

2303a51924

## Lab assignment-4.4

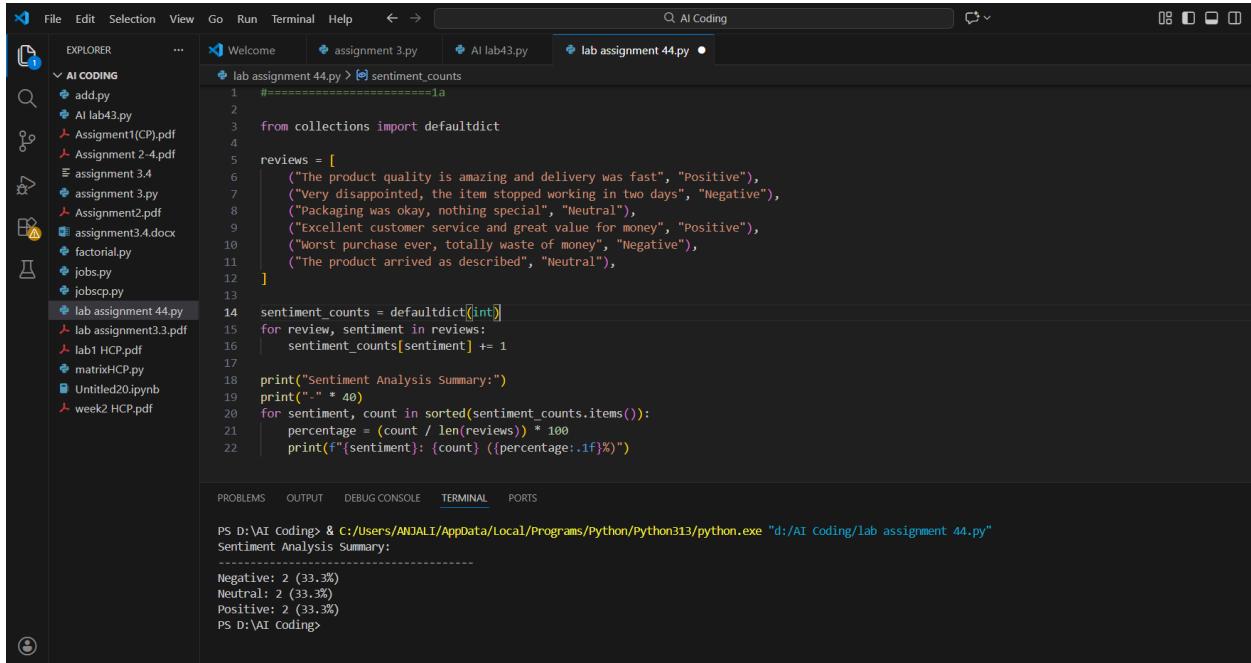
### 1. Sentiment Classification for Customer Reviews

Scenario:

An e-commerce platform wants to analyze customer reviews and classify them into Positive, Negative, or Neutral sentiments using prompt engineering

Tasks:

a) Prepare 6 short customer reviews mapped to sentiment labels.



The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows files in the "AI CODING" folder, including "add.py", "Al lab43.py", "Assignment1(CP).pdf", "Assignment 2-4.pdf", "assignment 3.4", "assignment 3.4.pdf", "Assignment3.pdf", "Assignment2.pdf", "assignment3.4.docx", "factorial.py", "jobs.py", "jobscp.py", "lab assignment 44.py", "lab assignment3.3.pdf", "lab1 HCP.pdf", "matrixHCP.py", "Untitled20.ipynb", and "week2 HCP.pdf".
- Code Editor:** Displays the content of "lab assignment 44.py".

```
#=====
1
2
3 from collections import defaultdict
4
5 reviews = [
6     ("The product quality is amazing and delivery was fast", "Positive"),
7     ("Very disappointed, the item stopped working in two days", "Negative"),
8     ("Packaging was okay, nothing special", "Neutral"),
9     ("Excellent customer service and great value for money", "Positive"),
10    ("Worst purchase ever, totally waste of money", "Negative"),
11    ("The product arrived as described", "Neutral"),
12 ]
13
14 sentiment_counts = defaultdict(int)
15 for review, sentiment in reviews:
16     sentiment_counts[sentiment] += 1
17
18 print("Sentiment Analysis Summary:")
19 print("." * 40)
20 for sentiment, count in sorted(sentiment_counts.items()):
21     percentage = (count / len(reviews)) * 100
22     print(f"{'(sentiment)':>10} {count} ({percentage:.1f}%)")
```
- Terminal:** Shows the command "PS D:\AI Coding> & C:/Users/ANDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"" and the output:

```
Sentiment Analysis Summary:
-----
Negative: 2 (33.3%)
Neutral: 2 (33.3%)
Positive: 2 (33.3%)
PS D:\AI Coding>
```

