

2303A51491

Batch-25

Assignment-4.4

## 1. Sentiment Classification for Customer Reviews

Scenario:

An e-commerce platform wants to analyze customer reviews and classify

Week2

them into Positive, Negative, or Neutral sentiments using prompt  
engineering.

Tasks:

- a) Prepare 6 short customer reviews mapped to sentiment labels.
- b) Design a Zero-shot prompt to classify sentiment.
- c) Design a One-shot prompt with one labeled example.
- d) Design a Few-shot prompt with 3–5 labeled examples.
- e) Compare the outputs and discuss accuracy differences.

The screenshot shows the Microsoft Visual Studio Code interface. The left sidebar displays a file tree with various files, including 'ass 4.4.py' which is currently open. The code in 'ass 4.4.py' is as follows:

```
24     """
25     #review": "It's okay. Does what it's supposed to do, nothing special."
26     #sentiment": "Neutral"
27   }
28 }
29 # Print reviews with sentiment labels
30 print("Commerce Customer Reviews - Sentiment Analysis")
31 print("-" * 70)
32 for idx, item in enumerate(reviews, 1):
33   print("\nReview #{}:\nText: {}\nSentiment: {}".format(item['review'], item['sentiment']))
34   print("-" * 70)
35
36 print("Summary:")
37 positive_count = sum(1 for item in reviews if item['sentiment'] == "Positive")
38 negative_count = sum(1 for item in reviews if item['sentiment'] == "Negative")
39 neutral_count = sum(1 for item in reviews if item['sentiment'] == "Neutral")
40
41 print("Positive: {} | Negative: {} | Neutral: {}".format(positive_count, negative_count, neutral_count))
42
43 print("-" * 70)
```

The terminal tab shows the output of running the script:

```
Review #1
Text: Terrible experience. Item arrived damaged and customer service was unhelpful.
Sentiment: Negative

Review #2
Text: The product arrived on time. It works as described.
Sentiment: Neutral

Review #3
Text: Love it! Exceeded my expectations and great value for money.
Sentiment: Positive

Review #4
Text: Not satisfied. Poor packaging and item doesn't match the description.
Sentiment: Negative

Review #5
Text: It's okay. Does what it's supposed to do, nothing special.
Sentiment: Neutral
```

A summary is provided at the bottom:

```
Summary:
Positive: 2 | Negative: 2 | Neutral: 2
PS C:\Users\parva\OneDrive\Desktop\AI Assted
```

The status bar indicates the date and time as 1/29/2026, 1:21 PM.

This screenshot shows the same Microsoft Visual Studio Code interface with a different Python script, 'ass 4.4.py'. The code is as follows:

```
1 def classify_sentiment(review):
2   """
3     # Classify based on keyword counts
4     if positive_count > negative_count:
5       return "Positive"
6     elif negative_count > positive_count:
7       return "Negative"
8     else:
9       return "Neutral"
10
11
12   # Test the classifier with user input
13   user_review = input("\nEnter a customer review: ")
14   sentiment = classify_sentiment(user_review)
15   print("User Review: {}\n".format(user_review))
16   print("Classified Sentiment: {}\n".format(sentiment))
17
18   # Optional: test classifier on existing reviews
19   print("-" * 70)
20   print("Testing Classifier on Existing Reviews")
21   print("-" * 70)
```

The terminal output shows the classifier being tested with user input:

```
Review #5
Text: Not satisfied. Poor packaging and item doesn't match the description.
Sentiment: Negative

Review #6
Text: It's okay. Does what it's supposed to do, nothing special.
Sentiment: Neutral

Review #7
Text: Love it! Exceeded my expectations and great value for money.
Sentiment: Positive

Review #8
Text: Not satisfied. Poor packaging and item doesn't match the description.
Sentiment: Negative

Review #9
Text: It's okay. Does what it's supposed to do, nothing special.
Sentiment: Neutral
```

The status bar indicates the date and time as 1/29/2026, 1:24 PM.

