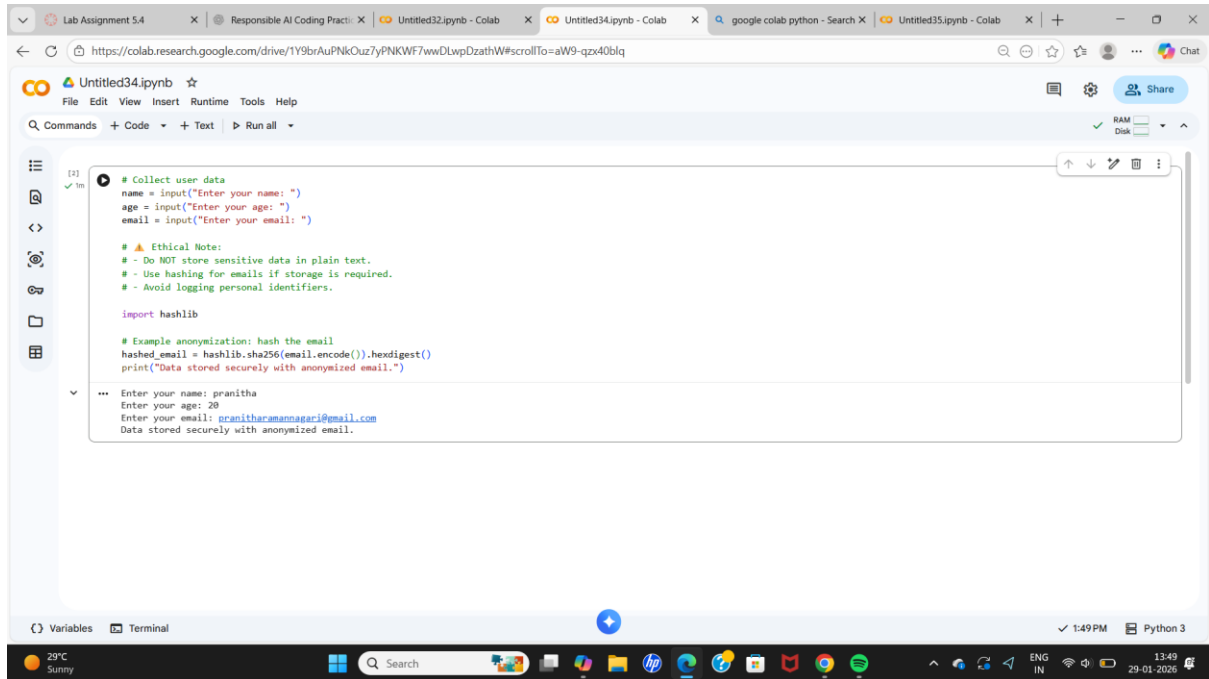


LAB ASSIGNMENT 5.4

NAME: A. Harshita Lakshmi Naga Durga

ID.NO: 2303A52211

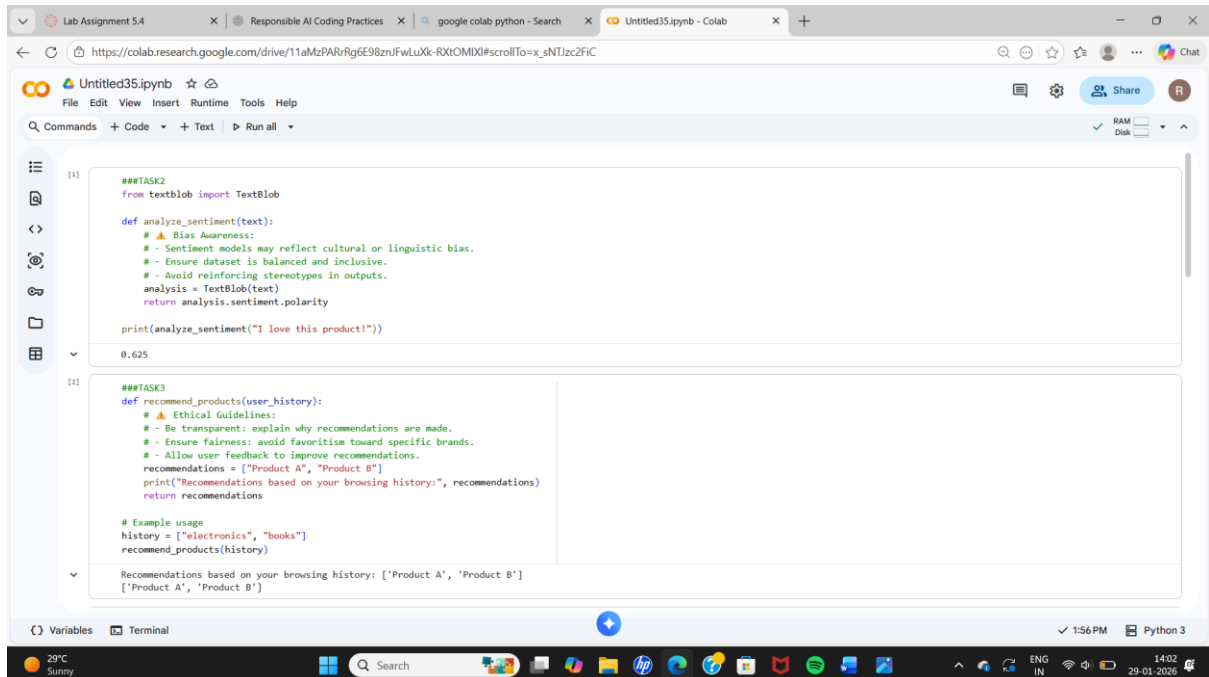
SUBJECT: AI ASST CODING



The screenshot shows a Google Colab notebook interface. The notebook is titled 'Untitled34.ipynb'. The code in the cell is as follows:

```
[2]:  
# Collect user data  
name = input("Enter your name: ")  
age = input("Enter your age: ")  
email = input("Enter your email: ")  
  
# ⚠️ Ethical Note:  
# - Do NOT store sensitive data in plain text.  
# - Use hashing for emails if storage is required.  
# - Avoid logging personal identifiers.  
  
import hashlib  
  
# Example anonymization: hash the email  
hashed_email = hashlib.sha256(email.encode()).hexdigest()  
print("Data stored securely with anonymized email.")  
  
*** Enter your name: pranitha  
Enter your age: 20  
Enter your email: pranitharamanagaci@gmail.com  
Data stored securely with anonymized email.
```

The output of the code is visible below the cell, showing the user input and the final print statement.



The screenshot shows a Google Colab notebook interface. The notebook is titled 'Untitled35.ipynb'. The code in the cell is as follows:

```
[1]:  
###TASK2  
from textblob import TextBlob  
  
def analyze_sentiment(text):  
    # ⚠️ Bias Awareness:  
    # - Sentiment models may reflect cultural or linguistic bias.  
    # - Ensure dataset is balanced and inclusive.  
    # - Avoid reinforcing stereotypes in outputs.  
    analysis = TextBlob(text)  
    return analysis.sentiment.polarity  
  
print(analyze_sentiment("I love this product!"))  
  
0.625  
  
[2]:  
###TASK3  
def recommend_products(user_history):  
    # ⚠️ Ethical Guidelines:  
    # - Be transparent: explain why recommendations are made.  
    # - Ensure fairness: avoid favoritism toward specific brands.  
    # - Allow user feedback to improve recommendations.  
    recommendations = ["Product A", "Product B"]  
    print("Recommendations based on your browsing history:", recommendations)  
    return recommendations  
  
# Example usage  
history = ["electronics", "books"]  
recommend_products(history)  
  
Recommendations based on your browsing history: ['Product A', 'Product B']  
['Product A', 'Product B']
```

The output of the code is visible below the cell, showing the sentiment analysis result and the product recommendations.

If the output is more or equal to 0.5 it is positive statement.

