

## Assignment-5.4

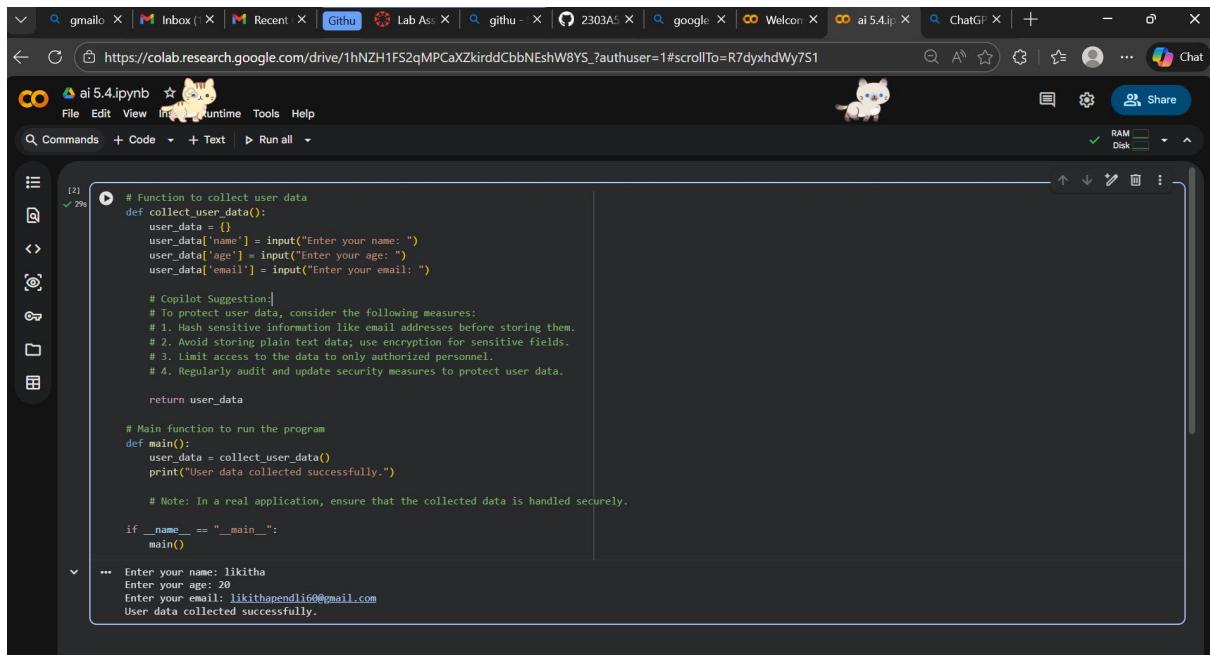
NAME:-pendli Likitha

HT NO:-2303A52393

BATCH:-43

### 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.



The screenshot shows a Google Colab notebook titled "ai 5.4.ipynb". The code cell contains a Python script for collecting user data. The script defines a function `collect\_user\_data` that prompts the user for their name, age, and email. It then prints a note about protecting sensitive information and provides four measures for security. The script also includes a main function that calls `collect\_user\_data` and prints a success message. The output pane shows the execution of the code and the user's input for name, age, and email, followed by the printed success message.

```
# Function to collect user data
def collect_user_data():
    user_data = {}
    user_data['name'] = input("Enter your name: ")
    user_data['age'] = input("Enter your age: ")
    user_data['email'] = input("Enter your email: ")

# Copilot Suggestion:
# To protect user data, consider the following measures:
# 1. Hash sensitive information like email addresses before storing them.
# 2. Avoid storing plain text data; use encryption for sensitive fields.
# 3. Limit access to the data to only authorized personnel.
# 4. Regularly audit and update security measures to protect user data.

return user_data

# Main function to run the program
def main():
    user_data = collect_user_data()
    print("User data collected successfully.")

    # Note: In a real application, ensure that the collected data is handled securely.

if __name__ == "__main__":
    main()

--- Enter your name: likitha
Enter your age: 20
Enter your email: likithapendli60@gmail.com
User data collected successfully.
```

