

ASSIGNMENT-4.5

NAME-TEENESWARI

BATCH-29

ROLL-NO:2303A51932

ADVANCED PROMPT ENGINEERING: ZERO-SHOT, ONE-SHOT & FEW-SHOT

TASK-1:

ZERO-SHOT

A. Preparing Sample data:

```
test_emails = [
```

```
    "My payment failed but money was deducted.",
```

```
    "The app is not opening on my phone.",
```

```
    "Great customer service, very satisfied.",
```

```
    "What is your customer care number?",
```

```
    "Invoice amount seems incorrect."
```

```
]
```

Expected Labels (for evaluation): true_labels =

```
[
```

```
    "Billing",
```

```
    "Technical Support",
```

```
"Feedback",  
"Others",  
"Billing"  
]
```

PROMPT:

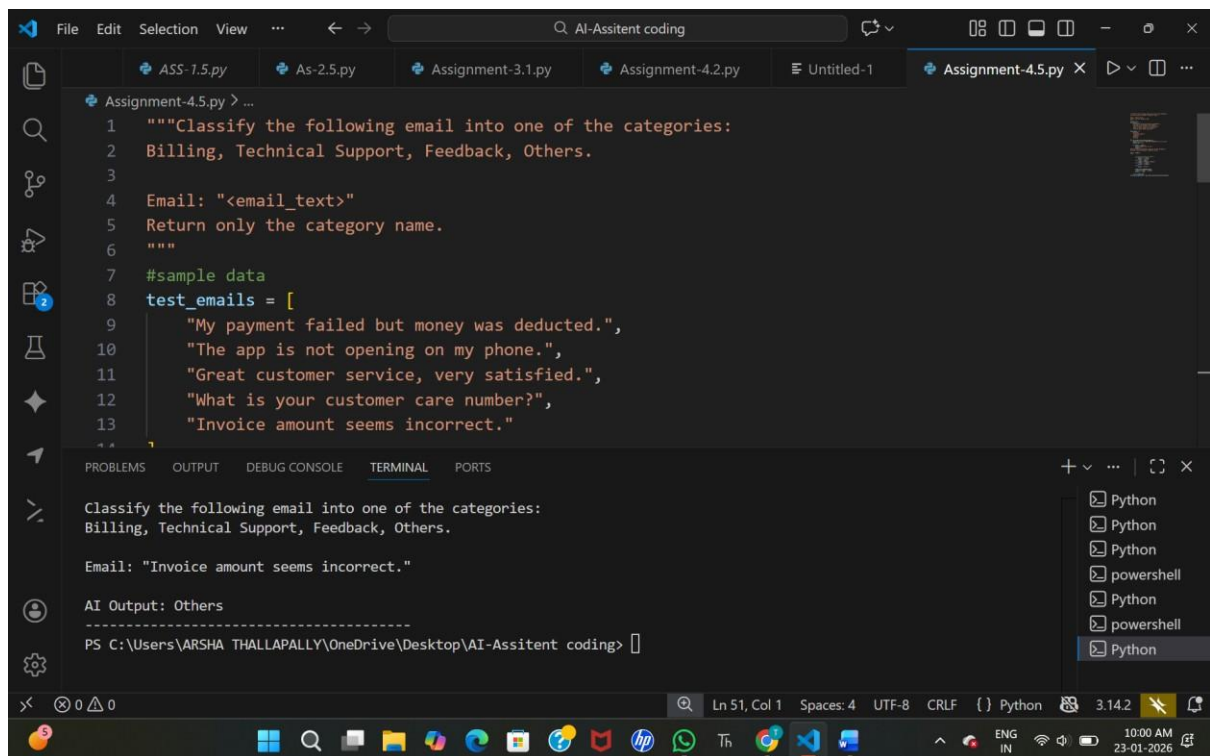
Classify the following email into one of the categories:

Billing, Technical Support, Feedback, Others.

Email: "<email_text>"

Return only the category name.

CODE:



The screenshot shows a Visual Studio Code editor window with a Python file named 'Assignment-4.5.py'. The script contains a prompt to classify an email into one of four categories: Billing, Technical Support, Feedback, or Others. It also includes sample data for testing. The terminal at the bottom shows the output of the script, which correctly classified the email 'Invoice amount seems incorrect.' as 'Others'.

```
1 """Classify the following email into one of the categories:  
2 Billing, Technical Support, Feedback, Others.  
3  
4 Email: "<email_text>"  
5 Return only the category name.  
6 """  
7 #sample data  
8 test_emails = [  
9     "My payment failed but money was deducted.",  
10    "The app is not opening on my phone.",  
11    "Great customer service, very satisfied.",  
12    "What is your customer care number?",  
13    "Invoice amount seems incorrect."  
14 ]  
15  
16 # Your code here
```

Terminal Output:

```
Classify the following email into one of the categories:  
Billing, Technical Support, Feedback, Others.  
  
Email: "Invoice amount seems incorrect."  
  
AI Output: Others  
-----  
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>
```

OBSERVATION:

Classifies emails using only instructions, without examples.