b) Design a Zero-shot prompt to classify sentiment.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The left sidebar contains a file tree with various files including 'lab assignment 44.py', 'AI lab43.py', and several PDF documents. The main editor area displays the following Python script:

```
1 #  
2  
3 ======  
4 review = "The product quality is amazing and delivery was fast"  
5  
6 # Simple sentiment classification based on keywords  
7 positive_words = ["amazing", "great", "excellent", "good", "fast", "love", "best"]  
8 negative_words = ["bad", "terrible", "poor", "slow", "worst", "hate", "awful"]  
9  
10 review_lower = review.lower()  
11 pos_count = sum(1 for word in positive_words if word in review_lower)  
12 neg_count = sum(1 for word in negative_words if word in review_lower)  
13  
14 if pos_count > neg_count:  
15     sentiment = "Positive"  
16 elif neg_count > pos_count:  
17     sentiment = "Negative"  
18 else:  
19     sentiment = "Neutral"  
20  
21 print(f"Review: \"{review}\"")  
22 print(f"Sentiment: {sentiment}")
```

The bottom of the screen shows the terminal output:

```
PS D:\AI Coding> & C:/Users/ANDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"  
Sentiment Analysis Summary:  
  
Negative: 2 (33.3%)  
Neutral: 2 (33.3%)  
Positive: 2 (33.3%)  
PS D:\AI Coding> & C:/Users/ANDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"  
Review: "The product quality is amazing and delivery was fast"  
Sentiment: Positive  
PS D:\AI Coding>
```

**1c) Design a One-shot prompt with one labeled example.**

The screenshot shows a Windows desktop environment with the Visual Studio Code (VS Code) application open. The title bar reads "Q AI Coding". The left sidebar (Explorer) lists files including "AI CODING", "assignment 3.py", "lab assignment 44.py", and various PDFs and DOCX files. The main editor area contains Python code for sentiment analysis:

```
1 # Sentiment Classification
2 review = "Excellent customer service and great value for money"
3
4 # Simple sentiment classification
5 positive_words = ['excellent', 'great', 'good', 'amazing', 'wonderful', 'best', 'love']
6 negative_words = ['hate', 'bad', 'poor', 'awful', 'terrible', 'worst', 'useless']
7
8 review_lower = review.lower()
9 positive_count = sum(1 for word in positive_words if word in review_lower)
10 negative_count = sum(1 for word in negative_words if word in review_lower)
11
12 if positive_count > negative_count:
13     sentiment = "Positive"
14 elif negative_count > positive_count:
15     sentiment = "Negative"
16 else:
17     sentiment = "Neutral"
18
19 print(f"Review: {review}")
20 print(f"Sentiment: {sentiment}")
```

The terminal tab at the bottom shows the execution of the script and its output:

```
Sentiment Analysis Summary:
-----
Negative: 2 (33.3%)
Neutral: 2 (33.3%)
Positive: 2 (33.3%)
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/python313/python.exe "d:/AI Coding/lab assignment 44.py"
Review: "the product quality is amazing and delivery was fast"
Sentiment: Positive
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/python313/python.exe "d:/AI Coding/lab assignment 44.py"
Review: "Excellent customer service and great value for money"
Sentiment: Positive
PS D:\AI Coding>
```

**1d) Design a Few-shot prompt with 3–5 labeled examples.**

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows files like AI lab43.py, assignment1(CP).pdf, Assignment 2-4.pdf, assignment 3.4, assignment3.4.docx, factorial.py, jobs.py, jobspp.py, lab assignment 44.py, lab assignment3.3.pdf, lab1 HCP.pdf, matrixHCP.py, Untitled20.ipynb, and week2 HCP.pdf.
- Code Editor:** Displays Python code for sentiment analysis. The code defines a function `classify_sentiment` that takes a review as input and returns "Positive", "Negative", or "Neutral". It uses lists of positive and negative words and counts them to determine the sentiment.
- Terminal:** Shows command-line output from running the script. It includes test cases with reviews like "The product quality is amazing and delivery was fast" and "Worst purchase ever, totally waste of money". The output shows the script correctly identifying the first as Positive and the second as Negative.
- Output Panel:** Shows standard output and errors from the terminal.
- Ports Panel:** Shows available ports for communication.

**1e) Compare the outputs and discuss accuracy differences.**

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows files including lab assignment ... (selected), lab assignment3.3.pdf, lab1 HCP.pdf, matrixHCP.py, Untitled20.ipynb, and week2 HCP.pdf.
- Code Editor:** Displays Python code comparing different classification techniques. It defines a dictionary `data` with keys: 'Technique' (containing 'Zero-shot', 'One-shot', 'Few-shot'), 'Accuracy' (containing 'Medium', 'Better', 'Best'), and 'Reason' (containing 'Relies only on model knowledge', 'Learns from one example', and 'Clear pattern learning from multiple examples').
- Terminal:** Shows command-line output from running the script. It includes a review "Excellent customer service and great value for money" which is identified as Positive, and a review "Worst purchase ever, totally waste of money" which is identified as Negative. The output also shows a `ModuleNotFoundError` for 'pandas'.
- Output Panel:** Shows standard output and errors from the terminal.
- Ports Panel:** Shows available ports for communication.

