d094'}

Anonymised Data:
protected_data {'name': 'varsha', 'age': '20', 'email_hash': '1fe2a6b684bc17fe11c0ea070168c73b62543f7f9bdbb17bdad24f3e92f6d094'}
PS C:\Users\sriva\OneDrive\Desktop\b-2 academics\AI Assisted> []

Ln 21, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.7 Go Live

EXPLANATION:

- This program collects a user's name, age, and email as input.
- To protect sensitive information, the email is converted into a hashed value using the algorithm from the hashlib library, so the real email is not stored directly.
- The program then stores only the necessary and protected data in a dictionary, replacing the email with its hash.
- Finally, it prints the secure user data and shows the anonymised protected version, demonstrating how personal data can be safely stored.

❖ TASK-2

PROMPT:

Generate a python program using function for to identify and handle potential biases in the data

CODE:

```
def analyze_sentiment(text):
    text = text.lower()

    positive_words = ["good", "great", "happy", "love", "excellent", "nice"]

    negative_words = ["bad", "sad", "hate", "terrible", "awful", "poor"]

    score = 0

    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"

# Bias Identification Function

def identify_bias(text):
    biased_words = ["hate", "terrible", "awful"]
    found_bias = []
    for word in biased_words:
        if word in text.lower():
            found_bias.append(word)
    return found_bias

# Bias Handling / Mitigation

def handle_bias(text):
    biased_words = ["hate", "terrible", "awful"]
    for word in biased_words:
        text = text.replace(word, "") # Neutralizing bias
    return text

# Example Usage

user_text = input("\nEnter a sentence for sentiment analysis: ")
bias_words = identify_bias(user_text)
```

```

if bias_words:

    print("Potential bias detected due to words:", bias_words)

    user_text = handle_bias(user_text)

    print("Text after bias handling:", user_text)

sentiment = analyze_sentiment(user_text)

print("Final Sentiment:", sentiment)

```

OUTPUT:

```

File Edit Selection View Go Run Terminal Help ← → Q AI Assisted
OPEN EDITORS 1 unsaved
EXPLORER lab4.py lab1.py lab1.4.py lab5.4.py lab6.3.py lab4.3.py lab3.3.py
AI ASSISTED
OUTLINE
TIMELINE
38     def identify_bias(text):
39         found_bias = []
40         for word in text.lower():
41             if word in bias_words:
42                 found_bias.append(word)
43         return found_bias
44
45     # Bias Handling / Mitigation
46     def handle_bias(text):
47         biased_words = ["hate", "terrible", "awful"]
48         for word in biased_words:
49             text = text.replace(word, "") # Neutralizing bias
50         return text
51
52     # Example Usage
53     user_text = input("\nEnter a sentence for sentiment analysis: ")
54     bias_words = identify_bias(user_text)
55     if bias_words:
56         print("Potential bias detected due to words:", bias_words)
57         user_text = handle_bias(user_text)
58         print("Text after bias handling:", user_text)
59         sentiment = analyze_sentiment(user_text)
60         print("Final Sentiment:", sentiment)
61
62
63
64
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
File "c:/Users/sriva/OneDrive/Desktop/3-2 academics/AI Assisted/lab5.4.py", line 163
    print("The password is not strong.")'''"
SyntaxError: unterminated triple-quoted string literal (detected at line 163)
PS C:\Users\sriva\OneDrive\Desktop\3-2 academics\AI Assisted > C:/Users/sriva/AppData/Local/Programs/Python/Python313/python.exe "c:/users/sriva/OneDrive/Desktop/3-2 academics/AI Assisted/lab5.4.py"

Enter a sentence for sentiment analysis: i hate this work
Potential bias detected due to words: ['hate']
Text after bias handling: i this work
Final Sentiment: Neutral

```

EXPLANATION:

- This program performs sentiment analysis and bias detection on a given sentence.
- First, it checks the text for biased or strongly negative words like *hate*, *terrible*. If such words are found, it reports them and removes them from the sentence to reduce bias.
- Then, the modified text is analyzed to determine whether its sentiment is Positive, Negative, or Neutral based on the presence of predefined positive and negative words.
- Finally, the program prints the cleaned text if bias was found and the final sentiment result.