Works if keywords are clear, may misclassify ambiguous emails.

Quick and simple, but less accurate for complex cases.

ONE-SHOT : PROMPT:

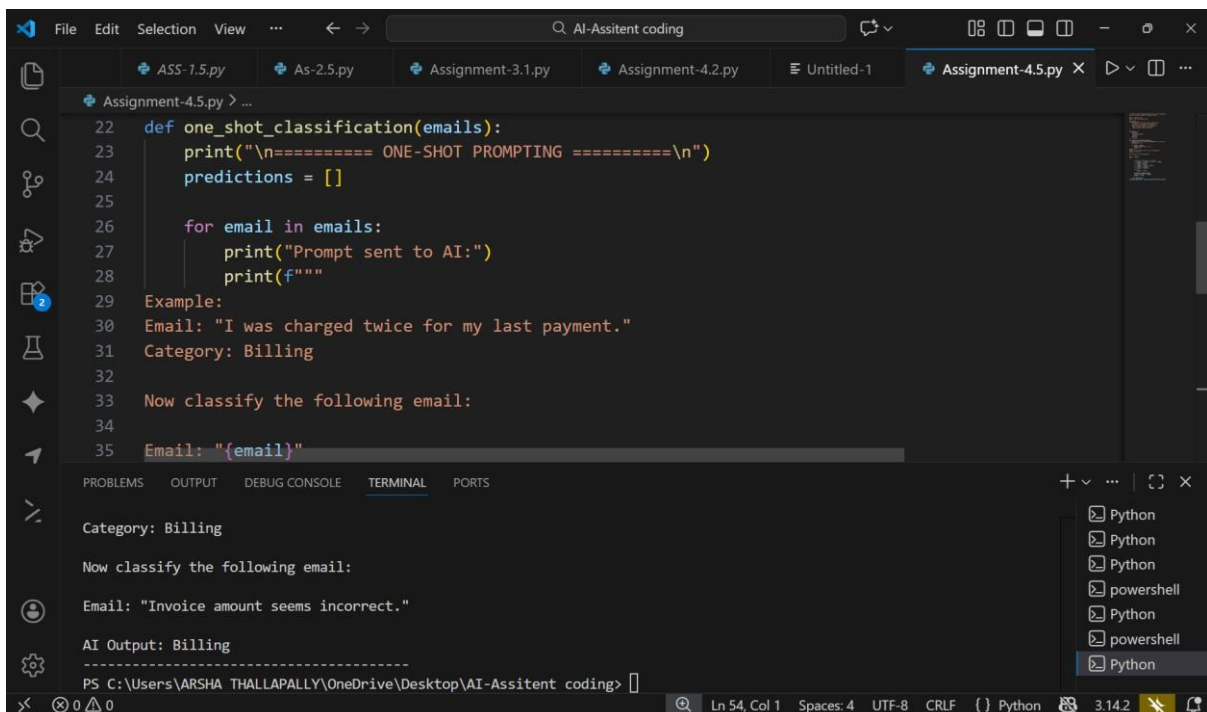
Example:

Email: "I was charged twice for my last payment."

Category: Billing

Now classify the following email:

Email: "<email_text>" CODE:



```
File Edit Selection View ... AI-Assitent coding
Assignment-4.5.py > ...
def one_shot_classification(emails):
    print("\n===== ONE-SHOT PROMPTING =====\n")
    predictions = []
    for email in emails:
        print("Prompt sent to AI:")
        print(f"""
29 Example:
30 Email: "I was charged twice for my last payment."
31 Category: Billing
32
33 Now classify the following email:
34
35 Email: "{email}"
Category: Billing

Now classify the following email:

Email: "Invoice amount seems incorrect."

AI Output: Billing
-----
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>
```

OBSERVATION:

Provides one example to guide the AI's reasoning. Improves accuracy over zero-shot and handles slightly ambiguous emails better.

Still limited; accuracy depends on how representative the example is.

One example helps the AI understand the expected format and category mapping.

Classification accuracy improves compared to zero-shot, especially for similar issues.

Performance depends heavily on how relevant the single example is to the new email.

