

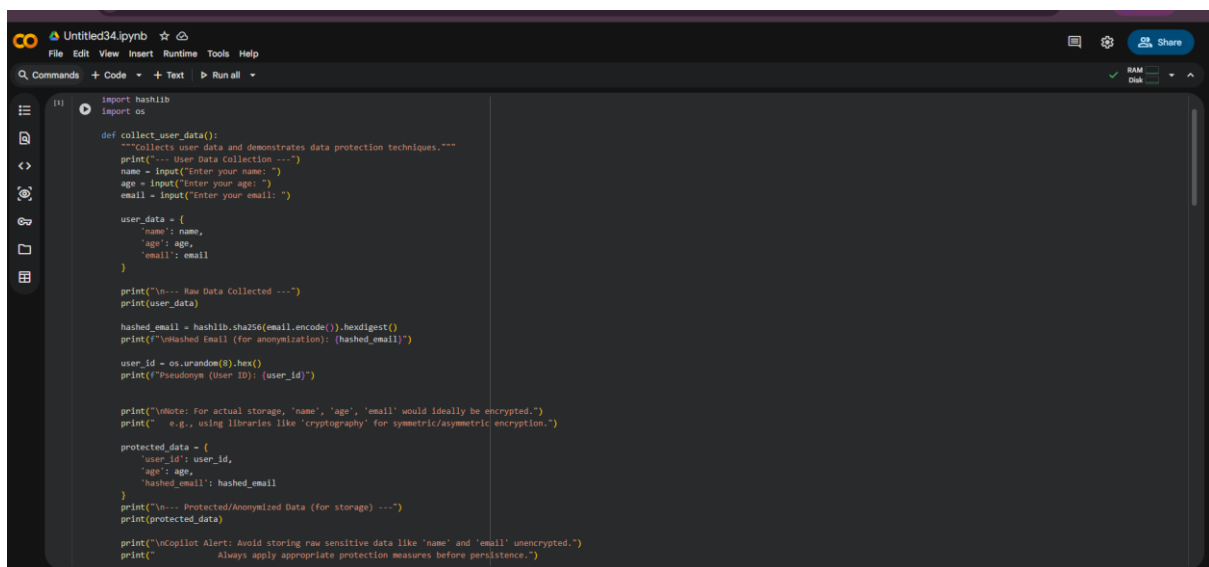
## Assignment-5.4

2303A52292

### Batch-43

Task 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.



```
import hashlib
import os

def collect_user_data():
    """Collects user data and demonstrates data protection techniques."""
    print("--- User Data Collection ---")
    name = input("Enter your name: ")
    age = input("Enter your age: ")
    email = input("Enter your email: ")

    user_data = {
        'name': name,
        'age': age,
        'email': email
    }

    print("\n--- Raw Data Collected ---")
    print(user_data)

    hashed_email = hashlib.sha256(email.encode()).hexdigest()
    print(f"\nHashed Email (for anonymization): {hashed_email}")

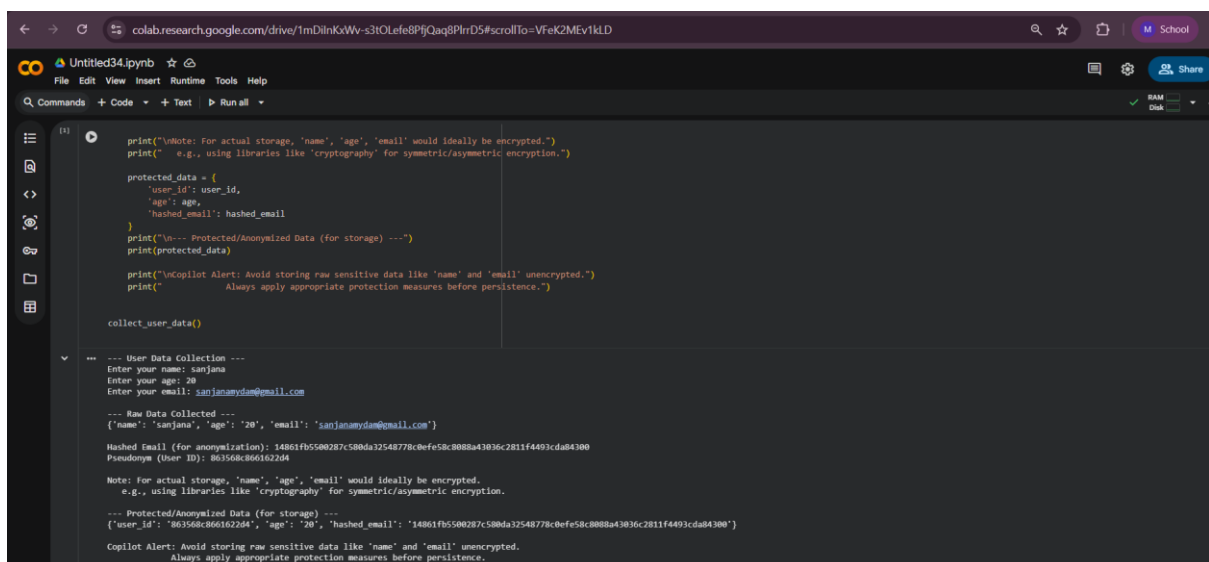
    user_id = os.urandom(8).hex()
    print(f"Pseudonym (User ID): {user_id}")

    print("\nNote: For actual storage, 'name', 'age', 'email' would ideally be encrypted.")
    print("      e.g., using libraries like 'cryptography' for symmetric/asymmetric encryption.")

    protected_data = {
        'user_id': user_id,
        'age': age,
        'hashed_email': hashed_email
    }

    print("\n--- Protected/Anonymized Data (for storage) ---")
    print(protected_data)

    print("\nCopilot Alert: Avoid storing raw sensitive data like 'name' and 'email' unencrypted.")
    print("      Always apply appropriate protection measures before persistence.")
```



```
print("\nNote: For actual storage, 'name', 'age', 'email' would ideally be encrypted.")