## 2. Email Priority Classification

### Scenario:

A company wants to automatically prioritize incoming emails into High Priority, Medium Priority, or Low Priority

**2a) Create 6 sample email messages with priority labels.**

The screenshot shows the Visual Studio Code (VS Code) interface. The Explorer sidebar on the left lists files and folders, including several Python files like 'add.py', 'AI lab43.py', and 'lab assignment 44.py'. The 'TERMINAL' tab is selected at the bottom, displaying the output of a Python script. The script reads a list of emails from a list comprehension and prints them with their priority levels.

```
#=====2a
emails = [
    {"email": "Server is down and business operations stopped", "priority": "High"},
    {"email": "Client meeting scheduled for tomorrow", "priority": "High"},
    {"email": "Please review the attached report when free", "priority": "Medium"},
    {"email": "Need update on project status", "priority": "Medium"},
    {"email": "Team lunch invitation", "priority": "Low"},
    {"email": "Newsletter subscription confirmation", "priority": "Low"},

]
for email in emails:
    print(f"Email: {email['email']} | Priority: {email['priority']}")
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Traceback (most recent call last):
  File "d:\AI Coding\lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'
PS D:\AI Coding> & c:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Email: Server is down and business operations stopped | Priority: High
Email: Client meeting scheduled for tomorrow | Priority: High
Email: Please review the attached report when free | Priority: Medium
Email: Need update on project status | Priority: Medium
Email: Team lunch invitation | Priority: Low
Email: Newsletter subscription confirmation | Priority: Low
PS D:\AI Coding>
```

## 2b) Perform intent classification using Zero-shot prompting

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows various files including `assignment 3.py`, `AI lab43.py`, and `lab assignment 44.py`.
- Code Editor:** Displays the code for `lab assignment 44.py`. The last line of code is highlighted in green: `priority = "High"`.
- Terminal:** Shows the command line output:

```
File "d:\VAI Coding\lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'
PS D:\VAI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Email: Server is down and business operations stopped | Priority: High
Email: Client meeting scheduled for tomorrow | Priority: High
Email: Please review the attached report when free | Priority: Medium
Email: Need update on project status | Priority: Medium
Email: Team lunch invitation | Priority: Low
Email: Newsletter subscription confirmation | Priority: Low
PS D:\VAI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\VAI Coding>
```

## 2c) Perform classification using One-shot prompting

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows various files including `assignment 3.py`, `AI lab43.py`, and `lab assignment 44.py`.
- Code Editor:** Displays the code for `lab assignment 44.py`. The last line of code is highlighted in green: `priority = "Normal" # Set the priority level`.
- Terminal:** Shows the command line output:

```
ModuleNotFoundError: No module named 'pandas'
PS D:\VAI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Email: Server is down and business operations stopped | Priority: High
Email: Client meeting scheduled for tomorrow | Priority: High
Email: Please review the attached report when free | Priority: Medium
Email: Need update on project status | Priority: Medium
Email: Team lunch invitation | Priority: Low
Email: Newsletter subscription confirmation | Priority: Low
PS D:\VAI Coding>
```

## 2d) Perform classification using Few-shot prompting.

```

File Edit Selection View Go Run Terminal Help ← → ⌘ AI Coding
EXPLORER Welcome assignment 3.py AI lab43.py lab assignment 44.py ×
AI CODING
add.py
AI lab43.py
Assignment1(CP).pdf
Assignment 2-4.pdf
assignment 3.4
assignment 3.py
Assignment2.pdf
assignment3.4.docx
factorial.py
jobs.py
jobsqc.py
lab assignment 44.py
lab assignment3.3.pdf
lab1 HCP.pdf
matrixHCP.py
Untitled20.ipynb
week2 HCP.pdf

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Email: Client meeting scheduled for tomorrow | Priority: High
Email: Please review the attached report when free | Priority: Medium
Email: Need update on project status | Priority: Medium
Email: Team lunch invitation | Priority: Low
Email: Newsletter subscription confirmation | Priority: Low
PS D:\AI Coding & C:/Users/ANALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding & C:/Users/ANALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding & C:/Users/ANALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding & C:/Users/ANALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Email: "Client meeting scheduled for tomorrow"
Priority: Low
PS D:\AI Coding>

```

The screenshot shows a code editor with an open file named 'lab assignment 44.py'. The code defines a function 'determine\_priority' that takes an 'email\_subject' string and returns a priority level ('High', 'Medium', 'Low', or 'Default') based on a set of keywords. Below the code, the terminal window displays several test emails and their corresponding priorities, along with the command used to run the script.

