

ASSIGNMENT 5.4

2303A51343

Batch 10

Task-1

Prompt : Generate a Python code that collects user data and add comments to anonymize or protect data. This script collects user data such as name, age, and email.

Code:

```
def collect_user_data():
    user_data = {}

    name = input("Please enter your name: ")
    user_data['name'] = f"{name[0]}{'*' * (len(name) - 1)}"

    age = input("Please enter your age: ")
    try:
        user_data['age'] = int(age)
    except ValueError:
        user_data['age'] = "Invalid age input"

    email = input("Please enter your email: ")
    # Anonymize the email by masking part of it
    email_parts = email.split('@')
    if len(email_parts) == 2:
        local_part = email_parts[0]
        domain_part = email_parts[1]

        masked_local = local_part[0] + '*' * (len(local_part) - 2) + local_part[-1] if len(local_part)
        > 2 else local_part

        user_data['email'] = f"{masked_local}@{domain_part}"
    else:
        user_data['email'] = "Invalid email input"
```

```

    return user_data
collected_data = collect_user_data()
print("Collected User Data:", collected_data)
def collect_user_data():
    user_data = {}
    name = input("Please enter your name: ")
    # Anonymize the name by storing only the first letter and the length of the name
    user_data['name'] = f"{name[0]}{'*' * (len(name) - 1)}"
    age = input("Please enter your age: ")
    # Store age as is, but ensure it's a number
    try:
        user_data['age'] = int(age)
    except ValueError:
        user_data['age'] = "Invalid age input"
    email = input("Please enter your email: ")
    # Anonymize the email by masking part of it
    email_parts = email.split('@')
    if len(email_parts) == 2:
        local_part = email_parts[0]
        domain_part = email_parts[1]
        masked_local = local_part[0] + '*' * (len(local_part) - 2) + local_part[-1] if len(local_part)
        > 2 else local_part
        user_data['email'] = f"{masked_local}@{domain_part}"
    else:
        user_data['email'] = "Invalid email input"
    return user_data

```

Output:

The screenshot shows the Visual Studio Code editor with the file `assignment5.4.py` open. The code is a Python script that collects user data (name, age, email) and anonymizes it. The terminal output shows the following interaction:

```
PS C:\Users\Rashmi\OneDrive\Desktop\AI ASSISTED> python3.13.exe "c:\Users\Rashmi\OneDrive\Desktop\AI ASSISTED\assignment5.4.py"
Please enter your name: varsha
Please enter your age: 20
Please enter your email: 123sty@gmail.com
Collected User Data: {'name': 'v*****', 'age': 20, 'email': '1*****y@gmail.com'}
```

This screenshot shows the same Python script as the previous one, but with line numbers visible on the left side of the editor. The code is as follows:

```
1 #Generate a Python code that collects user data and add comments to anonymize or protect data.
2 # This script collects user data such as name, age, and email.
3 def collect_user_data():
4     # Initialize an empty dictionary to store user data
5     user_data = {}
6     # collect user's name
7     name = input("Please enter your name: ")
8     # Anonymize the name by storing only the first letter and the length of the name
9     user_data['name'] = f"{name[0]}{'*' * (len(name) - 1)}"
10
11     # collect user's age
12     age = input("Please enter your age: ")
13     # Store age as is, but ensure it's a number
14     try:
15         user_data['age'] = int(age)
16     except ValueError:
17         user_data['age'] = "Invalid age input"
18
19     # collect user's email
20     email = input("Please enter your email: ")
21     # Anonymize the email by masking part of it
22     email_parts = email.split('@')
23     if len(email_parts) == 2:
24         local_part = email_parts[0]
25         domain_part = email_parts[1]
26         masked_local = local_part[0] + '*' * (len(local_part) - 2) + local_part[-1] if len(local_part) > 2 else local_part
27         user_data['email'] = f"{masked_local}@{domain_part}"
28     else:
29         user_data['email'] = "Invalid email input"
30
31     return user_data
32 # Call the function and print the collected user data
33 collect_user_data()
34 print(user_data)
```