The screenshot shows a Google Colab notebook titled 'Untitled35.ipynb'. The notebook contains two code cells. The first cell, labeled '##TASK4', defines a 'login' function that takes a 'user_id' and returns 'Login successful' after logging an event. The second cell, labeled '##TASK5', imports 'LogisticRegression' and 'load_iris' from 'sklearn', loads the Iris dataset, trains a logistic regression model, and prints the model. Below the code, there is a small output box showing 'LogisticRegression(LogisticRegression(max_iter=200))'. The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help), a toolbar with icons for file operations, and a status bar at the bottom showing '29°C Sunny' and system time.

```
##TASK4
import logging

# Configure logging
logging.basicConfig(filename="app.log", level=logging.INFO)

def login(user_id):
    # ⚠️ Ethical logging:
    # - Do NOT log passwords, emails, or personal identifiers.
    # - Only log non-sensitive events.
    logging.info("User login event recorded (user anonymized).")
    return "Login successful"

# Example run
print(login("user123"))

Login successful

##TASK5
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris

X, y = load_iris(return_X_y=True)
model = LogisticRegression(max_iter=200)
model.fit(X, y)

# ⚠️ Responsible Usage Documentation:
# - This model is trained on the Iris dataset (limited scope).
# - Accuracy may not generalize to other domains.
# - Explainability: coefficients can be inspected for transparency.
# - Fairness: avoid misuse in high-stakes decisions without validation.

LogisticRegression
LogisticRegression(max_iter=200)
```

The screenshot shows a Google Colab notebook titled 'Untitled35.ipynb'. The notebook contains a single code cell, labeled '##TASK6', which imports 'LogisticRegression' and 'load_iris' from 'sklearn', loads the Iris dataset, trains a logistic regression model, and prints the model. Below the code, there is a small output box showing 'Predicted class: [0]'. The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help), a toolbar with icons for file operations, and a status bar at the bottom showing '29°C Sunny' and system time.

```
##TASK6
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris

# Load dataset
X, y = load_iris(return_X_y=True)

# Train a simple logistic regression model
model = LogisticRegression(max_iter=200)
model.fit(X, y)

# ⚠️ Responsible Usage Documentation:
# - This model is trained on the Iris dataset (limited scope).
# - Accuracy may not generalize to other domains.
# - Explainability: coefficients can be inspected for transparency.
# - Fairness: avoid misuse in high-stakes decisions without validation.
# - Always disclose limitations when sharing predictions.

# Example prediction
sample = X[0].reshape(1, -1)
prediction = model.predict(sample)
print("Predicted class:", prediction)

Predicted class: [0]
```

Lab Assignment 5.4 X Responsible AI Coding Practices X google colab python - Search X Untitled35.ipynb - Colab X Pranitha3197/AI-Coding X +

https://github.com/Pranitha3197/AI-Coding/blob/main/lab5.4.ipynb

Files

- main
- Go to file
- LAB 4.4.ipynb
- LAB ASSIGNMENT 4(AI ASST CO...
- LAB ASSIGNMENT 4.4(AI ASST C...
- Lab_Assignment_3.1_AI_2248.pdf
- README.md
- lab 4.1.ipynb
- lab5.4.ipynb

AI-Coding / lab5.4.ipynb

Preview Code Blame 281 lines (281 loc) - 8.4 KB

```
In [1]: 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 collected securely.")

Enter your name: pranitha
Enter your age: 20
Enter your email: pranitharanamagari@gmail.com
User data collected securely.

In [6]: def sentiment_analysis(text):
    """
    Simple sentiment analysis using keyword matching.
    Bias Mitigation:
    - Uses neutral vocabulary
    - Avoids offensive terms
    - Requires balanced datasets for training
    """
    positive_words = ["good", "happy", "excellent", "nice"]
    negative_words = ["bad", "sad", "terrible", "poor"]

    score = 0
    for word in text.lower().split():
        if word in positive_words:
            score += 1
        elif word in negative_words:
```

Lab Assignment 5.4 X Responsible AI Coding Practices X google colab python - Search X Untitled35.ipynb - Colab X Pranitha3197/AI-Coding X +

https://github.com/Pranitha3197/AI-Coding/blob/main/lab5.4.ipynb

Files

- main
- Go to file
- LAB 4.4.ipynb
- LAB ASSIGNMENT 4(AI ASST CO...
- LAB ASSIGNMENT 4.4(AI ASST C...
- Lab_Assignment_3.1_AI_2248.pdf
- README.md
- lab 4.1.ipynb
- lab5.4.ipynb

