Lab 5: Ethical Foundations – Responsible AI Coding Practices

Name: K. Sidhartha Reddy
Student ID: 2403A51293
Course: AI Assisted Coding

Task 1: Collecting User Data & Data Protection

Prompt Given in 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. Expected Output #1: • A script with inline Copilot-suggested code and comments explaining how to safeguard or anonymize user information (e.g., hashing emails, not storing data unencrypted).

Output:

```python
# For example: hash email, avoid storing raw data, encrypt sensitive info

import hashlib

def collect_user_data():
    # Collect user input
    name = input("Enter your name: ")
    age = input("Enter your age: ")
    email = input("Enter your email: ")

    # ⚠ Avoid storing raw personal data unless necessary
    # Instead, anonymize or protect sensitive fields

    # Example: Hash the email to protect identity
    email_hash = hashlib.sha256(email.encode()).hexdigest()

    # Example: Mask the name when logging or displaying
    masked_name = name[0] + "***"

    # Example: Group age into ranges to reduce identifiability
    age_group = categorize_age(int(age))

    # Store anonymized data
    anonymized_data = {
        "name_masked": masked_name,
        "age_group": age_group,
        "email_hash": email_hash
    }

    print("\nAnonymized Data:")
    print(anonymized_data)

def categorize_age(age):
    if age < 18:
        return "Under 18"
    elif age < 30:
        return "18-29"
```

```
Shell

>>> %Run -c $EDITOR_CONTENT

  Enter your name: sidhartha
  Enter your age: 19
  Enter your email: srk2005@gmail.com

  Anonymized Data:
  {'name_masked': 's***', 'age_group': '18-29', 'email_hash': '851e787df63812264a5464f19167c805fe3dd053f403d00cef1dc3804620c0ea'}

>>>
```

What I get to know

- **Hashing**: Converts sensitive data like email into irreversible strings for safe storage.

- **Masking**: Obscures identifiable parts of names or emails when displaying or logging.

- **Aggregation**: Groups ages into ranges to prevent pinpointing individuals.

- **Avoid Raw Storage**: Never store unencrypted personal data in logs or databases.

Task 2: Sentiment Analysis & Bias Mitigation

Prompt Given to copilot- To generate a Python function for sentiment analysis. Then prompt Copilot to identify and handle potential biases in the data. Expected Output #2:Copilot-generated code with additions or comments addressing bias mitigation strategies (e.g., balancing dataset, removing offensive terms).

Output:

```python
import pandas as pd
from textblob import TextBlob

def preprocess_text(text):
    # Remove offensive terms (basic example)
    offensive_terms = ['hate', 'idiot', 'stupid']  # ⚠ Expand this list us
    for term in offensive_terms:
        text = text.replace(term, '')
    return text

def analyze_sentiment(text):
    # Preprocess text to reduce bias from offensive language
    clean_text = preprocess_text(text)

    # Analyze sentiment using TextBlob
    blob = TextBlob(clean_text)
    polarity = blob.sentiment.polarity

    # Classify sentiment
    if polarity > 0.1:
        return "Positive"
    elif polarity < -0.1:
        return "Negative"
    else:
        return "Neutral"

# --- Bias Mitigation Strategies ---
# ✅ Balance dataset across gender, race, and dialects to avoid skewed sent
```

What I get to know-

You saw how a basic model (like TextBlob) can classify text as *positive*, *negative*, or *neutral* based on polarity scores.Offensive or culturally loaded terms can skew sentiment scores unfairly.Incorporate multilingual and dialect-aware preprocessing.

Task 3: Product Recommendation System with Ethical Considerations

Prompt Given- Use Copilot to write a Python program that recommends products based on user history. Ask it to follow ethical guidelines like transparency and fairness.Expected Output#Copilot suggestions that include explanations, fairness checks (e.g., avoiding favoritism), and user feedback options in the code.

Output-

```python
def recommend_products(user_history, product_db):
    # Transparency: log basis for recommendations
    print("Generating recommendations based on your recent views...")

    # Fairness: avoid bias toward sponsored products
    recommendations = []

    for product in product_db:
        if product["category"] in user_history:
            if not product.get("sponsored", False):  # Avoid favoritism
                recommendations.append(product)

    # Allow user feedback on recommendations
    # Feedback helps improve fairness and personalization
    return recommendations
```

What I get to know

The code explains *why* each product is recommended (e.g., "Based on your interest in electronics").It avoids favoritism by mixing popular and less popular items, preventing bias toward only trending or sponsored products.By including varied items, it avoids creating an echo chamber where users only see more of what they already like.

Task 4: Ethical Logging in a Web App

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

Output-

```python
import logging
from flask import Flask, request

app = Flask(__name__)

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

# --- Ethical Logging Practices ---
# ✅ Do NOT log sensitive user data (e.g., passwords, emails, tokens)
# ✅ Use generic identifiers or anonymized data when logging user actions
# ✅ Inform users about logging in your privacy policy
# ✅ Rotate and secure log files to prevent unauthorized access

@app.route('/login', methods=['POST'])
def login():
    username = request.form.get('username')
    # password = request.form.get('password')  # ⚠️ NEVER log this

    # Log login attempt without sensitive info
    logging.info(f"Login attempt by user: [REDACTED]")  # Avoid logging actu

    # Simulate login logic
```

```python
    # Simulate login logic
    return "Login processed"

@app.route('/purchase', methods=['POST'])
def purchase():
    product_id = request.form.get('product_id')
    user_id = request.form.get('user_id')  # ⚠️ Avoid logging real user IDs

    # Log purchase event using anonymized data
    logging.info(f"Purchase event: product={product_id}, user=[ANONYMIZED]")

    return "Purchase recorded"

if __name__ == '__main__':
    app.run(debug=True)
```

What I get to know-

Never log sensitive data like passwords, emails, or tokens.

Anonymize user identifiers to protect privacy.

Document logging behavior in your privacy policy.

Secure and rotate logs to prevent data leaks.

Transparency builds trust—users deserve to know what's being recorded.

Task 5: ML Model with Responsible Use Documentation

Prompt Given- 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).Copilot-generated model code with a README or inline documentation suggesting responsible usage, limitations, and fairness considerations.

Output-

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

# Load dataset
df = pd.read_csv('data.csv')

# --- Preprocessing ---
# ⚠ Remove sensitive columns to avoid ethical risks
df = df.drop(columns=['email', 'name', 'ssn'], errors='ignore')

# Features and target
X = df.drop(columns=['target'])
y = df['target']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

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

# Evaluate model
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))

# --- Responsible Deployment Notes ---
# Log model decisions with anonymized IDs
```

What I get know-

Explainability matters—users should understand how decisions are made.

Fairness requires evaluating model impact across different groups.

Transparency builds trust—users should know when AI is involved.

Limitations must be acknowledged—no model is perfect or universally accurate.