FEW SHOT:

PROMPT:

Email: "I was charged twice for my last payment." → Billing

Email: "The app crashes on login." → Technical Support

Email: "I love the new update." → Feedback

Email: "What are your office hours?" → Others

CODE:

The screenshot shows a VS Code editor with a Python file named 'Assignment-4.5.py'. The code defines a function `few_shot_classification(emails)` that prints a prompt and then iterates over a list of emails, printing each email and its predicted category. The prompt includes two examples: 'I was charged twice for my last payment.' (Billing) and 'The app crashes on login.' (Technical Support). The terminal output shows the function being called with a new email: 'Invoice amount seems incorrect.', and the AI outputting 'Billing'.

```
22 def few_shot_classification(emails):
23     print("\n===== FEW-SHOT PROMPTING =====\n")
24     predictions = []
25
26     for email in emails:
27         print("Prompt sent to AI:")
28         print(f"""
29 Examples:
30 Email: "I was charged twice for my last payment."
31 Category: Billing
32
33 Email: "The app crashes on login."
34 Category: Technical Support
35 """)
36
37 # Test the function
38 emails = [
39     "I was charged twice for my last payment.",
40     "The app crashes on login.",
41     "Invoice amount seems incorrect."
42 ]
43
44 predictions = few_shot_classification(emails)
45
46 print(predictions)
```

Terminal Output:

```
Category: Others
Now classify the following email:
Email: "Invoice amount seems incorrect."
AI Output: Billing
-----
PS C:\Users\VARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>
```

OBSERVATION:

Provides multiple examples to show patterns to the AI.

Highest accuracy; AI can generalize better for unseen emails. Slightly longer prompts but most reliable for real-world use

TASK-2:

Sample travel queries (short & simple) travel_queries =

[

"Book a flight from Delhi to Mumbai.",

"Cancel my hotel reservation in Paris.",

"What is the baggage allowance?",

"I need a hotel in London for 2 nights.",

"Cancel my flight ticket to New York."

]

True labels for evaluation

```
true_labels = [    "Flight  
Booking",  
    "Cancellation",  
    "General Travel Info",  
    "Hotel Booking",  
    "Cancellation"  
]
```

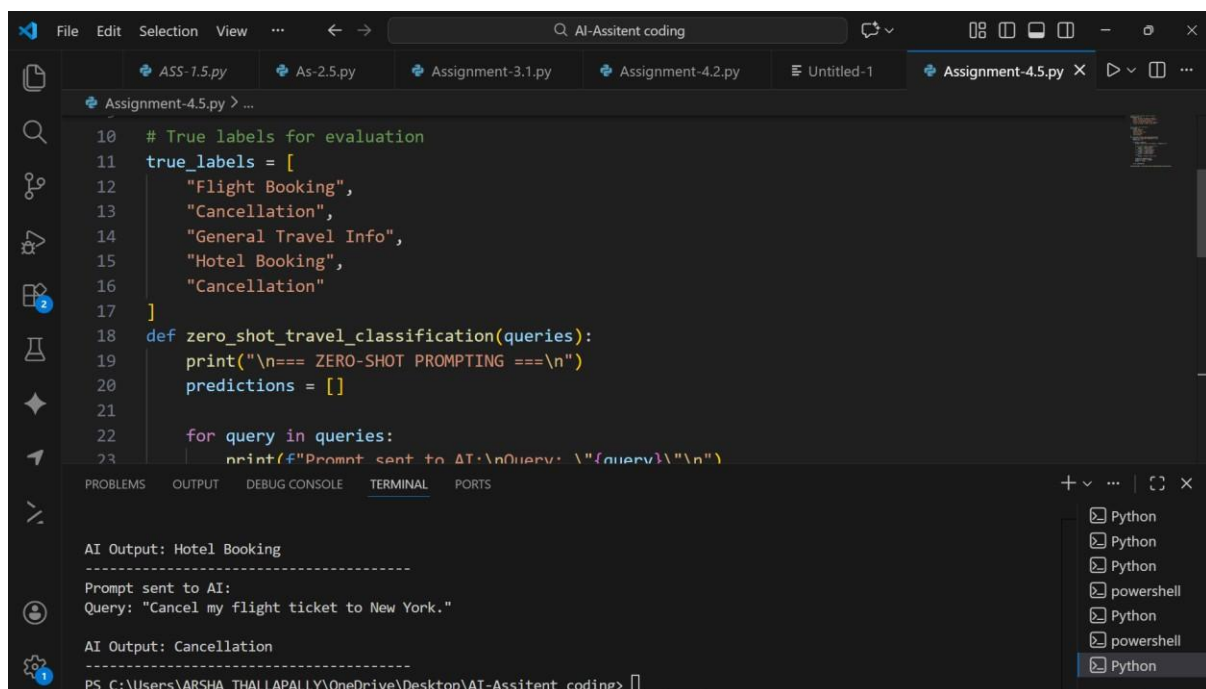
ZERO-SHOT:

PROMPT: Classify the following travel query into one of the categories:

Flight Booking, Hotel Booking, Cancellation, General Travel Info.

