2303A51744

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:

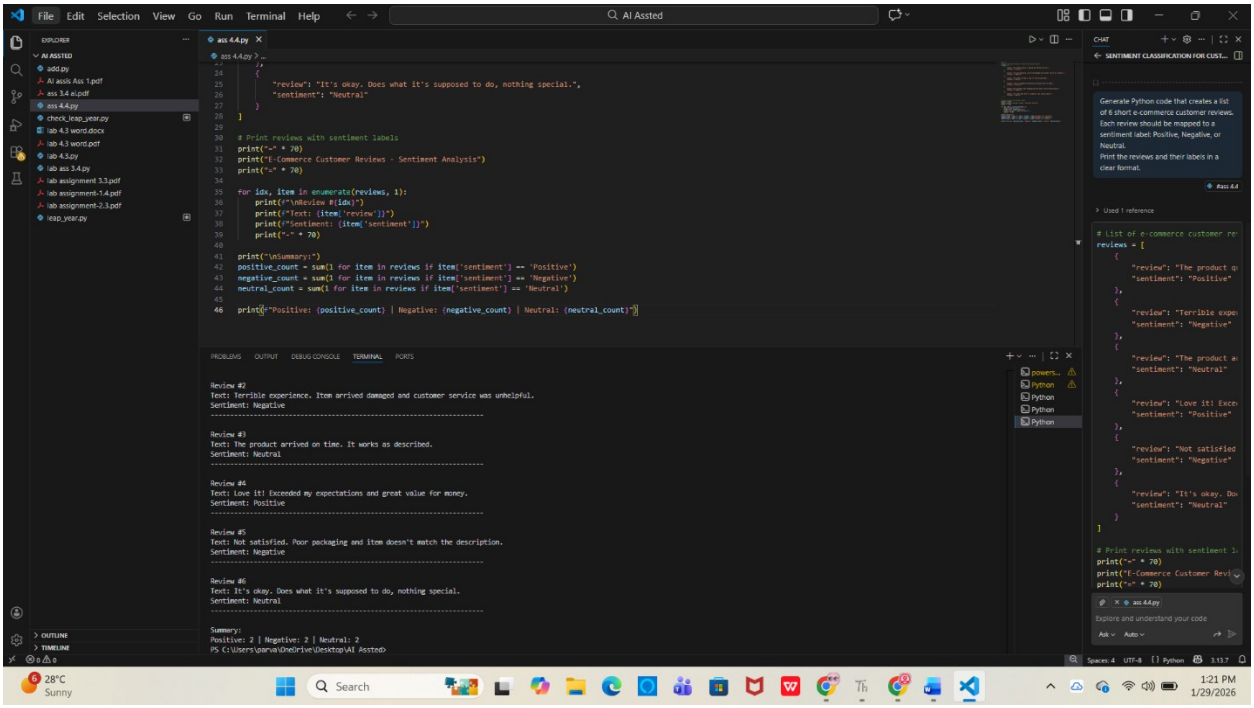
a) Prepare 6 short customer reviews mapped to sentiment labels.

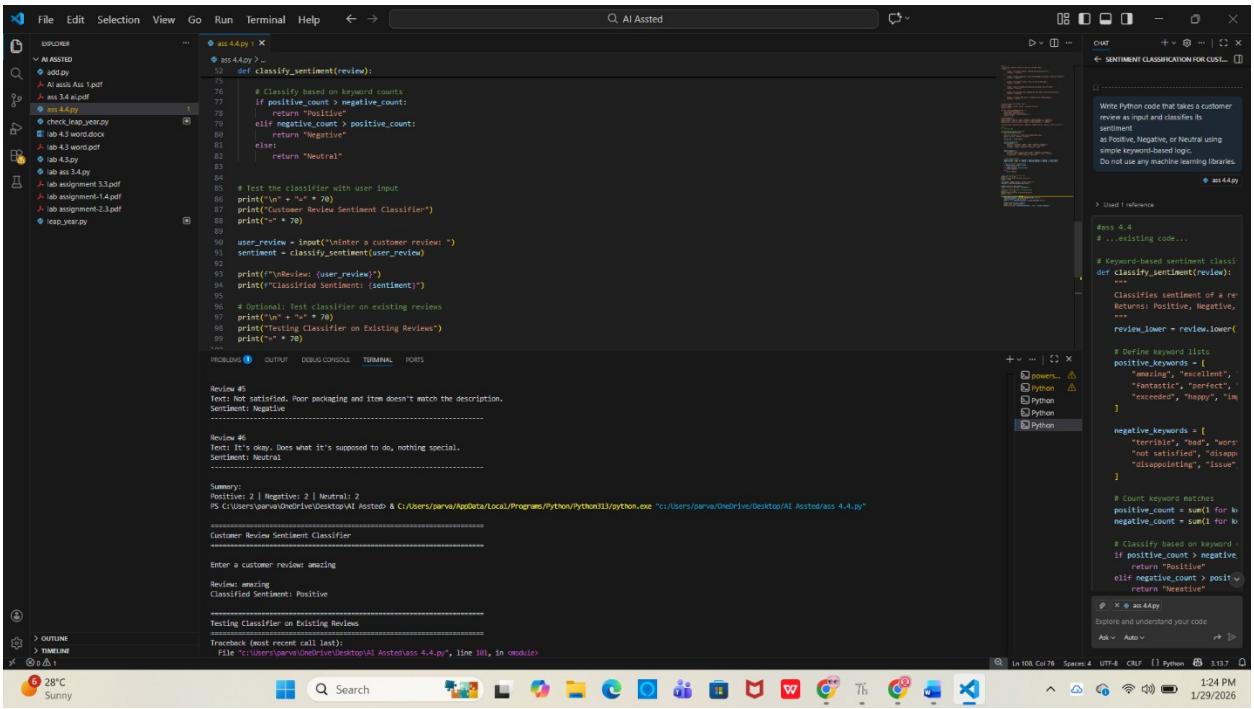
b) Design a Zero-shot prompt to classify sentiment.

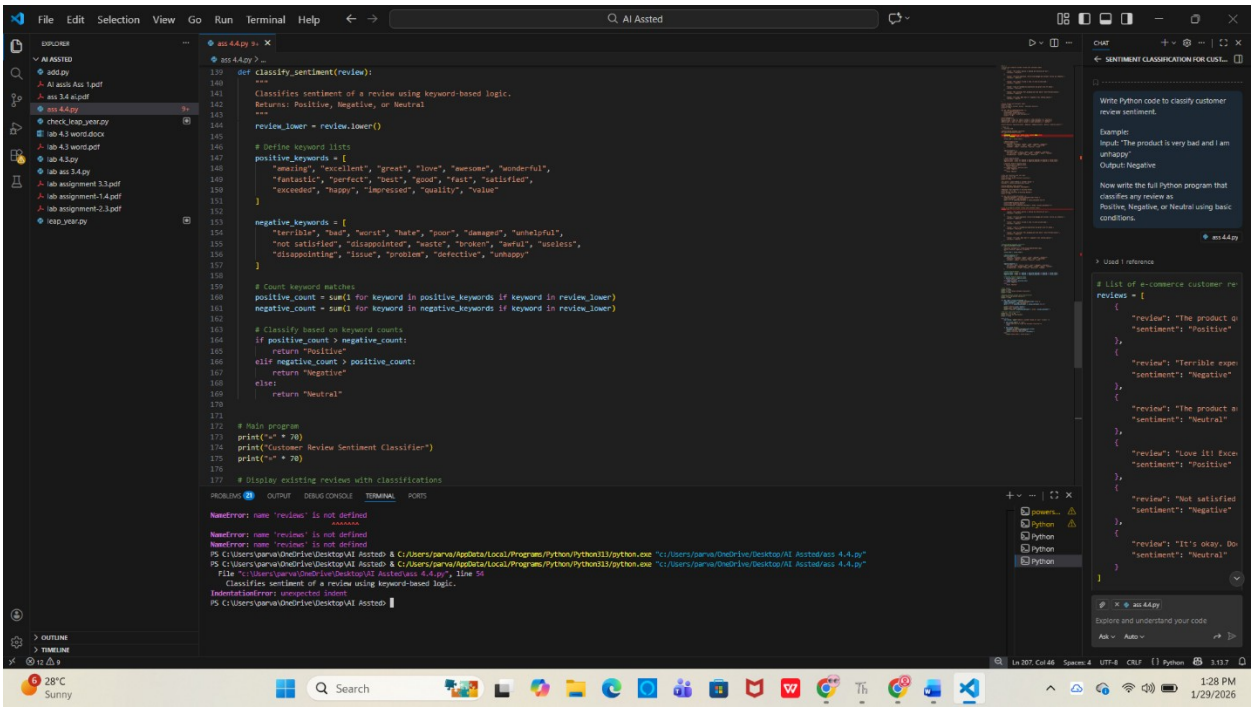
c) Design a One-shot prompt with one labeled example.