Screenshot of the Visual Studio Code interface showing Python code for sentiment classification. The code uses keyword-based logic to classify reviews as Positive, Negative, or Neutral. It defines lists of positive and negative keywords, counts matches, and returns the classification based on the count ratio. A main program prints the classifier name and calls it with a sample review.

```
139 def classify_sentiment(review):
140     """
141         Classifies sentiment of a review using keyword-based logic.
142         Returns: Positive, Negative, or Neutral
143         """
144     review_lower = review.lower()
145
146     # Define keyword lists
147     positive_keywords = [
148         "amazing", "excellent", "great", "love", "wonderful", "wonderfully",
149         "fantastic", "perfect", "loved", "good", "ext", "satisfied",
150         "exceeded", "happy", "impressed", "quality", "value"
151     ]
152
153     negative_keywords = [
154         "terrible", "bad", "worst", "hate", "sour", "damaged", "unhelpful",
155         "not satisfied", "disappointed", "waste", "broken", "awful", "useless",
156         "disappointing", "issue", "problem", "defective", "unhappy"
157     ]
158
159     # Count keyword matches
160     positive_count = sum(1 for keyword in positive_keywords if keyword in review_lower)
161     negative_count = sum(1 for keyword in negative_keywords if keyword in review_lower)
162
163     # Classify based on keyword counts
164     if positive_count > negative_count:
165         return "Positive"
166     elif negative_count > positive_count:
167         return "Negative"
168     else:
169         return "Neutral"
170
171     # Main program
172     print("= * *")
173     print("Customer Review Sentiment Classifier")
174     print("= * *")
175     print("= * *")
176
177     # Display existing reviews with classifications
```

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS

NameError: name 'review' is not defined

NameError: name 'review' is not defined

File "C:\Users\parva\Desktop\AI Assisted\ass 4.4.py", line 163, in <module>
 Classifies sentiment of a review using keyword-based logic.
 IndentationError: unexpected indent

PS C:\Users\parva\Desktop\AI Assisted>

Screenshot of the Visual Studio Code interface showing Python code for sentiment classification with user interaction. The code adds a loop to classify user input reviews. It prints examples, classifies existing reviews, and then enters a loop where it prompts the user for a review, classifies it, and prints the result. The user can exit by entering 'quit'.

```
276 print("Customer Review Sentiment Classifier")
277 print("= * *")
278 # Display example classifications
279 print("Example classifications:")
280 print("= * *")
281 examples = [
282     "The product is excellent and works perfectly",
283     "The item is okay, not great",
284     "Very disappointed with the quality",
285     "Average experience overall"
286 ]
287
288 for example in examples:
289     predicted = classify_sentiment(example)
290     print(f"\nPredicted: {example}")
291     print(f"Sentiment: {predicted}")
292
293 # Display existing reviews with classifications
294 print("= * *")
295 print("Existing Reviews Analysis")
296 print("= * *")
297
298 for idx, item in enumerate(reviews, 1):
299     predicted_sentiment = classify_sentiment(item["review"])
300     actual_sentiment = item["sentiment"]
301     match = "<" if predicted_sentiment == actual_sentiment else "X"
302
303     print(f"\nReview {idx}: {match}")
304     print(f"Text: {item['review']}")
305     print(f"Predicted: {predicted_sentiment} | Actual: {actual_sentiment}")
306
307 # Get user input and classify
308 print("= * *")
309 print("Classify Your Own Review")
310 print("= * *")
311
312 while True:
313     user_review = input("\nEnter a customer review (or 'quit' to exit): ")
314
315     if user_review.lower() == "quit":
316         print("Thank you for using the sentiment classifier!")
317         break
318
319     if user_review.strip():
320         sentiment = classify_sentiment(user_review)
```

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS

Classifies sentiment of a review using keyword-based logic.

IndentationError: unexpected indent

PS C:\Users\parva\Desktop\AI Assisted> 70

PS C:\Users\parva\Desktop\AI Assisted>

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

- Explorer View:** Shows files like "AI Assisted", "check\_leap\_year.py", "lab 4.3 word.doc", "lab 4.3 word.pdf", "lab 4.3 py", "lab 4.3 py", "lab assignment 3.3.pdf", "lab assignment 1.4.pdf", "lab assignment 1.4.pdf", and "lab assignment 2.3.pdf".
- Code Editor:** Displays a Python script named "ass 4.4.py" containing code for sentiment analysis. The code includes functions for classifying reviews, calculating accuracy, and comparing different approaches (Zero-shot, One-shot, Few-shot).
- Terminal:** Shows the command "PS C:\Users\parva\Desktop\AI Assisted & C:\Users\parva\appdata\Local\Programs\Python\Python311\python.exe %1" and the output of the script running.
- Output Panel:** Shows the output of the script execution.
- Right-hand Side:** Includes a "Sentiment Classification for Customers" panel with sample code and a "User Reference" panel with existing code snippets.

