

# Assignment-4.1

Name: S.VYSHNAV I

Ht No: 2303A52239

Batch: 35

## Task-1

### **Customer Email Classification**

A company receives a large number of customer emails every day and wants to automatically classify them into the following categories:

- Billing
- Technical Support
- Feedback
- Others

Instead of training a new machine learning model, the company decides to use prompt engineering techniques with an existing large language model.

#### Tasks

1. Prepare five short sample emails, each belonging to one of the above categories.
2. Write a zero-shot prompt to classify a given email into one of the categories without providing any examples.
3. Write a one-shot prompt by including one labeled email example and ask the model to classify a new email.

4. Write a few-shot prompt by including two or three labeled email examples and ask the model to classify a new email.

5. Compare the outputs obtained using zero-shot, one-shot, and few-shot prompting techniques and briefly comment on their effectiveness

The screenshot shows the VS Code interface with several files open in the Explorer sidebar, including `Y555.md`, `string_reversal_approaches.py`, `Email_Classification_Prompt_Engineering.ipynb`, and `email_classification_system.py`. The `email_classification_system.py` file contains the following code:

```
class EmailClassificationSystem:
    def __init__(self):
        self.prompts: Dict[str, str] = {}
        self.results: Dict[str, Any] = {}
        self.timestamp = datetime.datetime.now().isoformat()

    def prepare_sample_emails(self) -> None:
        """Prepare 5 sample emails, one for each category."""
        self.sample_emails = [
            {
                "id": 1,
                "category": EmailCategory.BILLING.value,
                "email": """Subject: Invoice #INV-2026-0145 - Payment Issue
Dear Support Team,
I received my invoice for the subscription renewal on January 15th, but I was charged twice for the same billing period. The first charge was on January 18th and the second on January 19th. Both transactions are showing on my account.
Could you please investigate this billing error and issue a refund for the duplicate charge? My account reference is CUST-78945.
Thank you for your prompt assistance.
Best regards,
Sarah Johnson"""
            },
            ...
        ]
```

The terminal tab shows the output of the script:

```
Subject: Invoice #INV-2026-0145 - Payment Issue
Dear Support Team,
I received my invoice for the subscription renewal on January 15th, but I was charged twice for the same billing period. The first charge was on January 18th and the second on January 19th. Both transactions are showing on my account.
Could you please investigate this billing error and issue a refund for the duplicate charge? My account reference is CUST-78945.
Thank you for your prompt assistance.
Best regards,
Sarah Johnson
```

The status bar at the bottom indicates the file is ready for build and deployment.

The screenshot shows the VS Code interface with the same files open. The `email_classification_system.py` file now includes a second email example:

```
class EmailClassificationSystem:
    def prepare_sample_emails(self) -> None:
        Best regards,
        Sarah Johnson"""
        ],
        {
            "id": 2,
            "category": EmailCategory.TECHNICAL_SUPPORT.value,
            "email": """Subject: Cannot Login - Error Code 502
Hello,
I've been unable to access my account for the past 2 hours. Every time I try to log in with my credentials, I receive error code 502 and the message "Bad Gateway". I've tried clearing my browser cache and using a different browser, but the issue persists.
My username is: mike.choe@gmail.com
Error occurs at: www.platform.com/login
Could you help me resolve this? This is affecting my work.
Thanks,
Mike Chen"""
        }
```

The terminal tab shows the output of the script:

```
Subject: Cannot Login - Error Code 502
Hello,
I've been unable to access my account for the past 2 hours. Every time I try to log in with my credentials, I receive error code 502 and the message "Bad Gateway". I've tried clearing my browser cache and using a different browser, but the issue persists.
My username is: mike.choe@gmail.com
Error occurs at: www.platform.com/login
Could you help me resolve this? This is affecting my work.
Thanks,
Mike Chen
```

The status bar at the bottom indicates the file is ready for build and deployment.

The screenshot shows the AIAC (AI Application Center) IDE interface. The left sidebar displays a file tree with various project files like `DO_START_HERE.md`, `EXECUTIVE_SUMMARY.md`, and `email_classification_system.py`. The main workspace shows a Python script (`email_classification_system.py`) with code related to email classification. Below the code editor is a terminal window showing command-line interactions. A right-hand sidebar provides navigation links for the project, including a deployment roadmap, customization guides, and deployment steps. The bottom status bar indicates the file is ready for production and shows system information like temperature and battery level.

```
File Edit Selection View Go Run Terminal Help < - > Q AIAC
EXPLORER -- YSI1.md string_reversal_approaches.py Email_Classification_Prompt_Engineering.ipynb email_classification_system.py I D V - OME
& [learn] Production deployment roadmap (4 phases)
COMPARISON ANALYSIS.md Decision framework (choose your technique)
COMPLETION_CHECKLIST.md Customization guide (adapt to your needs)
DELIVERY_SUMMARY.txt Quality verification (all tested)
Email_Classification_Comparison_R...
Email_Classification_Prompt_Engin...
email_classification_results.json
email_classification_system.py
EXECUTIVE_SUMMARY.md
physics_test_queries.csv
physics_test_queries.json
PROJECT_INDEX.md
README_email_Classification.md
reverse_string_function.py
reverse_string.py
string_reversal_approaches.py

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
SAMPLE #3 - (Category): Feedback
Subject: Great Experience with Your Product!
Hi Team,
I just wanted to reach out and say how impressed I am with your latest product update. The new dashboard interface is intuitive and the performance improvements are noticeable. The customer support team was also extremely helpful when I had questions during the setup process.
I'd love to see more integration options with third-party tools in the future, but overall, I'm very satisfied with my purchase and would definitely recommend your service to others.
Keep up the excellent work!
Regards,
Jennifer Martinez
Indexing completed.
29°C Sunny
Search
Ln 663, Col 1 Spaces: 4 UFT-B CRLF Python 3.12.5
EN IN 19-01-2020 14:19
```

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

- File Explorer:** Shows files like `00_DELIVERY_COMPLETE.txt`, `00_START_HERE.md`, `COMPARISON_ANALYSIS.md`, `COMPLETION_CHECKLIST.md`, `DELIVERY_SUMMARY.txt`, `Email_Classification_Comparison_R...`, `Email_Classification_Prompt_Engine...`, `email_classification_results...`, `email_classification_system.py` (selected), `EXECUTIVE_SUMMARY.md`, `physics_test_queries.csv`, `physics_test_queries.json`, `PROJECT_INDEX.md`, `README_email_classification.md`, `reverse_string_function.py`, and `using_reversed_approaches.py`.
- Code Editor:** Displays `email_classification_system.py` with code for generating sample emails. It includes a comment for Jennifer Martinez and a sample email template.
- Terminal:** Shows the output of running the script, indicating success with "SAMPLE #4 - Category: Others".
- Search Bar:** Located at the top center.
- Chat Panel:** On the right, it says "All files are ready in: AIAC" and "Status: PRODUCTION READY | Quality: ★★★★★".
- Status Bar:** At the bottom right, it shows "In 663, Col 1 Spaces: 4 UFT-B CRLF Python 3.12.5" and a timestamp "19-01-2024".

File Edit Selection View Go Run Terminal Help

YISI.md email\_classification\_system.py

```

    def prepare_sample_emails(self) -> None:
        ...
        {
            "id": 5,
            "category": EmailCategory.BILLING.value,
            "email": "Subject: Subscription Cancellation Request"
        }
        ...
    
```

TERMINAL

```

SAMPLE #5 - Category: Billing
Subject: Subscription Cancellation Request
Hello,
I would like to cancel my Premium subscription effective immediately. I no longer need the service due to my company's restructuring. Please confirm the cancellation and let me know if there are any remaining charges for this month.
My subscription ID: 50B-2024-98765
Thank you,
Robert Kim

```

TEST EMAILS - FOR CLASSIFICATION

Indexing completed.

29°C Sunny

File Edit Selection View Go Run Terminal Help

YISI.md email\_classification\_system.py

```

    def prepare_test_emails(self) -> None:
        ...
        print(f" Prepared {len(self.test_emails)} test emails for classification")
        ...
    
```

TERMINAL

```

ZEROSHOT PROMPT (No Examples)
Classify the following customer email into one of these categories:
- Billing
- Technical Support
- Feedback
- Others
Provide ONLY the category name as your response, nothing else.
Email:
Subject: App Crashes on Startup
Hi,
My app keeps crashing immediately after I open it. I've tried restarting my device and reinstalling the application, but the problem continues. The error message says "Segmentation Fault" but I don't know what that means.
Can you help me fix this issue?