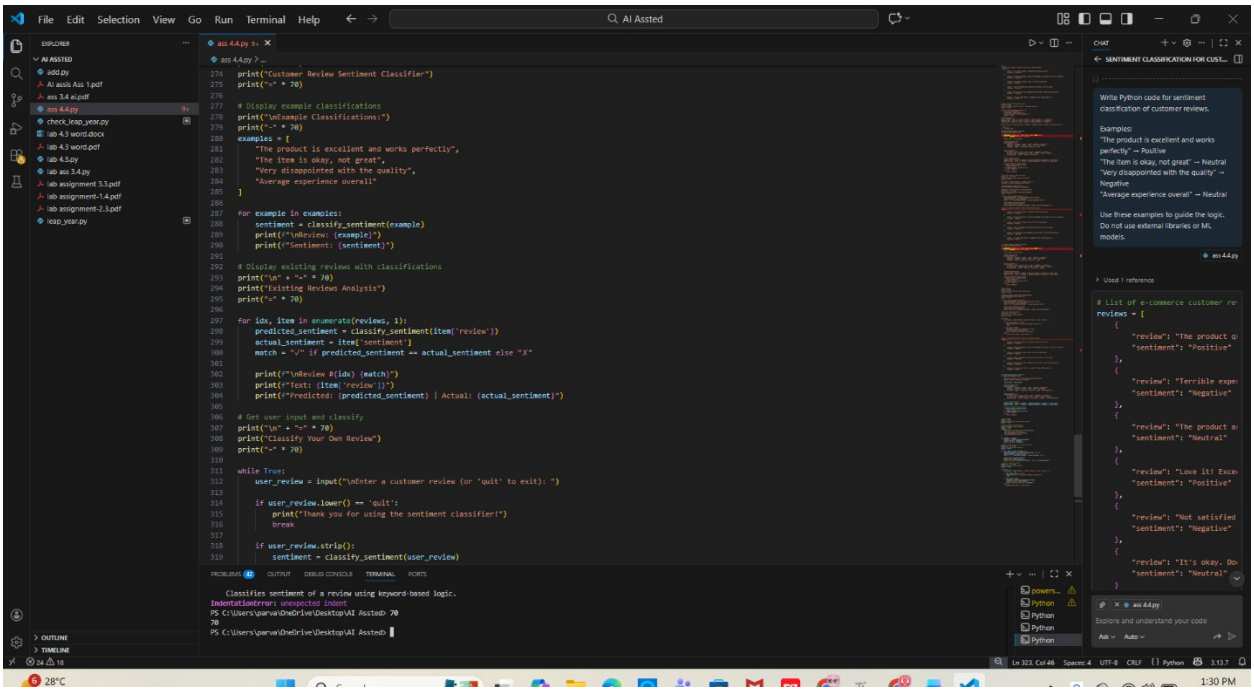
d) Design a Few-shot prompt with 3–5 labeled examples.

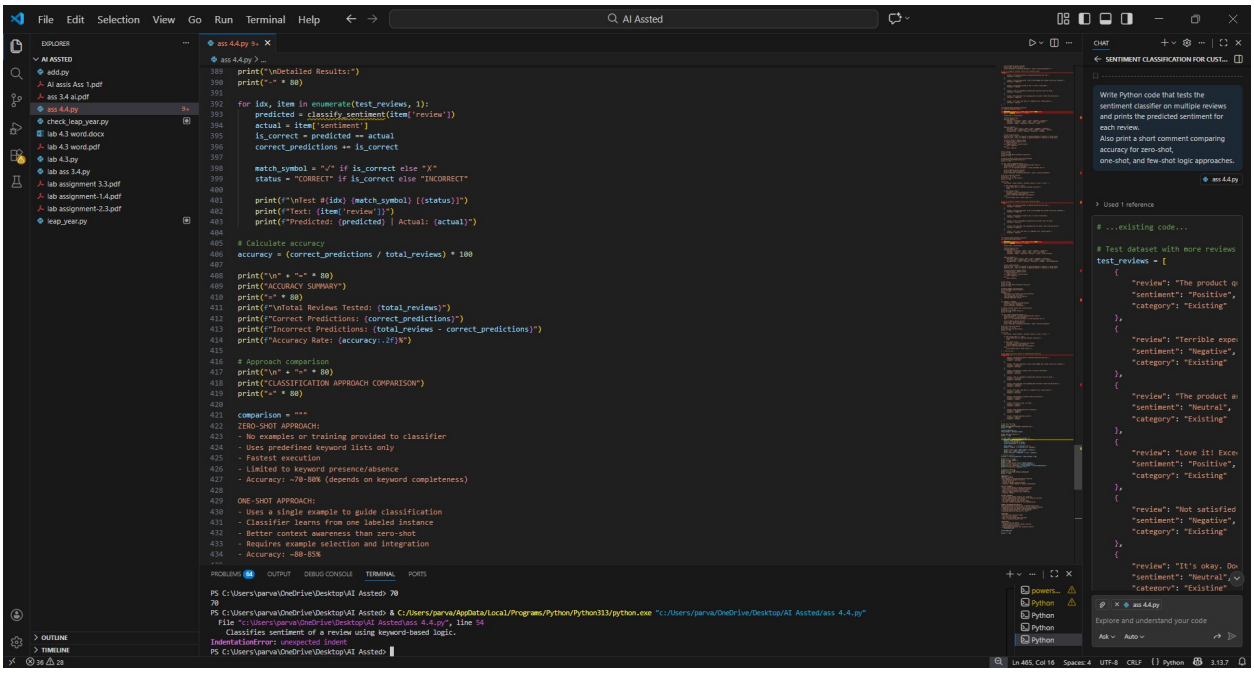
e) Compare the outputs and discuss accuracy differences.


File Edit Selection View Go Run Terminal Help
AI Assted
ass 4.4.py
Review #2
Text: Terrible experience. Item arrived damaged and customer service was unhelpful.
Sentiment: Negative
Review #3
Text: The product arrived on time. It works as described.
Sentiment: Neutral
Review #4
Text: Love it! Exceeded my expectations and great value for money.
Sentiment: Positive
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
Summary:
Positive: 2 | Negative: 2 | Neutral: 2
PS C:\Users\parva\OneDrive\Desktop\AI Assted>
28°C
Sunny
Search
1:21 PM
1/29/2026


File Edit Selection View Go Run Terminal Help
AI Assted
ass 4.4.py
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
Summary:
Positive: 2 | Negative: 2 | Neutral: 2
PS C:\Users\parva\OneDrive\Desktop\AI Assted> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass 4.4.py"
Customer Review Sentiment Classifier
Enter a customer review: amazing
Review: amazing
Classified Sentiment: Positive
Testing Classifier on Existing Reviews
Traceback (most recent call last):
File "c:\Users\parva\OneDrive\Desktop\AI Assted\ass 4.4.py", line 101, in <module>
28°C
Sunny
Search
1:24 PM
1/29/2026


File Edit Selection View Go Run Terminal Help
AI Assted
ass 4.4.py
def classify_sentiment(review):
"""
Classifies sentiment of a review using keyword-based logic.
Returns: Positive, Negative, or Neutral
"""
review_lower = review.lower()
# Define keyword lists
positive_keywords = [
"amazing", "excellent", "great", "love", "awesome", "wonderful",
"fantastic", "perfect", "best", "good", "fast", "satisfied",
"exceeded", "happy", "impressed", "quality", "value"
]
negative_keywords = [
"terrible", "bad", "worst", "hate", "poor", "damaged", "unhelpful",
"not satisfied", "disappointed", "waste", "broken", "awful", "useless",
"disappointing", "issue", "problem", "defective", "unhappy"
]
# Count keyword matches
positive_count = sum(1 for keyword in positive_keywords if keyword in review_lower)
negative_count = sum(1 for keyword in negative_keywords if keyword in review_lower)
# Classify based on keyword counts
if positive_count > negative_count:
return "Positive"
elif negative_count > positive_count:
return "Negative"
else:
return "Neutral"
# Main program
print("=" * 70)
print("Customer Review Sentiment Classifier")
print("=" * 70)
# Display existing reviews with classifications
PROBLEMS 21 OUTPUT DEBUG CONSOLE TERMINAL PORTS
NameError: name 'reviews' is not defined
NameError: name 'reviews' is not defined
NameError: name 'reviews' is not defined
PS C:\Users\parva\OneDrive\Desktop\AI Assted> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass 4.4.py"
PS C:\Users\parva\OneDrive\Desktop\AI Assted> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass 4.4.py"
File "c:\Users\parva\OneDrive\Desktop\AI Assted\ass 4.4.py", line 54
Classifies sentiment of a review using keyword-based logic.
IndentationError: unexpected indent
PS C:\Users\parva\OneDrive\Desktop\AI Assted>
CHAT
SENTIMENT CLASSIFICATION FOR CUST...
Write Python code to classify customer review sentiment.
Example:
Input: "The product is very bad and I am unhappy"
Output: Negative
Now write the full Python program that classifies any review as Positive, Negative, or Neutral using basic conditions.
ass 4.4.py
Used 1 reference
# List of e-commerce customer re
reviews = [
{
"review": "The product q
"sentiment": "Positive"
},
{
"review": "Terrible expe
"sentiment": "Negative"
},
{
"review": "The product a
"sentiment": "Neutral"
},
{
"review": "Love it! Exce
"sentiment": "Positive"
},
{
"review": "Not satisfied
"sentiment": "Negative"
},
{
"review": "It's okay. Do
"sentiment": "Neutral"
}
]
Explore and understand your code
Ask Auto
28°C Sunny
Search
1:28 PM 1/29/2026