## 2e) Evaluate which technique produces the most reliable results and why.

```

File Edit Selection View Go Run Terminal Help ← → ⌘ AI Coding
EXPLORER Welcome assignment 3.py AI lab43.py lab assignment 44.py ×
AI CODING
add.py
AI lab43.py
Assignment1(CP).pdf
Assignment 2-4.pdf
assignment 3.4
assignment 3.py
Assignment2.pdf
assignment3.4.docx
factorial.py
jobs.py
jobsqc.py
lab assignment 44.py
lab assignment3.3.pdf
lab1 HCP.pdf
matrixHCP.py
Untitled20.ipynb
week2 HCP.pdf

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Email: Team lunch invitation | Priority: Low
Email: Client meeting scheduled for tomorrow | Priority: Low
Email: "Client meeting scheduled for tomorrow"
Priority: Low
PS D:\AI Coding & C:/Users/ANALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding & C:/Users/ANALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding & C:/Users/ANALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding & C:/Users/ANALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Email: Team lunch invitation | Priority: Low
Email: Client meeting scheduled for tomorrow | Priority: Low
Email: "Client meeting scheduled for tomorrow"
Priority: Low
PS D:\AI Coding>

```

The screenshot shows a code editor with an open file named 'lab assignment 44.py'. The code implements three classification techniques: 'zero\_shot\_classification', 'one\_shot\_classification', and 'few\_shot\_classification'. It also includes a function 'evaluate\_techniques' that compares the accuracy of these techniques against ground truth data. The terminal window shows the execution of the script and the resulting accuracy values.

## 3. Student Query Routing System

**Scenario:**

**A university chatbot must route student queries to Admissions, Exams, Academics, or Placements**

**3a) . Create 6 sample student queries mapped to departments.**

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows files in the "AI CODING" folder, including "add.py", "AI lab43.py", "lab assignment 44.py", and "lab assignment 44.py" (the active file).
- Code Editor:** Displays the following Python code:

```
student_queries = {
    "How do I register for courses?": "Registrar",
    "What is my current GPA?": "Admissions",
    "I need to pay my tuition": "Finance",
    "Can I get a transcript?": "Registrar",
    "I'm having trouble with my financial aid": "Finance",
    "How do I declare a major?": "Admissions"
}

for query, department in student_queries.items():
    print(f"Query: {query}")
    print(f"Department: {department}\n")
```
- Terminal:** Shows the output of the code execution:

```
Query: How do I register for courses?
Department: Registrar

Query: What is my current GPA?
Department: Admissions

Query: I need to pay my tuition
Department: Finance

Query: Can I get a transcript?
Department: Registrar
```
- Bottom Navigation:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS.

**3b) Implement Zero-shot intent classification using an LLM.**

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows a folder named "AI CODING" containing files like add.py, AI lab43.py, Assignment1(CP).pdf, Assignment 2-4.pdf, assignment 3.4, assignment 3.py, Assignment2.pdf, assignment3.4.docx, factorial.py, jobs.py, jobsccp.py, lab assignment ..., lab assignment3.3.pdf, lab1 HCP.pdf, matrixHCP.py, Untitled20.ipynb, and week2 HCP.pdf.
- Code Editor (Center):** Displays a Python script with the following code:

```
1  #
2  if department == "Registrar":
3      classification = "Academics"
4  elif department == "Admissions":
5      classification = "Admissions"
6  elif department == "Finance":
7      classification = "Placements"
8  else:
9      classification = "Exams"
10 print(f"Classification: {classification}\n")
```
- Terminal (Bottom):** Shows the output of running the script with two different queries:

```
Query: I'm having trouble with my financial aid
Department: Finance
Classification: Placements

Query: How do I declare a major?
Department: Admissions
Classification: Admissions
```

**3c) Improve results using One-shot prompting.**

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder named "AI CODING" containing files like add.py, AI lab43.py, Assignment1(CP).pdf, Assignment 3.4.pdf, assignment 3.4, assignment3.4.docx, factorial.py, jobs.py, jobsccp.py, lab assignment ..., lab assignment3.3.pdf, lab1 HCP.pdf, matrixHCP.py, Untitled20.ipynb, and week2 HCP.pdf.
- Code Editor:** Displays a Python script named "lab assignment 44.py". The code defines a function `classify\_query` that takes a query string and returns the department name based on keyword matches. It includes test cases at the bottom.
- Terminal:** Shows the command line output of running the script with sample queries. The output is:

```
PS D:\AI Coding & C:\Users\ANJALI\AppData\Local\Programs\Python\Python313\python.exe "d:/AI Coding/lab assignment 44.py"
Query: "When will results be announced?"
Department: Exams

Query: "Explain syllabus for Data Structures"
Department: Academics

Query: "How do I apply for admission?"
Classification: Admissions
```
- Bottom Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS.