## 2. Email Priority Classification

Scenario:

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

Tasks:

1. Create 6 sample email messages with priority labels.
2. Perform intent classification using Zero-shot prompting.
3. Perform classification using One-shot prompting.
4. Perform classification using Few-shot prompting.
5. Evaluate which technique produces the most reliable results and why.

File Edit Selection View Go Run Terminal Help

AI Asst

EXPLORER AI ASSISTED add.py ass 4.pdf ass 4.py check\_leap\_year.py lab 4.3 word.doc lab 4.3 word.pdf lab 4.3 word.py lab 4.3.py lab assignment 3.3.pdf lab assignment 1.4.pdf lab assignment 2.3.pdf leap\_year.py

```

❶ ass 4.py >-
❷ # Simple list of email tuples: (subject, body, priority)
❸ emails = [
❹     ("Urgent: System Outage - Immediate Action Required", "The main database server is down. All operations are halted.", "High"),
❺     ("Q1 Budget Review Meeting - Next Friday at 2 PM", "Please review the attached budget documents.", "Medium"),
❻     ("Office Lunch - Catering Menu for Next week", "Please vote on your preferred lunch option.", "Low"),
❼     ("Client Presentation Delayed - Decision Needed Today", "Our major client has requested to reschedule the presentation.", "High"),
➋     ("Monthly Team Updates - Please Submit by End of Week", "Your monthly progress report by Friday.", "Medium"),
⌋     ("Office Supplies Restocking - New Printer Paper Available", "New printer paper has arrived in the supply closet.", "Low")
⌌ ]
⌍
⌎ # Print emails with priority
⌏ print("OFFICE EMAILS")
⌐ print("-" * 80)
⌑
⌒ for idx, (subject, body, priority) in enumerate(emails, 1):
⌓     print("\nEmail #{}: {}".format(idx, priority))
⌔     print("Subject: {}".format(subject))
⌕     print("Body: {}".format(body))
⌖
⌗ # Summary
⌘ print("-" * 80)
⌙ print("High: (sum[1 for e in emails if e[2] == 'High']) | Medium: (sum[1 for e in emails if e[2] == 'Medium']) | Low: (sum[1 for e in emails if e[2] == 'Low'])")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Body: The main database server is down. All operations are halted.  
 Email #2 (Medium Priority)  
 Subject: Q1 Budget Review Meeting - Next Friday at 2 PM  
 Body: Please review the attached budget documents.  
 Email #3 (Low Priority)  
 Subject: Office Lunch - Catering Menu for Next week  
 Body: Please vote on your preferred lunch option.  
 Email #4 (High Priority)  
 Subject: Critical! Client Presentation Delayed - Decision Needed Today  
 Body: Our major client has requested to reschedule the presentation.  
 Email #5 (Medium Priority)  
 Subject: Monthly Team Updates - Please Submit by End of Week  
 Body: Submit your monthly progress report by Friday.  
 Email #6 (Low Priority)  
 Subject: Office Supplies Restocking - New Printer Paper Available  
 Body: New printer paper has arrived in the supply closet.

-----  
 High: 2 | Medium: 2 | Low: 2  
 PS C:\Users\parva\Desktop\AI Asst>

28°C

File Edit Selection View Go Run Terminal Help

AI Asst

EXPLORER AI ASSISTED add.py ass 4.pdf ass 4.py check\_leap\_year.py lab 4.3 word.doc lab 4.3 word.pdf lab 4.3 word.py lab 4.3.py lab assignment 3.3.pdf lab assignment 1.4.pdf lab assignment 2.3.pdf leap\_year.py

File Edit Selection View Go Run Terminal Help

AI Asst

EXPLORER AI ASSISTED add.py ass 4.pdf ass 4.py check\_leap\_year.py lab 4.3 word.doc lab 4.3 word.pdf lab 4.3 word.py lab 4.3.py lab assignment 3.3.pdf lab assignment 1.4.pdf lab assignment 2.3.pdf leap\_year.py

