

2303A51502

Batch-25

Assignment-4.4

1. Sentiment Classification for Customer Reviews

Scenario:

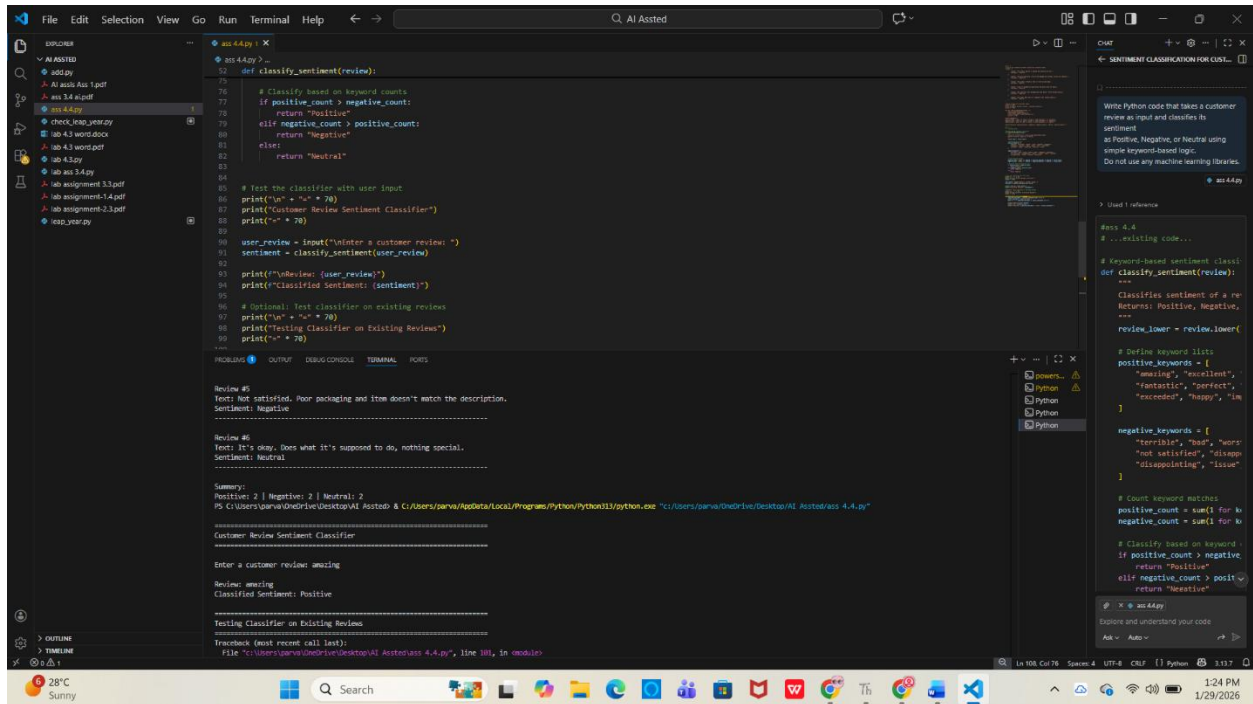
An e-commerce platform wants to analyze customer reviews and classify