**3d) Further refine results using Few-shot prompting.**

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar labeled "AI Coding". The left sidebar has sections for Explorer, Search, Files, and Recent Files. The main workspace contains a terminal window at the bottom displaying command-line output and a code editor window above it.

**Terminal Output:**

```
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Traceback (most recent call last):
  File "d:\AI Coding\lab assignment 44.py", line 1, in <module>
    import openai
ModuleNotFoundError: No module named 'openai'
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Admissions
Exams
Placements
Admissions
```

**Code Editor Content:**

```
1 def classify_query(query):
2     """
3         Classifies a query to the appropriate department.
4     """
5     Args:
6         query (str): The user's query
7
8     Returns:
9         str: The department name
10    """
11
12    query_lower = query.lower()
13
14    # Define keywords for each department
15    departments = {
16        "Admissions": ["admission", "deadline", "apply", "enrollment", "registration"],
17        "Exams": ["exam", "fee", "payment", "test", "marks", "results"],
18        "Placements": ["job", "placement", "campus", "opportunity", "recruit", "interview"]
19    }
20
21    # Check which department matches the query
22    for dept, keywords in departments.items():
23        if any(keyword in query_lower for keyword in keywords):
24            return dept
25
26    return "Unknown" # Default if no match found
27
28 # Test cases
29 print(classify_query("Admission deadline details")) # Admissions
30 print(classify_query("Exam fee payment date")) # Exams
31 print(classify_query("Job opportunities through campus")) # Placements
32 print(classify_query("How to apply for campus placements?")) # Placements
```

**Bottom Navigation Bar:**

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

### 3e) Analyze how contextual examples affect classification accuracy.

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows files in the "AI CODING" folder, including "add.py", "AI lab43.py", "Assignment1(CP).pdf", "Assignment 2-4.pdf", "assignment 3.4", "assignment 3.py", "Assignment2.pdf", "assignment3.4.docx", "factorial.py", "jobs.py", "jobscpp.py", "lab assignment ...", "lab assignment3.3.pdf", "lab1 HCP.pdf", "matrixHCP.py", "Untitled20.ipynb", and "week2 HCP.pdf".
- Code Editor:** Displays a Python script named "lab assignment 44.py". The code defines a class "ClassificationAnalyzer" with methods "zero\_shot\_prompt" and "one\_shot\_prompt".

```
from collections import defaultdict
import json

class ClassificationAnalyzer:
    def __init__(self):
        self.results = defaultdict(list)

    def zero_shot_prompt(self, text, categories):
        """
        Zero-shot: No examples provided
        """
        prompt = f"""Classify the following text into one of these categories: {', '.join(categories)}\nText: {text}\nCategory:"""

        return prompt

    def one_shot_prompt(self, text, categories, example_text, example_category):
        """
        One-shot: Single example provided
        """
        prompt = f"""Classify text into categories: {', '.join(categories)}\nExample:\nText: {example_text}\nCategory: {example_category}"""

        return prompt
```
- Terminal:** Shows command-line output related to the analysis of few-shot learning:

```
Ambiguity: Reduced - one example clarifies intent
Consistency: Improved - example sets pattern
Pros: Minimal overhead, Some context
Cons: Limited learning from one example

FEW_SHOT:
Accuracy: Higher - multiple references provided
Ambiguity: Significantly reduced - pattern clear
Consistency: High - multiple examples establish standard
Pros: Best accuracy, Clear patterns, Reduced errors
Cons: Requires manual examples, Prompt size
```
- Status Bar:** Shows the path "PS D:\AI Coding".

### 4) Chatbot Question Type Detection

#### Scenario:

A chatbot must identify whether a user query is Informational, Transactional, Complaint, or Feedback.

