Ai coding assignment test-1

Q1. Zero-shot Prompting in Healthcare [5M]
Scenario: A doctor uses an AI tool to quickly triage patient symptoms into “Mild,” “Moderate,” or “Severe.

• Task 1: Write a zero-shot prompt that classifies the severity of symptoms without giving any examples

Code:-

```
def classify_symptom_severity(symptom_description):
    # Convert to lowercase for uniform matching
    description = symptom_description.lower()

    # Define keyword sets for each severity level
    severe_keywords = {"chest pain", "shortness of breath", "loss of consciousness", "seizure", "bleeding", "high fever", "radiating pain", "dizziness"}
    moderate_keywords = {"persistent cough", "vomiting", "diarrhea", "fatigue", "moderate pain", "swelling", "headache"}
    mild_keywords = {"sore throat", "runny nose", "mild pain", "sneezing", "itching", "slight discomfort"}

    # Check for matches
    if any(keyword in description for keyword in severe_keywords):
        return "Severe"
    elif any(keyword in description for keyword in moderate_keywords):
        return "Moderate"
    elif any(keyword in description for keyword in mild_keywords):
        return "Mild"
    else:
        return "Unknown"

# Example usage
symptom_input = input("Describe your symptoms: ")
severity = classify_symptom_severity(symptom_input)
print("Severity:", severity)
```

OUTPUT:-

Describe your symptoms: chest pain

Severity: severe.

Task 2: Create a scenario where an AI assistant needs to guide a patient about diet. Write two prompts: one without context and one with detailed context (e.g., age, health condition, dietary restrictions).

CODE:-

```
def generate_diet_plan():
    diet = {
        "Breakfast": "Oatmeal with banana and almonds",
        "Lunch": "Grilled chicken salad with olive oil dressing",
        "Snack": "Greek yogurt with honey",
        "Dinner": "Steamed vegetables with quinoa and tofu",
        "Tips": [
            "Drink 2-3 liters of water daily",
            "Avoid processed foods and excess sugar",
            "Eat smaller meals more frequently"
        ]
    }
    return diet
```

#FOR DETAILED CONTEXT:-

```python
# prompt with detailed health context
def generate_personalized_diet(age, conditions, restrictions):
    if age > 60 and "type 2 diabetes" in conditions:
        diet = {
            "Breakfast": "Steel-cut oats with chia seeds and berries",
            "Lunch": "Grilled salmon with sautéed spinach and brown rice",
            "Snack": "Unsweetened almond yogurt with flaxseeds",
            "Dinner": "Lentil soup with roasted vegetables and millet",
            "Tips": [
                "Limit sodium and potassium due to kidney concerns",
                "Avoid dairy and high-glycemic foods",
                "Monitor blood sugar levels post-meal",
                "Stay hydrated with herbal teas and water"
            ]
        }
        return diet
    else:
        return "Please provide more specific health details."

# Patient context
age = 62
conditions = ["type 2 diabetes", "mild kidney dysfunction"]
restrictions = ["lactose intolerance"]
```

OUTPUT:-

```
PS D:\ai coding> & C:/Users/priya/AppData/Local/Microsoft/WindowsApps/python3.11.exe "d:/ai coding/AI CODING/ai test 1.2.py"
{'Breakfast': 'Oatmeal with banana and almonds', 'Lunch': 'Grilled chicken salad with olive oil dressing', 'Snack': 'Greek yogurt with honey', 'Dinner': 'Steamed vegetables with quinoa and tofu', 'Tips
': ['Drink 2-3 liters of water daily', 'Avoid processed foods and excess sugar', 'Eat smaller meals more frequently']}
{'Breakfast': 'Steel-cut oats with chia seeds and berries', 'Lunch': 'Grilled salmon with sautéed spinach and brown rice', 'Snack': 'Unsweetened almond yogurt with flaxseeds', 'Dinner': 'Lentil soup wi
h roasted vegetables and millet', 'Tips': ['Limit sodium and potassium due to kidney concerns', 'Avoid dairy and high-glycemic foods', 'Monitor blood sugar levels post-meal', 'Stay hydrated with herba
teas and water']}
PS D:\ai coding>
```

Q2. One-shot vs Few-shot for Customer Support [5M]
Scenario: An e-commerce company uses AI to classify support emails into “Refund,” “Order Status,”or “Technical Issue.”
• Task 1: Write:
o A one-shot prompt with 1 example of classification.
o A few-shot prompt with 3–4 examples..

CODE:-

#FOR ONE SHOT

```python
def classify_email(email):
    example = {
        "email": "I received the wrong item and would like my money back.",
        "category": "Refund"
    }

    if "arrive" in email or "shipping" in email or "delivery" in email:
        return "Order Status"
    elif "money back" in email or "refund" in email or "return" in email:
        return "Refund"
    elif "error" in email or "issue" in email or "problem" in email:
        return "Technical Issue"
    else:
        return "Unknown"

# Test email
email_to_classify = "Can you tell me when my package will arrive?"
print(classify_email(email_to_classify))
```

#FOR FEW SHOT:-

```python
def classify_email_few_shot(email):
    examples = [
        {"email": "I received the wrong item and would like my money back.", "category": "Refund"},
        {"email": "Can you tell me when my package will arrive?", "category": "Order Status"},
        {"email": "The website crashes every time I try to check out.", "category": "Technical Issue"},
        {"email": "I want to return the shoes I bought last week.", "category": "Refund"}
    ]

    if "tracking" in email or "where my order" in email or "delivery" in email:
        return "Order Status"
    elif "money back" in email or "refund" in email or "return" in email:
        return "Refund"
    elif "crash" in email or "error" in email or "problem" in email:
        return "Technical Issue"
    else:
        return "Unknown"

# Test email
email_to_classify = "My tracking number isn't working and I don't know where my order is."
print(classify_email_few_shot(email_to_classify))
```

OUTPUT:-

```
PS D:\ai coding> & C:/Users/priya/AppData/Local/Microsoft/WindowsApps/python3.11.exe "d:/ai coding/AI CODING/test task 2.py"
Order Status
Order Status
PS D:\ai coding>
```

Task 2: Use the same incoming email text for both prompts. Compare how the outputs differ and explain why.

```
**Task 2: Comparison of outputs**

Incoming Email Text (used for both prompts): "The screen on my new phone flickers constantly. What should I do?"

*Expected Output (One-shot):* Potentially misclassified if the one-shot example is not relevant. Could be classified as "Order Status" if the example used in the one-shot prompt was about order issues.

*Expected Output (Few-shot):* More likely to be correctly classified as "Technical Issue" because the multiple examples provide a better understanding of the different categories.

*Explanation:*

The few-shot prompt provides the AI with more context and examples, allowing it to better understand the nuances of each category. The one-shot prompt relies heavily on the single example provided, which may not be representative of all cases. Therefore, the few-shot prompt is generally more accurate and reliable for email classification.
```