```

❶ ass 4.py >-
❷ # Simple priority classifier function
❸ def classify_priority(subject, body):
❹     """Classify email priority using basic keywords"""
❺     text = (subject + " " + body).lower()
❻
⌎ if any(word in text for word in ["urgent", "critical", "immediate", "outage", "emergency"]):
⌏     return "High"
⌒ elif any(word in text for word in ["important", "meeting", "review", "deadline", "required"]):
⌓     return "Medium"
⌔ else:
⌕     return "Low"
⌖
⌗ # Test classifier on sample emails
⌘ print("PRIORITY CLASSIFIER TEST")
⌙ print("-" * 80)
⌚
⌛ for idx, (subject, body, actual_priority) in enumerate(emails, 1):
⌜     predicted = classify_priority(subject, body)
⌝     match "X" if predicted == actual_priority else "X"
⌞     print("Email #{}: {} | Predicted: {} | Actual: {} | Priority: {} |".format(idx, subject, predicted, actual_priority, priority))
⌟
⌠ # Test with custom email
⌡ print("-" * 80)
⌢ custom_subject = input("Enter email subject: ")
⌣ custom_body = input("Enter email body: ")
⌤ result = classify_priority(custom_subject, custom_body)
⌥ print("Classified Priority: (result)")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Email #1 (Low Priority)  
 Subject: Office Lunch - Catering Menu for Next week  
 Body: Please vote on your preferred lunch option.  
 Email #4 (High Priority)  
 Subject: Critical! Client Presentation Delayed - Decision Needed Today  
 Body: Our major client has requested to reschedule the presentation.  
 Email #5 (Medium Priority)  
 Subject: Monthly Team Updates - Please Submit by End of Week  
 Body: Submit your monthly progress report by Friday.  
 Email #6 (Low Priority)  
 Subject: Office Supplies Restocking - New Printer Paper Available  
 Body: New printer paper has arrived in the supply closet.

-----  
 High: 2 | Medium: 2 | Low: 2  
 PS C:\Users\parva\Desktop\AI Asst> & C:\Users\parva\AppData\Local\Programs\Python\Python311\python.exe
 Python 3.11.1 (tags/v3.11.1:8a3d23a, Aug 14 2023, 14:15:13) [MSC v.3994 64 bit (AMD64)] on win32
 Type "help", "copyright", "credits" or "license" for more information.
 Ctrl+C to launch Vt Code Native REPL
 >>> </c:/users/parva/appdata/local/programs/python/python311/python.exe c:/users/parva/Desktop/AI Asst/ass 4.4.py>
 File "ass 4.4.py", line 1

File Edit Selection View Go Run Terminal Help

AI Asst

EXPLORER AI ASSISTED add.py ass 4.pdf ass 4.py check\_leap\_year.py lab 4.3 word.doc lab 4.3 word.pdf lab 4.3 word.py lab 4.3.py lab assignment 3.3.pdf lab assignment 1.4.pdf lab assignment 2.3.pdf leap\_year.py

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

- File Explorer:** Shows files like `AI Assisted`, `add.py`, `AI ass1.pdf`, `ass 4.4.py`, `check_leap_year.py`, `lab 4.3 word.doc`, `lab 4.3 word.pdf`, `lab 4.3 word.py`, `lab 4.3 year.pdf`, `lab 4.3 year.py`, `lab assignment 3.3.pdf`, `lab assignment 1.4.pdf`, `lab assignment 2.3.pdf`, and `leap_year.py`.
- Code Editor:** Displays Python code for classifying email priority based on subject and body content. The code includes logic for handling custom emails and existing code snippets.
- Terminal:** Shows the command line output of running the script, indicating syntax errors.
- Output:** Shows the results of the email classification process.
- Search:** A search bar at the top.
- Bottom Bar:** Includes icons for file operations, a search bar, and system status.

This screenshot is nearly identical to the first one, showing the same code editor and file structure. However, the terminal output is different, reflecting a successful run of the script without syntax errors. The output shows the classification of various emails as High, Medium, or Low priority.

### 3. Student Query Routing System

Scenario:

A university chatbot must route student queries to Admissions, Exams,

Academics, or Placements.

## Tasks:

1. Create 6 sample student queries mapped to departments.
2. Implement Zero-shot intent classification using an LLM.
3. Improve results using One-shot prompting.
4. Further refine results using Few-shot prompting.
5. Analyze how contextual examples affect classification accuracy.