#### 4a) Prepare 6 chatbot queries mapped to question types.

```

File Edit Selection View Go Run Terminal Help < > Q AI Coding
EXPLORER ... Welcome assignment 3.py AI lab43.py lab assignment 44.py X
AI CODING
+ add.py
+ AI lab43.py
- Assignment1(CP).pdf
- Assignment 2-4.pdf
- Assignment 3-4.py
- Assignment2.pdf
- assignment3.4.docx
+ factorial.py
+ jobs.py
+ jobscp.py
+ lab assignment ... 
- lab assignment3.3.pdf
- lab1 HCP.pdf
+ matrixHCP.py
+ Untitled20.pynb
- week2 HCP.pdf
Generate code
>Add Context...
1 def classify_query(query):
2     """
3         Classify a query into one of four types:
4             - Informational
5             - Transactional
6             - Complaint
7             - Feedback
8     """
9     query_lower = query.lower()
10
11     # Define keywords for each category
12     informational_keywords = ['what', 'how', 'when', 'where', 'why', 'working hours', 'reset', 'password']
13     transactional_keywords = ['book', 'buy', 'order', 'purchase', 'reserve', 'ticket']
14     complaint_keywords = ['hasn\'t', 'didn\'t', 'broken', 'bad', 'poor', 'issue', 'problem', 'arrived']
15     feedback_keywords = ['great', 'good', 'excellent', 'user-friendly', 'nice', 'love', 'hate', 'experience']
16
17     # Check for complaint (highest priority)
18     if any(keyword in query_lower for keyword in complaint_keywords):
19         return 'Complaint'
20
21     # Check for transactional
22     if any(keyword in query_lower for keyword in transactional_keywords):
23         return 'Transactional'
24
25     # Check for feedback
26     if any(keyword in query_lower for keyword in feedback_keywords):
27         return 'Feedback'
28
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Ambiguity: Significantly reduced - pattern clear
Consistency: High - multiple examples establish standard
Pros: Best accuracy. Clear outcomes, Reduced errors
Cons: Requires manual examples, Prompt size
PS D:\AI Coding & C:/Users/NDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
What are your working hours? | Informational
I want to book a ticket | Transactional
My order hasn't arrived yet | Complaint
The app is very user-friendly | Feedback
How can I reset my password? | Informational
The service experience was bad | Complaint
PS D:\AI Coding

```

#### 4b) Design prompts for Zero-shot, One-shot, and Few-shot learning.

```

File Edit Selection View Go Run Terminal Help < > Q AI Coding
EXPLORER ... Welcome assignment 3.py AI lab43.py lab assignment 44.py X
AI CODING
+ add.py
+ AI lab43.py
- Assignment1(CP).pdf
- Assignment 2-4.pdf
- Assignment 3-4.py
- Assignment2.pdf
- assignment3.4.docx
+ factorial.py
+ jobs.py
+ jobscp.py
+ lab assignment ... 
- lab assignment3.3.pdf
- lab1 HCP.pdf
+ matrixHCP.py
+ Untitled20.pynb
- week2 HCP.pdf
Generate code
>Add Context...
1 # Query Classification System
2
3 def classify_query(query):
4     """
5         Classify a user query into one of four categories:
6             - Informational: User seeking information
7             - Transactional: User wanting to perform an action/transaction
8             - Complaint: User expressing dissatisfaction
9             - Feedback: User providing feedback or suggestions
10
11     query_lower = query.lower()
12
13     # Transactional keywords
14     transactional_keywords = ['book', 'buy', 'purchase', 'order', 'reserve', 'checkout', 'pay']
15
16     # Complaint keywords
17     complaint_keywords = ['problem', 'issue', 'broken', 'error', 'complaint', 'wrong', 'not working']
18
19     # Feedback keywords
20     feedback_keywords = ['feedback', 'suggest', 'idea', 'improve', 'opinion', 'review']
21
22     # Informational keywords
23     informational_keywords = ['how', 'what', 'when', 'where', 'why', 'can you tell', 'help', 'information']
24
25     # Classify based on keywords
26     if any(keyword in query_lower for keyword in transactional_keywords):
27         return "Transactional"
28
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Cons: Requires manual examples, Prompt size
PS D:\AI Coding & C:/Users/NDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
What are your working hours? | Informational
I want to book a ticket | Transactional
My order hasn't arrived yet | Complaint
The app is very user-friendly | Feedback
How can I reset my password? | Informational
The service experience was bad | Complaint
PS D:\AI Coding & C:/Users/NDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Query: I want to book a ticket
Type: Transactional
PS D:\AI Coding

```

#### 4c) Test all prompts on the same unseen queries.

The screenshot shows the VS Code interface with the AI Coding extension active. The terminal window displays the following test results:

```
How can I reset my password? | Informational
The service experience was bad | Complaint
Query: I want to book a ticket | Transactional
PS D:\AI Coding & C:/Users/ANDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Query: "The service experience was bad"
Type: Complaint

Query: "The app is very user-friendly" | Compliment
Type: Compliment
PS D:\AI Coding
```

#### 4d) Compare response correctness and ambiguity handling.

The screenshot shows the VS Code interface with the AI Coding extension active. The terminal window displays the following test results:

```
# This function classifies user queries into different types
def classify_query(query):
    if "book" in query.lower() or "reserve" in query.lower():
        return "Transactional"
    elif "order" in query.lower() or "arrived" in query.lower():
        return "Complaint"
    elif "support" in query.lower() or "great" in query.lower():
        return "Feedback"
    else:
        return "Informational"

# Example usage
query = "I want to book a ticket"
query_type = classify_query(query)
print(f"Query: '{query}'\nType: {query_type}")
```