Query: "<travel_query>"

CODE:



```
10 # True labels for evaluation
11 true_labels = [
12     "Flight Booking",
13     "Cancellation",
14     "General Travel Info",
15     "Hotel Booking",
16     "Cancellation"
17 ]
18 def zero_shot_travel_classification(queries):
19     print("\n=== ZERO-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for query in queries:
23         print(f"Prompt sent to AI:\nQuery: \"{query}\"")
24         # ... (AI interaction logic) ...
25         # ... (AI interaction logic) ...
26         # ... (AI interaction logic) ...
27         # ... (AI interaction logic) ...
28         # ... (AI interaction logic) ...
29         # ... (AI interaction logic) ...
30         # ... (AI interaction logic) ...
31         # ... (AI interaction logic) ...
32         # ... (AI interaction logic) ...
33         # ... (AI interaction logic) ...
34         # ... (AI interaction logic) ...
35         # ... (AI interaction logic) ...
36         # ... (AI interaction logic) ...
37         # ... (AI interaction logic) ...
38         # ... (AI interaction logic) ...
39         # ... (AI interaction logic) ...
40         # ... (AI interaction logic) ...
41         # ... (AI interaction logic) ...
42         # ... (AI interaction logic) ...
43         # ... (AI interaction logic) ...
44         # ... (AI interaction logic) ...
45         # ... (AI interaction logic) ...
46         # ... (AI interaction logic) ...
47         # ... (AI interaction logic) ...
48         # ... (AI interaction logic) ...
49         # ... (AI interaction logic) ...
50         # ... (AI interaction logic) ...
51         # ... (AI interaction logic) ...
52         # ... (AI interaction logic) ...
53         # ... (AI interaction logic) ...
54         # ... (AI interaction logic) ...
55         # ... (AI interaction logic) ...
56         # ... (AI interaction logic) ...
57         # ... (AI interaction logic) ...
58         # ... (AI interaction logic) ...
59         # ... (AI interaction logic) ...
60         # ... (AI interaction logic) ...
61         # ... (AI interaction logic) ...
62         # ... (AI interaction logic) ...
63         # ... (AI interaction logic) ...
64         # ... (AI interaction logic) ...
65         # ... (AI interaction logic) ...
66         # ... (AI interaction logic) ...
67         # ... (AI interaction logic) ...
68         # ... (AI interaction logic) ...
69         # ... (AI interaction logic) ...
70         # ... (AI interaction logic) ...
71         # ... (AI interaction logic) ...
72         # ... (AI interaction logic) ...
73         # ... (AI interaction logic) ...
74         # ... (AI interaction logic) ...
75         # ... (AI interaction logic) ...
76         # ... (AI interaction logic) ...
77         # ... (AI interaction logic) ...
78         # ... (AI interaction logic) ...
79         # ... (AI interaction logic) ...
80         # ... (AI interaction logic) ...
81         # ... (AI interaction logic) ...
82         # ... (AI interaction logic) ...
83         # ... (AI interaction logic) ...
84         # ... (AI interaction logic) ...
85         # ... (AI interaction logic) ...
86         # ... (AI interaction logic) ...
87         # ... (AI interaction logic) ...
88         # ... (AI interaction logic) ...
89         # ... (AI interaction logic) ...
90         # ... (AI interaction logic) ...
91         # ... (AI interaction logic) ...
92         # ... (AI interaction logic) ...
93         # ... (AI interaction logic) ...
94         # ... (AI interaction logic) ...
95         # ... (AI interaction logic) ...
96         # ... (AI interaction logic) ...
97         # ... (AI interaction logic) ...
98         # ... (AI interaction logic) ...
99         # ... (AI interaction logic) ...
100        # ... (AI interaction logic) ...
```

AI Output: Hotel Booking

Prompt sent to AI:
Query: "Cancel my flight ticket to New York."

AI Output: Cancellation

PS C:\Users\VARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>

OBSERVATION:

Classifies queries using only instructions, without examples. Works for obvious keywords like “flight” or “cancel”, may misclassify tricky queries.

Fast and simple, but accuracy is lower for ambiguous cases.

ONE-SHOT: PROMPT:

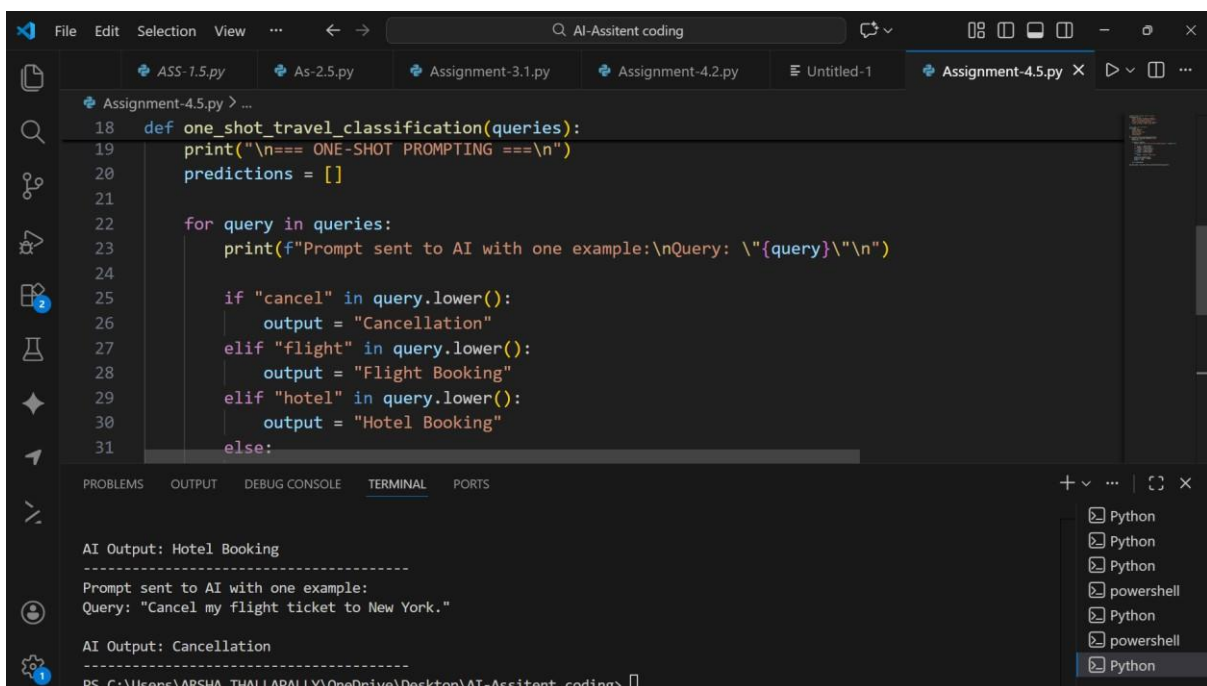
Example:

Query: "Cancel my flight ticket."

Category: Cancellation

Now classify the following query:

Query: "<travel_query>" CODE:



```
File Edit Selection View ... AI-Assitent coding
ASS-1.5.py As-2.5.py Assignment-3.1.py Assignment-4.2.py Untitled-1 Assignment-4.5.py X
Assignment-4.5.py > ...
18 def one_shot_travel_classification(queries):
19     print("\n=== ONE-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for query in queries:
23         print(f"Prompt sent to AI with one example:\nQuery: \"{query}\"")
24
25         if "cancel" in query.lower():
26             output = "Cancellation"
27         elif "flight" in query.lower():
28             output = "Flight Booking"
29         elif "hotel" in query.lower():
30             output = "Hotel Booking"
31         else:
32             output = "Unknown"
33         predictions.append(output)
34
35     return predictions
36
37 queries = ["Cancel my flight ticket.", "Book a hotel room in New York.", "I want to fly to London."]
38 result = one_shot_travel_classification(queries)
39 print("Predictions:", result)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

AI Output: Hotel Booking

Prompt sent to AI with one example:
Query: "Cancel my flight ticket to New York."

AI Output: Cancellation

PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>

OBSERVATION:

Provides one example to guide AI’s reasoning.

Improves accuracy and handles slightly ambiguous queries better.

Accuracy depends on how representative the example is.

FEW SHOT:

PROMPT:

Examples:

Query: "Book a flight to Mumbai."

Category: Flight Booking

Query: "Cancel my hotel reservation."

Category: Cancellation

Query: "I need a hotel in London."

Category: Hotel Booking

Query: "What is the baggage allowance?"

Category: General Travel Info

Now classify the following query:

Query: "<travel_query>" CODE:

The screenshot shows a VS Code editor window with a Python script named `Assignment-4.5.py`. The script defines a function `few_shot_travel_classification(queries)` that processes a list of queries. It prints a separator, then for each query, it prints the prompt and classifies it based on keywords: "cancel" for "Cancellation", "flight" for "Flight Booking", "hotel" for "Hotel Booking", and otherwise "Hotel Booking". The terminal output shows the function being called with a query "Cancel my flight ticket to New York.", resulting in "AI Output: Hotel Booking". A second query "Cancel my flight ticket to New York." is shown, resulting in "AI Output: Cancellation". The terminal path is `PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>`.

