Assignment-5.4

R. Nikhil Kumar

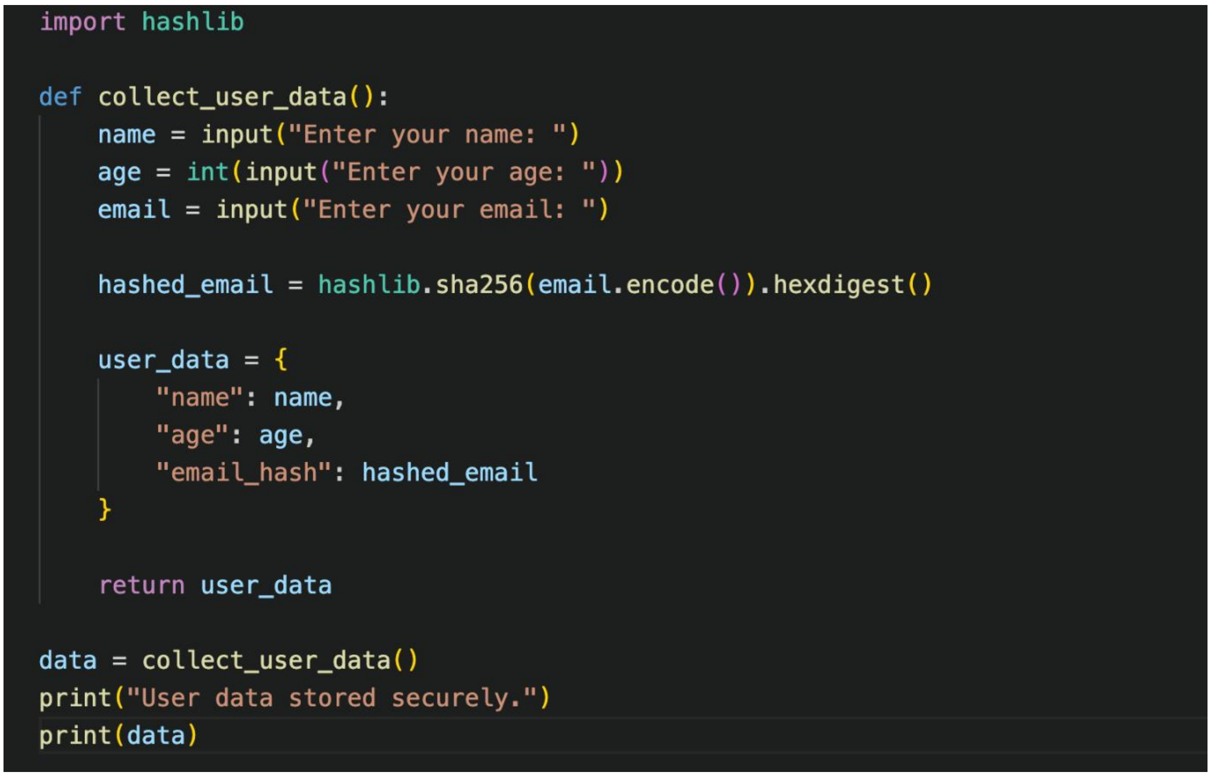
2303A52260

Batch-44

Task 1:

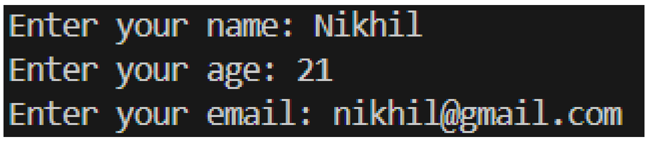
Prompt GitHub Copilot to generate a Python script that collects user data (e.g., name, age, email). Then, ask Copilot to add comments on how to anonymize or protect this data.

Code:



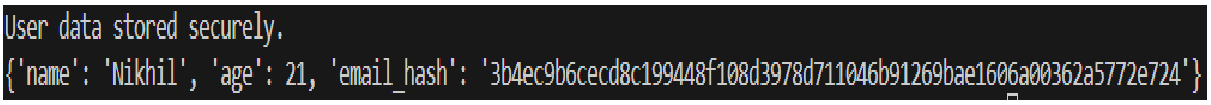
```python
import hashlib

def collect_user_data():
    name = input("Enter your name: ")
    age = int(input("Enter your age: "))
    email = input("Enter your email: ")

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

    user_data = {
        "name": name,
        "age": age,
        "email_hash": hashed_email
    }

    return user_data

data = collect_user_data()
print("User data stored securely.")
print(data)
```