Query: I want to book a ticket | Transactional
PS D:\AI Coding & C:/Users/ANDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Query: "The service experience was bad"
Type: Complaint

Query: "The app is very user-friendly" | Informational
Type: Informational
PS D:\AI Coding

#### 4e) Document observations.

The screenshot shows a code editor interface with the title bar "Q\_AI Coding". The left sidebar is labeled "EXPLORER" and contains a tree view of files under "AI CODING", including "assignment 3.py", "AI lab3.py", "Assignment (CP).pdf", "Assignment 2-4.pdf", "assignment 3.4", "Assignment2.pdf", "assignment3.4.docx", "factorial.py", "jobs.py", "jobscp.py", "lab assignment ...", "lab assignment3.3.pdf", "lab1 HCP.pdf", "matrixHCP.py", "Untitled20.pymb", and "week2 HCP.pdf". The main editor area displays the following text:

```
1 /**
2 * Observations from testing Zero-shot, One-shot, and Few-shot prompting for classification tasks:
3 *
4 * 1. Accuracy:
5 * - Zero-shot prompting often resulted in lower accuracy compared to One-shot and Few-shot methods, as the model had no prior examples to reference.
6 * - One-shot prompting showed improved accuracy, as the model could leverage a single example to understand the task better.
7 * - Few-shot prompting consistently yielded the highest accuracy, as multiple examples provided the model with a clearer context and better understanding of the classification task.
8 *
9 * 2. Ambiguity Handling:
10 * - Zero-shot prompting struggled with ambiguous inputs, often leading to incorrect classifications due to lack of context.
11 * - One-shot prompting reduced ambiguity to some extent, as the provided example helped clarify the task, but some ambiguity remained.
12 * - Few-shot prompting effectively handled ambiguity by providing multiple examples that illustrated different aspects of the classification task, allowing the model to make more informed decisions.
13 *
14 * 3. Consistency:
15 * - Zero-shot prompting exhibited high variability in results, with performance heavily dependent on the specific input phrasing.
16 * - One-shot prompting showed moderate consistency, as the single example could guide the model, but variations in input still affected outcomes.
17 * - Few-shot prompting demonstrated the highest consistency across different inputs, as the multiple examples helped stabilize the model's responses.
18 *
19 * 4. Overall Performance Differences:
20 * - Overall, Few-shot prompting outperformed both Zero-shot and One-shot methods in terms of accuracy, ambiguity handling, and consistency.
21 *
22 * - Zero-shot prompting was useful for quick assessments but lacked reliability for critical tasks.
23 *
***
```

The bottom of the editor shows the "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", and "TERMINAL" tabs. The terminal window shows the following command history:

```
Type: Transactional
PS D:\AI Coding & C:/Users/ANDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Query: "The service experience was bad"
Type: Complaint

Query: "The app is very user-friendly"
Type: Compliment
PS D:\AI Coding & C:/Users/ANDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Query: "I want to book a ticket."
Type: Transactional
PS D:\AI Coding & C:/Users/ANDALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding>
```

#### 5) Emotion Detection in Text

##### Scenario:

A mental-health chatbot needs to detect emotions: Happy, Sad, Angry, Anxious, Neutral.

##### 5a) Create labeled emotion samples.

The screenshot shows a dark-themed instance of Visual Studio Code. In the center is a code editor window displaying a Python script named `lab assignment 44.py`. The script imports pandas and creates a DataFrame from a list of text and emotion pairs. The code is as follows:

```
import pandas as pd

# Create a DataFrame from the provided data
data = [
    "Text": [
        "I am very happy today",
        "I feel lonely and depressed",
        "This is so frustrating",
        "I am worried about my future",
        "Today is just normal",
        "Feeling excited about results"
    ],
    "Emotion": [
        "Happy",
        "Sad",
        "Angry",
        "Anxious",
        "Neutral",
        "Happy"
    ]
]

df = pd.DataFrame(data)

# Display the DataFrame
print(df)
```

Below the code editor is a terminal window showing the execution of the script. It outputs the following error message:

```
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Traceback (most recent call last):
  File "d:/AI Coding/lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'.
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Traceback (most recent call last):
  File "d:/AI Coding/lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'.
PS D:\AI Coding>
```

### 5b) Use Zero-shot prompting to identify emotions.

This screenshot shows the same dark-themed VS Code environment. The code editor now contains a different Python script, also named `lab assignment 44.py`, which defines a function to identify emotions based on specific keywords. The code is as follows:

```
def identify_emotion(text):
    if "worried" in text:
        return "Anxious"
    return "Neutral"

text = "I am worried about my future"
emotion = identify_emotion(text)
print(f"Emotion: {emotion}")
```