```

Indexing completed.

29°C Sunny

The screenshot shows a Visual Studio Code (VS Code) interface with several tabs open at the top:

- File
- Edit
- Selection
- View
- Go
- Run
- Terminal
- Help

The main editor area contains Python code for an email classification system:

```
YAML.md
string_reversal.approaches.py
Email_Classification_Prompt_Engineering.ipynb
email_classification_system.pyx
00_DELIVERY_COMPLETED
DELIVERY.S

20 class EmailClassificationSystem:
21
22     def create_one_shot_prompt(self, email: str, example_email: str, example_category: str) -> str:
23         """Create a one-shot prompt (one labeled example provided)."""
24         prompt = f"""Classify customer emails into one of these categories:
25
26             - Billing
27             - Technical Support
28             - Feedback
29             - Others
30
31             Example:
32             Email: {example_email}
33             Category: {example_category}
34
35             Now classify this email:
36             Email:
37             {email}
38
39             Category:"""
40
41             
```

The sidebar on the right includes a "PROMPT REFINEMENT..." section with checkboxes for "adapt to your needs" and "Quality verification (all tested)". Below that is a "Getting Started" section with numbered steps:

1. READ →
  - + 00\_START\_HERE.m
  - d [0 min]
2. RUN → python
- email\_classification\_system.py (2 min)
3. DECIDE → Use
  - + EXECUTIVE\_SUMMA
  - RVM
  - devision
  - framework
4. DEPLOY → Follow
  - + README\_Email\_Classifications.md (4-5 days)

At the bottom, there's a "Status: PRODUCTION READY | Quality: ★★★★★" indicator, a "Describe what to build next" input field, and a "Python Deb..." entry in the "powerhell" dropdown.

File Edit Selection View Go Run Terminal Help

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

ONE-SHOT PROMPT (1 Example)

Classify customer emails into one of these categories:

- Billing
- Technical Support
- Feedback
- Others

Example:  
Email: Subject: Invoice #INV-2026-0045 - Payment Issue  
Dear Support Team,

I received my invoice for the subscription renewal on January 15th, but I was charged twice for the same billing period. The first charge was on January 15th and the second on January 16th. Both transactions are showing on my account.

Could you please investigate this billing error and issue a refund for the duplicate charge? My account reference is CUST-78945.

Thank you for your prompt assistance.

Best regards,  
Sarah Johnson  
Category: Billing

Now classify this email:  
Email:  
Subject: App crashes on startup

Hi,

My app keeps crashing immediately after I open it. I've tried restarting my device  
... [truncated for display]

All files are ready in AIAC

Status PRODUCTION  
READY | Quality ★★★★★

AIAC

Indexing completed.

File Edit Selection View Go Run Terminal Help

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

AIAC

powerShell PowerShell Python Debug

PROMPT REFINE... (adapt to your needs) Quality verification (all tested)

Getting Started:

1. READ → \* 00\_START\_HERE.m  
d (3 min)
2. RUN → python  
email\_classification.py (2 min)
3. DECIDE → Use  
\* EXECUTIVE\_SUMMARY  
REady decision framework
4. DEPLOY → Follow  
\* README\_email\_Classification.md (4-5 days)

All files are ready in AIAC

Status PRODUCTION  
READY | Quality ★★★★★

AIAC

email\_classification.py  
Describe what to build new A... v A... v B... v C... v D... v E... v F... v G... v H... v I... v J... v K... v L... v M... v N... v O... v P... v Q... v R... v S... v T... v U... v V... v W... v X... v Y... v Z... v

Indexing completed.

File Edit Selection View Go Run Terminal Help

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

AIAC

powerShell PowerShell Python

Indexing completed.

File Edit Selection View Go Run Terminal Help

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

AIAC

00\_ANALYSIS.md string reversal approaches.py Email\_Classification\_Prompt\_Engineering.ipynb email\_classification\_system.py 00\_DELIVERY\_COMPLETE.txt DELIVERY\_SUMMARY.txt COMPI

email\_classification\_system.py > ...

```
20 class EmailClassificationSystem:
```

```
224
225     def create_few_shot_prompt(self, email: str, examples: List[Tuple[str, str]]) -> str:
226         """Create a few-shot prompt (multiple labeled examples provided)."""
227         examples_text = ""
228         for i, (example_email, example_category) in enumerate(examples, 1):
229             examples_text += f"\nExample {i}: {example_email}\nCategory: {example_category}\n"
230
231         prompt = f"""Classify customer emails into one of these categories:
232 - Billing
233 - Technical Support
234 - Feedback
235 - Others
236
237 {examples_text}
238
239 Now classify this email:
240 Email:
241 {email}
242 Category:"""
243         return prompt
244
245
```

All files are ready in AIAC

Status PRODUCTION  
READY | Quality ★★★★★

AIAC

email\_classification.py  
Describe what to build new A... v A... v B... v C... v D... v E... v F... v G... v H... v I... v J... v K... v L... v M... v N... v O... v P... v Q... v R... v S... v T... v U... v V... v W... v X... v Y... v Z... v

Indexing completed.

File Edit Selection View Go Run Terminal Help

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

AIAC

powerShell PowerShell Python

Indexing completed.

File Edit Selection View Go Run Terminal Help ⌘ ⌘ AIAC

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

FEW-SHOT PROMPT (3 Examples)

Classify customer emails into one of these categories:

- Billing
- Technical Support
- Feedback
- Others

Example 1:  
Email Subject: Invoice #DW-2026-9845 - Payment Issue

Dear Support Team,

I received my invoice for the subscription renewal on January 15th, but I was charged twice for the same billing period. The first charge was on January 16th and the second on January 15th. Both transactions are showing on my account.

Could you please investigate this billing error and issue a refund for the duplicate charge? My account reference is CUST-78945.

Thank you for your prompt assistance.

Best regards,  
Sarah Johnson  
Category: Billing

Example 2:  
Email Subject: Cannot Login - Error Code 502

Hello,

I've been unable to access my account for the past 2 hours. Every time I try to log in ... [truncated for display]

Indexing completed.

29°C Sunny

File Edit Selection View Go Run Terminal Help ⌘ ⌘ AIAC

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

00\_DELIVERY\_COMPLETE.txt email\_classification\_system.py EmailClassificationAnalysis.py email\_classification\_system.py

In 661, Col 1 Spaces: 4 UTF-8 CRLF Python 3.12.5

File Edit Selection View Go Run Terminal Help ⌘ ⌘ AIAC

EXPLORER

AIAC

00\_START\_HERE.md COMPARISON\_ANALYSIS.md COMPLETION\_CHECKLIST.md DELIVERY\_SUMMARY.txt

EmailClassificationAnalysis.py EmailClassificationSystem.py EXECUTIVE\_SUMMARY.md physics\_test\_queries.csv physics\_test\_queries.json PROJECT\_INDEX.md README\_email\_Classification.md reverse\_string\_function.py reverse\_string.py string\_reversal\_approaches.py

37/0 class EmailClassificationAnalysis:

```

400     def generate_comparison_table(self) -> str:
401         """Generate a comparison table of all techniques."""
402         responses = self.system.results.get("responses", [])
403         accuracy = self.calculate_accuracy()
404
405         table_lines = []
406         table_lines.append("\n" + "="*140)
407         table_lines.append("EMAIL CLASSIFICATION RESULTS - ZERO-SHOT vs ONE-SHOT")
408         table_lines.append("="*140)
409
410         # Group by test email
411         responses_by_email = {}
412         for response in responses:
413             email_id = response["test_email_id"]
414             if email_id not in responses_by_email:
415                 responses_by_email[email_id] = []
416             responses_by_email[email_id].append(response)
417
418         # Display results per email
419         for email_id in sorted(responses_by_email.keys()):
420             email_responses = responses_by_email[email_id]
421             true_category = email_responses[0]["true_category"]
422
423             table_lines.append(f"\nTEST EMAIL: {email_id}")