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

- File Explorer:** Shows files including `ass 4.4.py`, `add.py`, `AI Assisted`, `check leap year.py`, `lab 4.3 word.docx`, `lab 4.3 word.pdf`, `lab 4.3 zip`, `lab ass 3.4.py`, `lab assignment 3.3.pdf`, `lab assignment 1.4.pdf`, `lab assignment 2.3.pdf`, and `leap year.py`.
- Code Editor:** Displays the `ass 4.4.py` file content:

```
# Simple list of student queries (query, department)
queries = [
    ("How do I apply for admissions?", "Admissions"),
    ("When are the exams scheduled?", "Academics"),
    ("How can I change my course?", "Academics"),
    ("What are the placement criteria?", "Placements"),
    ("How do I request a transcript?", "Academics"),
    ("What is the application deadline?", "Admissions"),
]

for q, dept in queries:
    print(f"{dept}: {q}")
```
- Terminal:** Shows the output of running the script:

```
SyntaxError: invalid syntax
>>> & C:/Users/perva/AppData/Local/Programs/Python/Python311/python.exe "c:/users/perva/OneDrive/Desktop/AI Assisted/ass 4.4.py"
  File "<stdin>", line 1
    & C:/Users/perva/AppData/Local/Programs/Python/Python311/python.exe "c:/users/perva/OneDrive/Desktop/AI Assisted/ass 4.4.py"
SyntaxError: invalid syntax
>>> & C:/Users/perva/AppData/Local/Programs/Python/Python311/python.exe "c:/users/perva/OneDrive/Desktop/AI Assisted/ass 4.4.py"
  File "<stdin>", line 1
    & C:/Users/perva/AppData/Local/Programs/Python/Python311/python.exe "c:/users/perva/OneDrive/Desktop/AI Assisted/ass 4.4.py"
SyntaxError: invalid syntax
>>> & C:/Users/perva/AppData/Local/Programs/Python/Python311/python.exe "c:/users/perva/OneDrive/Desktop/AI Assisted/ass 4.4.py"
  File "<stdin>", line 1
    & C:/Users/perva/AppData/Local/Programs/Python/Python311/python.exe "c:/users/perva/OneDrive/Desktop/AI Assisted/ass 4.4.py"
SyntaxError: invalid syntax
>>> exit()
```
- Output:** Shows the command prompt output:

```
PS C:\Users\perva\OneDrive\Desktop\AI Assisted> & C:/users/perva/AppData/Local/Programs/Python/Python311/python.exe "c:/users/perva/OneDrive/Desktop/AI Assisted/ass 4.4.py"
Admissions: How do I apply for admissions?
Exams: When are the exams scheduled?
Academics: How can I change my course?
Placements: What are the placement criteria?
Academics: How do I request a transcript?
Admissions: What is the application deadline?
```
- Right Panel:** Contains a sidebar with instructions for sentiment classification and a preview of the code.

The screenshot shows the Visual Studio Code interface with the "AI Assisted" extension open. The main editor window contains Python code for sentiment classification:

```

1 # ...existing code...
2
3 def classify_query(q):
4     t = q.lower()
5
6     if any(k in t for k in ("admission", "apply", "application", "enroll", "fee", "deadline")):
7         return "Admissions"
8
9     elif any(k in t for k in ("exam", "exams", "result", "results", "grade", "schedule")):
10        return "Exam"
11
12    elif any(k in t for k in ("placement", "placements", "internship", "job", "career", "interview")):
13        return "Placement"
14
15    else:
16        return "Academics"
17
18
19 if __name__ == "__main__":
20     q = input("Query: ").strip()
21     print("Department:", classify_query(q))
22
23 # ...existing code...

```

The status bar at the bottom right shows the date as 1/29/2026 and the time as 1:50 PM.

This screenshot shows a similar setup in VS Code, but with a different file structure. The main editor window contains the same Python code for sentiment classification:

```

1 # ...existing code...
2
3 def classify_query(q):
4     t = q.lower()
5
6     if any(k in t for k in ("admission", "apply", "application", "enroll", "fee", "deadline")):
7         return "Admissions"
8
9     elif any(k in t for k in ("exam", "exams", "result", "results", "grade", "schedule")):
10        return "Exam"
11
12    elif any(k in t for k in ("placement", "placements", "internship", "job", "career", "interview")):
13        return "Placement"
14
15    else:
16        return "Academics"
17
18
19 if __name__ == "__main__":
20     q = input("Query: ").strip()
21     print("Department:", classify_query(q))
22
23 # ...existing code...