AI-Coding / lab5.4.ipynb

Preview Code Blame 281 lines (281 loc) - 8.4 KB

```
        score -= 1

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

# ----- OUTPUT TEST -----
sentence = "This product is good and excellent"
result = sentiment_analysis(sentence)
print("Input Text:", sentence)
print("Sentiment:", result)

Input Text: This product is good and excellent
Sentiment: Positive

In [7]: def recommend_products(user_history, product_catalog):
    """
    Ethical Recommendation System
    - Transparency: Explains recommendation logic
    - Fairness: No sponsored prioritization
    """

    recommendations = []

    for product in product_catalog:
        if product["category"] in user_history:
            recommendations.append(product["name"])

    print("Recommendations are based on your browsing history.")
    print("You can change preferences anytime.")

    return recommendations
```

Lab Assignment 5.4 Responsible AI Coding Practices google colab python - Search Untitled35.ipynb - Colab Pranitha3197/AI-Coding

https://github.com/Pranitha3197/AI-Coding/blob/main/lab5.4.ipynb

Files

- main
- Go to file
- LAB 4.4.ipynb
- LAB ASSIGNMENT 4(AI ASST CO...
- LAB ASSIGNMENT 4.4(AI ASST C...
- Lab_Assignment_3.1_AI_2248.pdf
- README.md
- lab 4.1.ipynb
- lab5.4.ipynb

AI-Coding / lab5.4.ipynb

Preview Code Blame 281 lines (281 loc) · 8.4 KB

```
return recommendations

# ---- OUTPUT TEST ----
user_history = ["electronics"]
product_catalog = [
    {"name": "Laptop", "category": "electronics"},
    {"name": "Book", "category": "education"},
    {"name": "Headphones", "category": "electronics"}
]

recommended = recommend_products(user_history, product_catalog)
print("Recommended Products:", recommended)

Recommendations are based on your browsing history.
You can change preferences anytime.
Recommended Products: ['Laptop', 'Headphones']

In [8]: import logging

logging.basicConfig(
    filename="app.log",
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(message)s"
)

def log_user_action(action):
    # Ethical Logging:
    # - No passwords
    # - No emails
    # - No personal identifiers
    logging.info(f"User action recorded: {action}")
    print("Action logged successfully.")

# ---- OUTPUT TEST ----
log_user_action("User viewed product page")

Action logged successfully.
```

29°C Sunny 14:08 29-01-2026

Lab Assignment 5.4 Responsible AI Coding Practices google colab python - Search Untitled35.ipynb - Colab Pranitha3197/AI-Coding

https://github.com/Pranitha3197/AI-Coding/blob/main/lab5.4.ipynb

Files

- main
- Go to file
- LAB 4.4.ipynb
- LAB ASSIGNMENT 4(AI ASST CO...
- LAB ASSIGNMENT 4.4(AI ASST C...
- Lab_Assignment_3.1_AI_2248.pdf
- README.md
- lab 4.1.ipynb
- lab5.4.ipynb

AI-Coding / lab5.4.ipynb

Preview Code Blame 281 lines (281 loc) · 8.4 KB

```
print("Action logged successfully.")

# ---- OUTPUT TEST ----
log_user_action("User viewed product page")

Action logged successfully.

In [9]: from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Sample dataset (dummy data for demonstration)
# Features: [hours_studied, attendance]
X = [[2, 60], [4, 70], [6, 80], [8, 90], [1, 50], [9, 95]]

# Labels: 1 = Pass, 0 = Fail
y = [0, 0, 1, 1, 0, 1]

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42
)

# Train model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict
predictions = model.predict(X_test)

# Accuracy
accuracy = accuracy_score(y_test, predictions)

print("Predicted Results:", predictions)
print("Model Accuracy:", accuracy)

Predicted Results: [0 1]
Model Accuracy: 0.5
```

29°C Sunny 14:08 29-01-2026