AI ASSISTED CODING:

ASSIGNMENT:5.4

M.KEERTHANA
2403A51259
BATCH-11(CSE-GEN)

TASK1:

PROMPT:

create a Python script that collects user data like name, age, email, add comments on how to protect/anonymize the data.

CODE & OUTPUT:

TASK1.py > ...

```python
# Collect user data: name, age, email
name = input("Enter your name: ")
age = input("Enter your age: ")
email = input("Enter your email: ")

print("\n--- User Data Collected ---")
print(f"Name: {name}")
print(f"Age: {age}")
print(f"Email: {email}")

# To anonymize or protect this data:
# - Do not store raw personal data; use pseudonyms or user IDs.
# - Hash or encrypt sensitive fields like email before saving.
# - Limit access to data and use secure storage solutions.
# - Remove or mask identifying information before
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS CODE REFERENCE LOG

```
PS C:\Users\P\Desktop\AIcoding> & C:\Users\P\AppData\Local\Programs\Python\Python313\python.exe c:/Users/P/Desktop/AIcoding/TASK1.py
Enter your name: KITTU
Enter your age: 19
Enter your email: adc@gmail.com

--- User Data Collected ---
Name: KITTU
Age: 19
Email: adc@gmail.com
PS C:\Users\P\Desktop\AIcoding>
```

powershell
pow...

TASK2:

PROMPT:

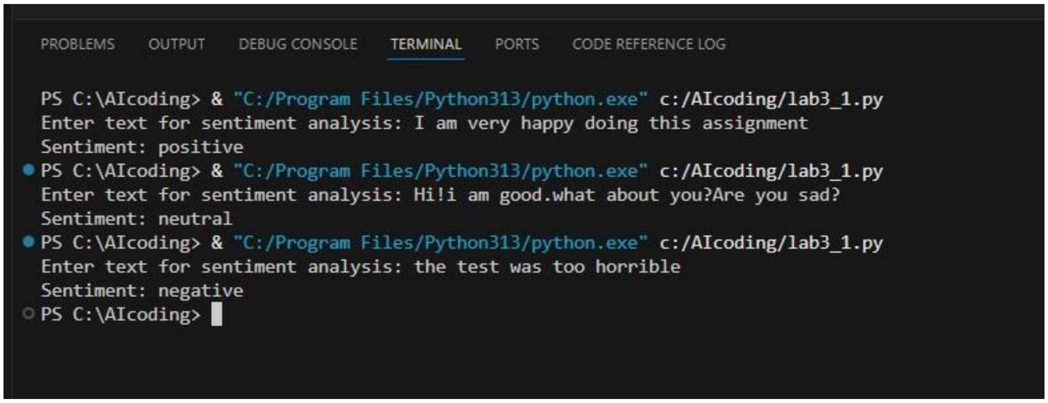
write a Python function that does sentiment analysis, identify and handle biases in training data or results.

CODE:

```
p3_1.py > ...
# Simple sentiment analysis function (rule-based, no external libraries)
Tabnine | Edit | Test | Explain | Document
def simple_sentiment(text):
    """Return sentiment label for the given text using a basic rule-based approach."""
    positive_words = ['good', 'happy', 'excellent', 'great', 'love', 'wonderful', 'best', 'fantastic']
    negative_words = ['bad', 'sad', 'terrible', 'worst', 'hate', 'awful', 'poor', 'horrible']
    text_lower = text.lower()
    pos = sum(word in text_lower for word in positive_words)
    neg = sum(word in text_lower for word in negative_words)
    if pos > neg:
        return 'positive'
    elif neg > pos:
        return 'negative'
    else:
        return 'neutral'

# Example usage
if __name__ == "__main__":
    text = input("Enter text for sentiment analysis: ")
    print(f"Sentiment: {simple_sentiment(text)}")

# --- Identifying and Handling Potential Biases in Data ---
# 1. Check for class imbalance (e.g., more positive than negative samples).
# 2. Review data sources for demographic or topical bias.
# 3. Use diverse and representative datasets for training and testing.
# 4. Regularly evaluate model predictions for fairness and accuracy.
# 5. Apply techniques like re-sampling, re-weighting, or data augmentation to mitigate bias.
```

```
# Example usage
if __name__ == "__main__":
    text = input("Enter text for sentiment analysis: ")
    print(f"Sentiment: {simple_sentiment(text)}")

# --- Identifying and Handling Potential Biases in Data ---
# 1. Check for class imbalance (e.g., more positive than negative samples).
# 2. Review data sources for demographic or topical bias.
# 3. Use diverse and representative datasets for training and testing.
# 4. Regularly evaluate model predictions for fairness and accuracy.
# 5. Apply techniques like re-sampling, re-weighting, or data augmentation to mitigate bias.
# Example: Check class distribution in a dataset
#
# from collections import Counter
# labels = ['positive', 'negative', 'neutral', 'positive', 'positive']  # Example labels
# print(Counter(labels))  # Shows class counts
# If imbalance is found, consider collecting more data for underrepresented classes.
```

OUTPUT:



```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    CODE REFERENCE LOG

PS C:\AIcoding> & "C:/Program Files/Python313/python.exe" c:/AIcoding/lab3_1.py
Enter text for sentiment analysis: I am very happy doing this assignment
Sentiment: positive
PS C:\AIcoding> & "C:/Program Files/Python313/python.exe" c:/AIcoding/lab3_1.py
Enter text for sentiment analysis: Hi!i am good.what about you?Are you sad?
Sentiment: neutral
PS C:\AIcoding> & "C:/Program Files/Python313/python.exe" c:/AIcoding/lab3_1.py
Enter text for sentiment analysis: the test was too horrible
Sentiment: negative
PS C:\AIcoding>
```

TASK 3:

PROMPT:

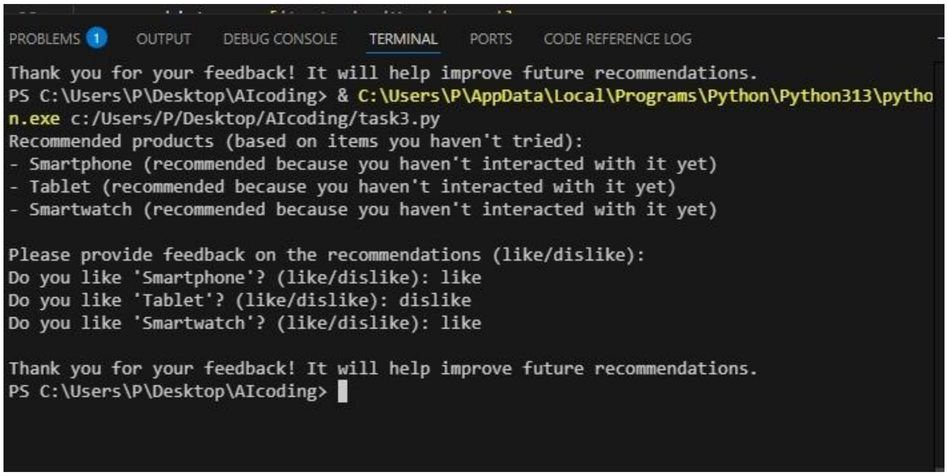
Generate a Python program that recommends products based on user history,Add comments about ethical guidelines like transparency and fairness and give users feedback options.

CODE:

```
task3.py > ...
# Simple product recommendation system based on user history

Tabnine | Edit | Test | Explain | Document
def recommend_products(user_history, all_products):
    """
    Recommend products based on user's purchase/view history.
    Ensures transparency by showing why products are recommended.
    """
    # Recommend products not already in user history
    recommendations = [product for product in all_products if product not in user_histo
    print("Recommended products (based on items you haven't tried):")
    for product in recommendations:
        print(f"- {product} (recommended because you haven't interacted with it yet)")
    return recommendations

Tabnine | Edit | Test | Explain | Document
def get_user_feedback(recommendations):
    """
    Allow users to provide feedback on recommendations.
    """
    print("\nPlease provide feedback on the recommendations (like/dislike):")
    feedback = {}
    for product in recommendations:
        response = input(f"Do you like '{product}'? (like/dislike): ").strip().lower()
        feedback[product] = response
    print("\nThank you for your feedback! It will help improve future recommendations."
    return feedback

```

```
task3.py > ...
def get_user_feedback(recommendations):
    feedback = {}
    for product in recommendations:
        response = input(f"Do you like '{product}'? (like/dislike): ").strip().lower()
        feedback[product] = response
    print("\nThank you for your feedback! It will help improve future recommendations."
    return feedback

# Example usage
if __name__ == "__main__":
    user_history = ['Laptop', 'Headphones']
    all_products = ['Laptop', 'Headphones', 'Smartphone', 'Tablet', 'Smartwatch']
    recommendations = recommend_products(user_history, all_products)
    feedback = get_user_feedback(recommendations)

# --- Ensuring Transparency and Fairness ---
# - Recommendations are explained to the user.
# - Products are not filtered by demographic or personal attributes.
# - Users can give feedback to improve fairness and relevance.
# - Regularly review recommendation logic for bias or unfairness.
```

OUTPUT:



TASK4:

PROMPT:

generate logging for a Python web app,Then ensure no sensitive information is logged.

CODE:

```
ta4.py > ...
import logging

# Configure logging
logging.basicConfig(
    filename='app.log',
    level=logging.INFO,
    format='%(asctime)s %(levelname)s %(message)s'
)

# Ethical Logging Practices:
# 1. Do NOT log sensitive information such as passwords, emails, or personal identifier
# 2. Log only necessary information for debugging and monitoring.
# 3. Regularly review logs for accidental sensitive data exposure.
# 4. Restrict access to log files to authorized personnel only.