```

The status bar at the bottom right shows the date as 1/29/2026 and the time as 1:51 PM.

The screenshot shows the Visual Studio Code interface with the "AI Assisted" extension open. The left sidebar shows files like "ass 4.4.py" and "check\_leap\_year.py". The main editor window displays Python code for sentiment classification. A sidebar panel titled "SENTIMENT CLASSIFICATION FOR CUSTOMER QUERIES" provides examples and instructions. The bottom status bar shows the date and time.

```

# ...existing code...
def classify_query(q):
    t = q.lower()
    if "exam" in t or "semester" in t:
        return "Exams"
    elif "course" in t or "syllabus" in t:
        return "Academics"
    elif "placement" in t or "campus" in t:
        return "Placements"
    elif "admission" in t or "apply" in t:
        return "Admissions"
    else:
        return "General"

if __name__ == "__main__":
    q = input().strip()
    print("Department:", classify_query(q))
# ...existing code...

```

This screenshot shows the same setup as above, but with more examples in the "SENTIMENT CLASSIFICATION FOR CUSTOMER QUERIES" sidebar. It includes examples for "course syllabus details", "course placement updates", "course syllabus", "placement updates", and "internship opportunities". The bottom status bar shows the date and time.

```

# ...existing code...
def classify_query(q):
    t = q.lower()
    if "exam" in t or "semester" in t:
        return "Exams"
    elif "course" in t or "syllabus" in t:
        return "Academics"
    elif "finals" in t or "semesters" in t:
        return "Exams"
    elif "placement" in t or "campus" in t:
        return "Placements"
    elif "admission" in t or "apply" in t:
        return "Admissions"
    else:
        return "Academics"

one_example = "admission process"
def zero_shot():
    t = q.lower()
    if one_example in t:
        return "Admissions"
    return zero_shot()

few_examples = [
    "admission process",
    "how to apply for admission",
    "finals scheduled next semester",
    "course syllabus",
    "placement updates",
    "internship opportunities",
]
def few_shot():
    t = q.lower()
    for k,v in few_examples.items():
        if k in t:
            return v
    return zero_shot()

def score(fm):
    return sum(1 for q,a in tests if fm(q)==a)/len(tests)
# ...existing code...

```

## 4. Chatbot Question Type Detection

Scenario:

A chatbot must identify whether a user query is Informational,

Transactional, Complaint, or Feedback.

Tasks:

1. Prepare 6 chatbot queries mapped to question types.
2. Design prompts for Zero-shot, One-shot, and Few-shot learning.
3. Test all prompts on the same unseen queries.
4. Compare response correctness and ambiguity handling.
5. Document observations.

The screenshot shows a code editor interface with several files open in the sidebar and tabs at the top. The active tab is 'ass 4.4.py'. The code contains functions for zero-shot, one-shot, and few-shot classification based on user input. A right-hand panel displays a 'CHAT' window with a conversation about sentiment classification, and a 'TODO' list with numbered items related to the tasks.

```
ass 4.4.py
# Sample queries and types
queries = [
    "What's the status of my order",
    "What's my billing address",
    "The product arrived broken and unusable",
    "How do I use the new dashboard feature",
    "Thanks for the quick support, great job!"
]

# Zero-shot classification
def zero_shot(q):
    t = q.lower()
    if any(k in t for k in ("broken", "damaged", "not working", "complain", "issue", "problem")):
        return "Complaint"
    if any(k in t for k in ("reset", "change", "cancel", "status", "buy", "buy", "billing")):
        return "Transactional"
    if any(k in t for k in ("love", "great", "thanks", "thank", "feedback", "suggest")):
        return "Feedback"
    return "Informational"

# One-shot classification
def one_shot(q):
    if "reset my password" in q.lower(): return "Transactional"
    return zero_shot(q)

# Few-shot classification
def few_shot(q):
    examples = {
        "reset my password": "Transactional",
        "order arrived damaged": "Complaint",
        "love the new interface": "Feedback",
        "user guide": "Informational"
    }
    t = q.lower()
    for k, v in examples.items():
        if k in t: return v
    return zero_shot(q)

# Run all queries
for q in queries:
    print("Input: ", q)
    print("Zero-shot: ", zero_shot(q))
    print("One-shot: ", one_shot(q))
    print("Few-shot: ", few_shot(q))
```

## 5. Emotion Detection in Text

Scenario:

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

Anxious, Neutral.