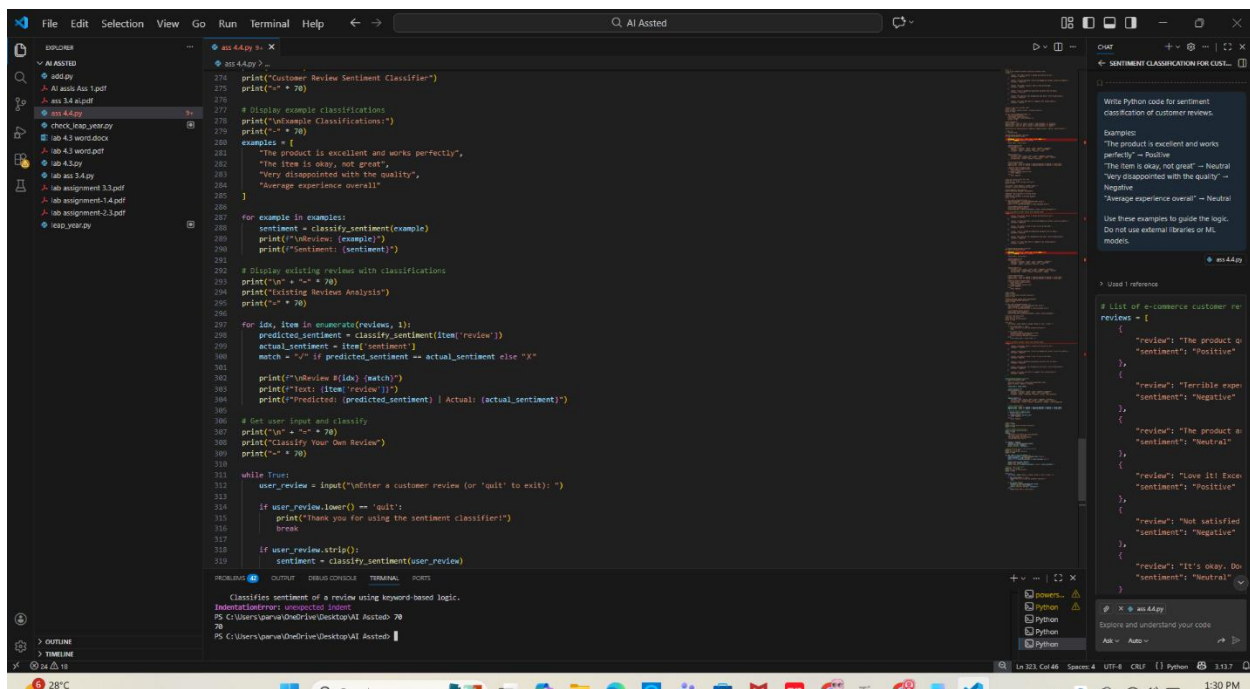
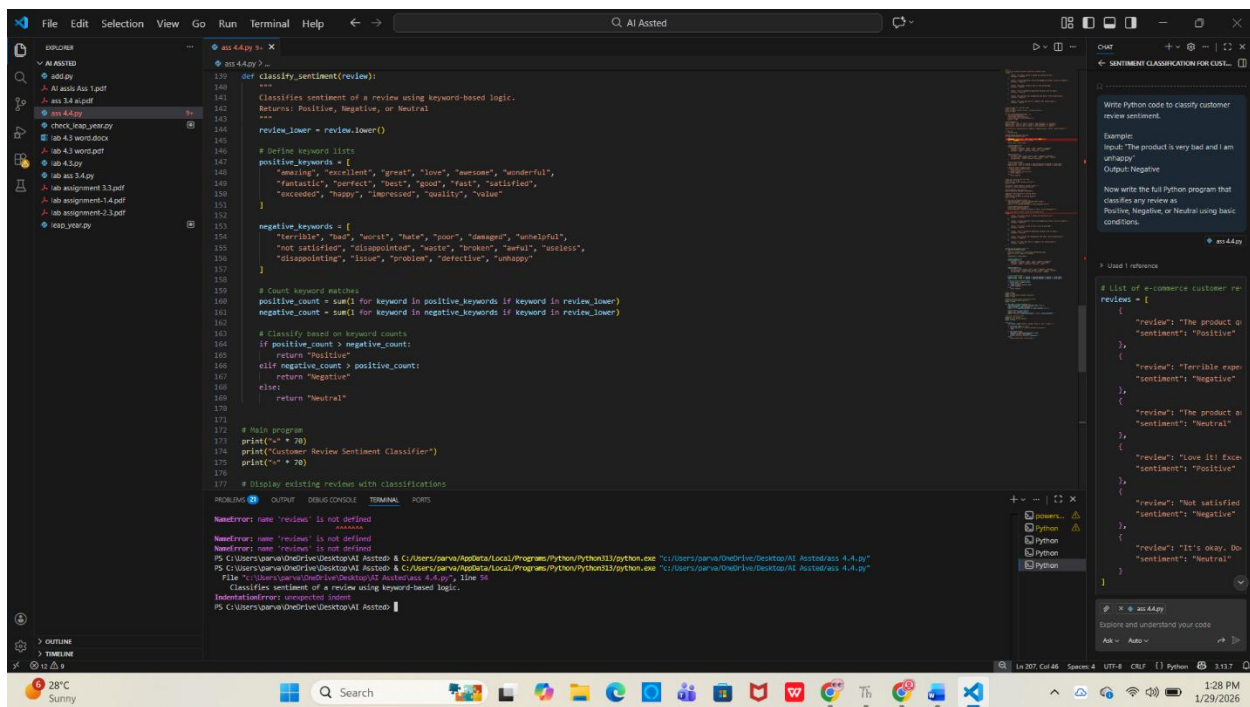
Week2