File Edit Selection View Go Run Terminal Help
AI Assted
ass 4.4.py
print("Customer Review Sentiment Classifier")
print("=" * 70)
# Display example classifications
print("\nExample Classifications:")
print("-" * 70)
examples = [
"The product is excellent and works perfectly",
"The item is okay, not great",
"Very disappointed with the quality",
"Average experience overall"
]
for example in examples:
sentiment = classify_sentiment(example)
print(f"\nReview: {example}")
print(f"Sentiment: {sentiment}")
# Display existing reviews with classifications
print("\n" + "=" * 70)
print("Existing Reviews Analysis")
print("=" * 70)
for idx, item in enumerate(reviews, 1):
predicted_sentiment = classify_sentiment(item['review'])
actual_sentiment = item['sentiment']
match = "✓" if predicted_sentiment == actual_sentiment else "✗"
print(f"\nReview #{idx} {match}")
print(f"Text: {item['review'][:]}")
print(f"Predicted: {predicted_sentiment} | Actual: {actual_sentiment}")
# Get user input and classify
print("\n" + "=" * 70)
print("Classify Your Own Review")
print("=" * 70)
while True:
user_review = input("\nEnter a customer review (or 'quit' to exit): ")
if user_review.lower() == 'quit':
print("Thank you for using the sentiment classifier!")
break
if user_review.strip():
sentiment = classify_sentiment(user_review)
PROBLEMS 42 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Classifies sentiment of a review using keyword-based logic.
IndentationError: unexpected indent
PS C:\Users\parva\OneDrive\Desktop\AI Assted> 70
70
PS C:\Users\parva\OneDrive\Desktop\AI Assted>
CHAT
SENTIMENT CLASSIFICATION FOR CUST...
Write Python code for sentiment classification of customer reviews.
Examples:
"The product is excellent and works perfectly" → Positive
"The item is okay, not great" → Neutral
"Very disappointed with the quality" → Negative
"Average experience overall" → Neutral
Use these examples to guide the logic. Do not use external libraries or ML models.
ass 4.4.py
Used 1 reference
# List of e-commerce customer re
reviews = [
{
"review": "The product q
"sentiment": "Positive"
},
{
"review": "Terrible expe
"sentiment": "Negative"
},
{
"review": "The product a
"sentiment": "Neutral"
},
{
"review": "Love it! Exce
"sentiment": "Positive"
},
{
"review": "Not satisfied
"sentiment": "Negative"
},
{
"review": "It's okay. Do
"sentiment": "Neutral"
}
]
Explore and understand your code
Ask Auto
28°C
1:30 PM



2. Email Priority Classification

Scenario:

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

Tasks:

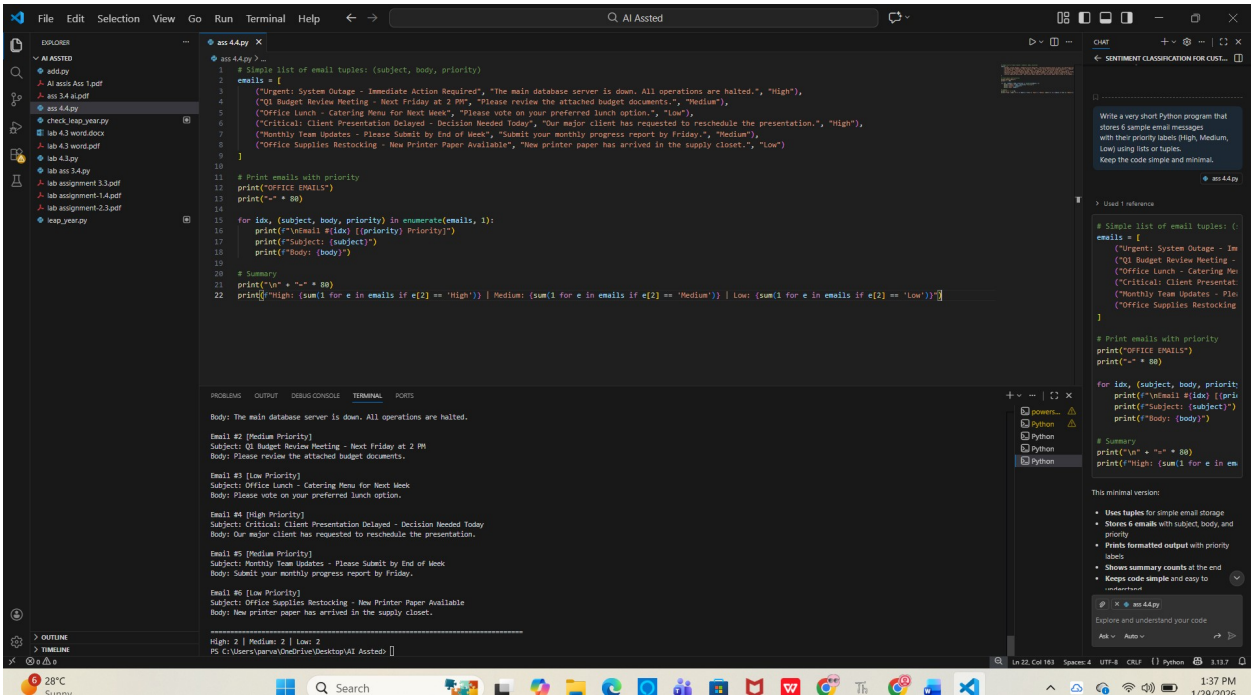
1. Create 6 sample email messages with priority labels.

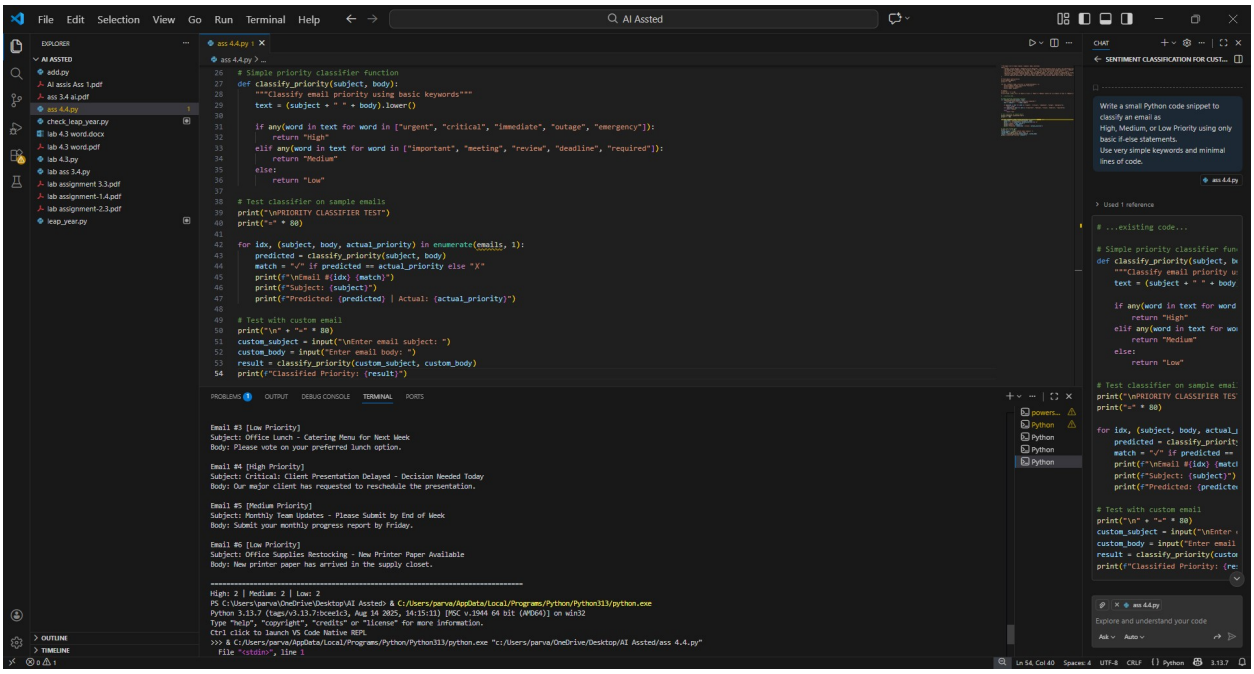
2. Perform intent classification using Zero-shot prompting.

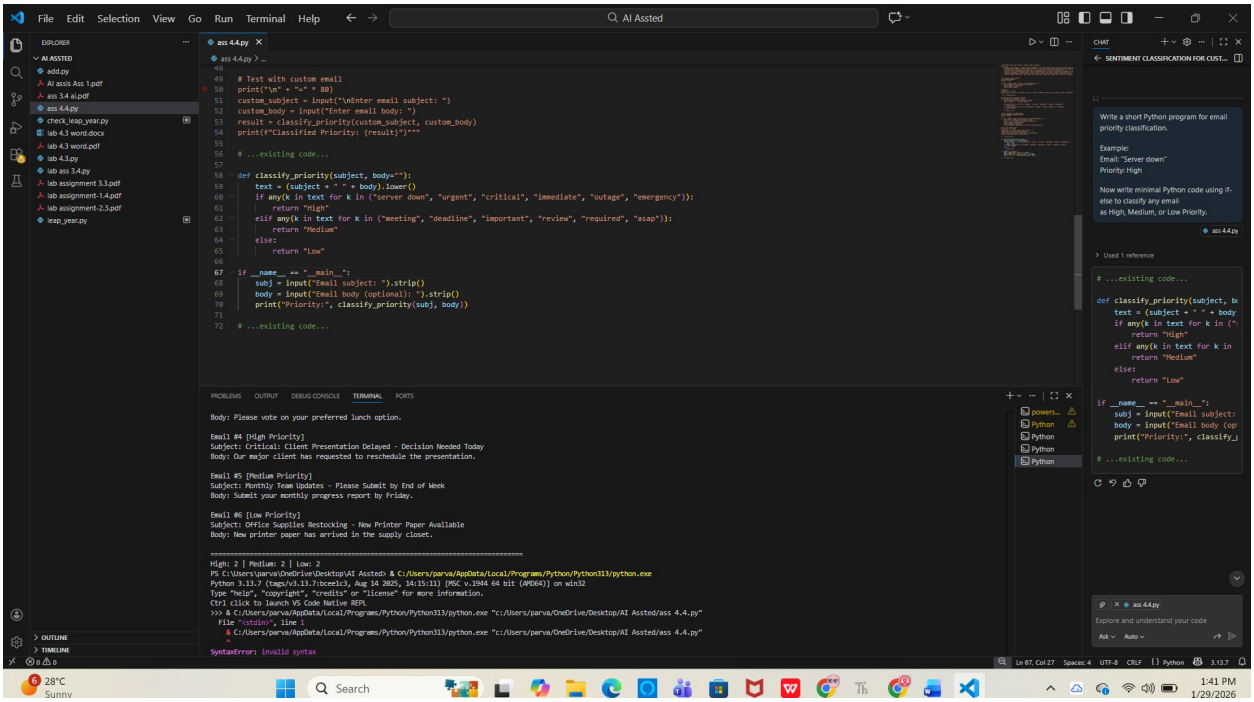
3. Perform classification using One-shot prompting.