```

In 519, Col 47 Spaces: 4 UTF-8 CRLF Python 3.12.5

The screenshot shows a Jupyter Notebook interface with several open files. The current file is `email_classification_system.py`, which contains Python code for email classification analysis. The code includes functions for generating comparison tables and displaying analysis results. Other visible files include `string_reversal_approaches.py`, `Email_Classification_Prompt_Engineering.ipynb`, and `email_classification_system.py`. The interface includes a sidebar with project navigation and a status bar at the bottom.

```
370 class EmailClassificationAnalysis:
380     def generate_comparison_table(self) -> str:
381         table_lines = ["=" * 140]
382
383         for technique in ["zero_shot", "one_shot", "few_shot"]:
384             if technique in accuracy:
385                 stats = accuracy[technique]
386                 row = f'{technique.replace("_", "-").upper():<20} | {stats["com"}'
387                 table_lines.append(row)
388
389         table_lines.append("\n" + "="*140)
390
391         return "\n".join(table_lines)
392
393     def display_analysis(self) -> None:
394         """Display comprehensive analysis."""
395         print(self.generate_comparison_table())
396
397         accuracy = self.calculate_accuracy()
398
399         print("\n" + "="*140)
400         print("DETAILED EFFECTIVENESS ANALYSIS")
401         print("="*140)
402
403         print("\n🕒 ZERO-SHOT PROMPTING")
404         print("=" * 140)
```



```
File Edit Selection View Go Run Terminal Help <- > Q_AIMC email_classification_system.py > EmailClassificationAnalysis > display_recommendations
class EmailClassificationAnalysis:
    def display_recommendations(self) -> None:
        """Display recommendations based on analysis."""
        print("\n💡 WHEN TO USE EACH TECHNIQUE:\n")
        print("ZERO-SHOT:")
        print("✓ Quick classification for straightforward emails")
        print("✓ When computational resources are limited")
        print("✓ For high-level category detection")
        print("✗ Avoid for mission-critical classifications")
        print("✗ Not suitable when categories are similar or ambiguous")

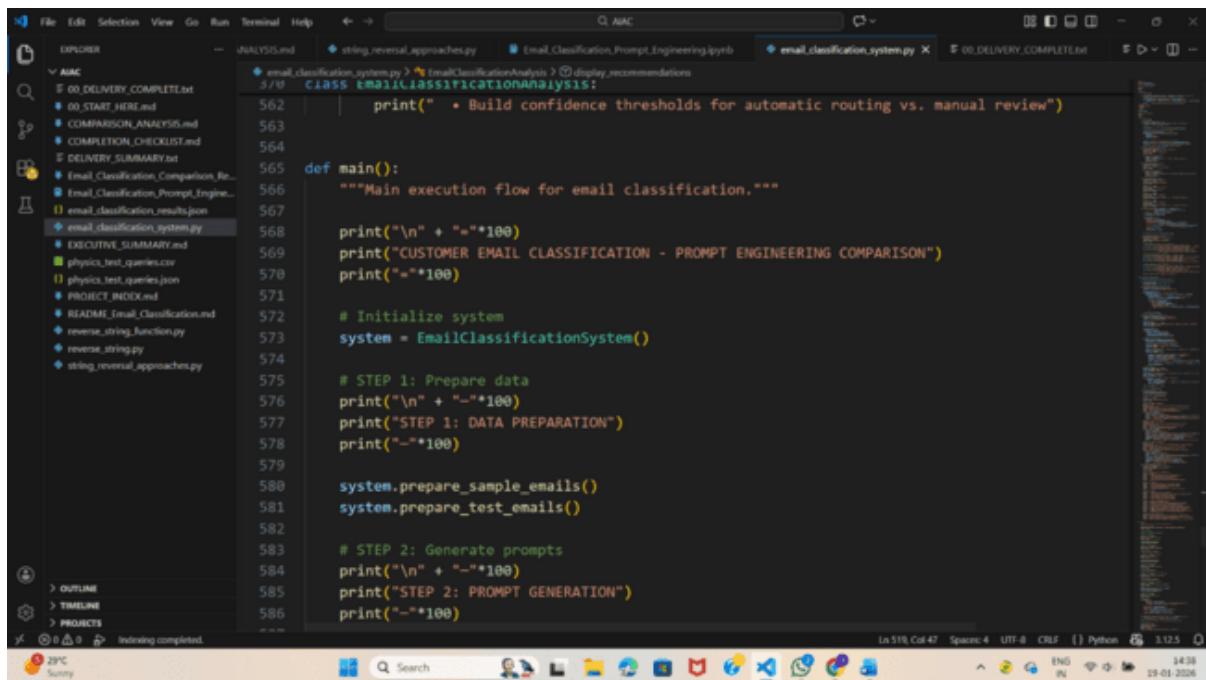
        print("\nONE-SHOT:")
        print("✓ Good balance between context and efficiency")
        print("✓ When slight improvement in accuracy is needed")
        print("✓ For moderately complex classification tasks")
        print("✗ May still miss edge cases")
        print("✗ Single example may not cover all category variations")

        print("\nFEW-SHOT:")
        print("✓ Highest accuracy and confidence")
```

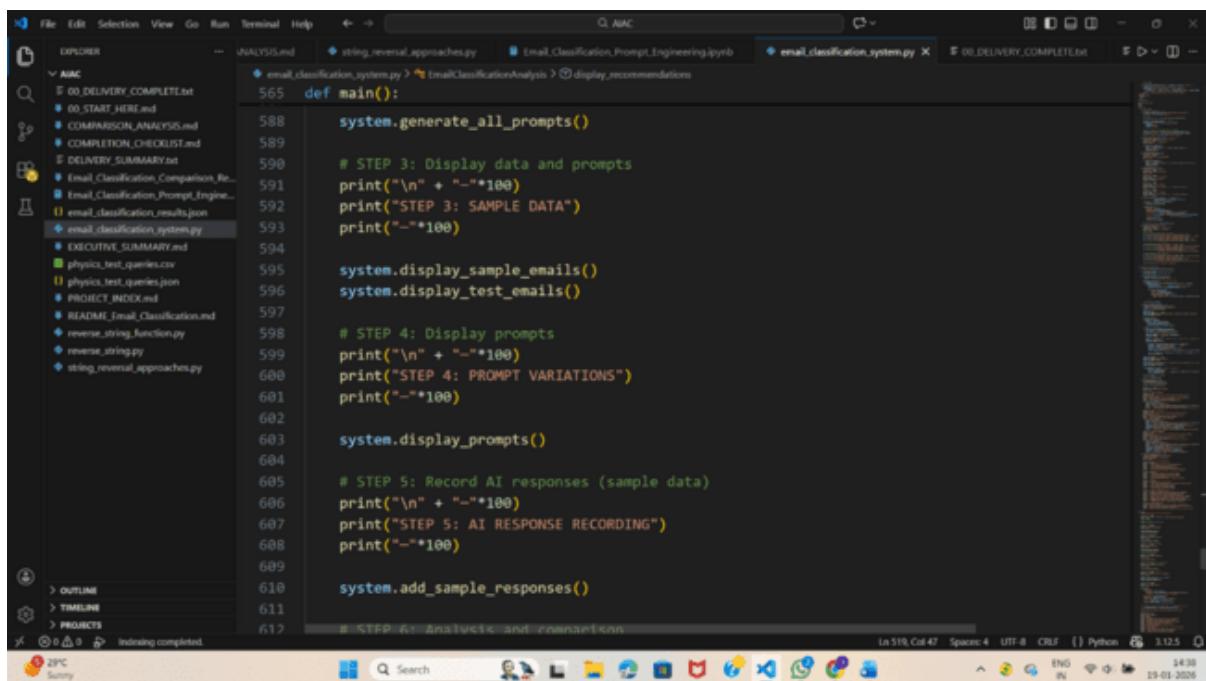
```
File Edit Selection View Go Run Terminal Help <- > Q_AIMC email_classification_system.py > EmailClassificationAnalysis > display_recommendations
class EmailClassificationAnalysis:
    def display_recommendations(self) -> None:
        print("✓ For critical classification (e.g., high-value customer issues)")
        print("✓ When category boundaries are fuzzy")
        print("✓ Most reliable for production systems")
        print("✗ Larger prompt size increases latency")
        print("✗ Higher computational cost per request")

        print("\n💡 PRACTICAL RECOMMENDATIONS:\n")
        print("1. START with zero-shot for rapid prototyping and validation")
        print("2. MEASURE accuracy and identify problematic email types")
        print("3. USE one-shot when zero-shot shows 85-90% accuracy")
        print("4. EMPLOY few-shot for production systems requiring >95% accuracy")
        print("5. SELECT diverse, representative examples for few-shot prompts")
        print("6. REGULARLY update examples as new email patterns emerge")
        print("7. COMBINE with confidence scores to flag uncertain classifications")
        print("8. CONSIDER human review for low-confidence predictions")

        print("\n💡 HYBRID APPROACH:\n")
        print("• Use zero-shot as initial filter for obvious classifications")
        print("• Escalate ambiguous cases (low confidence) to one-shot or few-shot")
        print("• Maintain human review queue for edge cases")
        print("• Build confidence thresholds for automatic routing vs. manual review")
```



```
562     print(" + Build confidence thresholds for automatic routing vs. manual review")
563
564
565 def main():
566     """Main execution flow for email classification."""
567
568     print("\n" + "="*100)
569     print("CUSTOMER EMAIL CLASSIFICATION - PROMPT ENGINEERING COMPARISON")
570     print("="*100)
571
572     # Initialize system
573     system = EmailClassificationSystem()
574
575     # STEP 1: Prepare data
576     print("\n" + "="*100)
577     print("STEP 1: DATA PREPARATION")
578     print("="*100)
579
580     system.prepare_sample_emails()
581     system.prepare_test_emails()
582
583     # STEP 2: Generate prompts
584     print("\n" + "="*100)
585     print("STEP 2: PROMPT GENERATION")
586     print("="*100)
```



```
588     system.generate_all_prompts()
589
590     # STEP 3: Display data and prompts
591     print("\n" + "="*100)
592     print("STEP 3: SAMPLE DATA")
593     print("="*100)
594
595     system.display_sample_emails()
596     system.display_test_emails()
597
598     # STEP 4: Display prompts
599     print("\n" + "="*100)
600     print("STEP 4: PROMPT VARIATIONS")
601     print("="*100)
602
603     system.display_prompts()
604
605     # STEP 5: Record AI responses (sample data)
606     print("\n" + "="*100)
607     print("STEP 5: AI RESPONSE RECORDING")
608     print("="*100)
609
610     system.add_sample_responses()
611
612     # STEP 6: ANALYTICS AND ITERATION
```