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

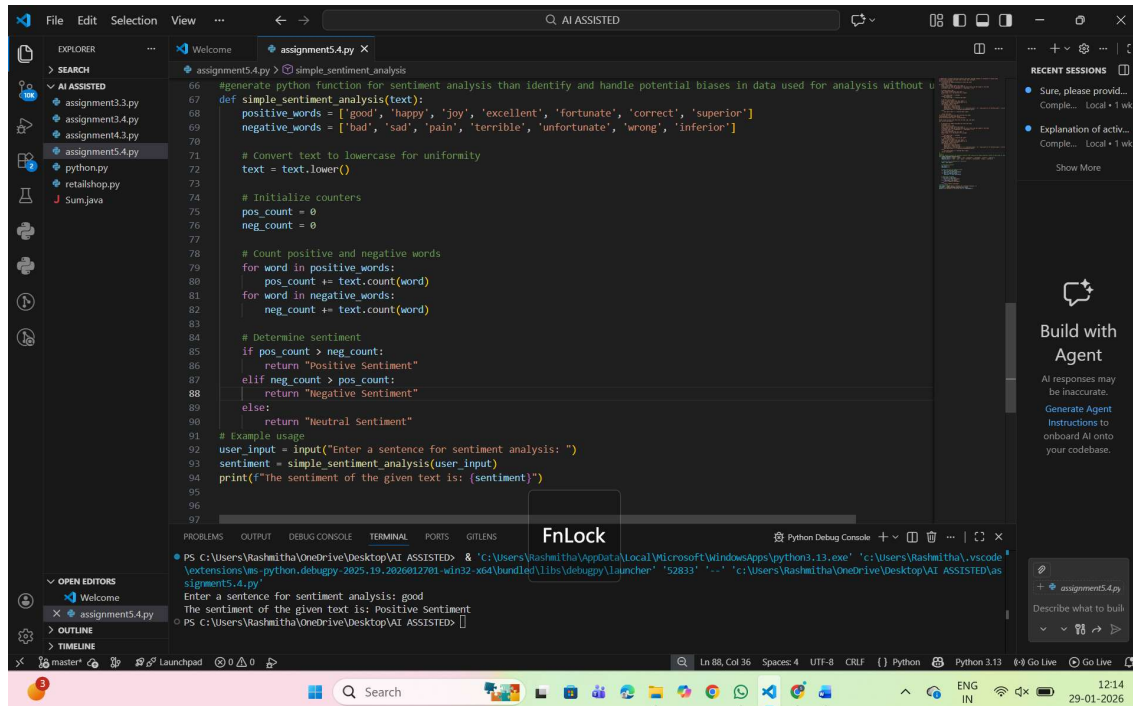
Prompt: generate python function for sentiment analysis than identify and handle potential biases in data used for analysis without using modules

Code:

generate python function for sentiment analysis than identify and handle potential biases in data used for analysis without using modules

```
def simple_sentiment_analysis(text):  
    positive_words = ['good', 'happy', 'joy', 'excellent', 'fortunate', 'correct', 'superior']  
    negative_words = ['bad', 'sad', 'pain', 'terrible', 'unfortunate', 'wrong', 'inferior']  
    text = text.lower()  
    pos_count = 0  
    neg_count = 0  
    for word in positive_words:  
        pos_count += text.count(word)  
    for word in negative_words:  
        neg_count += text.count(word)  
    if pos_count > neg_count:  
        return "Positive Sentiment"  
    elif neg_count > pos_count:  
        return "Negative Sentiment"  
    else:  
        return "Neutral Sentiment"  
  
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:



```
66 #generate python function for sentiment analysis than identify and handle potential biases in data used for analysis without u
67 def simple_sentiment_analysis(text):
68     positive_words = ['good', 'happy', 'joy', 'excellent', 'fortunate', 'correct', 'superior']
69     negative_words = ['bad', 'sad', 'pain', 'terrible', 'unfortunate', 'wrong', 'inferior']
70
71     # Convert text to lowercase for uniformity
72     text = text.lower()
73
74     # Initialize counters
75     pos_count = 0
76     neg_count = 0
77
78     # Count positive and negative words
79     for word in positive_words:
80         pos_count += text.count(word)
81     for word in negative_words:
82         neg_count += text.count(word)
83
84     # Determine sentiment
85     if pos_count > neg_count:
86         return "Positive Sentiment"
87     elif neg_count > pos_count:
88         return "Negative Sentiment"
89     else:
90         return "Neutral Sentiment"
91
92 # Example usage
93 user_input = input("Enter a sentence for sentiment analysis: ")
94 sentiment = simple_sentiment_analysis(user_input)
95 print(f"The sentiment of the given text is: {sentiment}")
96
97
```

Enter a sentence for sentiment analysis: good
The sentiment of the given text is: Positive Sentiment

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

Code: 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 = []

for category in user_history:

if category in products:

```

        recommendations.extend(products[category])

    if not recommendations:

        return "No recommendations available based on your history."

    return recommendations

user_history_input = ['electronics', 'books']

recommended_items = recommend_products(user_history_input)

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

print(recommended_items)

```

Output:

```

95 print(f"The sentiment of the given text is: {sentiment}")
96 #'''#'''Task 3'''#'''
97 ## Generate python program that recommends products based on user history and follow ethical guidelines to avoid manipulative pr
98 def recommend_products(user_history):
99     # Sample product database
100     products = {
101         'electronics': ['Smartphone', 'Laptop', 'Headphones'],
102         'books': ['Fiction Novel', 'Science Textbook', 'Biography'],
103         'clothing': ['T-Shirt', 'Jeans', 'Jacket']
104     }
105
106     recommendations = []
107
108     # Recommend products based on user history
109     for category in user_history:
110         if category in products:
111             recommendations.extend(products[category])
112
113     # Ethical guideline: Avoid recommending products that are not relevant to user's interests
114     if not recommendations:
115         return "No recommendations available based on your history."
116
117     return recommendations
118
119 # Example usage
120 user_history_input = ['electronics', 'books']
121 recommended_items = recommend_products(user_history_input)
122 print("Recommended Products based on your history:")
123 print(recommended_items)
124
125
126

```

Terminal Output:

```

PS C:\Users\Rashmi\OneDrive\Desktop\AI ASSISTED> python3.13.exe "c:\Users\Rashmi\OneDrive\Desktop\AI ASSISTED\assignment5.4.py"
Recommended Products based on your history:
['Smartphone', 'Laptop', 'Headphones', 'Fiction Novel', 'Science Textbook', 'Biography']

```

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-4:

Prompt: Generate python program that logging fuctionality in python web application and logs do not record senstive 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 file named `assignment5.4.py` open. The code in the editor is as follows:

```
120 recommended_items = recommend_products(user_history_input)
121 print("Recommended Products based on your history:")
122 print(recommended_items)
123
124 ##Task-4
125 #Generate python program that logging functionality in python web application and logs do not record sensitive information
126 import logging
127 # Configure logging
128 logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
129 def log_user_action(action, user_id=None):
130     # Avoid logging sensitive information like user_id
131     logging.info(f"User performed action: {action}")
132 # Example usage
133 log_user_action("Login")
134 log_user_action("Viewed Product Page")
135 log_user_action("Logout")
136
137
138
139
```

The terminal output at the bottom shows the execution of the script:

```
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED> & 'c:\Users\Rashmitha\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\Rashmitha\vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundle\vlibs\debugpy\launcher' '63849' '-' 'c:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED\assignment5.4.py'
2026-01-29 12:38:31,223 - INFO - User performed action: Login
2026-01-29 12:38:31,224 - INFO - User performed action: Viewed Product Page
2026-01-29 12:38:31,224 - INFO - User performed action: Logout
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED>
```

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

#

```

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.

```

"""