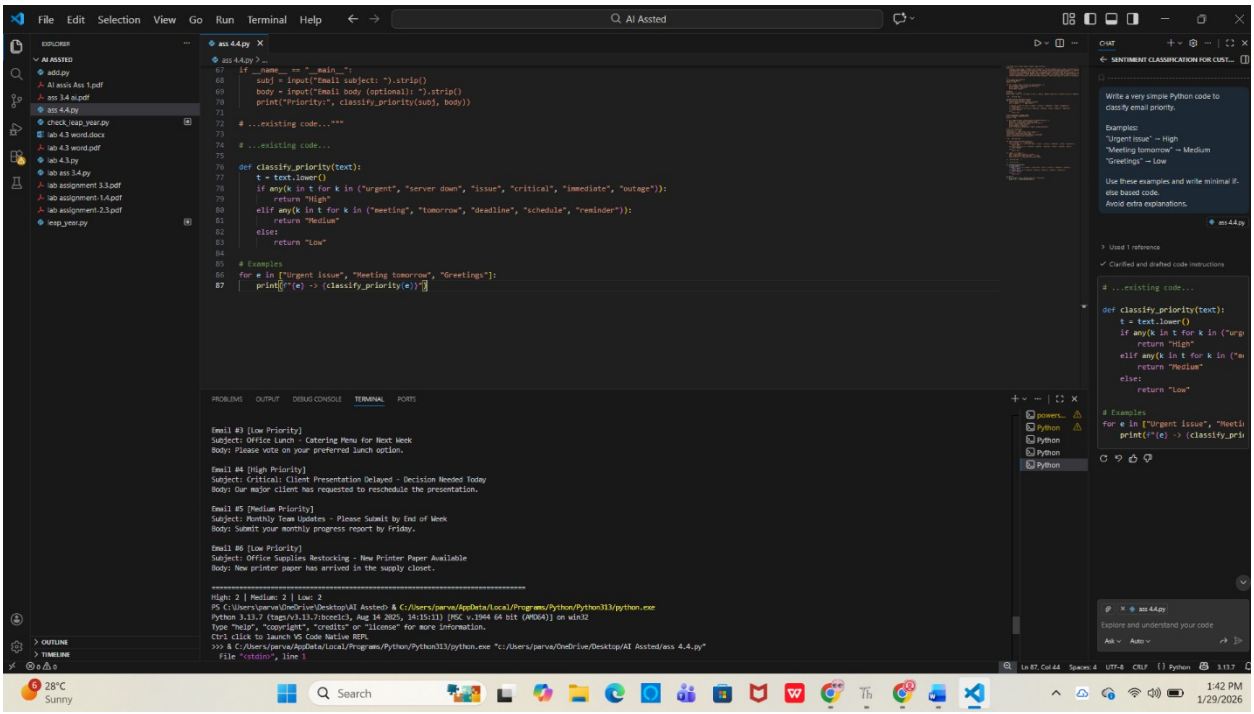
4. Perform classification using Few-shot prompting.

5. Evaluate which technique produces the most reliable results and why.


File Edit Selection View Go Run Terminal Help
AI Assted
ass 4.4.py
# Simple list of email tuples: (subject, body, priority)
emails = [
("Urgent: System Outage - Immediate Action Required", "The main database server is down. All operations are halted.", "High"),
("Q1 Budget Review Meeting - Next Friday at 2 PM", "Please review the attached budget documents.", "Medium"),
("Office Lunch - Catering Menu for Next Week", "Please vote on your preferred lunch option.", "Low"),
("Critical: Client Presentation Delayed - Decision Needed Today", "Our major client has requested to reschedule the presentation.", "High"),
("Monthly Team Updates - Please Submit by End of Week", "Submit 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(f"\nEmail #{idx} [{priority} Priority]")
print(f"Subject: {subject}")
print(f"Body: {body}")
# Summary
print("\n" + "=" * 80)
print(f"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')}")
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\OneDrive\Desktop\AI Assted>
Write a very short Python program that stores 6 sample email messages with their priority labels (High, Medium, Low) using lists or tuples.
Keep the code simple and minimal.
This minimal version:
Uses tuples for simple email storage
Stores 6 emails with subject, body, and priority
Prints formatted output with priority labels
Shows summary counts at the end
Keeps code simple and easy to
Explore and understand your code
1:37 PM


File Edit Selection View Go Run Terminal Help
AI Assted
ass 4.4.py
# Simple priority classifier function
def classify_priority(subject, body):
"""Classify email priority using basic keywords"""
text = (subject + " " + body).lower()
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("\nPRIORITY CLASSIFIER TEST")
print("=" * 80)
for idx, (subject, body, actual_priority) in enumerate(emails, 1):
predicted = classify_priority(subject, body)
match = "✓" if predicted == actual_priority else "X"
print(f"\nEmail #{idx} {match}")
print(f"Subject: {subject}")
print(f"Predicted: {predicted} | Actual: {actual_priority}")
# Test with custom email
print("\n" + "=" * 80)
custom_subject = input("\nEnter email subject: ")
custom_body = input("Enter email body: ")
result = classify_priority(custom_subject, custom_body)
print(f"Classified Priority: {result}")
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\OneDrive\Desktop\AI Assted> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
Ctrl-click to launch VS Code Native REPL
>>> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass 4.4.py"
File "<stdin>", line 1
Write a small Python code snippet to classify an email as High, Medium, or Low Priority using only basic if-else statements.
Use very simple keywords and minimal lines of code.
Explore and understand your code





3. Student Query Routing System

Scenario:

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

Academics, or Placements.

Tasks:

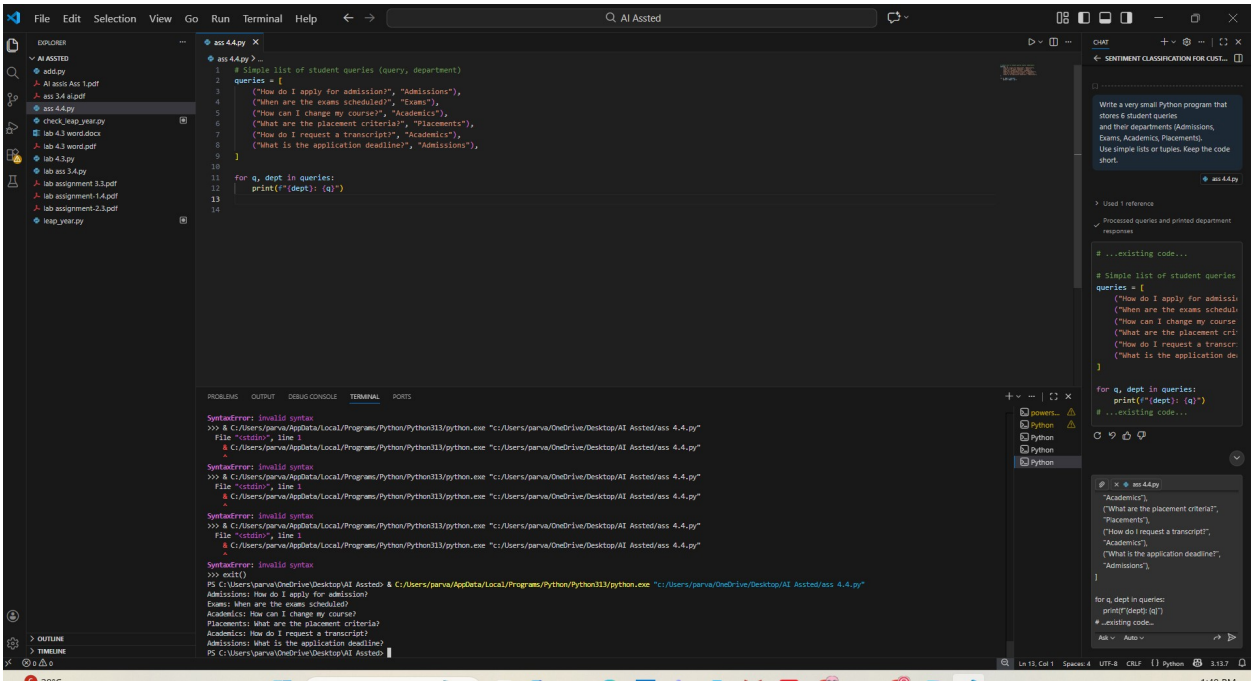
1. Create 6 sample student queries mapped to departments.