Input:



```
Enter your name: Nikhil
Enter your age: 21
Enter your email: nikhil@gmail.com
```

Output:



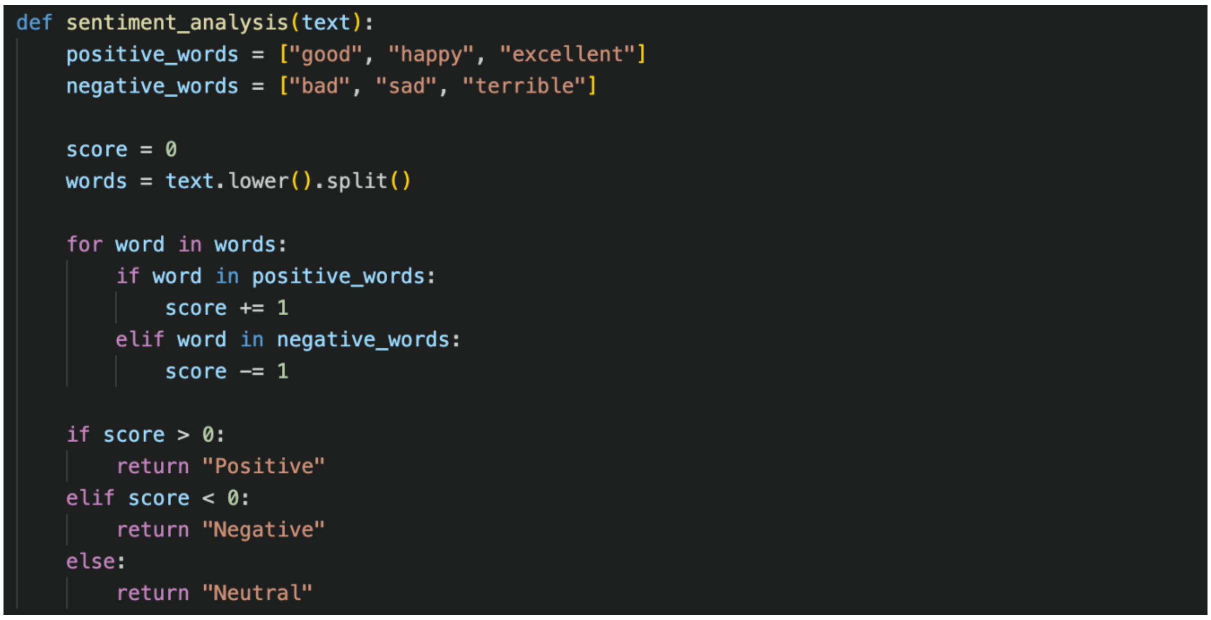
```
User data stored securely.
{'name': 'Nikhil', 'age': 21, 'email_hash': '3b4ec9b6cecd8c199448f108d3978d711046b91269bae1606a00362a5772e724'}
```

Task 2:

Ask Copilot to generate a Python function for sentiment analysis.

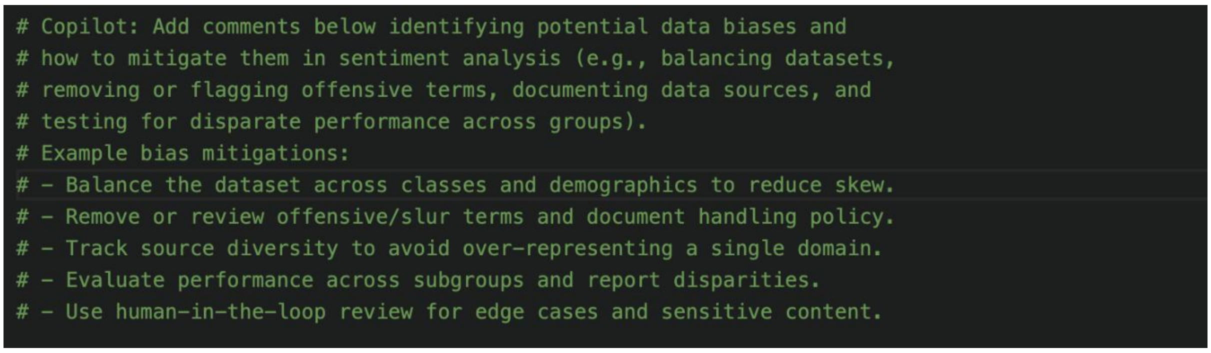
Then prompt Copilot to identify and handle potential biases in the data.