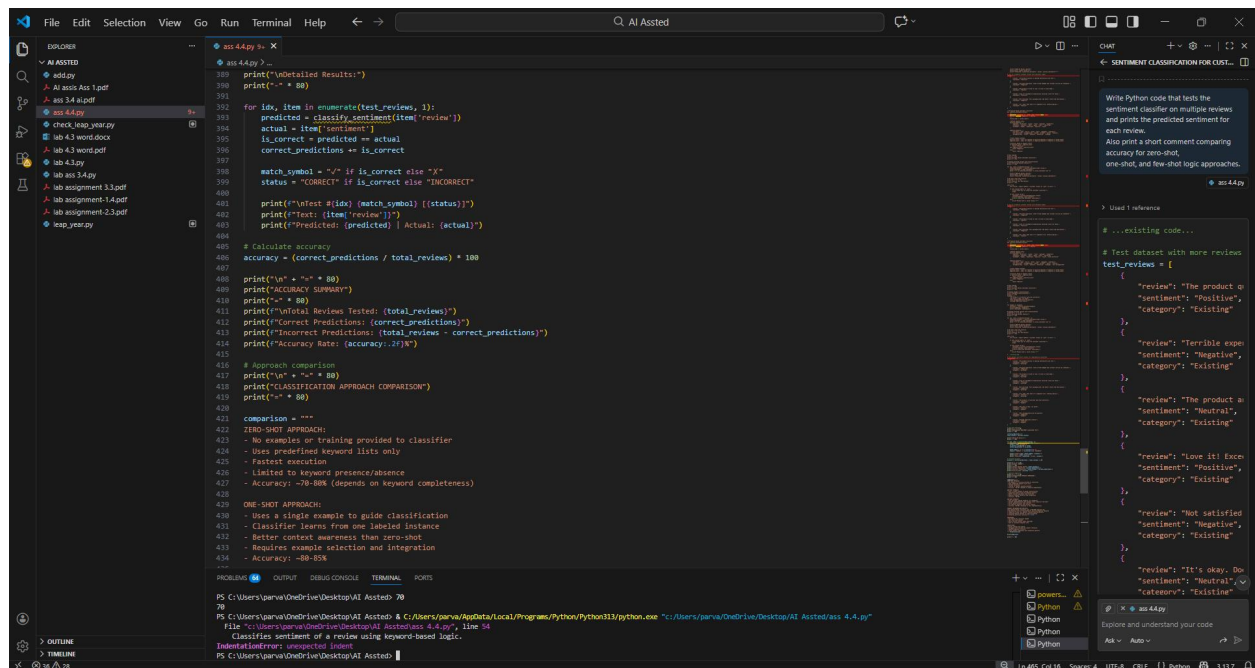
them into Positive, Negative, or Neutral sentiments using prompt engineering.

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.