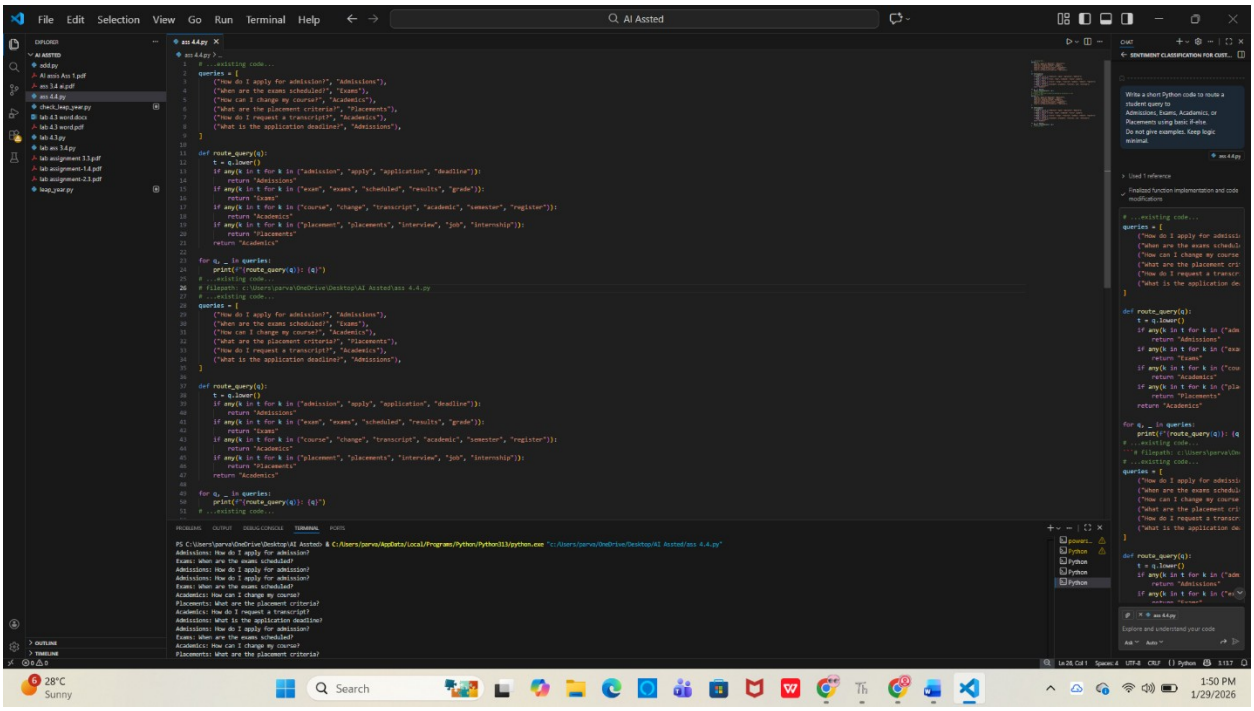
2. Implement Zero-shot intent classification using an LLM.

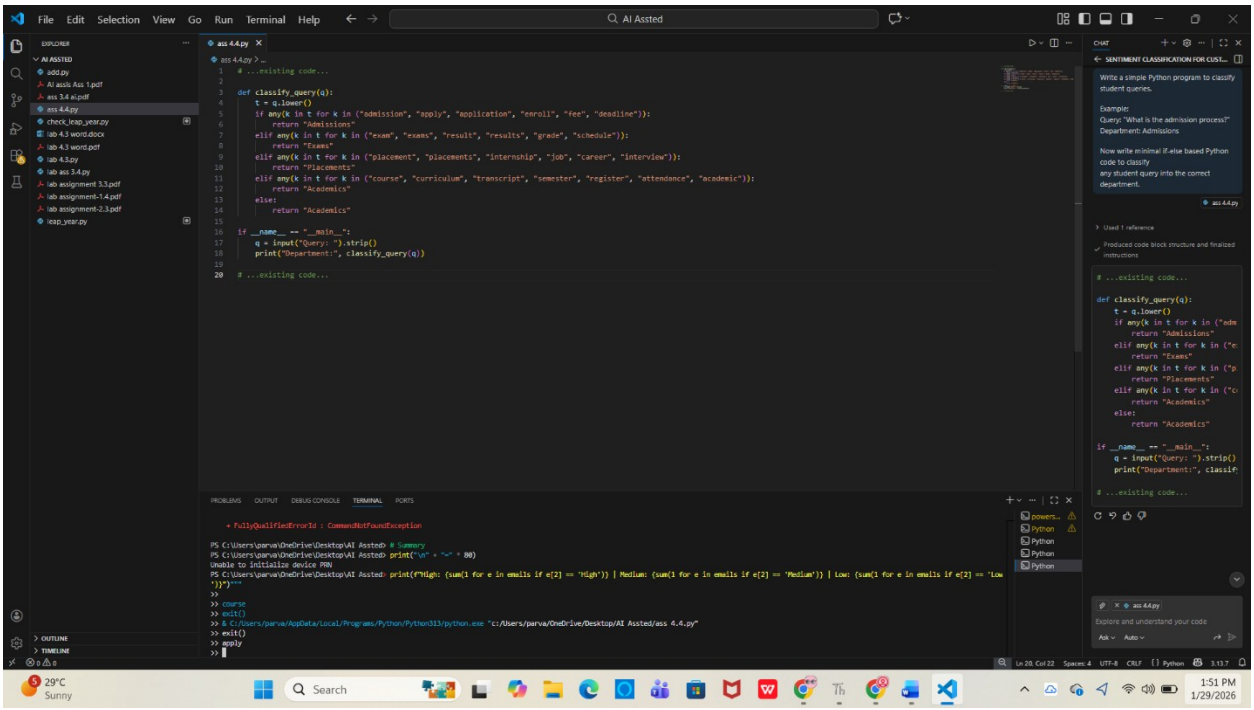
3. Improve results using One-shot prompting.

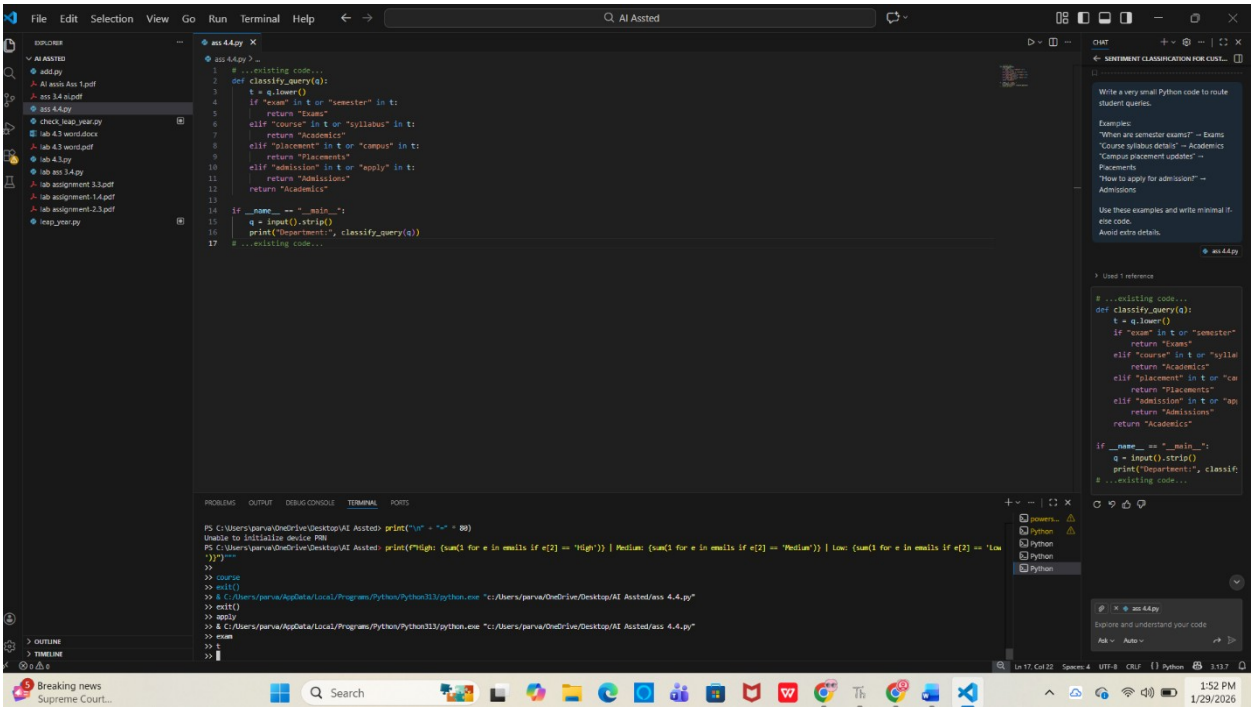
4. Further refine results using Few-shot prompting.