```
18 def few_shot_travel_classification(queries):
19     print("\n=== FEW-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for query in queries:
23         print(f"Prompt sent to AI with multiple examples:\nQuery: \"{query}\"")
24
25         if "cancel" in query.lower():
26             output = "Cancellation"
27         elif "flight" in query.lower():
28             output = "Flight Booking"
29         elif "hotel" in query.lower():
30             output = "Hotel Booking"
31         else:
32             output = "Hotel Booking"
```

AI Output: Hotel Booking

Prompt sent to AI with multiple examples:
Query: "Cancel my flight ticket to New York."

AI Output: Cancellation

PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>

OBSERVATION:

Provides multiple examples to show patterns to AI.

Highest accuracy; AI generalizes better for unseen queries.

Slightly longer prompts but most reliable for real-world use. **TASK-**

3: SAMPLE DATA:

Sample coding queries (short & simple) `coding_queries = [`

"Why am I getting IndexError in my Python list?",

"My sorting algorithm is too slow for large inputs.",

"I wrote a function but it returns wrong results.",

"Explain the difference between list and tuple in Python.",

"How can I optimize my recursive Fibonacci function?"

`]`

True labels for evaluation

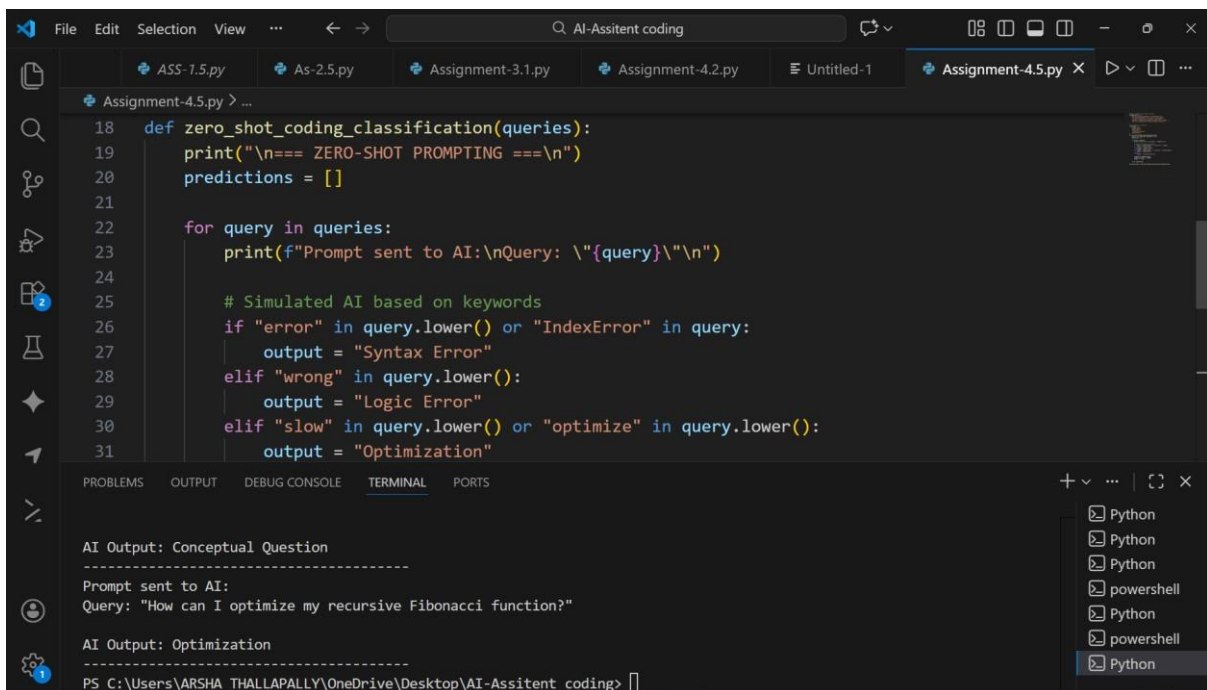
```
true_labels = [    "Syntax  
Error",  
    "Optimization",  
    "Logic Error",  
    "Conceptual Question",  
    "Optimization"  
]
```

ZERO-SHOT

PROMPT: Classify the following coding query into one of the categories:

Syntax Error, Logic Error, Optimization, Conceptual Question.

Query: "<coding_query>" CODE:



```
File Edit Selection View ... AI-Assitent coding
Assignment-4.5.py > ...
18 def zero_shot_coding_classification(queries):
19     print("\n=== ZERO-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for query in queries:
23         print(f"Prompt sent to AI:\nQuery: \"{query}\"")
24
25         # Simulated AI based on keywords
26         if "error" in query.lower() or "IndexError" in query:
27             output = "Syntax Error"
28         elif "wrong" in query.lower():
29             output = "Logic Error"
30         elif "slow" in query.lower() or "optimize" in query.lower():
31             output = "Optimization"

```

AI Output: Conceptual Question

Prompt sent to AI:
Query: "How can I optimize my recursive Fibonacci function?"

AI Output: Optimization

PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>

OBSERVATION:

ONE SHOT PROMPT:

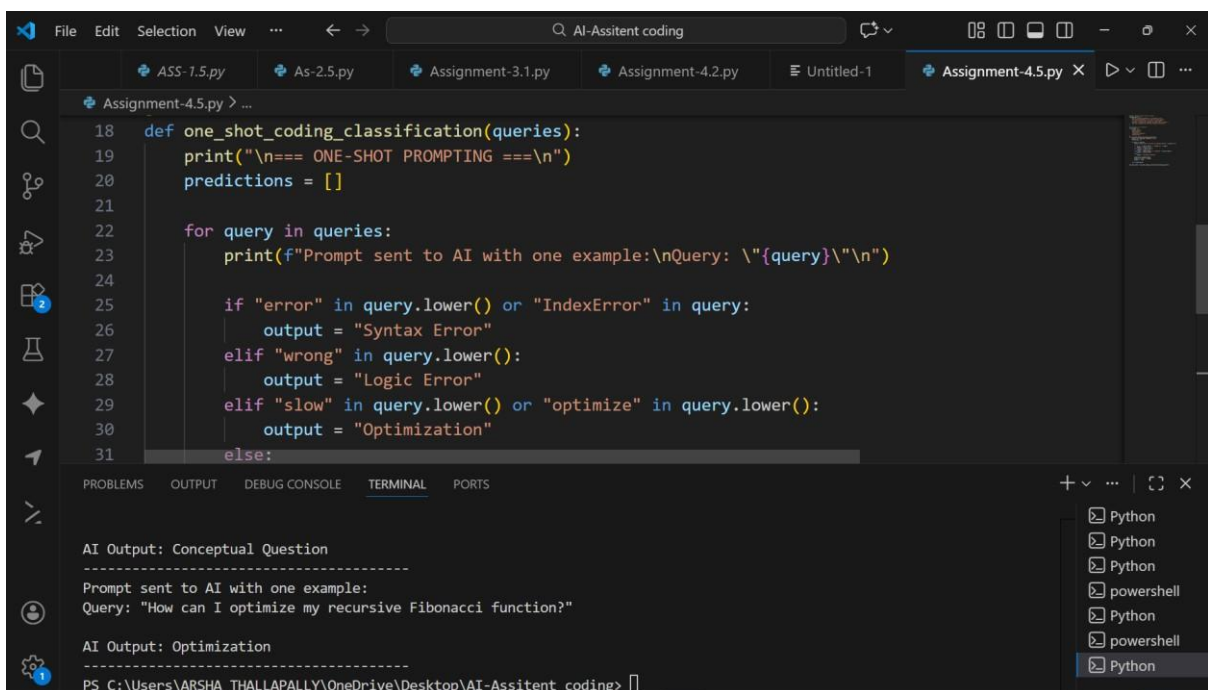
Example:

Query: "I want to cancel my Python function."

Category: Logic Error

Now classify the following coding query:

Query: "<coding_query>" CODE:



```
File Edit Selection View ... AI-Assitent coding
Assignment-4.5.py > ...
18 def one_shot_coding_classification(queries):
19     print("\n=== ONE-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for query in queries:
23         print(f"Prompt sent to AI with one example:\nQuery: \"{query}\"")
24
25         if "error" in query.lower() or "IndexError" in query:
26             output = "Syntax Error"
27         elif "wrong" in query.lower():
28             output = "Logic Error"
29         elif "slow" in query.lower() or "optimize" in query.lower():
30             output = "Optimization"
31         else:
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

AI Output: Conceptual Question

Prompt sent to AI with one example:
Query: "How can I optimize my recursive Fibonacci function?"

AI Output: Optimization

PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>

Python
Python
Python
powershell
Python
powershell
Python

OBSERVATION:

Provides one example to guide AI's reasoning.

Improves accuracy and handles slightly ambiguous queries better.

Accuracy depends on how representative the single example is.

FEW SHOT PROMPT:

Examples:

Query: "Why does my Python list give IndexError?"

Category: Syntax Error

Query: "My function returns wrong output."

Category: Logic Error

Query: "My loop is too slow for large data."

Category: Optimization

Query: "Explain Python variable scopes."

Category: Conceptual Question

Now classify the following coding query:

Query: "<coding_query>" CODE:

The screenshot shows a VS Code editor window with a Python script named `Assignment-4.5.py`. The script defines a function `few_shot_coding_classification(queries)` that processes a list of queries. It prints a separator, then for each query, it prints the prompt sent to an AI and the AI's output. The AI's output is classified into one of four categories: "Syntax Error", "Logic Error", "Optimization", or "Conceptual Question". The terminal output shows the function being called with a query about optimizing a recursive Fibonacci function, resulting in an "Optimization" classification.

```
18 def few_shot_coding_classification(queries):
19     print("\n=== FEW-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for query in queries:
23         print(f"Prompt sent to AI with multiple examples:\nQuery: \"{query}\"")
24
25         if "error" in query.lower() or "IndexError" in query:
26             output = "Syntax Error"
27         elif "wrong" in query.lower():
28             output = "Logic Error"
29         elif "slow" in query.lower() or "optimize" in query.lower():
30             output = "Optimization"
31         else:
32             output = "Conceptual Question"
```

Terminal Output:

```
AI Output: Conceptual Question
-----
Prompt sent to AI with multiple examples:
Query: "How can I optimize my recursive Fibonacci function?"

AI Output: Optimization
-----
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>
```

OBSERVATION:

Provides multiple examples showing patterns to AI.