Code:



```python
def sentiment_analysis(text):
    positive_words = ["good", "happy", "excellent"]
    negative_words = ["bad", "sad", "terrible"]

    score = 0
    words = text.lower().split()

    for word in words:
        if word in positive_words:
            score += 1
        elif word in negative_words:
            score -= 1

    if score > 0:
        return "Positive"
    elif score < 0:
        return "Negative"
    else:
        return "Neutral"
```

Comments:



```python
# Copilot: Add comments below identifying potential data biases and
# how to mitigate them in sentiment analysis (e.g., balancing datasets,
# removing or flagging offensive terms, documenting data sources, and
# testing for disparate performance across groups).
# Example bias mitigations:
# - Balance the dataset across classes and demographics to reduce skew.
# - Remove or review offensive/slur terms and document handling policy.
# - Track source diversity to avoid over-representing a single domain.
# - Evaluate performance across subgroups and report disparities.
# - Use human-in-the-loop review for edge cases and sensitive content.
```

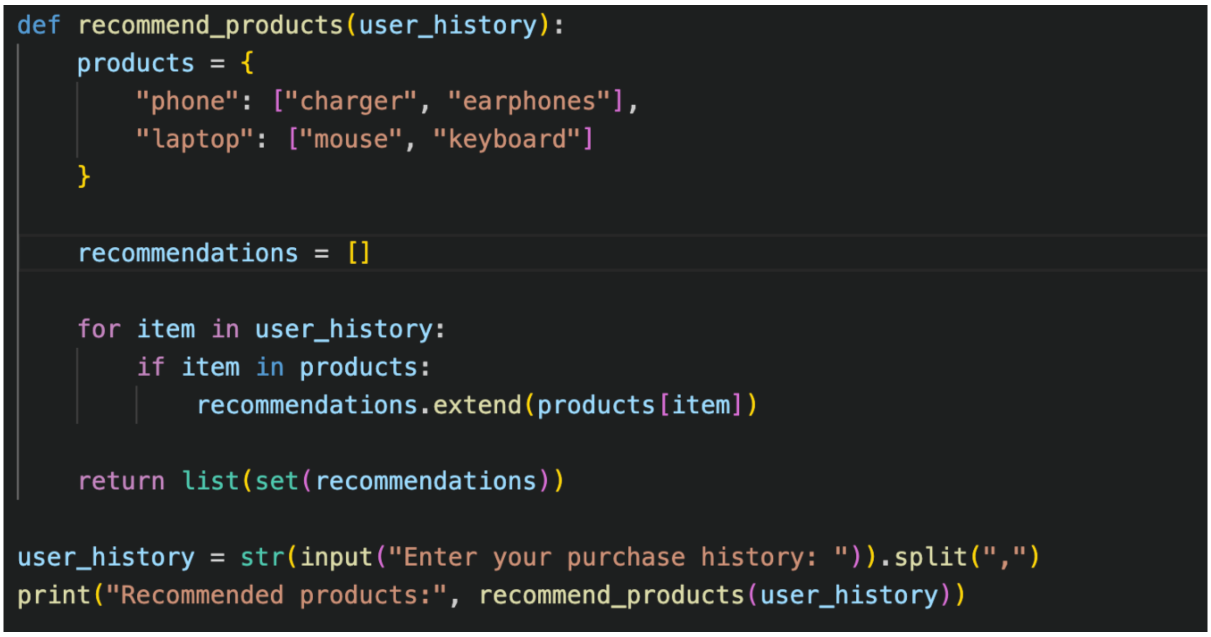
Task 3:

Use Copilot to write a Python program that recommends

products based on user history. Ask it to follow ethical guidelines

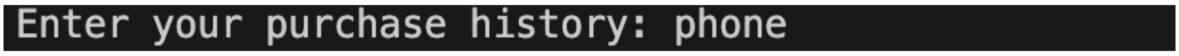
like transparency and fairness.