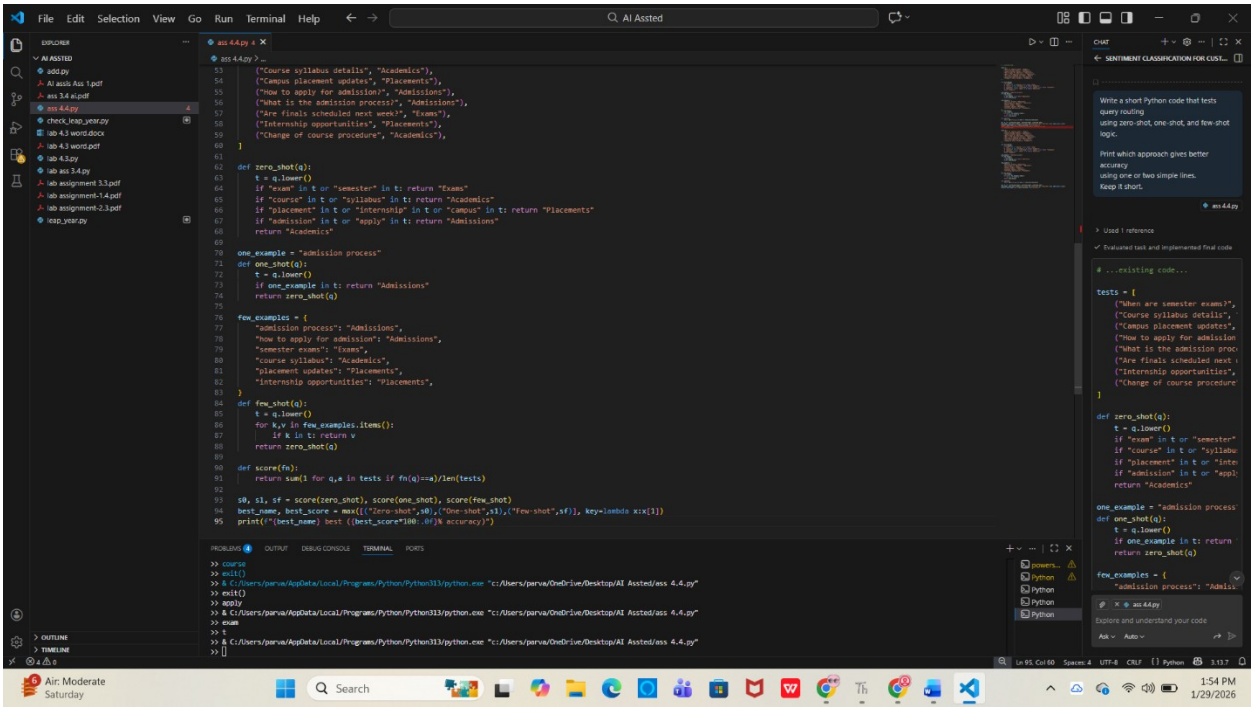
5. Analyze how contextual examples affect classification accuracy.




File Edit Selection View Go Run Terminal Help
AI Assted
ass 4.4.py
# ...existing code...
queries = [
("How do I apply for admission?", "Admissions"),
("When are the exams scheduled?", "Exams"),
("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"),
]
def route_query(q):
t = q.lower()
if any(k in t for k in ("admission", "apply", "application", "deadline")):
return "Admissions"
if any(k in t for k in ("exam", "exams", "scheduled", "results", "grade")):
return "Exams"
if any(k in t for k in ("course", "change", "transcript", "academic", "semester", "register")):
return "Academics"
if any(k in t for k in ("placement", "placements", "interview", "job", "internship")):
return "Placements"
return "Academics"
for q, _ in queries:
print(f"{route_query(q)}: {q}")
# ...existing code...
# Filepath: c:\Users\parva\OneDrive\Desktop\AI Assted\ass 4.4.py
# ...existing code...
queries = [
("How do I apply for admission?", "Admissions"),
("When are the exams scheduled?", "Exams"),
("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"),
]
def route_query(q):
t = q.lower()
if any(k in t for k in ("admission", "apply", "application", "deadline")):
return "Admissions"
if any(k in t for k in ("exam", "exams", "scheduled", "results", "grade")):
return "Exams"
if any(k in t for k in ("course", "change", "transcript", "academic", "semester", "register")):
return "Academics"
if any(k in t for k in ("placement", "placements", "interview", "job", "internship")):
return "Placements"
return "Academics"
for q, _ in queries:
print(f"{route_query(q)}: {q}")
# ...existing code...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\parva\OneDrive\Desktop\AI Assted> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass 4.4.py"
Admissions: How do I apply for admission?
Exams: When are the exams scheduled?
Admissions: How do I apply for admission?
Admissions: How do I apply for admission?
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?
Admissions: How do I apply for admission?
Exams: When are the exams scheduled?
Academics: How can I change my course?
Placements: What are the placement criteria?
CHAT
SENTIMENT CLASSIFICATION FOR CUST...
Write a short Python code to route a student query to Admissions, Exams, Academics, or Placements using basic if-else. Do not give examples. Keep logic minimal.
Used 1 reference
Finalized function implementation and code modifications
Explore and understand your code
28°C
Sunny
Search
1:50 PM
1/29/2026


File Edit Selection View Go Run Terminal Help
AI Assted
ass 4.4.py
# ...existing code...
def classify_query(q):
t = q.lower()
if any(k in t for k in ("admission", "apply", "application", "enroll", "fee", "deadline")):
return "Admissions"
elif any(k in t for k in ("exam", "exams", "result", "results", "grade", "schedule")):
return "Exams"
elif any(k in t for k in ("placement", "placements", "internship", "job", "career", "interview")):
return "Placements"
elif any(k in t for k in ("course", "curriculum", "transcript", "semester", "register", "attendance", "academic")):
return "Academics"
else:
return "Academics"
if __name__ == "__main__":
q = input("Query: ").strip()
print("Department:", classify_query(q))
# ...existing code...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+ FullyQualifiedErrorId : CommandNotFoundException
PS C:\Users\parva\OneDrive\Desktop\AI Assted> # Summary
PS C:\Users\parva\OneDrive\Desktop\AI Assted> print("\n" + "=" * 80)
Unable to initialize device PRN
PS C:\Users\parva\OneDrive\Desktop\AI Assted> print(f"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
')}")***
>>
>> course
>> exit()
>> & C:/Users/parva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/parva/OneDrive/Desktop/AI Assted/ass 4.4.py"
>> exit()
>> apply
>>
CHAT
SENTIMENT CLASSIFICATION FOR CUST...
Write a simple Python program to classify student queries.
Examples:
Query: "What is the admission process?"
Department: Admissions
Now write minimal if-else based Python code to classify any student query into the correct department.
Used 1 reference
Produced code block structure and finalized instructions
Explore and understand your code
29°C
Sunny
Search
1:51 PM
1/29/2026





4. Chatbot Question Type Detection

Scenario:

A chatbot must identify whether a user query is Informational,

Transactional, Complaint, or Feedback.

Tasks:

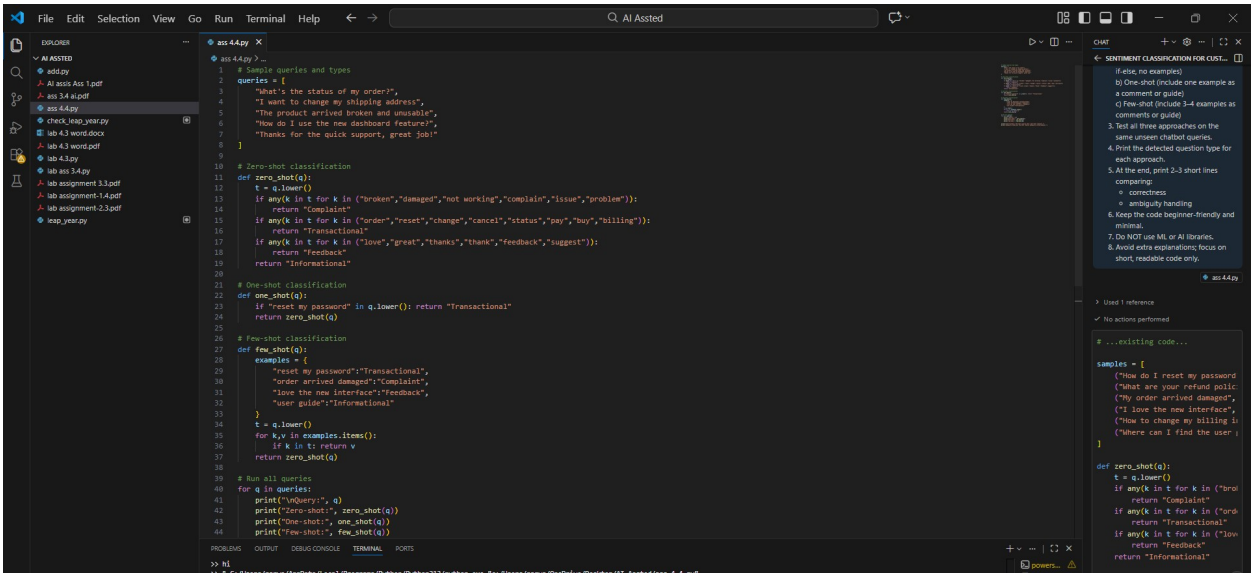
1. Prepare 6 chatbot queries mapped to question types.

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.



5. Emotion Detection in Text

Scenario:

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

Anxious, Neutral.