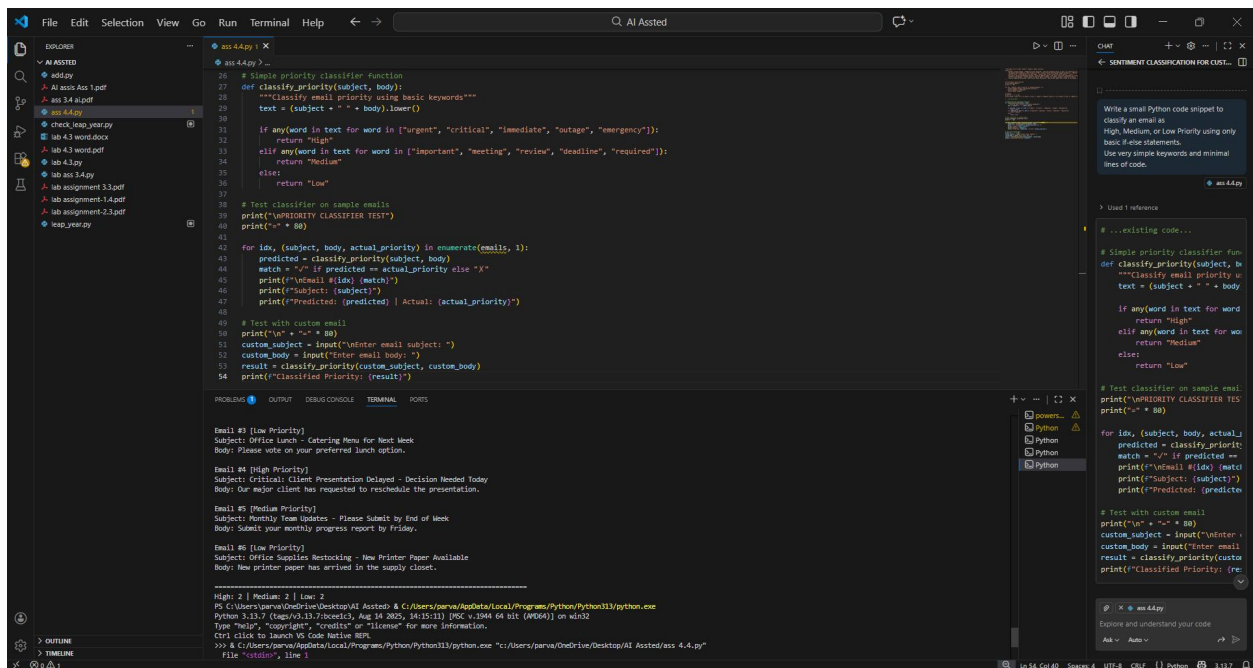
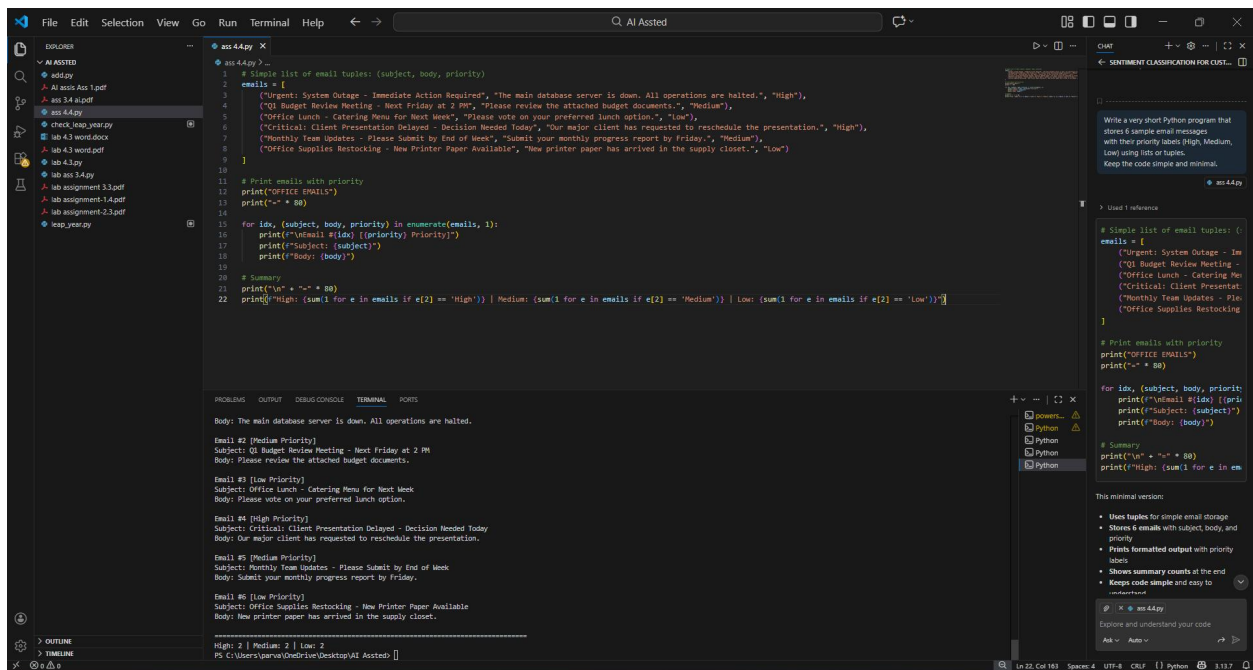
2. Email Priority Classification