❖ TASK-3:

PROMPT:

Generate a python program that recommends products based on user history

Ask the user about their past behavior (this is their history)

CODE:

```
print("To recommend products, we use your past activity.")

viewed = input("What categories have you viewed before? (comma separated): ")

purchased = input("What categories have you purchased from? (comma separated): ")

# Normalize and combine user history

viewed_list = [v.strip().lower() for v in viewed.split(",")]

purchased_list = [p.strip().lower() for p in purchased.split(",")]

user_history = set(viewed_list + purchased_list)

# Product database kept diverse to avoid favoritism

products = [

    {"name": "Laptop", "category": "electronics"},

    {"name": "Headphones", "category": "electronics"},

    {"name": "Novel", "category": "books"},

    {"name": "Notebook", "category": "books"},

    {"name": "T-shirt", "category": "fashion"},

    {"name": "Jacket", "category": "fashion"},

    {"name": "Running Shoes", "category": "sports"},

    {"name": "Football", "category": "sports"},

]

# Recommendation logic with transparency & fairness

def recommend_products(history, product_list):

    # Copilot: Be transparent about how recommendations work

    print("\n Recommendations are based on the categories in your history.")

    recommendations = []

    # Match by category only (no brand or seller favoritism)

    for product in product_list:

        if product["category"] in history:
```

```

recommendations.append(product)

# Fairness check — avoid empty or biased results

if not recommendations:
    print("⚠️ No direct matches. Showing a balanced set instead.")

    recommendations = product_list[:3]

return recommendations

# Generate and show recommendations

final_recs = recommend_products(user_history, products)

print("\n🛒 Recommended Products:")

for item in final_recs:
    print(f"- {item['name']} ({item['category']})")

# Ask for feedback (user control + ethical AI)

feedback = input("\nDid you find these recommendations useful? (yes/no): ").lower()

print("Thank you! Your feedback helps improve fairness and transparency.")

```

OUTPUT:

The screenshot shows a code editor interface with several tabs open. The active tab contains Python code for generating recommendations based on user history and asking for user feedback. The code includes comments explaining the logic, such as 'Fairness check – avoid empty or biased results' and 'Ask for feedback (user control + ethical AI)'. The output pane at the bottom shows the execution of the code, including the recommended products and the user's feedback input.

```

File Edit Selection View Go Run Terminal Help ↺ → AI Assisted
EXPLORER ... lab4.py lab1.py lab1.4.py lab54.py x lab6.3.py lab4.3.py lab3.3.py
> OPEN EDITORS
> AI ASSISTED
> OUTLINE
> TIMELINE
86     def recommend_products(history, product_list):
87         recommendations = []
88         for product in product_list:
89             if product["category"] in history:
90                 recommendations.append(product)
91
92             # Fairness check – avoid empty or biased results
93             if not recommendations:
94                 print("⚠️ No direct matches. Showing a balanced set instead.")
95                 recommendations = product_list[:3]
96
97             return recommendations
98
99         # Generate and show recommendations
100
101     final_recs = recommend_products(user_history, products)
102     print("\n🛒 Recommended Products:")
103     for item in final_recs:
104         print(f"- {item['name']} ({item['category']})")
105     # Ask for feedback (user control + ethical AI)
106     feedback = input("\nDid you find these recommendations useful? (yes/no): ").lower()
107
108     print("Thank you! Your feedback helps improve fairness and transparency.")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
To recommend products, we use your past activity.
What categories have you viewed before? (comma separated): book
What categories have you purchased from? (comma separated): book
💡 Recommendations are based on the categories in your history.
⚠️ No direct matches. Showing a balanced set instead.

💡 Recommended Products:
- Laptop (electronics)
- Headphones (electronics)
- Novel (books)

Did you find these recommendations useful? (yes/no): yes
Thank you! Your feedback helps improve fairness and transparency.
PS C:\Users\sriva\OneDrive\Desktop\3-2 academics\AI Assisted>

```

EXPLANATION:

❖ TASK-4:

PROMPT:

Create a Python logging system for a web app that avoids logging sensitive user data like passwords and emails, and follows ethical logging practices.

CODE:

```
import logging

# Configure logging
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s',
    handlers=[

        logging.FileHandler("webapp.log"),

        logging.StreamHandler()

    ]
)

def log_user_action(user_id, action, details=None):
    log_message = f"User ID: {user_id} | Action: {action}"
    if details:
        log_message += f" | Details: {details}"
    logging.info(log_message)

# Example usage
if __name__ == "__main__":
    log_user_action(user_id=12345, action="Login Attempt")
    log_user_action(user_id=12345, action="Viewed Profile", details="Profile ID: 67890")
    log_user_action(user_id=12345, action="Password Change Request")
    log_user_action(user_id=12345, action="Logout")
    password = input("Enter a password to check its strength: ")
    logging.info("Password strength check performed.") # Avoid logging the actual
password
    print("Logging complete. Check webapp.log for details.")
```

OUTPUT:

The screenshot shows a code editor interface with several tabs open. The active tab is 'lab5.4.py' which contains Python code for a logging system. The code imports the 'logging' module and configures it to log messages to both a file ('webapp.log') and the console. It defines a function 'log_user_action' that takes a user ID and action, and logs a message with the timestamp, user ID, action, and profile details. The terminal below shows the execution of the script and the resulting log entries.

```
# TASK - 4 :Create a Python logging system for a web app that avoids logging sensitive user data like passwords and emails, and fol
import logging
# Configure logging
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s',
    handlers=[
        logging.FileHandler("webapp.log"),
        logging.StreamHandler()
    ]
)
def log_user_action(user_id, action, details=None):
    log_message = f"User ID: {user_id} | Action: {action}"
    if details:
        log_message += f" | Details: {details}"
    logging.info(log_message)

Thank you! Your feedback helps improve fairness and transparency.
PS C:\Users\sriva\OneDrive\Desktop\3-2 academics\AI Assisted & C:/Users/sriva/AppData/Local/Programs/Python/Python313/python.exe "c:/users/sri
va/OneDrive/Desktop/3-2 academics/AI Assisted/labs.4.py"
2026-02-04 10:32:14,488 - INFO - User ID: 12345 | Action: Login Attempt
2026-02-04 10:32:14,488 - INFO - User ID: 12345 | Action: Viewed Profile | Details: Profile ID: 67890
2026-02-04 10:32:14,488 - INFO - User ID: 12345 | Action: Password Change Request
2026-02-04 10:32:14,488 - INFO - User ID: 12345 | Action: Logout
Enter a password to check its strength: 12345
2026-02-04 10:32:19,785 - INFO - Password strength check performed.
Logging complete. Check webapp.log for details.
The password is not strong.
PS C:\Users\sriva\OneDrive\Desktop\3-2 academics\AI Assisted & C:/Users/sriva/AppData/Local/Programs/Python/Python313/python.exe "c:/users/sri
va/OneDrive/Desktop/3-2 academics/AI Assisted/labs.4.py"
2026-02-04 10:32:25,149 - INFO - User ID: 12345 | Action: Login Attempt
2026-02-04 10:32:25,150 - INFO - User ID: 12345 | Action: Viewed Profile | Details: Profile ID: 67890
2026-02-04 10:32:25,150 - INFO - User ID: 12345 | Action: Password Change Request
2026-02-04 10:32:25,150 - INFO - User ID: 12345 | Action: Logout
Enter a password to check its strength: varsha@989087
2026-02-04 10:32:39,095 - INFO - Password strength check performed.
Logging complete. Check webapp.log for details.
```

EXPLANATION:

- This program uses Python's logging module to record user actions in a safe and organized way. It is configured to write logs both to a file (webapp.log) and to the console, including the time and type of each message.
- The function `log_user_action()` creates a log entry for actions such as login, viewing a profile, changing a password, and logout. Sensitive information like the actual password is not logged, which helps protect user privacy.
- Finally, the program prints a message telling the user that logging is complete and where the log file can be checked.

❖ TASK-5:

PROMPT:

Create a Python ML model with responsible usage guidelines, including fairness, explainability, accuracy limits, and privacy. Include a simple example prediction.

CODE:

```
def simple_model(value):
    if value >= 5:
        return "High"
```

```

else:
    return "Low"

num = float(input("Enter a number: "))

result = simple_model(num)

print("Model Prediction:", result)

```

OUTPUT:

The screenshot shows the Visual Studio Code interface. The code editor displays a Python script named lab54.py. The terminal at the bottom shows the execution of the script and its output. The file browser on the left shows other files in the project directory.

```

136 """
137 #task5
138 #Create a Python ML model with responsible usage guidelines, including fairness, explainability, accuracy limits, and privacy. Incl
139 def simple_model(value):
140     if value >= 5:
141         return "High"
142     else:
143         return "Low"
144 num = float(input("Enter a number: "))
145 result = simple_model(num)
146
147 print("Model Prediction:", result)
148

```

```

va/OneDrive/Desktop/3-2 academics/AI Assisted/lab5.4.py"
Enter a number: 25
Model Prediction: High
PS C:\Users\sriva\OneDrive\Desktop\3-2 academics\AI Assisted> & c:/users/sriva/appdata/local/programs/python/python313/python.exe "c:/users/sri
va/OneDrive/Desktop/3-2 academics/AI Assisted/lab5.4.py"
Enter a number: 2
Model Prediction: Low
PS C:\Users\sriva\OneDrive\Desktop\3-2 academics\AI Assisted>

```

EXPLANATION:

- This program acts like a very simple prediction model. It takes a number as input from the user and passes it to the `simple_model()` function.
- The function checks the value and returns high if the number is 5 or greater, otherwise it returns low.
- The result is then printed as the model's prediction. This shows how a basic decision rule can be used to make a simple classification.