Tasks:

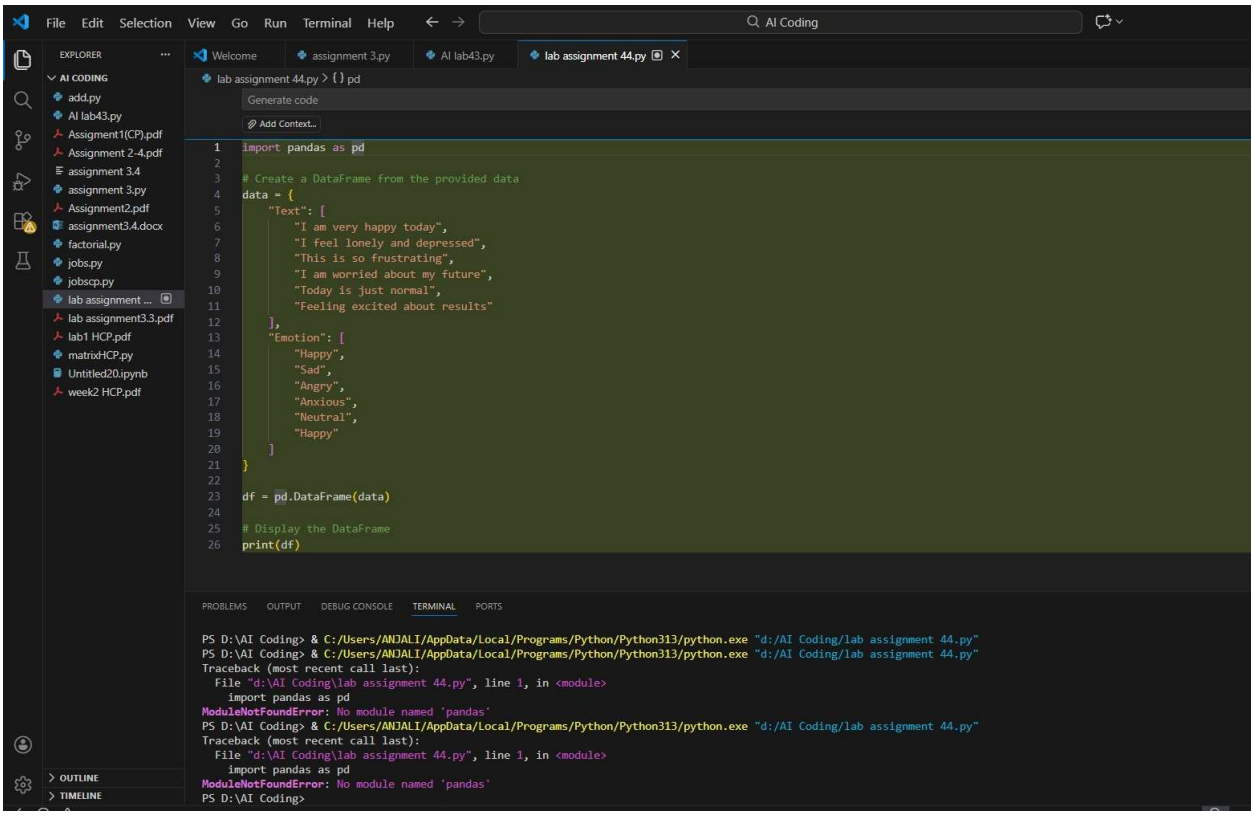
1. Create labeled emotion samples.

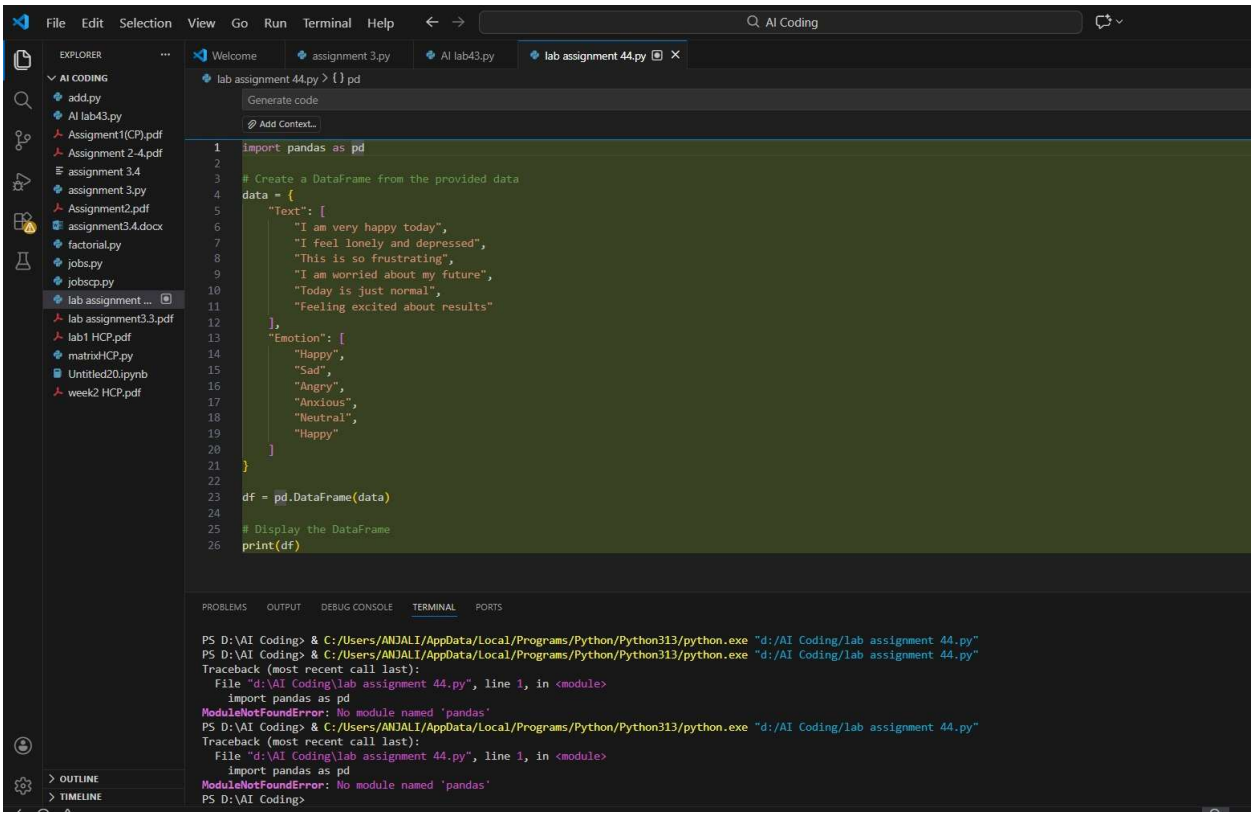
2. Use Zero-shot prompting to identify emotions.

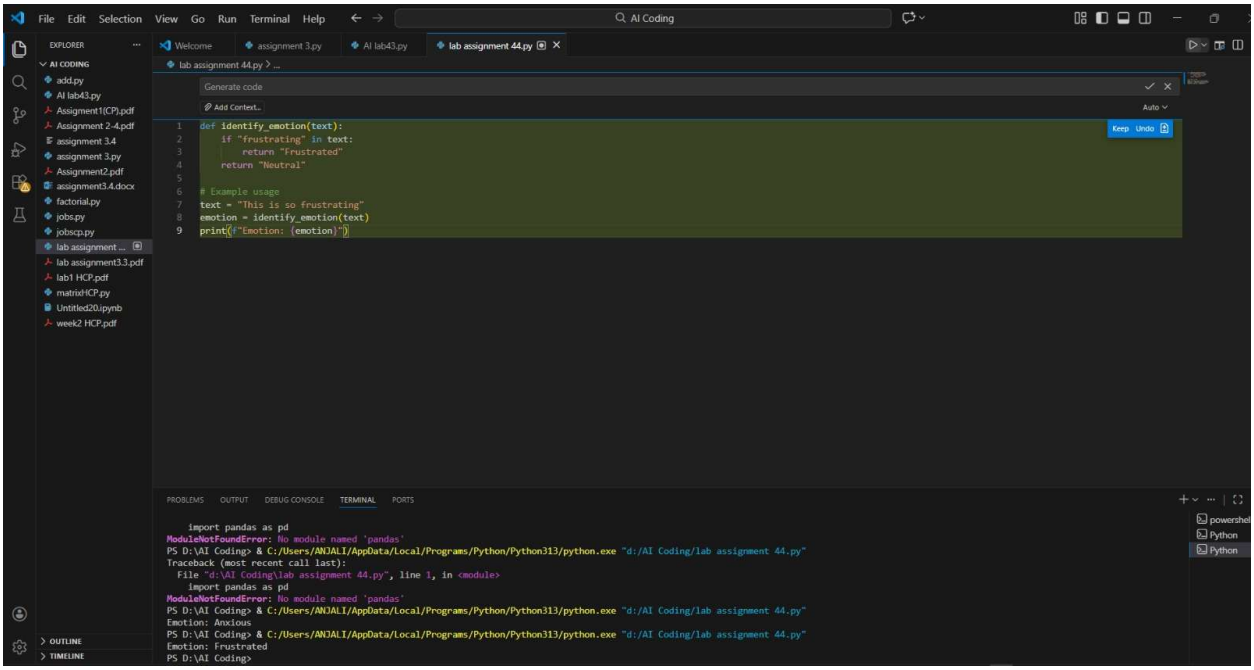
3. Use One-shot prompting with an example.