Highest accuracy; AI generalizes better for unseen queries. Slightly longer prompts but most reliable for technical classification.

TASK-4

ZERO-SHOT

PROMPT:

Classify the following social media post into one of the categories:

Promotion, Complaint, Appreciation, Inquiry.

Post: "<social_post>" CODE:

The screenshot shows a VS Code editor window with a Python file named 'Assignment-4.5.py'. The code defines a function `zero_shot_social_classification(posts)` that prints a separator, iterates over posts, prints each post, and classifies it based on keywords: 'disappointed' or 'not arrived' leads to 'Complaint', and 'discount' or 'offer' leads to another classification. The terminal output shows the function being called with the post 'Thanks for the quick customer support!', resulting in an 'Inquiry' classification. The file explorer on the right shows a list of files including Python and PowerShell scripts.

```
18 def zero_shot_social_classification(posts):
19     print("\n=== ZERO-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for post in posts:
23         print(f"Prompt sent to AI:\nPost: \"{post}\"")
24
25         # AI decides based on keywords
26         if "disappointed" in post.lower() or "not arrived" in post.lower():
27             output = "Complaint"
28         elif "discount" in post.lower() or "offer" in post.lower():
```

AI Output: Inquiry

Prompt sent to AI:
Post: "Thanks for the quick customer support!"

AI Output: Appreciation

PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>

OBSERVATION:

Classifies posts using only instructions, without examples. Works for clear keywords but may misinterpret informal or slang language.

Fast and simple, lower accuracy for ambiguous or sarcastic posts.

ONE-SHOT PROMPT:

Example:

Post: "My order is late."

Category: Complaint

Now classify the following social media post:

Post: "<social_post>" CODE:

The screenshot shows a VS Code editor window with a file named 'Assignment-4.5.py'. The code defines a function `one_shot_social_classification(posts)` that iterates through a list of posts and classifies them based on keywords. The terminal output shows three examples: 'Can someone help me with the login issue?' is classified as 'Promotion', 'Thanks for the quick customer support!' as 'Inquiry', and 'Loved the new feature!' as 'Appreciation'.

```
18 def one_shot_social_classification(posts):
19     print("\n=== ONE-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for post in posts:
23         print(f"Prompt sent to AI with one example:\nPost: \"{post}\"")
24
25         if "disappointed" in post.lower() or "not arrived" in post.lower():
26             output = "Complaint"
27         elif "discount" in post.lower() or "offer" in post.lower():
28             output = "Promotion"
```

Terminal Output:

```
AI Output: Promotion
-----
Prompt sent to AI with one example:
Post: "Can someone help me with the login issue?"

AI Output: Inquiry
-----
Prompt sent to AI with one example:
Post: "Thanks for the quick customer support!"

AI Output: Appreciation
-----
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>
```

OBSERVATION:

Provides one example to guide AI reasoning.

Improves accuracy and handles some informal expressions better.

Depends on how representative the example is for informal language.

FEW-SHOT PROMPT:

Examples:

Post: "Loved the new feature!" → Appreciation

Post: "My order hasn't arrived." → Complaint

Post: "Check out our discount offer!" → Promotion

Post: "How can I reset my password?" → Inquiry

Now classify the following social media post:

Post: "<social_post>" CODE:

The screenshot shows a Visual Studio Code editor window with the file explorer on the left and the terminal at the bottom. The active file is 'Assignment-4.5.py', which contains a Python function named `few_shot_social_classification(posts)`. The function prints a separator, iterates over a list of posts, and classifies each post based on keywords. The terminal shows the output of the function for three different posts, resulting in 'Promotion', 'Inquiry', and 'Appreciation' classifications.

```
18 def few_shot_social_classification(posts):
19     print("\n=== FEW-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for post in posts:
23         print(f"Prompt sent to AI with multiple examples:\nPost: \"{post}\"")
24
25         if "disappointed" in post.lower() or "not arrived" in post.lower():
26             output = "Complaint"
27         elif "discount" in post.lower() or "offer" in post.lower():
28             output = "Promotion"
```

Terminal Output:

```
AI Output: Promotion
-----
Prompt sent to AI with multiple examples:
Post: "Can someone help me with the login issue?"

AI Output: Inquiry
-----
Prompt sent to AI with multiple examples:
Post: "Thanks for the quick customer support!"

AI Output: Appreciation
-----
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>
```

OBSERVATION:

Provides multiple examples showing patterns to AI. Highest accuracy; better handles informal, slang, or mixed language posts.

Slightly longer prompts but most reliable for social media classification.