print("      e.g., using libraries like 'cryptography' for symmetric/asymmetric encryption.")

protected_data = {
    'user_id': user_id,
    'age': age,
    'hashed_email': hashed_email
}

print("\n--- Protected/Anonymized Data (for storage) ---")
print(protected_data)

print("\nCopilot Alert: Avoid storing raw sensitive data like 'name' and 'email' unencrypted.")
print("      Always apply appropriate protection measures before persistence.")

collect_user_data()

--- User Data Collection ---
Enter your name: sanjana
Enter your age: 20
Enter your email: sanjanamydam@gmail.com

--- Raw Data Collected ---
{'name': 'sanjana', 'age': '20', 'email': 'sanjanamydam@gmail.com'}

Hashed Email (for anonymization): 14861f5508287c580da32548778c0efe58c8088a43036c2811f4493da84300
Pseudonym (User ID): 863568c866162204

Note: For actual storage, 'name', 'age', 'email' would ideally be encrypted.
      e.g., using libraries like 'cryptography' for symmetric/asymmetric encryption.

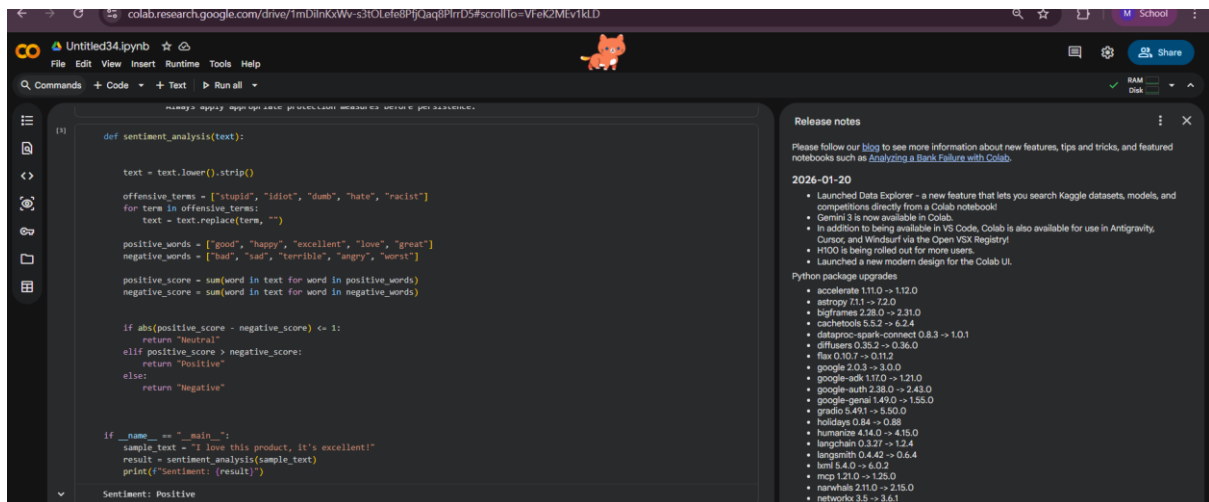
--- Protected/Anonymized Data (for storage) ---
{'user_id': '863568c866162204', 'age': '20', 'hashed_email': '14861f5508287c580da32548778c0efe58c8088a43036c2811f4493da84300'}

Copilot Alert: Avoid storing raw sensitive data like 'name' and 'email' unencrypted.
      Always apply appropriate protection measures before persistence.
```

Task 2:

- Ask Copilot to generate a Python function for sentiment analysis.