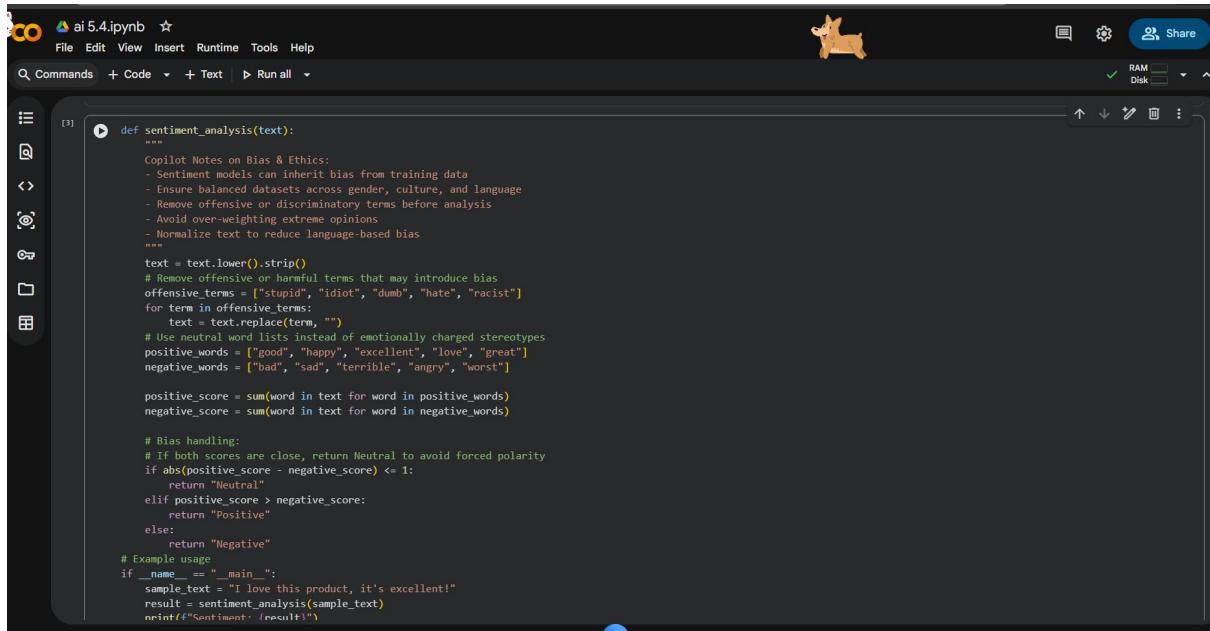
### Expected Output #1:

- 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).

### Task Description #2:

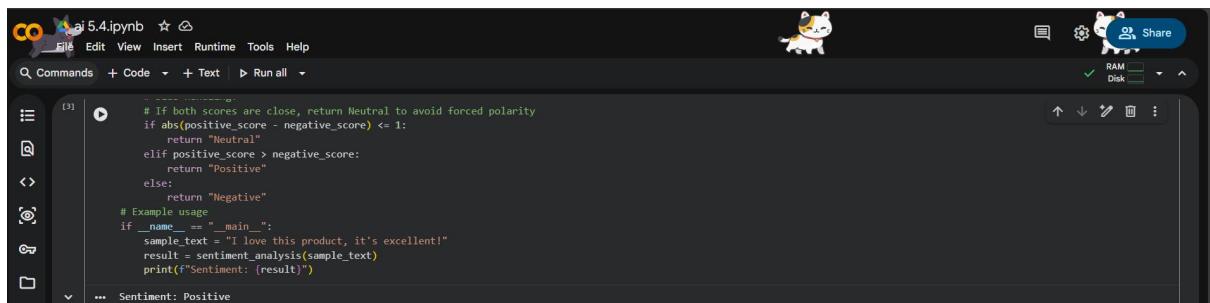
- Ask Copilot to generate a Python function for sentiment analysis. Then prompt Copilot to identify and handle potential biases in the