The screenshot shows a Jupyter Notebook interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** AIAC.
- Left Sidebar:** Explorer, File tree showing various .md and .py files, and a Projects section with OUTLINE, TIMELINE, and PROJECTS.
- Current Cell:** cell\_1, titled "string\_reversal\_approaches.py".
- Code Content:** The code performs the following steps:
  - Imports and initializes the system.
  - Prints analysis results and recommendations.
  - Exports results to a JSON-like structure.
- Right Panel:** A large panel displaying the execution results of the code, showing the output of each cell.
- Bottom Bar:** Includes a search bar, a progress bar indicating "Indirect completed.", and a status bar showing "In 510 Col 47 Spaces: 4 LIT: 8 CRU: 0 Python 3.12.5".

The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** On the left, it shows a tree view of files under the "AIAC" folder. The "email\_classification\_system.py" file is currently selected.
- Code Editor:** The main area displays the Python code for "email\_classification\_system.py". The code includes a main function that prints four steps for AI integration. It also contains a conditional check for the script's name.
- Search Bar:** At the top center, there is a search bar with the text "AIAC".
- Terminal:** A terminal window at the bottom left shows the message "Indexing completed."
- Bottom Status Bar:** The status bar at the bottom right shows the following information: Line 510, Column 47; Spaces 4; UTF-8; CR/LF; Python; and the date/time 09-01-2024 3:12:55.

The screenshot shows a Jupyter Notebook interface with several open files:

- `EXPLORER`: Shows files like `DO_DELIVERY_COMPLETE.txt`, `DO_START_HERE.rmd`, `COMPARISON_ANALYSIS.rmd`, `COMPLETION_CHECKLIST.rmd`, `DELIVERY_SUMMARY.txt`, `Email_Classification_Comparison_R...`, `Email_Classification_Prompt_Engine...`, `email_classification_results.json`, `email_classification_system.py`, `EXECUTIVE_SUMMARY.rmd`, `physics_test_queries.csv`, `physics_test_queries.json`, `PROJECT_INDEX.rmd`, `README_email_classification.rmd`, `reverse_string_function.py`, `reverse_string.py`, and `string_reversal_approaches.py`.
- `AIAC`: The active notebook contains sections for `SAMPLE AI RESPONSES (PLACEHOLDERS)` and `STEP 6: RESULTS ANALYSIS & COMPARISON`. It includes a table comparing `EMAIL CLASSIFICATION RESULTS - ZERO-SHOT vs ONE-SHOT vs FEW-SHOT` across different techniques.
- `email_classification_system.py`: A Python script for email classification.
- `email_classification_system.ipynb`: A Jupyter Notebook for email classification.
- `email_classification_systems.py`: A Python script for email classification systems.

The right sidebar displays a `PROMPT REFERENCE` with examples for `DO_START_HERE.rmd`, `EXECUTIVE_SUMMARY.rmd`, `physics_test_queries.csv`, and `PROJECT_INDEX.rmd`. It also shows a `Getting Started` section with steps for `READ`, `TEST`, `DECIDE`, and `DEPLOY`. The status bar at the bottom indicates the notebook is ready and has 4 stars.

File Edit Selection View Go Run Terminal Help

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

1 ISSUE... | Technical Support | ✓ YES | 0.92 | Correctly identified from keywords like 'App Cras...  
hes', ...

TEST (EMAIL) TEST\_2  
True Category: Feedback

Technique	Predicted	Correct	Confidence	Notes
FEW-SHOT	Feedback	✓ YES	0.96	Few examples provided diverse feedback patterns;
Very C...				
ONE-SHOT	Feedback	✓ YES	0.93	One example strengthened classification; better c...
on file...				
ZERO-SHOT	Feedback	✓ YES	0.88	Keywords 'suggestion', 'feature' helped; could ha...
ords; ...				

TEST (EMAIL) TEST\_3  
True Category: Billing

Technique	Predicted	Correct	Confidence	Notes
FEW-SHOT	Billing	✓ YES	0.97	Multiple billing examples provided clear pattern;
High...				
ONE-SHOT	Billing	✓ YES	0.91	One example improved accuracy; reduced ambiguity
ord; ...				
ZERO-SHOT	Billing	✓ YES	0.85	Identified from 'refund', 'order', 'invoice' keyw...

SUMMARY STATISTICS

All files are ready in AIAC

Status: PRODUCTION  
READY | Quality: ★★★★★

Describe what to build next

AIAC

Indexing completed.

29°C Sunny

File Edit Selection View Go Run Terminal Help

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

00\_DELIVERY\_COMPLETE.txt  
00\_START\_HERE.md  
COMPARISON\_ANALYSIS.md  
COMPLETION\_CHECKLIST.md  
DELIVERY\_SUMMARY.txt  
Email\_Classification\_Comparison\_Re...  
Email\_Classification\_Prompt\_Engine...  
email\_classification\_results.json  
email\_classification\_system.py  
EXECUTIVE\_SUMMARY.md  
physics\_test\_queries.csv  
physics\_test\_queries.json  
PROJECT\_INDEX.md  
README\_email\_Classification.md  
reverse\_string\_function.py  
reverse\_string.py  
string\_reversal\_approaches.py

SUMMARY STATISTICS

Technique	correct/total	Accuracy %	Avg Confidence
ZERO-SHOT	1/3	100.0	0.88
ONE-SHOT	1/3	100.0	0.93
FEW-SHOT	1/3	100.0	0.97

DETAILED EFFECTIVENESS ANALYSIS

**ZERO-SHOT PROMPTING**

Definition: Classification without providing any labeled examples.  
Characteristics:

- Relies on model's intrinsic knowledge of category definitions
- Fastest approach (minimal prompt length)
- Most general; may struggle with ambiguous emails

Results for this dataset:

- Accuracy: 100.0% (3/3 correct)
- Average Confidence: 0.88/1.0
- Effectiveness: BASELINE - Lower confidence, may require clarification in borderline cases

All files are ready in AIAC

Status: PRODUCTION  
READY | Quality: ★★★★★

Describe what to build next

AIAC

Indexing completed.

29°C Sunny

**AIAC**

File Edit Selection View Go Run Terminal Help

email\_classification\_system.py

RECOMMENDATIONS & BEST PRACTICES

WHEN TO USE EACH TECHNIQUE:

**ZERO-SHOT:**

- ✓ Quick classification for straightforward emails
- ✓ When computational resources are limited
- ✓ For high-level category detection
- X Not ideal for mission-critical classifications
- X Not suitable when categories are similar or ambiguous

**ONE-SHOT:**

- ✓ Good balance between context and efficiency
- ✓ When slight improvement in accuracy is needed
- ✓ For moderately complex classification tasks
- X May still miss edge cases
- X Single example may not cover all category variations

**FEW-SHOT:**

- ✓ Highest accuracy and confidence
- ✓ For critical classification (e.g., high-value customer issues)
- ✓ When category boundaries are fuzzy
- ✓ Most reliable for production systems
- X Larger prompt size increases latency
- X Higher computational cost per request

PRACTICAL RECOMMENDATIONS:

1. START with zero-shot for rapid prototyping and validation
2. MEASURE accuracy and identify problematic email types

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

powerShell PowerShell Python Deb...