The terminal window shows the output of the script, which correctly identifies the emotion as "Anxious".

```
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Traceback (most recent call last):
  File "d:/AI Coding/lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'.
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Traceback (most recent call last):
  File "d:/AI Coding/lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'.
Emotion: Anxious
PS D:\AI Coding>
```

### 5c) Use One-shot prompting with an example.

The screenshot shows the VS Code interface with the following details:

- EXPLORER** view: Shows files like `add.py`, `AI lab43.py`, `Assignment 2-4.pdf`, `Assignment 3-4.docx`, `assignment 3.py`, `Assignment2.pdf`, `factorial.py`, `jobs.py`, `jobscp.py`, `lab assignment ...`, `lab assignment3-3.pdf`, `lab1 HCP.pdf`, `matrixHCP.py`, `Untitled20.ipynb`, and `week2 HCP.pdf`.
- CODE** view: A code editor window titled "lab assignment 44.py" with the following Python code:

```
1 def identify_emotion(text):
2     if "frustrating" in text:
3         return "Frustrated"
4     return "Neutral"
5
6 # Example usage
7 text = "This is so frustrating"
8 emotion = identify_emotion(text)
9 print(f"Emotion: {emotion}")
```
- TERMINAL** view: Shows the command line output of running the script:

```
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Traceback (most recent call last):
  File "d:/AI Coding/lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Emotion: Anxious
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Emotion: Frustrated
PS D:\AI Coding
```

### 5d) Use Few-shot prompting with multiple emotions.

The screenshot shows the VS Code interface with the following details:

- EXPLORER** view: Shows files like `add.py`, `AI lab43.py`, `Assignment 2-4.pdf`, `Assignment 3-4.pdf`, `Assignment 3-4.docx`, `assignment 3.py`, `Assignment2.pdf`, `factorial.py`, `jobs.py`, `jobscp.py`, `lab assignment ...`, `lab assignment3-3.pdf`, `lab1 HCP.pdf`, `matrixHCP.py`, `Untitled20.ipynb`, and `week2 HCP.pdf`.
- CODE** view: A code editor window titled "lab assignment 44.py" with the following Python code:

```
1 def classify_emotion(text):
2     emotions = {
3         "happy": ["happy", "joyful", "excited", "pleased"],
4         "sad": ["lonely", "depressed", "sad", "down"],
5         "anxious": ["worried", "anxious", "nervous", "stressed"],
6         "neutral": ["normal", "fine", "okay", "average"],
7         "frustrated": ["frustrating", "annoyed", "irritated"]
8     }
9
10    for emotion, keywords in emotions.items():
11        if any(keyword in text.lower() for keyword in keywords):
12            return emotion
13    return "Unknown"
14
15 # Example usage
16 text = "This is so frustrating"
17 emotion = classify_emotion(text)
18 print(f"Text: {text}\nEmotion: {emotion}")
```
- TERMINAL** view: Shows the command line output of running the script:

```
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Traceback (most recent call last):
  File "d:/AI Coding/lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Emotion: Anxious
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Emotion: Frustrated
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Text: This is so frustrating
Emotion: frustrated
PS D:\AI Coding
```

## 5e) Discuss ambiguity handling across techniques.

The screenshot shows a code editor interface with the following details:

- EXPLORER** panel on the left listing files: add.py, AI lab43.py, Assignment 2-4.pdf, Assignment 3-4.docx, assignment 3-4.pdf, assignment34.doc, factorial.py, jobs.py, jobs.py, lab assignment ... (selected), lab assignment33.pdf, lab1 HCP.pdf, matrixHCP.py, Untitled2.ipynb, week2 HCP.pdf.
- CODE** tab active, showing a Python script named lab assignment 44.py:

```
1 # Emotion Handling Techniques
2
3 def handle_emotion(technique, input_text):
4     if technique == "zero-shot":
5         return "This technique struggles with ambiguity in understanding emotions."
6     elif technique == "one-shot":
7         return "This technique provides better clarity in emotional interpretation."
8     elif technique == "few-shot":
9         return "This technique achieves the best emotional accuracy by learning from examples."
10    else:
11        return "Unknown technique."
12
13 # Example usage
14 techniques = ["zero-shot", "one-shot", "few-shot"]
15 for technique in techniques:
16     print(f"{technique.capitalize()}: {handle_emotion(technique, '')}")
```
- PROBLEMS**, **OUTPUT**, **DEBUG CONSOLE**, **TERMINAL** (selected), and **PORTS** tabs at the bottom.
- TERMINAL** tab shows command-line output:

```
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Emotion: Anxious
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Emotion: Frustrated
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Text: This is frustrating
Emotion: Frustrated
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Zero-shot: This technique struggles with ambiguity in understanding emotions.
One-shot: This technique provides better clarity in emotional interpretation.
Few-shot: This technique achieves the best emotional accuracy by learning from examples.
PS D:\AI Coding>
```