```

```

# Example usage

```

```

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

```

```

accuracy = simple_ml_model(input_data)

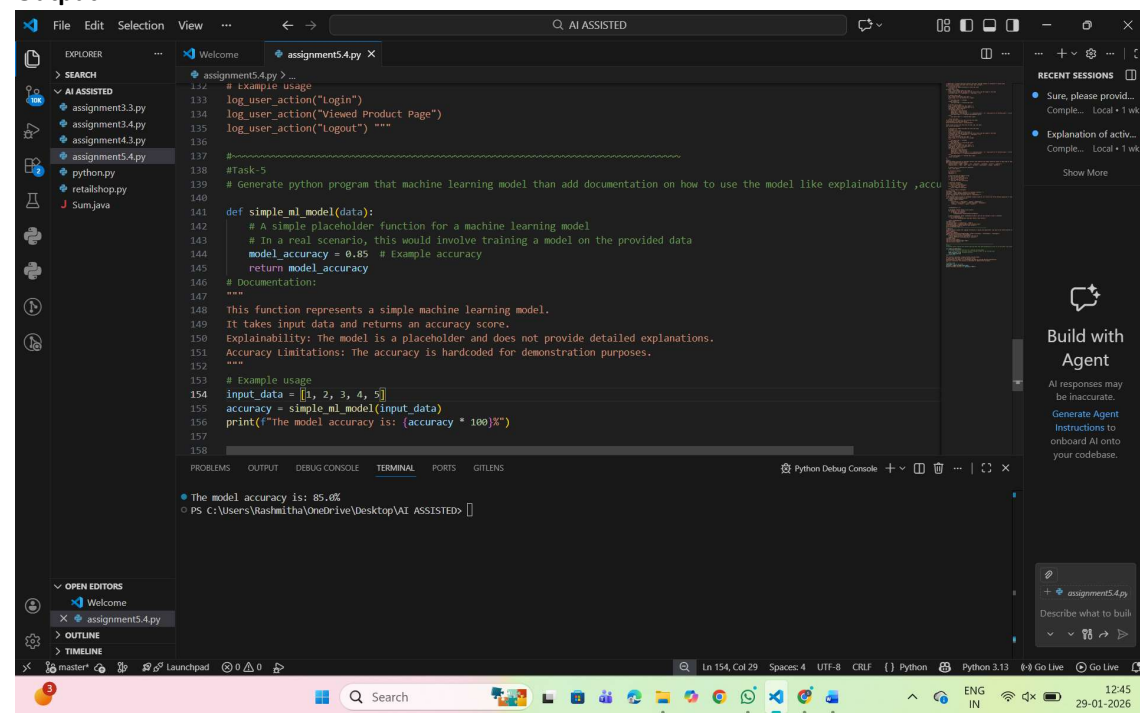
```

```

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

```

Output:



```

132 # Example usage
133 log_user_action("Login")
134 log_user_action("Viewed Product Page")
135 log_user_action("Logout") """
136
137 #-----
138 #Task-5
139 # Generate python program that machine learning model than add documentation on how to use the model like explainability ,accuracy
140
141 def simple_ml_model(data):
142     # A simple placeholder function for a machine learning model
143     # In a real scenario, this would involve training a model on the provided data
144     model_accuracy = 0.85 # Example accuracy
145     return model_accuracy
146 # Documentation:
147 """
148 This function represents a simple machine learning model.
149 It takes input data and returns an accuracy score.
150 Explainability: The model is a placeholder and does not provide detailed explanations.
151 Accuracy Limitations: The accuracy is hardcoded for demonstration purposes.
152 """
153 # Example usage
154 input_data = [1, 2, 3, 4, 5]
155 accuracy = simple_ml_model(input_data)
156 print(f"The model accuracy is: {accuracy * 100}%")
157
158

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GIT LENS

Python Debug Console

```

• The model accuracy is: 85.0%
• PS C:\Users\Vaishali\OneDrive\Desktop\AI ASSISTED>

```

RECENT SESSIONS

- Sure, please provid...
Complete... Local • 1 wk
- Explanation of activ...
Complete... Local • 1 wk

Show More

Build with Agent

AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

assignment5.4.py
Describe what to build

Ln 154, Col 29 Spaces: 4 UTF-8 CRLF Python Python 3.13 Go Live Go Live

12:45 29-01-2026

Analysis:It 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 explainations. Documentaionclearly mentions limitations in accuracy and explainability.