data.



```
def sentiment_analysis(text):
    """
    Copilot Notes on Bias & Ethics:
    - Sentiment models can inherit bias from training data
    - Ensure balanced datasets across gender, culture, and language
    - Remove offensive or discriminatory terms before analysis
    - Avoid over-weighting extreme opinions
    - Normalize text to reduce language-based bias
    """
    text = text.lower().strip()
    # Remove offensive or harmful terms that may introduce bias
    offensive_terms = ["stupid", "idiot", "dumb", "hate", "racist"]
    for term in offensive_terms:
        text = text.replace(term, "")
    # Use neutral word lists instead of emotionally charged stereotypes
    positive_words = ["good", "happy", "excellent", "love", "great"]
    negative_words = ["bad", "sad", "terrible", "angry", "worst"]
    positive_score = sum(word in text for word in positive_words)
    negative_score = sum(word in text for word in negative_words)

    # Bias handling:
    # If both scores are close, return Neutral to avoid forced polarity
    if abs(positive_score - negative_score) <= 1:
        return "Neutral"
    elif positive_score > negative_score:
        return "Positive"
    else:
        return "Negative"
    # Example usage
if __name__ == "__main__":
    sample_text = "I love this product, it's excellent!"
    result = sentiment_analysis(sample_text)
    print(f"Sentiment: {result}")
```



```
# If both scores are close, return Neutral to avoid forced polarity
if abs(positive_score - negative_score) <= 1:
    return "Neutral"
elif positive_score > negative_score:
    return "Positive"
else:
    return "Negative"
# Example usage
if __name__ == "__main__":
    sample_text = "I love this product, it's excellent!"
    result = sentiment_analysis(sample_text)
    print(f"Sentiment: {result}")
... Sentiment: Positive
```

## Expected Output #2:

- Copilot-generated code with additions or comments addressing bias mitigation strategies (e.g., balancing dataset, removing offensive terms).

## 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.

The screenshot shows the Copilot interface with a dark theme. A file named "ai 5.4.ipynb" is open. The code defines a function "recommend\_products" that takes "user\_history" as input. It includes Copilot ethical guidelines, a fairness check (not promoting a single brand), transparency messages, and example usage. The code also includes a warning about avoiding favoritism and a note about user feedback.

```
def recommend_products(user_history):
    """
    Recommends products based on past user behavior.

    Copilot ethical guidelines:
    - Avoid favoritism toward paid or sponsored products
    - Explain recommendations to users
    - Allow feedback or opt-out
    """

    recommendations = []

    # Fairness check: do not over-promote a single brand
    for product in user_history:
        if product not in recommendations:
            recommendations.append(product)

    # Transparency message
    print("Recommendations are based on your browsing history.")
    print("You can reset or customize recommendations anytime.")

    return recommendations

# Example usage
history = ["Laptop", "Headphones", "Laptop"]
print(recommend_products(history))

...
... Recommendations are based on your browsing history.
You can reset or customize recommendations anytime.
['Laptop', 'Headphones']
```

### Expected Output #3:

- Copilot suggestions that include explanations, fairness checks (e.g., avoiding favoritism), and user feedback options in the code.