Then prompt Copilot to identify and handle potential biases in the data.

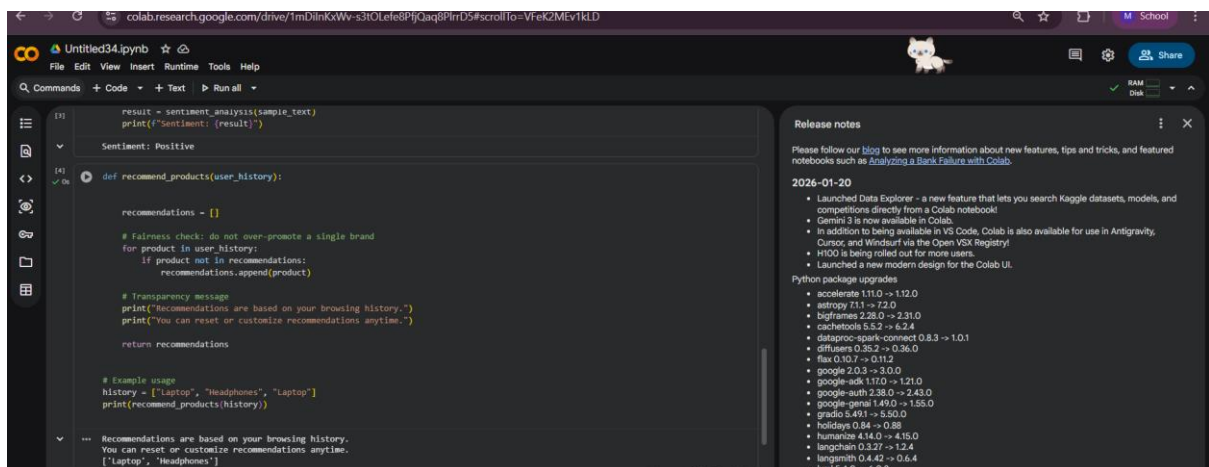


The screenshot shows a Google Colab notebook titled 'Untitled34.ipynb'. The code defines a `sentiment_analysis(text)` function that processes a string of text to determine its sentiment. It uses lists of offensive, positive, and negative words. The function returns 'Neutral', 'Positive', or 'Negative' based on the scores. The output of the function is 'Sentiment: Positive'. On the right side, there is a 'Release notes' panel with information about new features and package updates.

```
[1] def sentiment_analysis(text):  
  
    text = text.lower().strip()  
  
    offensive_terms = ["stupid", "idiot", "dumb", "hate", "racist"]  
    for term in offensive_terms:  
        text = text.replace(term, "")  
  
    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)  
  
    if abs(positive_score - negative_score) <= 1:  
        return "Neutral"  
    elif positive_score > negative_score:  
        return "Positive"  
    else:  
        return "Negative"  
  
if __name__ == "__main__":  
    sample_text = "I love this product, it's excellent!"  
    result = sentiment_analysis(sample_text)  
    print(f"Sentiment: {result}")  
  
Sentiment: Positive
```

Task 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.