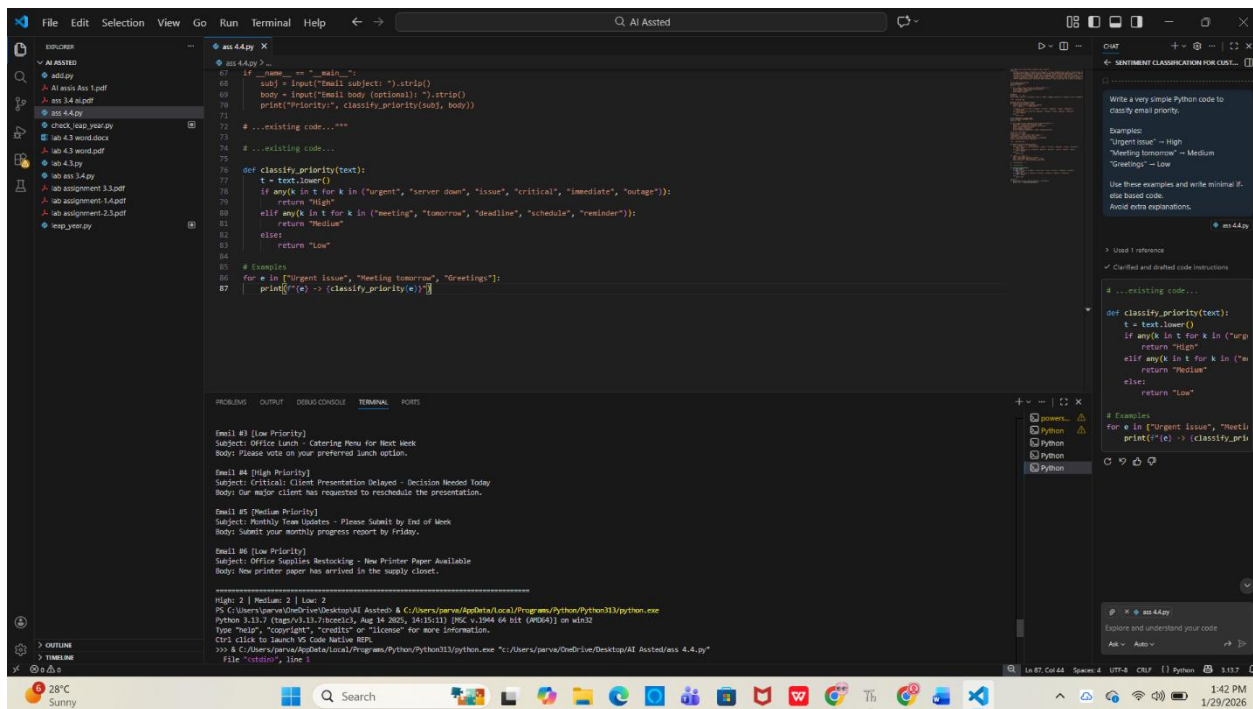
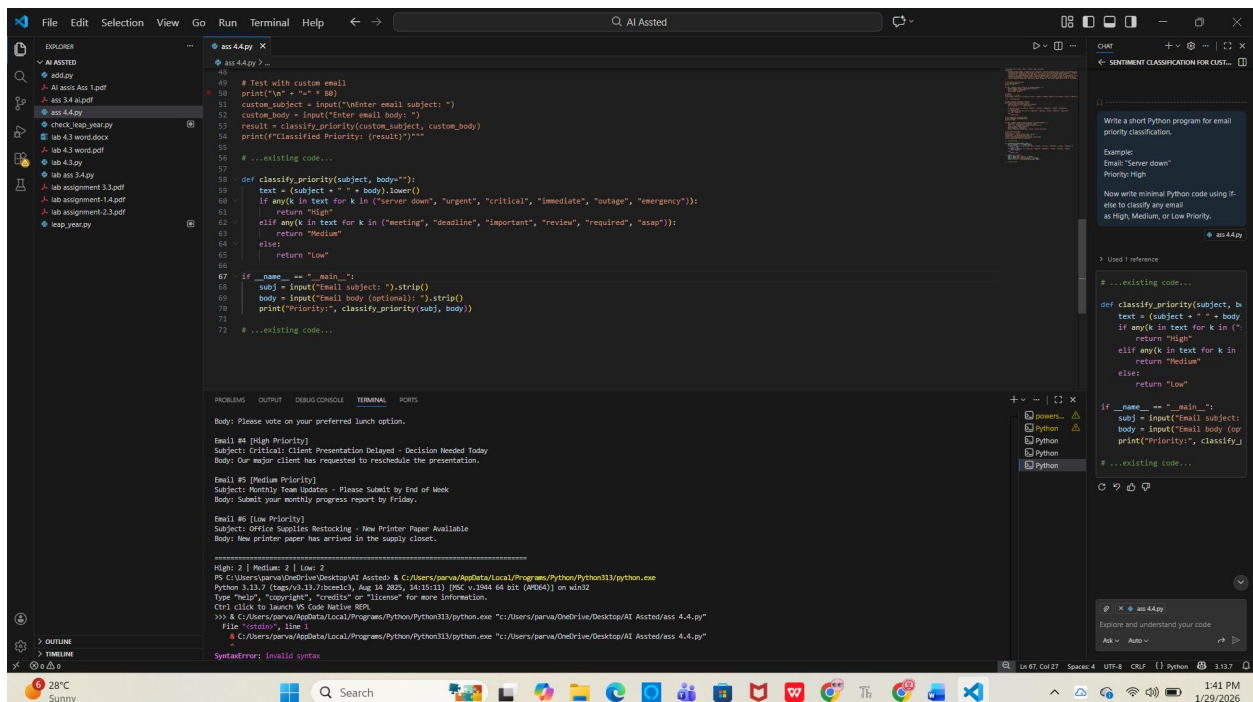
Scenario:

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.