### 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.

The screenshot shows the Copilot interface with a dark theme. A file named "ai 5.4.ipynb" is open. The code imports logging and defines a "login" function. It logs an info message for successful logins and a warning message for failed logins. It also includes example usage where it prompts for username and password and prints "Welcome!" or "Invalid credentials." based on the password.

```
import logging
# Configure Logging
logging.basicConfig(
    filename='app.log',
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s'
)
def login(username, password):
    # Copilot warning:
    # NEVER log passwords, emails, or personal identifiers

    logging.info("Login attempt for user") # Do not log username

    if password == "admin123":
        logging.info("Login successful")
        return True
    else:
        logging.warning("Login failed")
        return False
# Example usage
if __name__ == "__main__":
    username = input("Enter username: ")
    password = input("Enter password: ")

    if login(username, password):
        print("Welcome!")
    else:
        print("Invalid credentials.")

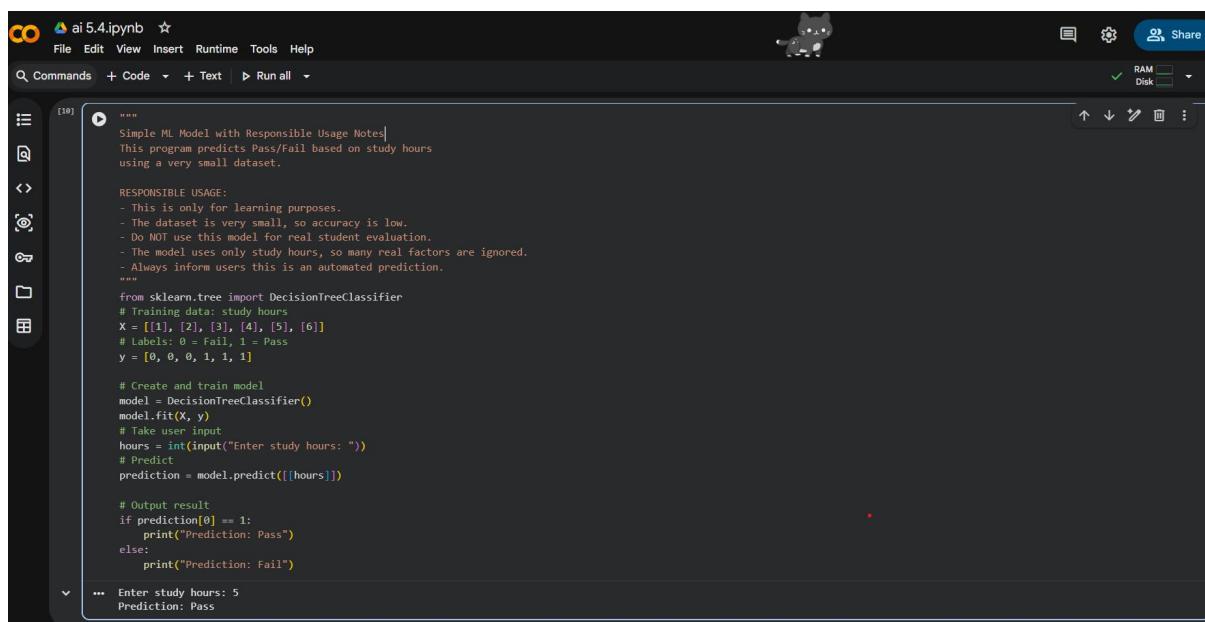
...
... Enter username: likitha
Enter password: admin123
Welcome!
```

## Expected Output #4:

- Logging code that avoids saving personal identifiers (e.g., passwords, emails), and includes comments about ethical logging practices.

## 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).



The screenshot shows a Jupyter Notebook interface with a dark theme. The title bar says "ai 5.4.ipynb". The code cell contains the following Python script:

```
"""
Simple ML Model with Responsible Usage Notes
This program predicts Pass/Fail based on study hours
using a very small dataset.

RESPONSIBLE USAGE:
- This is only for learning purposes.
- The dataset is very small, so accuracy is low.
- Do NOT use this model for real student evaluation.
- The model uses only study hours, so many real factors are ignored.
- Always inform users this is an automated prediction.

from sklearn.tree import DecisionTreeClassifier
# Training data: study hours
X = [[1], [2], [3], [4], [5], [6]]
# Labels: 0 = Fail, 1 = Pass
y = [0, 0, 0, 1, 1, 1]

# Create and train model
model = DecisionTreeClassifier()
model.fit(X, y)
# Take user input
hours = int(input("Enter study hours: "))
# Predict
prediction = model.predict([[hours]])

# Output result
if prediction[0] == 1:
    print("Prediction: Pass")
else:
    print("Prediction: Fail")
...
Enter study hours: 5
Prediction: Pass

```

## Expected Output #5:

- Copilot-generated model code with a README or inline documentation suggesting responsible usage, limitations, and fairness considerations.