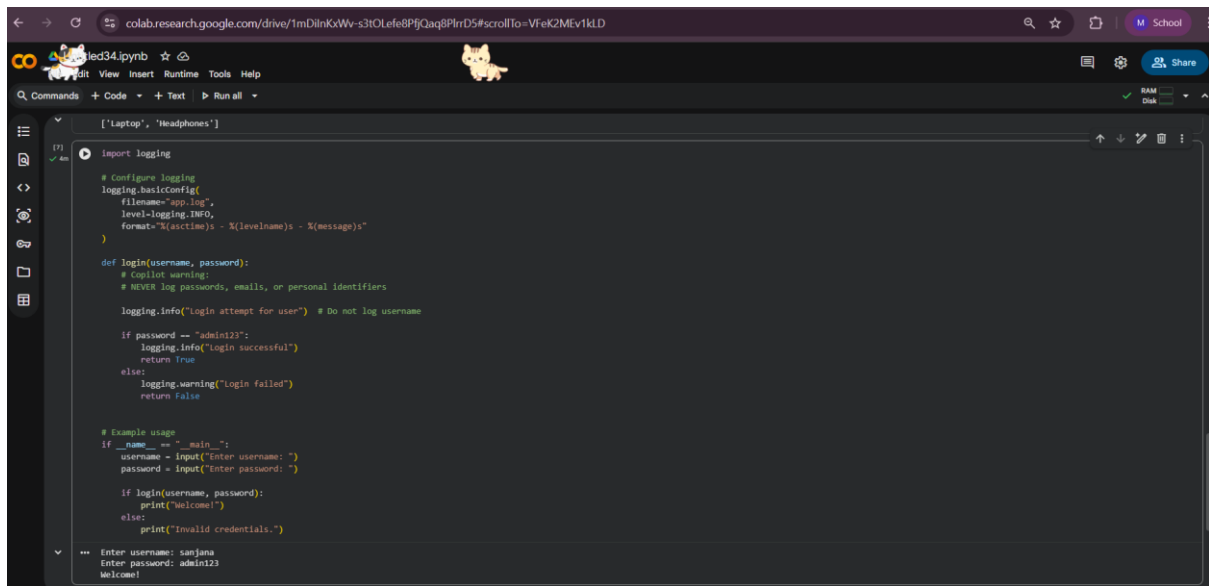


The screenshot shows a Google Colab notebook titled 'Untitled34.ipynb'. The code defines a `recommend_products(user_history)` function that takes a list of user history and returns a list of recommended products. It includes a fairness check to avoid promoting a single brand and a transparency message. The output of the function is 'Recommendations are based on your browsing history. You can reset or customize recommendations anytime. ['Laptop', 'Headphones']'. On the right side, there is a 'Release notes' panel with information about new features and package updates.

```
[1] result = sentiment_analysis(sample_text)  
print(f"Sentiment: {result}")  
  
Sentiment: Positive  
  
[4] def recommend_products(user_history):  
  
    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']
```

Task 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 a Google Colab notebook with a dark theme. The code defines a logging module with a login function. The output shows a successful login for 'sanjana'.

```
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

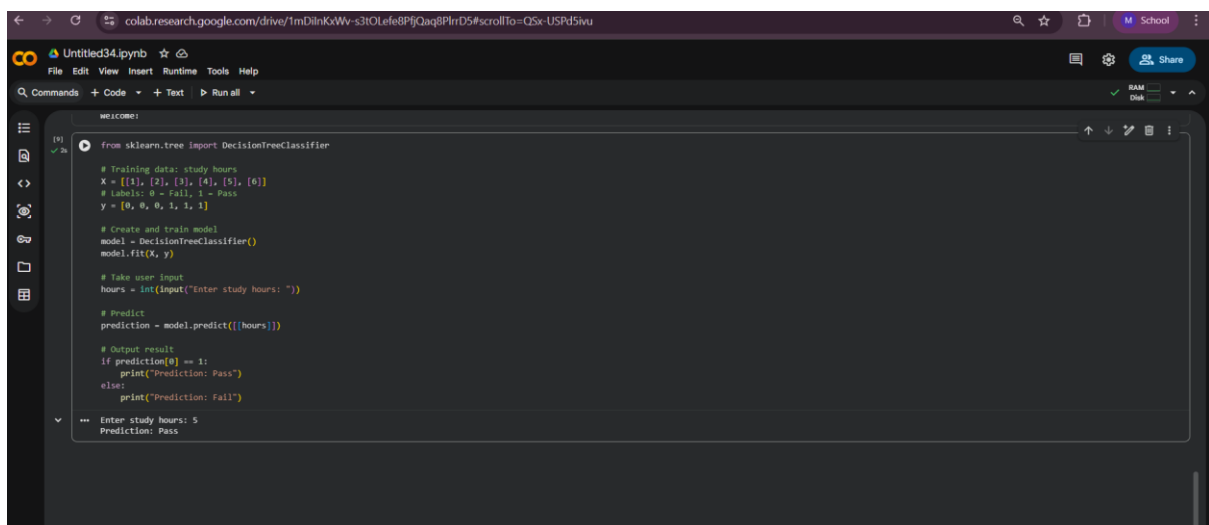
# Example usage
if __name__ == "__main__":
    username = input("Enter username: ")
    password = input("Enter password: ")

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

Enter username: sanjana
Enter password: admin123
Welcome!
```

## Task 5:

- Ask Copilot to generate a machine learning model. Then, prompt it to add documentation on how to use the model responsibly



The screenshot shows a Google Colab notebook with a dark theme. The code implements a simple machine learning model using a DecisionTreeClassifier. The output shows a prediction of 'Pass' for 5 study hours.

```
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
```