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 VS Code editor with the following components:

- EXPLORER:** A file tree on the left showing a project named 'AI Assted' with files like 'add.py', 'AI ass 1.pdf', 'ass 3.4.pdf', 'ass 4.4.py', 'check_knap_year.py', 'lab 4.3 word.docx', 'lab 4.3 word.pdf', 'lab 4.3.py', 'lab ass 3.4.py', 'lab assignment 3.4.pdf', 'lab assignment 3.4.pdf', 'lab assignment 2.3.pdf', and 'knap_year.py'.
- EDITOR:** The main window displays the code for 'ass 4.4.py'. The code defines a list of student queries mapped to departments and a function to process them.
- TERMINAL:** The bottom panel shows the output of running the script. It displays several 'SyntaxError: invalid syntax' messages for the first four queries, followed by a successful execution for the last two queries, which prints the department names 'Academic' and 'Admissions'.
- CHAT:** A chat window on the right is titled 'SENTIMENT CLASSIFICATION FOR CUST...'. It contains a prompt asking for a Python program to classify student queries and their departments. The chat also shows a list of the queries and departments used in the script.


```
1 # ...existing code...
2 def classify_query(q):
3     t = q.lower()
4     if "exam" in t or "semester" in t:
5         return "Exams"
6     elif "course" in t or "syllabus" in t:
7         return "Academics"
8     elif "placement" in t or "campus" in t:
9         return "Placements"
10    elif "admission" in t or "apply" in t:
11        return "Admissions"
12    return "Academics"
13
14 if __name__ == "__main__":
15     q = input().strip()
16     print("Department:", classify_query(q))
17 # ...existing code...
```

```
53 ("Course syllabus details", "Academics"),
54 ("Campus placement updates", "Placements"),
55 ("How to apply for admission", "Admissions"),
56 ("What is the admission process", "Admissions"),
57 ("Are finals scheduled next week", "Exams"),
58 ("Internship opportunities", "Placements"),
59 ("Change of course procedure", "Academics"),
60 ]
61
62 def zero_shot(q):
63     t = q.lower()
64     if "exam" in t or "semester" in t: return "Exams"
65     if "course" in t or "syllabus" in t: return "Academics"
66     if "placement" in t or "internship" in t or "campus" in t: return "Placements"
67     if "admission" in t or "apply" in t: return "Admissions"
68     return "Academics"
69
70 one_example = "admission process"
71 def one_shot(q):
72     t = q.lower()
73     if one_example in t: return "Admissions"
74     return zero_shot(q)
75
76 few_examples = [
77     "admission process": "Admissions",
78     "how to apply for admission": "Admissions",
79     "semester exam": "Exams",
80     "course syllabus": "Academics",
81     "placement updates": "Placements",
82     "internship opportunities": "Placements",
83 ]
84
85 def few_shot(q):
86     t = q.lower()
87     for k,v in few_examples.items():
88         if k in t: return v
89     return zero_shot(q)
90
91 def score(q):
92     return sum(1 for q,a in tests if fn(q)==a)/len(tests)
93
94 s0, s1, s2 = score(zero_shot), score(one_shot), score(few_shot)
95 best_name, best_score = max([(zero_shot,s0), (one_shot,s1), (few_shot,s2)], key=lambda x:x[1])
96 print("Best name:", best, "(best_score=%f%% accuracy)"
```