# Example function in a web application

Tabnine | Edit | Test | Explain | Document
def login_user(username, password):
    # DO NOT log the password or any sensitive data
    # logging.info(f"Login attempt: username={username}, password={password}")  # BAD P
    logging.info(f"Login attempt: username={username}")  # Acceptable if username is no
    # ... authentication logic ...
    return True

# Example usage
if __name__ == "__main__":
    user = input("Enter username: ")
    pwd = input("Enter password: ")
    login_user(user, pwd)
    print("Login attempted. Check app.log for log entry (no sensitive data recorded).")
```

OUTPUT:

```
2025-08-28 16:34:30,814 INFO Login attempt: username=keerthana

PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   CODE REFERENCE LOG

Enter username: keerthana
Enter password: kittu
Login attempted. Check app.log for log entry (no sensitive data recorded).
PS C:\Users\P\Desktop\AIcoding>
```

TASK 5:

PROMPT:

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

OUTPUT&CODE:

```python
# ai_ml_model.py
"""
Copilot-Generated Machine Learning Model with Responsible Usage Guidelines
==========================================================================

This example uses a Logistic Regression classifier.
The documentation explains:
- How to use the model responsibly
- Accuracy limits
- Fairness and explainability considerations
"""

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
import pandas as pd

# Example dataset (tiny and artificial, not suitable for production)
data = {
    "hours_studied": [1, 2, 3, 4, 5, 6, 7, 8],
    "sleep_hours": [8, 7, 6, 5, 4, 6, 7, 5],
    "passed_exam": [0, 0, 0, 1, 1, 1, 1, 1]
}

# Load into DataFrame
df = pd.DataFrame(data)

X = df[["hours_studied", "sleep_hours"]]
y = df["passed_exam"]
```

```python
# Load into DataFrame
df = pd.DataFrame(data)

X = df[["hours_studied", "sleep_hours"]]
y = df["passed_exam"]

# Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# Initialize Logistic Regression model
model = LogisticRegression()

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

"""
Responsible Usage & Documentation
----------------------------------------
```

```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

"""
Responsible Usage & Documentation
-----------------------------------
1. Explainability:
   - Logistic Regression is a linear model: predictions are based on learned coefficients.
   - Coefficients can be inspected using `model.coef_` for interpretability.

2. Accuracy Limits:
   - Accuracy depends heavily on dataset size and quality.
   - Small or biased datasets may lead to misleading performance metrics.
   - This toy dataset is for demonstration only → DO NOT use in real-world exams prediction.

3. Fairness Considerations:
   - Ensure the dataset is diverse and representative to avoid bias.
   - Regularly validate the model on unseen data to prevent unfair outcomes.
   - Provide transparency: communicate that predictions are probabilistic, not absolute truths.

4. Human Oversight:
   - This model should *assist* decision-making, not replace human judgment.
   - Always provide users with explanations of predictions and allow feedback.

 Summary:
This model is a simple demo of how ML works.
Use responsibly with awareness of accuracy limits, dataset bias, and fairness issues.
"""
```

```
Accuracy: 1.0

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2

PS C:\Users\sonti\OneDrive\Documents>
```

EPLANATION&DOCUMENTATION:

Overview

This project provides a basic template for training and evaluating a logistic regression model using scikit-learn.

Responsible Usage Guidelines

Explainability:

This model uses logistic regression, which is relatively interpretable. You can check model.coef_ and model.intercept_ to understand feature importance and decision boundaries.

- **Accuracy Limits:**
  - The model's performance depends on the quality, balance, and representativeness of the provided dataset.
  - Always validate metrics (precision, recall, F1, etc.) using the provided classification report.
  - Do not use the model in critical applications without thorough evaluation.

- *Fairness Considerations:*
  - Ensure your dataset does not contain biases (e.g., demographic or gender imbalance) that might be learned by the model and reflected in its predictions.
  - Regularly audit predictions for disparate impact and retrain with a balanced dataset if necessary.

- **Transparency:**
  - Document the data sources and preprocessing steps.
  - Clearly communicate the model's intended use cases and limitations to stakeholders.

- **Privacy:**
  - Do not include personally identifiable information (PII) in training data without appropriate consent and anonymization.

Limitations

- This model is a simple baseline and may not capture complex relationships in data.
- It may perform poorly on unbalanced datasets or in the presence of outliers.
- The model is not robust to adversarial examples or novel, out-of-distribution data.

Usage

1. Place your CSV data in the project directory as data.csv.
2. Run python simple_ml_model.py to train and evaluate the model.
3. Adapt the code for your own datasets, being mindful of responsible ML practices.