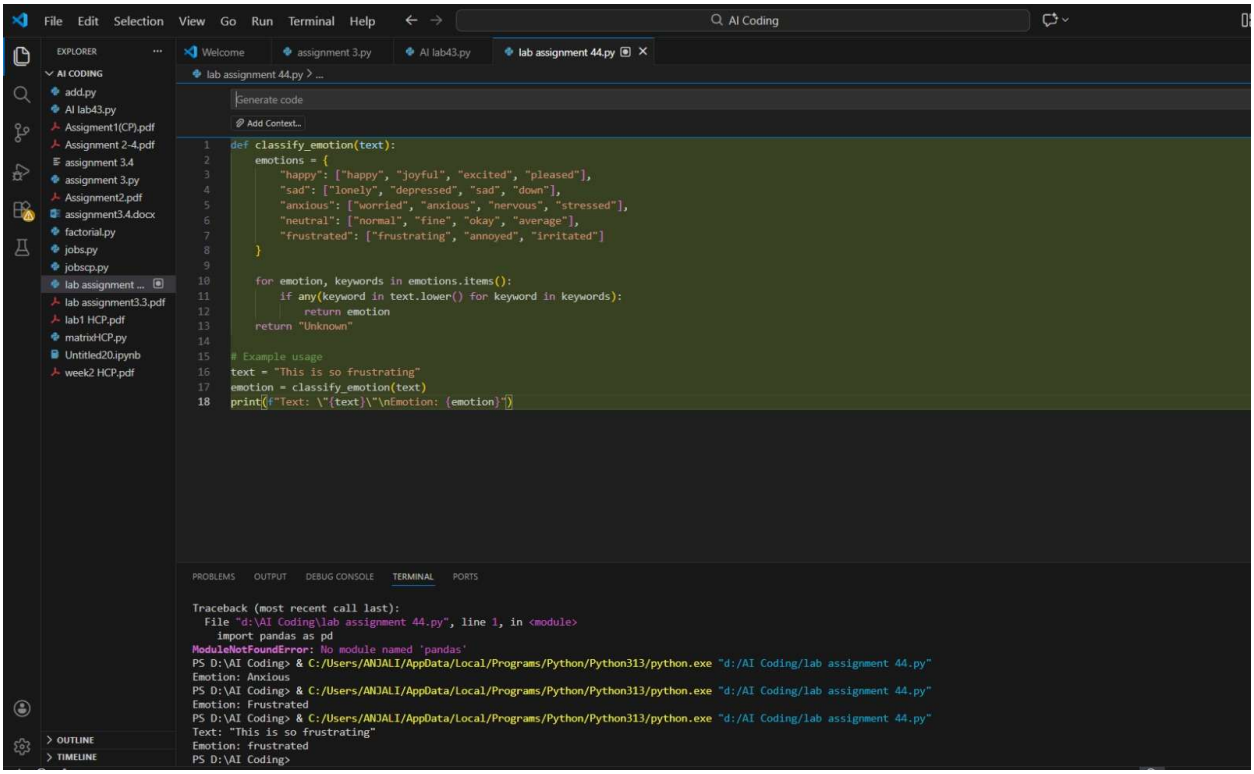
4. Use Few-shot prompting with multiple emotions.

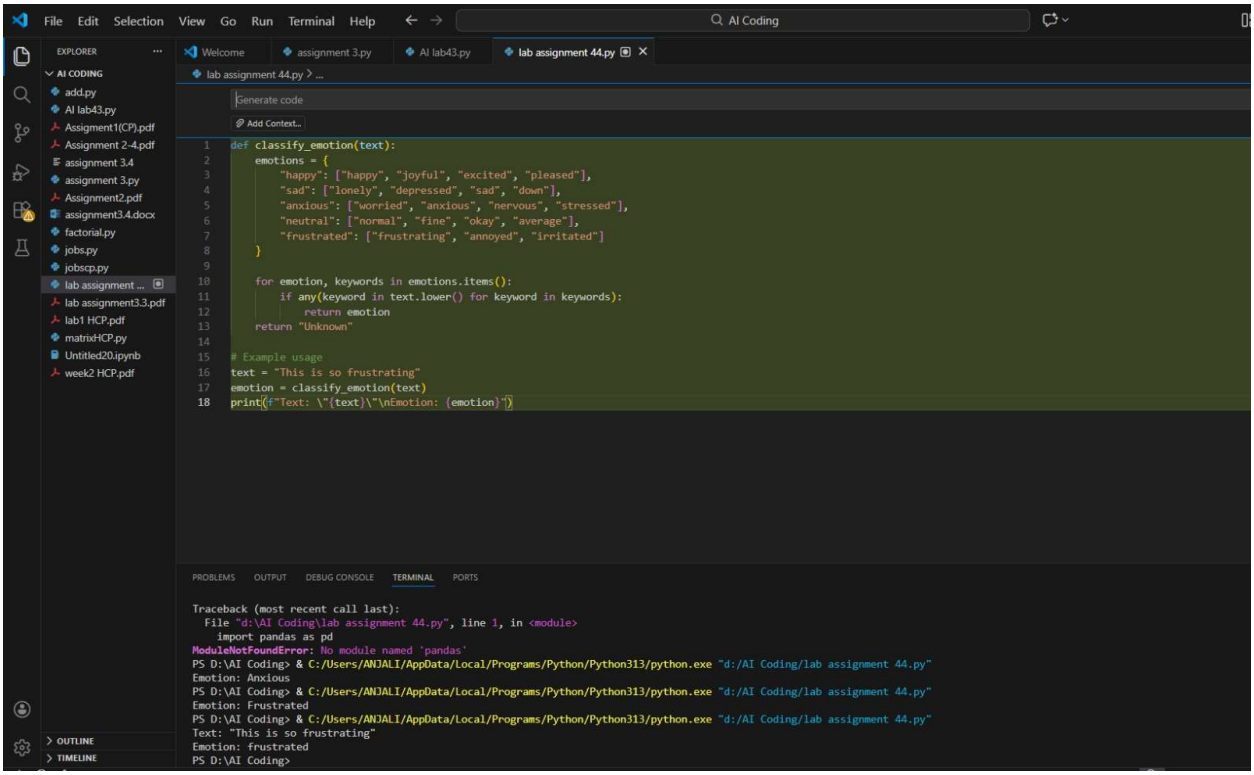
5. Discuss ambiguity handling across techniques.


File Edit Selection View Go Run Terminal Help
AI Coding
EXPLORER
AI CODING
add.py
AI lab43.py
Assigment1(CP).pdf
Assignment 2-4.pdf
assignment 3.4
assignment 3.py
Assignment2.pdf
assignment3.4.docx
factorial.py
jobs.py
jobscp.py
lab assignment ...
lab assignment3.3.pdf
lab1 HCP.pdf
matrixHCP.py
Untitled20.ipynb
week2 HCP.pdf
Welcome
assignment 3.py
AI lab43.py
lab assignment 44.py
lab assignment 44.py > { } pd
Generate code
Add Context..
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)
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>
OUTLINE
TIMELINE


File Edit Selection View Go Run Terminal Help
AI Coding
EXPLORER
AI CODING
add.py
AI lab43.py
Assigment1(CP).pdf
Assignment 2-4.pdf
assignment 3.4
assignment 3.py
Assignment2.pdf
assignment3.4.docx
factorial.py
jobs.py
jobscp.py
lab assignment ...
lab assignment3.3.pdf
lab1 HCP.pdf
matrixHCP.py
Untitled20.ipynb
week2 HCP.pdf
Welcome
assignment 3.py
AI lab43.py
lab assignment 44.py
lab assignment 44.py > { } pd
Generate code
Add Context..
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)
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>
OUTLINE
TIMELINE


File Edit Selection View Go Run Terminal Help
AI Coding
EXPLORER
AI CODING
add.py
AI lab43.py
Assigment1(CP).pdf
Assignment 2-4.pdf
assignment 3.4
assignment 3.py
Assignment2.pdf
assignment3.4.docx
factorial.py
jobs.py
jobscp.py
lab assignment ...
lab assignment3.3.pdf
lab1 HCP.pdf
matrixHCP.py
Untitled20.ipynb
week2 HCP.pdf
Welcome
assignment 3.py
AI lab43.py
lab assignment 44.py
lab assignment 44.py > ...
Generate code
Add Context..
Auto
Keep Undo
def identify_emotion(text):
if "frustrating" in text:
return "Frustrated"
return "Neutral"
# Example usage
text = "This is so frustrating"
emotion = identify_emotion(text)
print(f"Emotion: {emotion}")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
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> & 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>
powershell
Python
Python
OUTLINE
TIMELINE


File Edit Selection View Go Run Terminal Help
AI Coding
EXPLORER
AI CODING
add.py
AI lab43.py
Assigment1(CP).pdf
Assignment 2-4.pdf
assignment 3.4
assignment 3.py
Assignment2.pdf
assignment3.4.docx
factorial.py
jobs.py
jobscp.py
lab assignment ...
lab assignment3.3.pdf
lab1 HCP.pdf
matrixHCP.py
Untitled20.ipynb
week2 HCP.pdf
Welcome
assignment 3.py
AI lab43.py
lab assignment 44.py
lab assignment 44.py > ...
Generate code
Add Context..
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}")
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>
OUTLINE
TIMELINE


File Edit Selection View Go Run Terminal Help
AI Coding
EXPLORER
AI CODING
add.py
AI lab43.py
Assigment1(CP).pdf
Assignment 2-4.pdf
assignment 3.4
assignment 3.py
Assignment2.pdf
assignment3.4.docx
factorial.py
jobs.py
jobscp.py
lab assignment ...
lab assignment3.3.pdf
lab1 HCP.pdf
matrixHCP.py
Untitled20.ipynb
week2 HCP.pdf
OUTLINE
TIMELINE
Welcome
assignment 3.py
AI lab43.py
lab assignment 44.py
lab assignment 44.py > ...
Generate code
Add Context...
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}")
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>