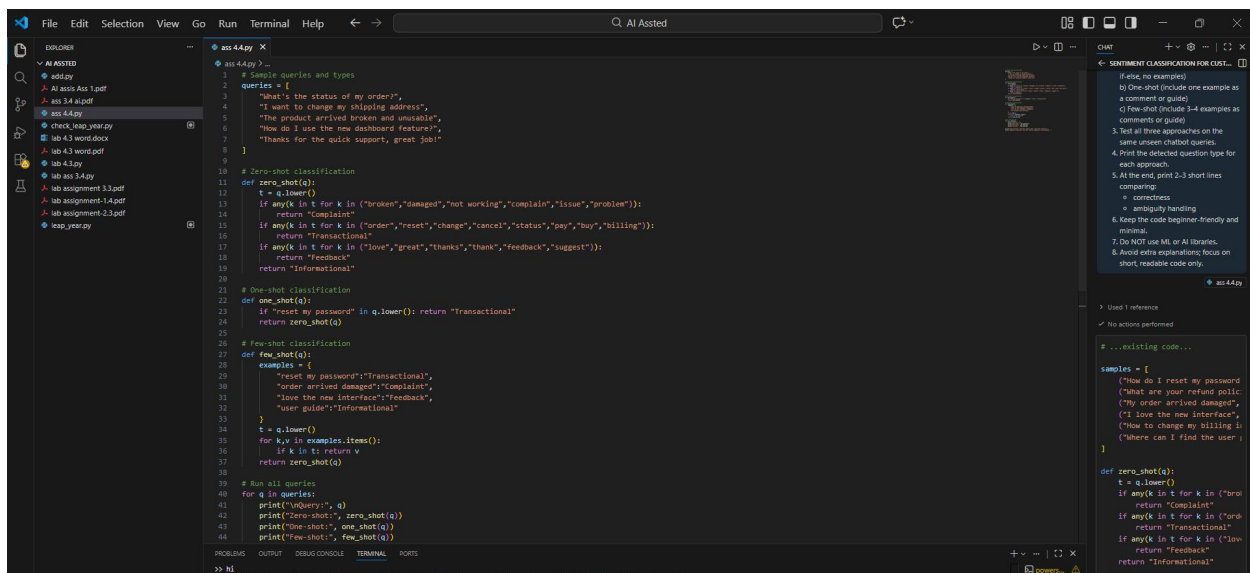
4. Chatbot Question Type Detection

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.



```
1 # Small queries and types
2 queries = [
3     "What's the status of my order?",
4     "I want to change my shipping address",
5     "The product arrived broken and unusable",
6     "How do I use the new dashboard feature?",
7     "Thanks for the quick support, great job!"
8 ]
9
10 # Zero-shot classification
11 def zero_shot(a):
12     t = q_lower()
13     if any(k in t for k in ("broken", "damaged", "not working", "complain", "issue", "problem")):
14         return "Complaint"
15     if any(k in t for k in ("order", "reset", "change", "cancel", "status", "pay", "buy", "billing")):
16         return "Transactional"
17     if any(k in t for k in ("love", "great", "thanks", "thank", "feedback", "suggest")):
18         return "Feedback"
19     return "Informational"
20
21 # One-shot classification
22 def one_shot(a):
23     if "reset my password" in q_lower(): return "Transactional"
24     return zero_shot(a)
25
26 # Few-shot classification
27 def few_shot(a):
28     examples = [
29         "reset my password": "Transactional",
30         "order arrived damaged": "complaint",
31         "love the new interface": "feedback",
32         "user guide": "Informational"
33     ]
34     t = q_lower()
35     for k, v in examples.items():
36         if k in t: return v
37     return zero_shot(a)
38
39 # Run all queries
40 for q in queries:
41     print("Query:", q)
42     print("Zero-shot:", zero_shot(a))
43     print("One-shot:", one_shot(a))
44     print("Few-shot:", few_shot(a))
45
46 # ...existing code...
47
48 def zero_shot(a):
49     t = q_lower()
50     if any(k in t for k in ("bro",
51                             "return "Complaint"
52     if any(k in t for k in ("ord",
53                             "return "Transactional"
54     if any(k in t for k in ("love",
55                             "return "Feedback"
56     return "Informational"
```

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.

