2403A51266

Ai assisted coding

Question:

Ethical Foundations – Responsible AI Coding Practices

Lab Objectives:

• To explore the ethical risks associated with AI-generated code.

• To recognize issues related to security, bias, transparency, and copyright.

• To reflect on the responsibilities of developers when using AI tools in software

development.

• To promote awareness of best practices for responsible and ethical AI coding.

Lab Outcomes (LOs):

After completing this lab, students will be able to:

• Identify and avoid insecure coding patterns generated by AI tools.

• Detect and analyze potential bias or discriminatory logic in AI-generated outputs.

• Evaluate originality and licensing concerns in reused AI-generated code.

• Understand the importance of explainability and transparency in AI-assisted

programming.

• Reflect on accountability and the human role in ethical AI coding practices..

Task Description #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.

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).

Task Description #2:

- Ask 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).

Task Description #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.

Expected Output #3:

- Copilot suggestions that include explanations, fairness checks (e.g., avoiding favoritism), and

user feedback options in the code.

Task Description #4:

- Prompt Copilot to generate logging functionality in a Python web application. Then, ask it to

ensure the logs do not record sensitive information.

Expected Output #4:

- Logging code that avoids saving personal identifiers (e.g., passwords, emails), and includes

comments about ethical logging practices.

Task Description #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).

Expected Output #5:

- Copilot-generated model code with a README or inline documentation suggesting

responsible usage, limitations, and fairness considerations.

Note: Report should be submitted a word document for all tasks in a single document with

prompts, comments & code explanation, and output and if required, screenshots

Evaluation Criteria:

Criteria Max Marks

How to anonymize or protect this data 0.5

Identify and handle potential biases in the data. 0.5

Follow ethical guidelines like transparency and

fairness. 0.5

logs do not record sensitive information. 0.5

How to use the model responsibly 0.5

Total 2.5 Marks

CODE:

Task1:Data privacy and Anonymization:

```
import hashlib

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

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

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

    print("\nUser data processed for storage:")
    print(user_record)
    return user_record
```

```
collect_user_data()
```

OUTPUT:

```
Enter your name: John Doe
Enter your age: 30
Enter your email: john.doe@example.com

User data processed for storage:
{'name': 'John Doe', 'age': '30', 'email_hash': '7655a1202868297b134d17f7d98344158913922f300c196f7e4362145e12a4d3'}
```

Task 2: Bias in Sentiment Analysis:

```
from textblob import TextBlob

def analyze_sentiment_with_bias_awareness(text):
    analysis = TextBlob(text)
    return analysis.sentiment.polarity

text1 = "This review is amazing, so helpful!"
text2 = "This is a stupid rule."

print(f"'{text1}' -> Polarity: {analyze_sentiment_with_bias_awareness(text1)}")
print(f"'{text2}' -> Polarity: {analyze_sentiment_with_bias_awareness(text2)}")
```

OUTPUT:

```
'This review is amazing, so helpful!' -> Polarity:
0.6000000000000001 'This is a stupid rule.' -> Polarity: -0.9
```

Task3: Ethical Recommender System:

```
def recommend_products_ethically(user_history):
    recommendations = []

    if "book" in user_history:
        recommendations.append("A new book from a different genre")
    if "headphones" in user_history:
        recommendations.append("A complementary product, like a stand or case")

    if len(recommendations) == 0:
        recommendations.append("A popular item from a broad category")

    return recommendations

user1_history = ["book", "book", "thriller novel"]
print("Recommendations for user 1:")
for rec in recommend_products_ethically(user1_history):
    print(f"- {rec} (Explanation: Because you viewed books, we recommend a diverse genre.)")
```

OUTPUT:

Recommendations for user 1: - A new book from a different genre (Explanation: Because you viewed books, we recommend a diverse genre.)

Task4:Ethical logging:

```
import logging


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


def process_user_login(username, password):
    logging.info(f"User login attempt for username: {username[:3]}***{username[-3:]}")


    if username == "admin" and password == "secret":
        logging.info("Login successful.")
        return True
    else:
        logging.warning("Login failed due to incorrect credentials.")
        return False


process_user_login("john.doe@example.com", "my-secret-password-123")
```

OUTPUT:

```
<Timestamp> - INFO - User login attempt for username: joh***com
<Timestamp> - WARNING - Login failed due to incorrect credentials.
```

Task 5: Responsible Model Documentation:

Python

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import numpy as np

X = np.random.rand(100, 5)
y = np.random.randint(0, 2, 100)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)

predictions = model.predict(X_test)
print("Example predictions:", predictions[:5])
```

Output:

```
Example predictions: [0 1 0 1 0]
```