

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):
    anonymized_name = "User123"
    anonymized_age = "25-34" # Example age range
    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 which collects user data like name,age,email than add comments on how to anonymize the data
3 # Collecting user data
4 name = input("Enter your name: ")
5 age = input("Enter your age: ")
6 email = input("Enter your email: ")
7 # Anonymizing the data
8 print("anonymized Data")
9 print(f"Name: {name}")
10 print(f"Age: {age}")
11 print(f"Email: {email}")
12 def anonymize_data(name, age, email):
13     anonymized_name = "User123"
14     anonymized_age = "25-34" # Example age range
15     anonymized_email = "user@example.com"
16     return anonymized_name, anonymized_age, anonymized_email
17 anon_name, anon_age, anon_email = anonymize_data(name, age, email)
18 print("anonymized Data:")
19 print(f"Name: {anon_name}")
20 print(f"Age: {anon_age}")
21 print(f"Email: {anon_email}")
22

```

PS C:\Users\NIRNAYA\Desktop\AI_ASSISTANT_CODING & C:\Users\NIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRNAYA/Desktop/AI_ASSISTANT_CODING/lab-5.4.py
Enter your name: mula nirnaya
Enter your age: 20
Enter your email: chinni@gmail.com
Anonymized Data:
Name: mula nirnaya
Age: 20
Email: chinni@gmail.com
Anonymized Data:
Name: User123
Age: 25-34
Email: user@example.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 function for sentiment analysis than identify and handle potential biases in data used for analysis without using modules

Code :

```

def simple_sentiment_analysis(text):

    positive_words = ['good', 'happy', 'joy', 'excellent', 'fortunate', 'correct', 'superior']

    negative_words = ['bad', 'sad', 'pain', 'terrible', 'unfortunate', 'wrong', 'inferior']

```

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

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

```
# Initialize counters
```

```
pos_count = 0
```

```
neg_count = 0
```

```

# Count positive and negative words

for word in positive_words:
    pos_count += text.count(word)

for word in negative_words:
    neg_count += text.count(word)

# Determine sentiment

if pos_count > neg_count:
    return "Positive Sentiment"
elif neg_count > pos_count:
    return "Negative Sentiment"
else:
    return "Neutral Sentiment"

# Example usage

user_input = input("Enter a sentence for sentiment analysis: ")

sentiment = simple_sentiment_analysis(user_input)

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

```

Output :

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows files in the "AI ASSISTANT_CODING" folder: lab-3.3.py, lab-3.4.py, lab-4.3.py, lab-5.4.py (the active file), lab-1.py, and lab2.py.
- Code Editor:** Displays the Python code for sentiment analysis. The code defines a function `simple_sentiment_analysis` that takes a string `text`. It initializes two lists of words: `positive_words` (good, happy, joy, excellent, fortunate, correct, superior) and `negative_words` (bad, sad, pain, terrible, unfortunate, wrong, inferior). It converts the input text to lowercase and initializes counters for positive and negative counts. It then iterates through the words in the text, updating the counters based on whether they are in the positive or negative list. Finally, it compares the counts to determine the sentiment and returns it as a string.
- Terminal:** At the bottom, the terminal window shows the command `PS C:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTANT_CODING> & C:\Users\NIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRNAYA/OneDrive/Desktop/AI_ASSISTANT_CODING/lab-5.4.py` being run. The output of the script is displayed, asking for user input and then printing "The sentiment of the given text is: Positive 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 the Visual Studio Code interface. The Explorer sidebar on the left lists files in the 'AI_ASSISTANT_CODING' folder: lab-3.3.py, lab-3.4.py, lab-4.3.py, lab-5.4.py (which is the active file), and lab1.py, lab2.py. The terminal at the bottom shows the execution of the script 'lab-5.4.py'. The output in the terminal is:

```

PS C:\Users\NIRNAYA\Desktop\AI_ASSISTANT_CODING> & c:\Users\NIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRNAYA/Desktop/AI_ASSISTANT_CODING/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 Microsoft Visual Studio Code (VS Code) interface. The title bar reads "AI_ASSISTANT_CODING". The left sidebar has a "EXPLORER" tab open, showing a file tree with files like "lab-3.3.py", "lab-3.4.py", "lab-5.4.py" (which is the active file), "lab-4.3.py", "lab1.py", and "lab2.py". The main editor area contains the following Python code:

```
# Task-4
# Generate python program that logging functionality in python web application and logs do not record sensitive information
import logging
# Configure logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
# Log user action
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")
```

The bottom status bar shows the path "C:\Users\NIRWAYA\OneDrive\Desktop\AI_ASSISTANT_CODING & C:\Users\NIRWAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRWAYA/OneDrive/Desktop/AI_ASSISTANT_CODING\lab-5.4.py" and icons for powershell, terminal, ports, query results, and Azure.

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 documentation on how to use the model like explainability ,accuracy limkits .

code :

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

```
# A simple placeholder function 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
```

Documentation:

11

This function 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.

11

```
# Example usage
```

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

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

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

Output :

The screenshot shows the Visual Studio Code interface with the 'AI ASSISTANT CODING' extension active. The left sidebar displays a tree view of files under 'EXPLORER' for a project named 'AI_ASSISTANT_CODING'. The main editor area shows a Python script 'lab-5.py' with code for a simple machine learning model. A tooltip from the extension is overlaid on the code, providing documentation for the 'simple_ml_model' function. The tooltip includes a detailed description of the function's purpose, parameters, and return value, along with explanations for 'Explainability' and 'Accuracy limitations'. The bottom status bar shows the file path 'C:\Users\NIRWANA\Desktop\AI_ASSISTANT_CODING & C:\Users\NIRWANA\AppData\Local\Microsoft\WindowsApps\python3.13.exe' and the command prompt 'powershell'.

```
def simple_ml_model(data):
    """A simple placeholder function 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
    """
    This function 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("The model accuracy is: (accuracy * 100)%")
```

Code Analysis :

- The function represents a basic machine learning model using a placeholder.
 - It returns a fixed accuracy value for demonstration purposes.
 - Comments explain that the model does not show real predictions or explanations.
 - Documentation clearly mentions limitations in accuracy and explainability.