Getting Started:

1. READ →
  - 00\_START\_HERE.md
  - d (3 min)
2. RUN → python email\_classification\_system.py (2 min)
3. DECIDE → Use
  - + EXECUTIVE\_SUMMARY.md decision framework
4. DEPLOY → Follow
  - + README\_email\_classification.md (4-5 days)

All files are ready in AIAC

Status PRODUCTION READY | Quality ★★★★★

Describe what to build next

AIAC

File Edit Selection View Go Run Terminal Help

email\_classification\_system.py

ONE-SHOT PROMPTING

Definition: Classification with ONE labeled example provided.

Characteristics:

- Provides single reference point for pattern matching
- Moderate improvement with minimal context
- Helps disambiguate similar categories

Results for this dataset:

- Accuracy: 100.0% (3/3 correct)
- Average Confidence: 0.99/1.0
- Improvement over zero-shot: +0.0%
- Effectiveness: MODERATE - Better confidence and accuracy than zero-shot

FEW-SHOT PROMPTING

Definition: Classification with TWO OR MORE labeled examples provided.

Characteristics:

- Demonstrates broader pattern coverage across categories
- Higher confidence and accuracy
- More context increases prompt size but improves reliability

Results for this dataset:

- Accuracy: 100.0% (3/3 correct)
- Average Confidence: 0.97/1.0
- Improvement over zero-shot: +0.0%
- Effectiveness: HIGHEST - Best accuracy and confidence

File Edit Selection View Go Run Terminal Help

email\_classification\_system.py

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

powerShell PowerShell Python Deb...

Getting Started:

1. READ →
  - 00\_START\_HERE.md
  - d (3 min)
2. RUN → python email\_classification\_system.py (2 min)
3. DECIDE → Use
  - + EXECUTIVE\_SUMMARY.md decision framework
4. DEPLOY → Follow
  - + README\_email\_classification.md (4-5 days)

All files are ready in AIAC

Status PRODUCTION READY | Quality ★★★★★

Describe what to build next

AIAC

The screenshot shows the AIAC (AI Assistant) interface. The left sidebar displays a project structure under 'EXPLORER' for 'AIAC'. The main area shows two terminal tabs: 'email\_classification\_system.py' and 'string\_reversal\_approaches.py'. The 'email\_classification\_system.py' tab contains a summary of the project's approach and execution results. It includes sections for 'START', 'MEASURE', 'EMPLOY', 'SELECT', 'COMBINE', and 'HUMAN REVIEW'. Below this is a 'MIXED APPROACH' section with steps for zero-shot, one-shot, and few-shot classification, along with a note about maintaining a review queue for edge cases. A 'STEP 7: EXPORT RESULTS' section shows that 'results exported to email\_classification\_results.json'. The 'EXECUTION COMPLETE' section lists generated files: 'email\_classification\_results.json - All results and data'. The right sidebar provides a 'Getting Started' guide with four numbered steps: 1. READ (00\_START\_HERE.md), 2. RUN (python email\_classification\_system.py), 3. DECIDE (EXECUTIVE\_SUMMARY.md), and 4. DEPLOY (README\_email\_Classification.md). A status bar at the bottom indicates 'All files are ready' and shows system information like CPU, RAM, and date.

This screenshot is nearly identical to the one above, showing the same AIAC interface. The project structure in the Explorer panel is identical. The terminals show the same output for 'email\_classification\_system.py' and 'string\_reversal\_approaches.py'. The main content area shows the project summary, execution results, and generated files. The right sidebar's 'Getting Started' guide remains the same. The status bar at the bottom also remains consistent, showing 'All files are ready' and system details.

## Task-2

### Intent Classification for Chatbot Queries

A company wants to deploy a chatbot to handle customer queries.

Each query must be classified into one of the following intents:

Account Issue, Order Status, Product Inquiry, or General Question

using prompt engineering techniques.

### Tasks to be Completed

#### 1. Prepare Sample Data

Create 6 short chatbot user queries, each mapped to one of the four intents.

#### 2. Zero-shot Prompting

Design a prompt that asks the LLM to classify a user query into the given intent categories without examples.

#### 3. One-shot Prompting

Provide one labeled query in the prompt before classifying a new query.

#### 4. Few-shot Prompting

Include 3–5 labeled intent examples to guide the LLM before classifying a new query.

#### 5. Evaluation

Apply all three techniques to the same set of test queries and document differences in performance.

The screenshot shows a Jupyter Notebook interface with several tabs open. The main code editor contains the following Python code:

```
20 class ChatbotIntentClassifier:
21     def prepare_sample_queries(self) -> None:
22         """Prepare 6 sample chatbot queries, each for one intent."""
23         self.sample_queries = [
24             {
25                 "id": 1,
26                 "intent": IntentType.ACCOUNT_ISSUE.value,
27                 "query": "I forgot my password and can't log into my account. How do I reset it?"
28             },
29             {
30                 "id": 2,
31                 "intent": IntentType.ORDER_STATUS.value,
32                 "query": "Where is my order #ORD-2026-0845? I placed it 3 days ago."
33             },
34             {
35                 "id": 3,
36                 "intent": IntentType.PRODUCT_INQUIRY.value,
37                 "query": "Does the premium model come in blue? What's the size of the screen?"
38             },
39             {
40                 "id": 4,
41                 "intent": IntentType.GENERAL_QUESTION.value,
42                 "query": "What are your business hours today?"
43             },
44             {
45                 "id": 5,
46                 "intent": IntentType.GENERAL_QUESTION.value,
47                 "query": "I need to update my billing address on file."
48             }
49         ]
50
51         print(f"\n\n{len(self.sample_queries)} sample queries across all intents")
```

The notebook sidebar shows a list of cells with status indicators (green checkmarks). A message at the bottom right says: "Now let me run the notebook to verify all cells execute properly." Below that, there are buttons for "Run Notebook Cell", "Running imports and setup", and "Allow" or "Skip".

This screenshot is nearly identical to the one above, showing the same code in the main editor and the same execution status in the sidebar. The only difference is the final line of code, which has been modified to:

```
51         print(f"\n\n{len(self.sample_queries)} sample queries across all intents")
```

The notebook sidebar shows a list of cells with status indicators (green checkmarks). A message at the bottom right says: "Now let me run the notebook to verify all cells execute properly." Below that, there are buttons for "Run Notebook Cell", "Running imports and setup", and "Allow" or "Skip".

File Edit Selection View Go Run Terminal Help

intent\_classification\_system.ipynb Intent\_Classification\_Prompt\_Engineering.ipynb intent\_classification\_system.ipynb Chatbot\_Intent\_Classification.ipynb

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

SAMPLE QUERIES - TRAINING DATA

\* SAMPLE #1 - Intent: Account Issue  
Query: I forgot my password and can't log into my account. How do I reset it?

\* SAMPLE #2 - Intent: Order Status  
Query: Where is my order #ORD-2026-0845? I placed it 3 days ago.

\* SAMPLE #3 - Intent: Product Inquiry  
Query: Does the premium model come in blue? What's the size of the screen?

\* SAMPLE #4 - Intent: General Question  
Query: What are your business hours today?

\* SAMPLE #5 - Intent: Account Issue  
Query: I need to update my billing address on file.

\* SAMPLE #6 - Intent: Order Status  
Query: Can I cancel or modify my recent order?

Indexing completed.

File Edit Selection View Go Run Terminal Help

intent\_classification\_system.ipynb Intent\_Classification\_Prompt\_Engineering.ipynb intent\_classification\_system.ipynb Chatbot\_Intent\_Classification.ipynb

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

CHAT PROMPT REFINEMENT AND EVALUATION

powerShell PowerShell Python Debug

Intent Classification Prompt Engineering.ipynb

Intent Classification Prompt Engineering.ipynb

Intent Classification Prompt Engineering.ipynb

Intent Classification Prompt Engineering.ipynb

Now let me run the notebook to verify all cells execute properly:

Retrieving Notebook summary.