Tasks:

1. Create labeled emotion samples.
2. Use Zero-shot prompting to identify emotions.
3. Use One-shot prompting with an example.
4. Use Few-shot prompting with multiple emotions.
5. Discuss ambiguity handling across techniques.

```
File Edit Selection View Go Run Terminal Help < > Q AI Coding

EXPLORER ... Welcome assignment 3.py AI lab43.py lab assignment 44.py X
Generate code
Add Context...
1 import pandas as pd
2
3 # Create a DataFrame from the provided data
4 data = [
5     "Text": [
6         "I am very happy today",
7         "I feel lonely and depressed",
8         "This is so frustrating",
9         "I am worried about my future",
10        "Today is just normal",
11        "Feeling excited about results"
12    ],
13    "Emotion": [
14        "Happy",
15        "Sad",
16        "Angry",
17        "Anxious",
18        "Neutral",
19        "Happy"
20    ]
21 }
22
23 df = pd.DataFrame(data)
24
25 # Display the DataFrame
26 print(df)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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>
```

```
File Edit Selection View Go Run Terminal Help < > Q AI Coding

EXPLORER ... Welcome assignment 3.py AI lab43.py lab assignment 44.py X
Generate code
Add Context...
1 import pandas as pd
2
3 # Create a DataFrame from the provided data
4 data = [
5     "Text": [
6         "I am very happy today",
7         "I feel lonely and depressed",
8         "This is so frustrating",
9         "I am worried about my future",
10        "Today is just normal",
11        "Feeling excited about results"
12    ],
13    "Emotion": [
14        "Happy",
15        "Sad",
16        "Angry",
17        "Anxious",
18        "Neutral",
19        "Happy"
20    ]
21 }
22
23 df = pd.DataFrame(data)
24
25 # Display the DataFrame
26 print(df)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

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>
```

The screenshot shows the VS Code interface with the 'AI Coding' workspace selected. In the center editor area, the file 'lab assignment 44.py' is open, displaying 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}")
```

A code completion tooltip is displayed above the word 'identify\_emotion'. The tooltip contains the function signature and a snippet of code demonstrating its use. The status bar at the bottom shows the path 'D:\AI Coding'.

The screenshot shows the VS Code interface with the 'AI Coding' workspace selected. In the center editor area, the file 'lab assignment 44.py' is open, displaying 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}")
```

A code completion tooltip is displayed above the word 'classify\_emotion'. The tooltip contains the function signature and a snippet of code demonstrating its use. The status bar at the bottom shows the path 'D:\AI Coding'.

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists various files and folders under the 'AI CODING' category, including 'add.py', 'AI lab43.py', 'Assignment1(CP).pdf', 'Assignment 2-4.pdf', 'assignment 3.4', 'Assignment2.pdf', 'assignment3.4.docx', 'factorial.py', 'jobs.py', 'jobs.py', 'lab assignment ...', 'lab assignment3.3.pdf', 'lab1 HCP.pdf', 'mathttHCP.py', 'Untitled20.py', and 'week2 HCP.pdf'. The main workspace displays a Python script named 'lab assignment 44.py'. The script defines a function 'classify\_emotion' that takes a string 'text' and returns an emotion based on a dictionary of keywords. A sample usage example is shown at the bottom. Below the code editor is a terminal window showing a stack trace for a 'ModuleNotFoundError' due to missing the 'pandas' module. The terminal also shows the execution of the script and its output.

```
def classify_emotion(text):
    emotions = {
        "happy": ["happy", "joyful", "excited", "pleased"],
        "sad": ["lonely", "depressed", "sad", "down"],
        "anxious": ["worried", "anxious", "nervous", "stressed"],
        "neutral": ["normal", "fine", "okay", "average"],
        "frustrated": ["frustrating", "annoyed", "irritated"]
    }

    for emotion, keywords in emotions.items():
        if any(keyword in text.lower() for keyword in keywords):
            return emotion
    return "Unknown"

# Example usage
text = "This is so frustrating"
emotion = classify_emotion(text)
print(f"Text: '{text}'\nEmotion: {emotion}")
```

```
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>
```