Code:



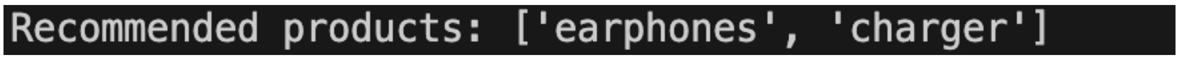
```python
def recommend_products(user_history):
    products = {
        "phone": ["charger", "earphones"],
        "laptop": ["mouse", "keyboard"]
    }

    recommendations = []

    for item in user_history:
        if item in products:
            recommendations.extend(products[item])

    return list(set(recommendations))

user_history = str(input("Enter your purchase history: ")).split(",")
print("Recommended products:", recommend_products(user_history))
```

Input 1:



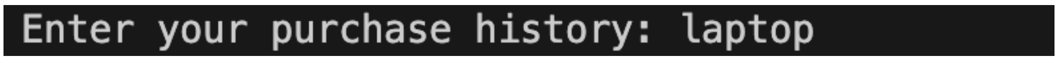
```
Enter your purchase history: phone
```

Output 1:



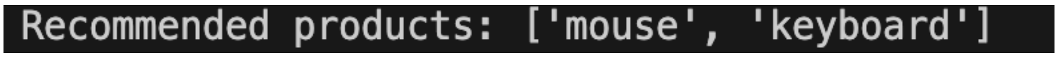
```
Recommended products: ['earphones', 'charger']
```

Input 2:



```
Enter your purchase history: laptop
```

Output 2:

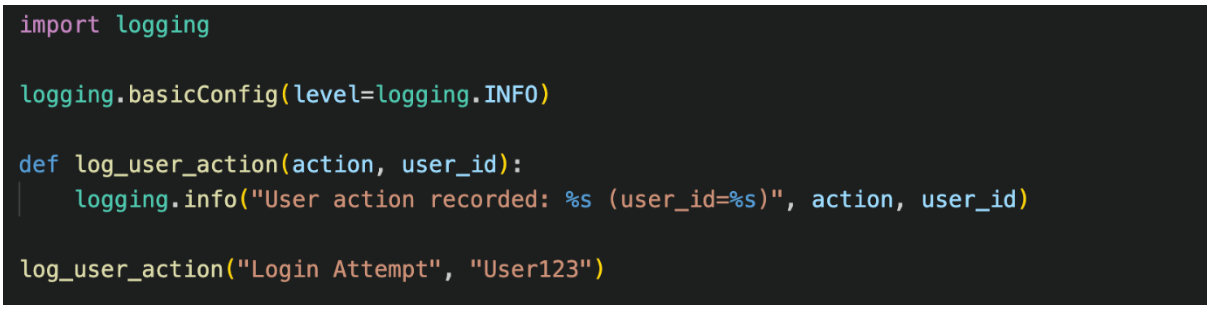


```
Recommended products: ['mouse', 'keyboard']
```

Task 4:

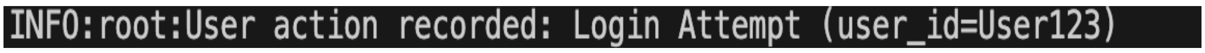
Prompt Copilot to generate logging functionality in a Python web application. Then, ask it to ensure the logs do not record sensitive information.

Code:



```python
import logging

logging.basicConfig(level=logging.INFO)

def log_user_action(action, user_id):
    logging.info("User action recorded: %s (user_id=%s)", action, user_id)

log_user_action("Login Attempt", "User123")
```

Output:

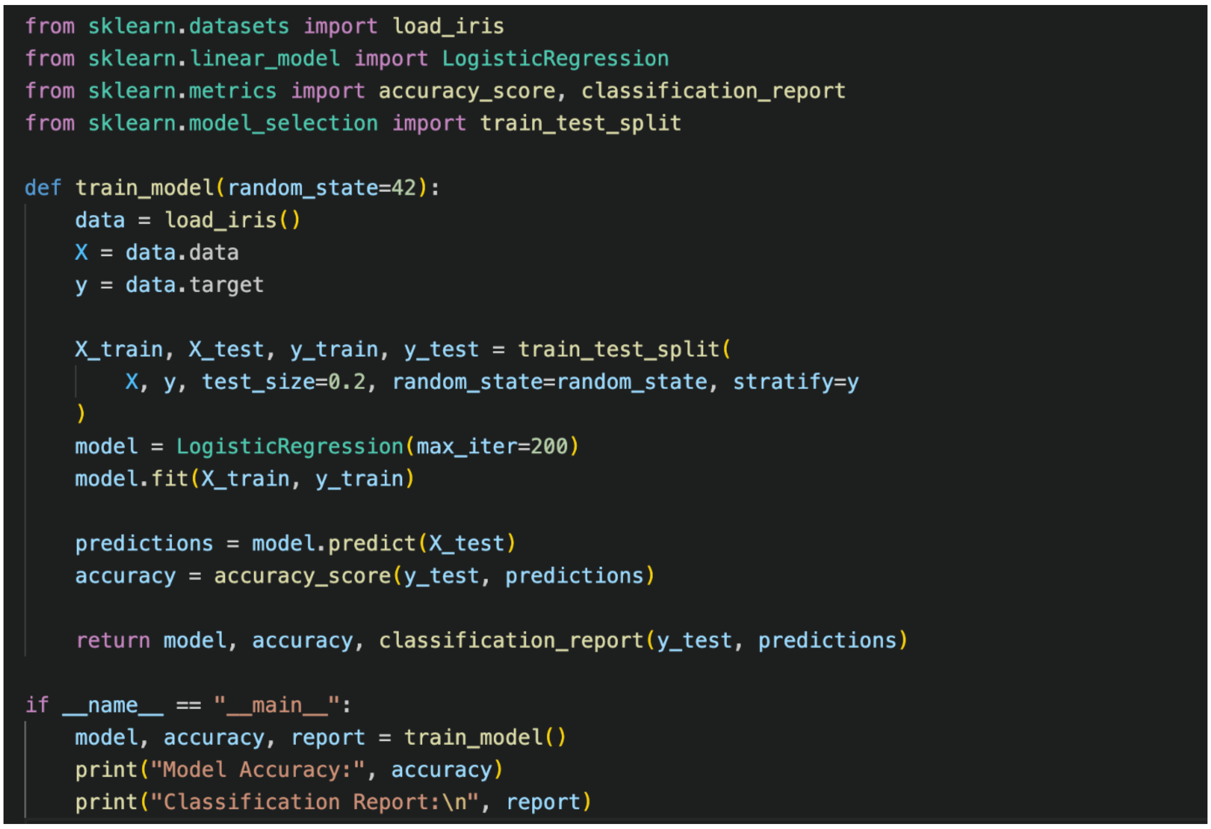


```
INFO:root:User action recorded: Login Attempt (user_id=User123)
```

Task 5:

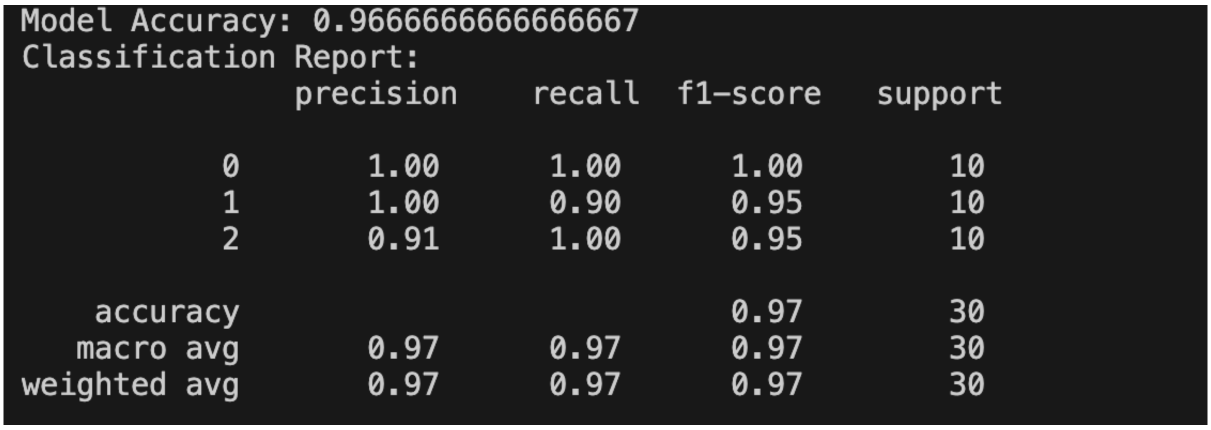
Ask Copilot to generate a machine learning model. Then, prompt it to add documentation on how to use the model responsibly (e.g., explainability, accuracy limits).

Code:



```python
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split

def train_model(random_state=42):
    data = load_iris()
    X = data.data
    y = data.target

    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=random_state, stratify=y
    )
    model = LogisticRegression(max_iter=200)
    model.fit(X_train, y_train)

    predictions = model.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)

    return model, accuracy, classification_report(y_test, predictions)

if __name__ == "__main__":
    model, accuracy, report = train_model()
    print("Model Accuracy:", accuracy)
    print("Classification Report:\n", report)
```

Output:



```
Model Accuracy: 0.9666666666666667
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      0.90      0.95        10
           2       0.91      1.00      0.95        10

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30
```