Good! The notebook is now complete with 19 cells. Let me run the key cells to verify everything works. I'll start with the imports:

Run Notebook Cell

Running imports and setup

# Intent Classification Show More

Allow Skip

Describe what to build next

Agent Auto Python 3.12.5

ENGLISH IN 13-01-2026

File Edit Selection View Go Run Terminal Help

email\_Classification\_Prompt\_Engineering.ipynb intent\_classification\_system.ipynb Intent\_Classification\_Prompt\_Engineering.ipynb intent\_classification\_system.ipynb Chatbot\_Intent\_Classification.ipynb

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

RECENT SESSIONS

- Baseline prompt testing for physics... Local + 1 hr
- Prompt Refinement and Evaluation S... Completed Local + 16 mins
- Python program to reverse a string ... Completed Local + 1 wk

Show More

Build with Agent

All responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

ZEPHYRUS Sunny

File Edit Selection View Go Run Terminal Help

intent\_classification\_system.ipynb Intent\_Classification\_Prompt\_Engineering.ipynb intent\_classification\_system.ipynb Chatbot\_Intent\_Classification.ipynb

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

ZERO-SHOT PROMPT (No Examples)

Classify the following chatbot user query into one of these intents:  
 - Account Issue (account problems, password reset, profile updates, security concerns)  
 - Order Status (order tracking, delivery status, order modifications)  
 - Product Inquiry (product features, specifications, compatibility, availability)  
 - General Question (business info, hours, policies, contact information)

Provide ONLY the intent category name as your response, nothing else.

User Query: `{query}`

Intent:  
 return prompt

Provide ONLY the intent category name as your response, nothing else.

User Query: My account shows a suspicious login from another location. What should I do?

Indexing completed.

File Edit Selection View Go Run Terminal Help

intent\_classification\_system.ipynb Intent\_Classification\_Prompt\_Engineering.ipynb intent\_classification\_system.ipynb Chatbot\_Intent\_Classification.ipynb

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

CHAT PROMPT REFINEMENT AND EVALUATION

powerShell PowerShell Python Debug

Intent Classification Prompt Engineering.ipynb

Intent Classification Prompt Engineering.ipynb

Intent Classification Prompt Engineering.ipynb

Intent Classification Prompt Engineering.ipynb

Describe what to build next

Agent Auto Python 3.12.5

ENGLISH IN 13-01-2026

Screenshot of the Visual Studio Code (VS Code) interface. The main area shows Python code for a ChatbotIntentClassifier class. The terminal at the bottom right shows the command "Indexing completed". The status bar indicates the file is 121 lines long, has 4 spaces per tab, is in UTF-8 encoding, and is using Python 3.12.5. A sidebar on the right lists recent sessions, and a "Build with Agent" panel is visible.

```
def create_one_shot_prompt(self, query: str, example_query: str, example_intent: str) -> str:
    """Create a one-shot prompt (one labeled example provided)."""
    prompt = f"""Classify chatbot user queries into one of these intents:
- Account Issue
- Order Status
- Product Inquiry
- General Question
Example:
Query: {example_query}
Intent: {example_intent}
Now classify this query:
Query: {query}
Intent:"""
    return prompt
```

Screenshot of the Visual Studio Code (VS Code) interface, identical to the first one but with a different set of terminal output. The terminal at the bottom right shows the command "Indexing completed". The status bar indicates the file is 121 lines long, has 4 spaces per tab, is in UTF-8 encoding, and is using Python 3.12.5. A sidebar on the right lists recent sessions, and a "Build with Agent" panel is visible.

```
def create_one_shot_prompt(self, query: str, example_query: str, example_intent: str) -> str:
    """Create a one-shot prompt (one labeled example provided)."""
    prompt = f"""Classify chatbot user queries into one of these intents:
- Account Issue
- Order Status
- Product Inquiry
- General Question
Example:
Query: {example_query}
Intent: {example_intent}
Now classify this query:
Query: My account shows a suspicious login from another location. What should I do?
Intent:
... [truncated for display]
```

The screenshot shows a Python file named `ChatbotIntentClassifier` in the main editor area. The code defines a class `ChatbotIntentClassifier` with a method `create_few_shot_prompt`. This method generates a prompt for a few-shot learning scenario based on provided examples. It also includes a block of code to classify user queries into intents: Account Issue, Order Status, Product Inquiry, and General Question. The code uses f-strings and tuple unpacking. Below the code, there's a snippet for classifying a query. The bottom left shows a sidebar with recent sessions and a 'Build with Agent' feature. The bottom right shows a terminal window with the command `python intent_classification_system.py` and a status bar indicating the file is indexed.

```
File Edit Selection View Go Run Terminal Help < > Q AI AC & Email_Classification_Prompt_Engineering.pyw email_classification_system.py Intent_Classification_Prompt_Engineering.pyw intent_classification_system.py ChatbotIntentClassifier > create_few_shot_prompt 20 class ChatbotIntentClassifier:  
129     def create_few_shot_prompt(self, query: str, examples: List[Tuple[str, str]]) -> str:  
130         """Create a few-shot prompt (multiple labeled examples provided)."""  
131         examples_text = ""  
132         for i, (example_query, example_intent) in enumerate(examples, 1):  
133             examples_text += f"\nExample {i}:\nQuery: {example_query}\nIntent: {example_intent}  
134  
135         prompt = f"""Classify chatbot user queries into one of these intents:  
136 - Account Issue  
137 - Order Status  
138 - Product Inquiry  
139 - General Question  
140  
141 {examples_text}  
142  
143 Now classify this query:  
144 Query: {query}  
145  
146 Intent: """  
147         return prompt  
148  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + v - | x x  
Account Issue  
Order Status  
Product Inquiry  
General Question  
powershell powershell Python Deb...  
Describe what to build next  
Agent Auto & 29-01-2020  
Indexing completed. In 12L, Col 1 Spaces: 4 UTF-8 CR/LF {} Python 3.12.5 15:16  
29PC Sunny
```

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with several windows open. The main editor window displays Python code for a 'ChatbotIntentClassifier' class, specifically the 'create\_few\_shot\_prompt' method. Below the code, a 'PROBLEMS' panel is visible. To the right, a sidebar titled 'RECENT SESSIONS' lists three entries: 'Baseline prompt testing for physics q...', 'Prompt Refinement and Evaluation S...', and 'Python program to reverse a string ...'. A large central area contains a 'Few-shot Prompt (3 Examples)' section, which lists four examples for classifying chatbot user queries into intents like Account Issue, Order Status, Product Inquiry, and General Question. Each example includes a query, response, and intent label. At the bottom right, there's a 'Build with Agent' feature with a message about AI responses being inaccurate and instructions to onboard AI into your database. The status bar at the bottom shows file statistics and system information.

File Edit Selection View Go Run Terminal Help ⏪ ⏩ Q AIAC

Email\_Classification\_Prompt\_Engineering.ipynb email\_classification\_system.py Intent\_Classification\_Prompt\_Engineering.ipynb intent\_classification\_system.py

intent\_classification\_system.py > ChatbotIntentClassifier > create\_few\_shot\_prompt

20 class ChatbotIntentClassifier:

129 def create\_few\_shot\_prompt(self, query: str, examples: List[Tuple[str, str]]) -> str:

130 """Create a few-shot prompt (multiple labeled examples provided)."""

131 prompt = f"

132 FEW-SHOT PROMPT (3 Examples)

133 Classify chatbot user queries into one of these intents:

134 - Account Issue

135 - Order Status

136 - Product Inquiry

137 - General Question

138

139 Example 1:

140 Query: I forgot my password and can't log into my account. How do I reset it?

141 Intent: Account Issue

142

143 Example 2:

144 Query: Where is my order #ORD-2026-0845? I placed it 3 days ago.

145 Intent: Order Status

146

147 Example 3:

148 Query: Does the premium model come in blue? What's the size of the screen?

149 Intent: Product Inquiry

150

151 Now classify this query:

152 Query: My account shows a suspicious login from another location. What should I do?

153 Intent:

154 ... [truncated for display]

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

RECENT SESSIONS

Baseline prompt testing for physics q... Input needed Local + 1 hr

Prompt Refinement and Evaluation S... Completed Local + 22 mins

Python program to reverse a string ... Completed Local + 1 wk

Show More

Few-shot Prompt (3 Examples)

Classify chatbot user queries into one of these intents:

- Account Issue
- Order Status
- Product Inquiry
- General Question

Example 1:

Query: I forgot my password and can't log into my account. How do I reset it?

Intent: Account Issue

Example 2:

Query: Where is my order #ORD-2026-0845? I placed it 3 days ago.

Intent: Order Status

Example 3:

Query: Does the premium model come in blue? What's the size of the screen?

Intent: Product Inquiry

Now classify this query:

Query: My account shows a suspicious login from another location. What should I do?

Intent:

... [truncated for display]

Build with Agent

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your database.

intent\_classification\_system.py

Describe what to build next

Agent Auto

Indexing completed.