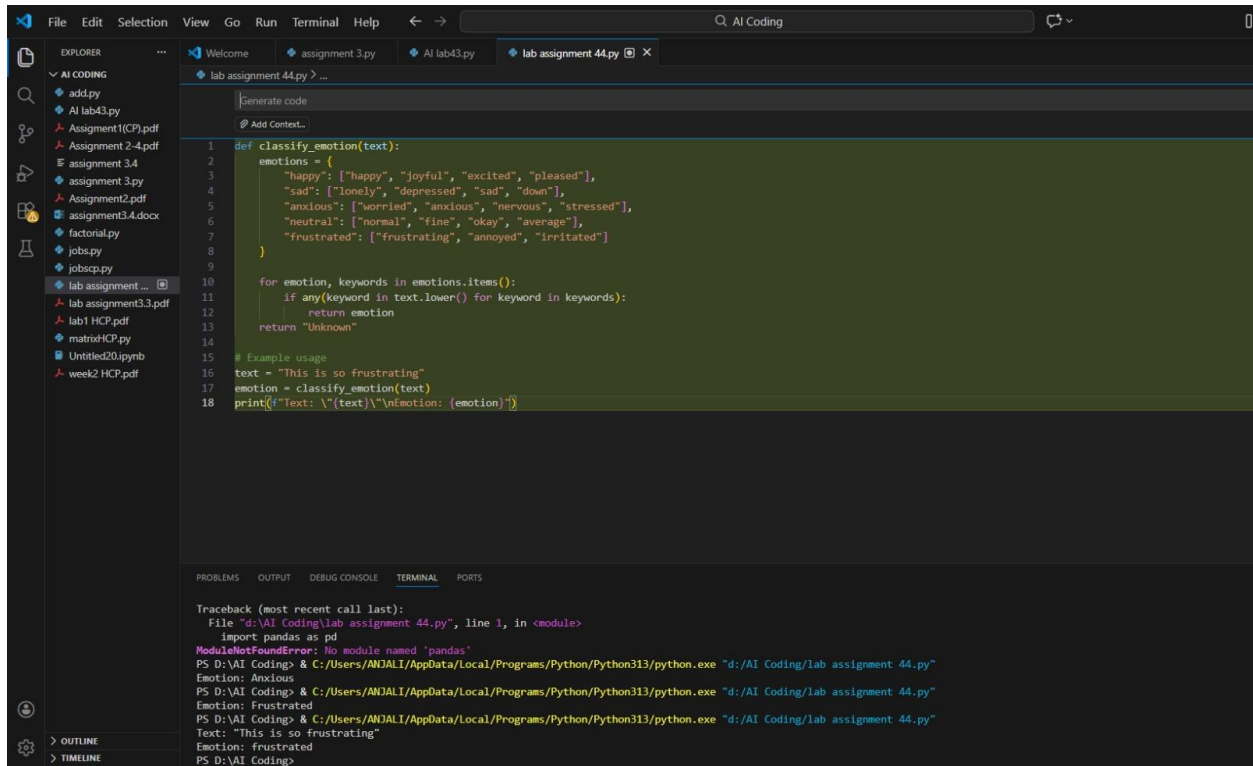
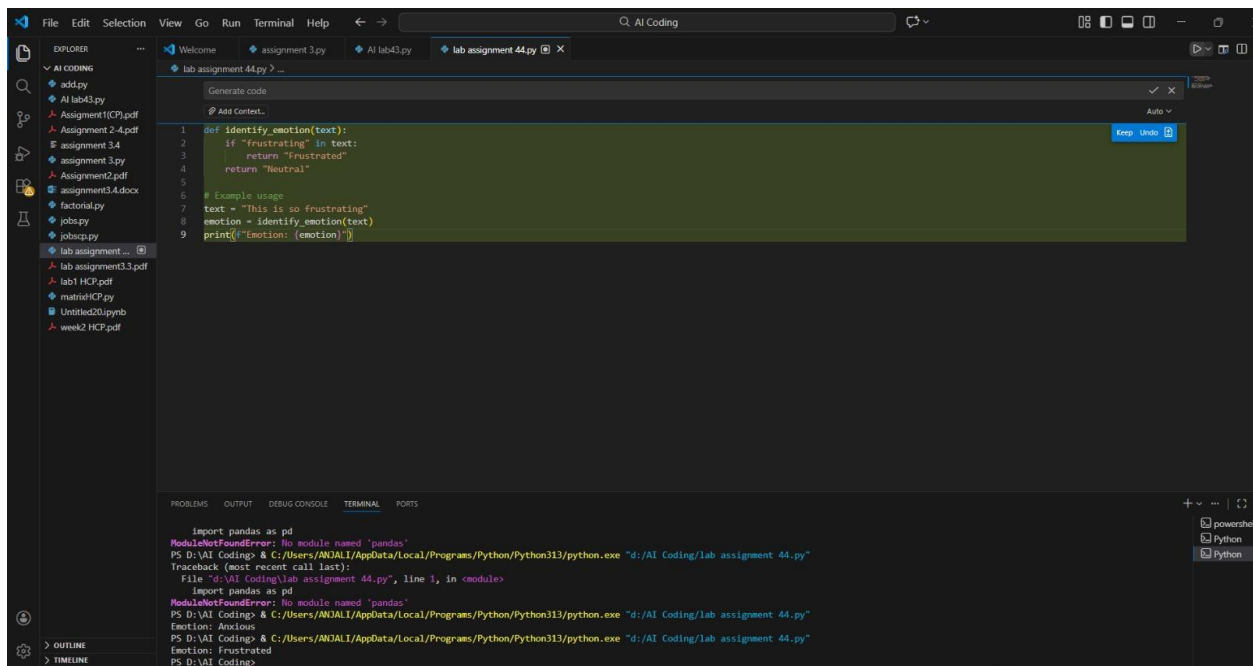
The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files under 'AI CODING', including 'add.py', 'AI lab43.py', and 'lab assignment 44.py'. The main editor displays the code for 'lab assignment 44.py'. The code is as follows:

```
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)
```

Below the code editor, the TERMINAL tab is active, showing the command prompt output:

```
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>
```

This screenshot is a duplicate of the one above, showing the same Visual Studio Code interface with the 'lab assignment 44.py' file open. The code and the terminal output, which displays a 'ModuleNotFoundError: No module named 'pandas'', are identical to the first image.



File Edit Selection View Go Run Terminal Help

AI Coding

EXPLORER

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

jobs.py

lab assignment ...

lab assignment3.3.pdf

lab1 HCP.pdf

matrixHCP.py

Untitled20.ipynb

week2 HCP.pdf

lab assignment 44.py

lab assignment 44.py

lab assignment 44.py

Generate code

Add Context...

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}")

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"

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>