

# LAB ASSIGNMENT 5.4

NAME: A. Harshita Lakshmi Naga Durga

ID.NO: 2303A52211

SUBJECT: AI ASST CODING

The screenshot shows a Google Colab notebook titled "Untitled34.ipynb". The code collects user data (name, age, email) and demonstrates how to hash an email using the hashlib library. It includes an ethical note about data storage and hashing. The output shows sample inputs and the resulting hashed email.

```
# 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: aharshita.lakshmi.naga.durga@gmail.com
Data stored securely with anonymized email.
```

The screenshot shows a Google Colab notebook titled "Untitled35.ipynb". The code uses the TextBlob library to analyze sentiment and recommend products based on user history. It includes ethical guidelines for sentiment analysis, such as avoiding bias and ensuring fairness. The output shows the sentiment score and recommended products.

```
##TASK1
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

##TASK2
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']
```

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

Untitled35.ipynb

```
##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)
```

Variables Terminal 1:56 PM Python 3

29°C Sunny 14:02 29-01-2026

Untitled35.ipynb

```
##TASK4
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]
```

Start coding or generate with AI.

Variables Terminal 1:56 PM Python 3

29°C Sunny 14:03 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

AI-Coding / 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

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": email
    }

    return user_data

data = collect_user_data()
print("User data collected securely.")

Enter your name: pranitha
Enter your age: 20
Enter your email: pranitharanamangari@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:
            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)
```

29°C Sunny

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

AI-Coding / 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

In [1]:

```
score = 0
for word in text.lower().split():
    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"

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

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
```

29°C Sunny

The screenshot shows a Google Colab notebook titled "AI-Coding / lab5.4.ipynb". The code implements a recommendation system based on user history and a product catalog. It includes a logging module to track user actions.

```
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 email
    # - 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.
```

The screenshot shows a Google Colab notebook titled "AI-Coding / lab5.4.ipynb". The code demonstrates a logistic regression model for predicting student pass/fail status based on study hours and attendance.

```
# Add header to dataset (dummy data for demonstration)
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
```