29PC Sunn

Search

ENG IN 19-01-2026

13:12:55

```
20 class ChatbotIntentClassifier:
241     def add_sample_classifications(self) -> None:
242         """Add sample classification results."""
243         print("\n" + "="*100)
244         print("SAMPLE CLASSIFICATIONS (PLACEHOLDERS)")
245         print("="*100)
246
247         # Sample classifications for TEST_1 (Account Issue)
248         self.record_classification("zero_shot", "TEST_1", "Account Issue", 0.94,
249                                     "Correctly identified from 'suspicious login' and 'security'")
250         self.record_classification("one_shot", "TEST_1", "Account Issue", 0.96,
251                                     "Example improved confidence; security concern pattern recog")
252         self.record_classification("few_shot", "TEST_1", "Account Issue", 0.98,
253                                     "Multiple examples provided strong security pattern")
254
255         # Sample classifications for TEST_2 (Product Inquiry)
256         self.record_classification("zero_shot", "TEST_2", "Product Inquiry", 0.89,
257                                     "Keywords 'compatible', 'devices' identified; good accuracy")
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

-----  
INTENT CLASSIFICATION RESULTS - COMPARISON TABLE  
-----

TEST 1: My account shows a suspicious login from another i...  
True Intent: Account Issue

Indexing completed.

Build with Agent

AI responses may be inaccurate.  
Generate Agent Instructions to onboard AI onto your codebase.

Python Dev...

In 121, Col 1 Spaces: 4 UTT-B CRLF { Python 11.2.5 ENG IN 15:19 29-01-2026

```
20 class ChatbotIntentClassifier:
241     def add_sample_classifications(self) -> None:
242         """Add sample classification results."""
243         self.record_classification("one_shot", "TEST_2", "Product Inquiry", 0.92,
244                                     "One example improved pattern recognition")
245         self.record_classification("few_shot", "TEST_2", "Product Inquiry", 0.95,
246                                     "Few examples provided comprehensive product inquiry patterns")
247
248         # Sample classifications for TEST_3 (Order Status)
249         self.record_classification("zero_shot", "TEST_3", "Order Status", 0.87,
250                                     "Identified from 'delivered', 'package' keywords")
251         self.record_classification("one_shot", "TEST_3", "Order Status", 0.91,
252                                     "Example strengthened delivery tracking pattern")
253         self.record_classification("few_shot", "TEST_3", "Order Status", 0.96,
254                                     "Multiple order tracking examples highly effective")
255
256         # Sample classifications for TEST_4 (General Question)
257         self.record_classification("zero_shot", "TEST_4", "General Question", 0.85,
258                                     "Correctly classified location/store inquiry")
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

-----  
INTENT CLASSIFICATION RESULTS - COMPARISON TABLE  
-----

TEST 1: My account shows a suspicious login from another i...  
True Intent: Account Issue

Indexing completed.

Build with Agent

AI responses may be inaccurate.  
Generate Agent Instructions to onboard AI onto your codebase.

Python Dev...

In 121, Col 1 Spaces: 4 UTT-B CRLF { Python 11.2.5 ENG IN 15:19 29-01-2026

The screenshot shows a Jupyter Notebook interface with multiple code cells and an AI sidebar.

**Code Cells:**

```
File Edit Selection View Go Run Terminal Help <--> QI AAC  
Email_Classification_Prompt_Engineering.ipynb email_classification_system.py Intent_Classification_Prompt_Engineering.ipynb intent.classification.system.py
```

```
Intent_Classification_System.ipynb > ChatbotIntentClassifier > create_one_shot_prompt  
20 class ChatbotIntentClassifier:  
241     def add_sample_classifications(self) -> None:  
272         self.record_classification("zero_shot", "TEST_4", "General Question", 0.85,  
273                                  "Correctly classified location/store inquiry")  
274         self.record_classification("one_shot", "TEST_4", "General Question", 0.89,  
275                                  "Example helped with general question pattern")  
276         self.record_classification("few_shot", "TEST_4", "General Question", 0.93,  
277                                  "Few examples clarified general inquiry patterns")  
278  
279         # Sample classifications for TEST_5 (Account Issue)  
280         self.record_classification("zero_shot", "TEST_5", "Account Issue", 0.90,  
281                                  "Identified from 'delete account' keywords")  
282         self.record_classification("one_shot", "TEST_5", "Account Issue", 0.93,  
283                                  "Account management pattern reinforced")  
284         self.record_classification("few_shot", "TEST_5", "Account Issue", 0.97,  
285                                  "Strong account action pattern from examples")  
286  
287         print("\nAdded sample classification results")
```

**PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS**

**INTENT CLASSIFICATION RESULTS - COMPARISON TABLE**

TEST\_3: My account shows a suspicious login from another location. True Intent: Account Issue

**AI Assistant:**

CHAT + - X

RECENT SESSIONS

- Baseline prompt testing for physics questions... Input needed. Local + 1 hr
- Prompt Refinement and Evaluation Script... Completed Local + 24 mins
- Python program to reverse a string... Completed Local + 1 hr

Show More

Build with Agent

All responses may be inaccurate.  
Generate Agent Instructions to onboard AI onto your codebase.

intent.classification.system.py

Describe what to build next  
Agent Auto

Natty w/ncap 0.84% Indexing completed.

Search

10 121 Col 1 Space: 4 UFT-B CRLF { } Python 3.12.5 Q

The screenshot shows a Jupyter Notebook interface with several tabs open. The active tab contains Python code for analyzing intent classification results. The code defines a class `IntentClassificationAnalysis` with methods for initializing a classifier, calculating metrics, and comparing results across different techniques (zero\_shot, one\_shot, few\_shot). The interface includes a sidebar for recent sessions, a bottom navigation bar with tabs like PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS, and a status bar at the bottom.

```
File Edit Selection View Go Run Terminal Help < > QI_AAC Email_Classification_Prompt_Engineering.ipynb email_classification_system.py Intent_Classification_Prompt_Engineering.ipynb intent_classification_system.py CHAT RECENT SESSIONS Baseline prompt testing for physics q... Input needed. Local + 1 hr Prompt Refinement and Evaluation S... Completed Local + 24 mins Python program to reverse a string ... Completed Local + 1 wk Show More
```

```
290 class IntentClassificationAnalysis:
```

```
291     """Analyze and compare intent classification results."""
```

```
292
```

```
293     def __init__(self, classifier: ChatbotIntentClassifier):
```

```
294         self.classifier = classifier
```

```
295
```

```
296     def calculate_metrics(self) -> Dict[str, Any]:
```

```
297         """Calculate accuracy and performance metrics."""
```

```
298         classifications = self.classifier.results.get("classifications", [])
```

```
299
```

```
300         metrics_by_technique = {}
```

```
301         for technique in ["zero_shot", "one_shot", "few_shot"]:
```

```
302             tech_classifications = [c for c in classifications if c["technique"] == technique]
```

```
303             if tech_classifications:
```

```
304                 correct_count = sum(1 for c in tech_classifications if c["correct"])
```

```
305                 accuracy = correct_count / len(tech_classifications) * 100
```

```
306                 avg_confidence = sum(c["confidence"] for c in tech_classifications) / len(tech,
```

```
307
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

INTENT CLASSIFICATION RESULTS - COMPARISON TABLE

TEST\_3: My account shows a suspicious login from another location. True Intent: Account issue

Indexing completed.

Natty ariapac -0.84%

Search

ENGLISH IN 19-01-2026 15:19

The screenshot shows a developer's workspace with multiple tabs open in an IDE. The active tab contains Python code for intent classification analysis. The code includes methods for calculating metrics and displaying comparison tables. A sidebar on the right provides AI-generated suggestions and session history. The bottom of the screen features a terminal window showing command-line interactions and a status bar indicating file paths and system information.

```
File Edit Selection View Go Run Terminal Help ⇧ ⌘ AIAC

Email_Classification_Prompt_Engineering.pyw email_classification_system.py Intent_Classification_Prompt_Engineering.pyw intent_classification_system.py

intents_classification_system.py > ChatbotIntentClassifier > create_one_shot_prompt
290 class IntentClassificationAnalysis:
291     def calculate_metrics(self) -> Dict[str, Any]:
292         metrics_by_technique[technique] = {
293             "correct": correct_count,
294             "total": len(tech_classifications),
295             "accuracy_percentage": round(accuracy, 2),
296             "average_confidence": round(avg_confidence, 3),
297             "classifications": tech_classifications
298         }
299
300         return metrics_by_technique
301
302     def display_comparison_table(self) -> None:
303         """Display comparison results."""
304         metrics = self.calculate_metrics()
305         classifications = self.classifier.results.get("classifications", [])
306
307         # Group by test query
308
309         print("INTENT CLASSIFICATION RESULTS - COMPARISON TABLE")
310
311         for test_query, metrics in metrics.items():
312             print(f"\nTest Query: {test_query}")
313             print("Technique\tCorrect\tTotal\tAccuracy (%)")
314             for technique, values in metrics.items():
315                 print(f"{technique}\t{values['correct']}\t{values['total']}\t{values['accuracy_percentage']}%")
316
317             print("\nOverall Metrics:")
318             print(f"Total Test Queries: {len(classifications)}")
319             print(f"Total Correct Predictions: {sum([metric['correct'] for metric in metrics.values()])}")
320             print(f"Average Accuracy: {round((sum([metric['correct'] for metric in metrics.values()]) / len(classifications)) * 100, 2)}%")
321
322             print("\nClassification Results:")
323             for classification in classifications:
324                 print(f"Test Query: {classification['test_query']}, Intent: {classification['intent']}, Prediction: {classification['predicted_intent']}, Confidence: {classification['confidence']}")
325
326             print("\n---")
327
328         print("Overall Summary:")
329         print(f"Total Test Queries: {len(classifications)}")
330         print(f"Total Correct Predictions: {sum([metric['correct'] for metric in metrics.values()])}")
331         print(f"Average Accuracy: {round((sum([metric['correct'] for metric in metrics.values()]) / len(classifications)) * 100, 2)}%")
332
333         print("\nClassification Results:")
334         for classification in classifications:
335             print(f"Test Query: {classification['test_query']}, Intent: {classification['intent']}, Prediction: {classification['predicted_intent']}, Confidence: {classification['confidence']}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + - x
INTENT CLASSIFICATION RESULTS - COMPARISON TABLE
TEST_1: My account shows a suspicious login from another ...
TRUE INTENT: ACCOUNT ISSUE

CMD RECENT SESSIONS
Baseline prompt testing for physics q...
Input needed. Local + 1 hr
Prompt Refinement and Evaluation 5...
Compiled Local + 24 mins
Python program to reverse a string ...
Compiled Local + 1 wk
Show More

Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

intent_classification_system.pyw
Describe what to build next
Agent Auto ↴

Natty wncap 0.84% Indexing completed.
Search
In 121, Col 1 Spaces: 4 UFT-8 CR/LF {} Python 3.12.5
ENGLISH IN 15:19
```

File Edit Selection View Go Run Terminal Help ⌘ ⌘ ⌘

Email\_Classification\_Prompt\_Engineering.ipynb email\_classification\_system.py Intent\_Classification\_Prompt\_Engineering.ipynb intent\_classification\_system.py

```

299 class IntentClassificationAnalysis:
318     def display_comparison_table(self) -> None:
329
340         query_clfs = by_query[query_id]
341         true_intent = query_clfs[0]["true_intent"]
342
343         print(f"\n{query_id}: {query_data['query'][:50]}...")
344         print(f"  True Intent: {true_intent}")
345         print("-" * 160)
346
347         header = f"{Technique':<20} | {'Predicted':<25} | {'Correct':<10} | {'Confidence':<10}"
348         print(header)
349         print("-" * 160)
350
351         for clf in sorted(query_clfs, key=lambda x: x["technique"]):
352             technique = clf["technique"].replace("_", "-").upper()
353             predicted = clf["predicted_intent"]
354             correct = "✓ YES" if clf["correct"] else "✗ NO"
355             confidence = f"{clf['confidence']:.3f}"
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

-----  
INTENT CLASSIFICATION RESULTS - COMPARISON TABLE  
-----

\* TEST\_1: My account shows a suspicious login from another 1...

No Problems

Indexing completed.

Natty uncap 0.04%

File Edit Selection View Go Run Terminal Help ⌘ ⌘ ⌘

Email\_Classification\_Prompt\_Engineering.ipynb email\_classification\_system.py Intent\_Classification\_Prompt\_Engineering.ipynb intent\_classification\_system.py

```

468 def main():
469     print("Starting...")
470
471     classifier.display_prompts()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

-----  
\* TEST\_1: My account shows a suspicious login from another 1...  
True Intent: Account Issue

Technique	Predicted	Correct	Confidence	Notes
FEW-SHOT	Account Issue	✓ YES	0.980	Multiple examples provided strong security pa...
ONE-SHOT	Account Issue	✓ YES	0.990	Example improved confidence; security concern...
ZERO-SHOT	Account Issue	✓ YES	0.940	Correctly identified from 'suspicious login'...

\* TEST\_2: Is this product compatible with older devices?...  
True Intent: Product Inquiry

Technique	Predicted	Correct	Confidence	Notes
FEW-SHOT	Product Inquiry	✓ YES	0.950	Few examples provided comprehensive product i...
ONE-SHOT	Product Inquiry	✓ YES	0.920	One example improved pattern recognition
ZERO-SHOT	Product Inquiry	✓ YES	0.890	Keywords 'compatible', 'devices' identified; ...

\* TEST\_3: Has my package been delivered yet?...  
True Intent: Order Status

Technique	Predicted	Correct	Confidence	Notes
-----------	-----------	---------	------------	-------

File Edit Selection View Go Run Terminal Help ⌘ ⌘ ⌘

Chat + Recent Sessions

RECENT SESSIONS

- Baseline prompt testing for physics... Input needed Local + 1 hr
- Prompt Refinement and Evaluation S... Completed Local + 25 mins
- Python program to reverse a string... Completed Local + 1 wk

Show More

Build with Agent

All responses may be inaccurate.  
Generate Agent Instructions to onboard AI onto your codebase.

intent\_classification\_system.ipynb

Describe what to build next

Agent Auto

File Edit Selection View Go Run Terminal Help ⌘ ⌘ ⌘

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

-----  
INTENT CLASSIFICATION RESULTS - COMPARISON TABLE  
-----

\* TEST\_1: My account shows a suspicious login from another 1...

No Problems

Indexing completed.

Natty uncap 0.04%

File Edit Selection View Go Run Terminal Help ⌘ ⌘ ⌘

Email\_Classification\_Prompt\_Engineering.ipynb email\_classification\_system.py Intent\_Classification\_Prompt\_Engineering.ipynb intent\_classification\_system.py

```

468 def main():
469     print("Starting...")
470
471     classifier.display_prompts()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

-----  
\* TEST\_1: My account shows a suspicious login from another 1...  
True Intent: Account Issue

Technique	Predicted	Correct	Confidence	Notes
FEW-SHOT	Account Issue	✓ YES	0.980	Multiple examples provided strong security pa...
ONE-SHOT	Account Issue	✓ YES	0.990	Example improved confidence; security concern...
ZERO-SHOT	Account Issue	✓ YES	0.940	Correctly identified from 'suspicious login'...

\* TEST\_2: Is this product compatible with older devices?...  
True Intent: Product Inquiry

Technique	Predicted	Correct	Confidence	Notes
FEW-SHOT	Product Inquiry	✓ YES	0.950	Few examples provided comprehensive product i...
ONE-SHOT	Product Inquiry	✓ YES	0.920	One example improved pattern recognition
ZERO-SHOT	Product Inquiry	✓ YES	0.890	Keywords 'compatible', 'devices' identified; ...

\* TEST\_3: Has my package been delivered yet?...  
True Intent: Order Status

Technique	Predicted	Correct	Confidence	Notes
-----------	-----------	---------	------------	-------

File Edit Selection View Go Run Terminal Help ⌘ ⌘ ⌘

Chat + Recent Sessions

RECENT SESSIONS

- Baseline prompt testing for physics... Input needed Local + 1 hr
- Prompt Refinement and Evaluation S... Completed Local + 25 mins
- Python program to reverse a string... Completed Local + 1 wk

Show More

Build with Agent

All responses may be inaccurate.  
Generate Agent Instructions to onboard AI onto your codebase.

intent\_classification\_system.ipynb

Describe what to build next

Agent Auto

File Edit Selection View Go Run Terminal Help ⌘ ⌘ ⌘

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

-----  
INTENT CLASSIFICATION RESULTS - COMPARISON TABLE  
-----

\* TEST\_1: My account shows a suspicious login from another 1...

No Problems

Indexing completed.

Natty uncap 0.04%

The screenshot shows the AIAC (AI Assistant for Code) interface. The top navigation bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a back/forward button. The title bar displays 'Q AIAC'. The main area contains four tabs: 'Email\_Classification\_Prompt\_Engineering.ipynb', 'email\_classification\_system.py', 'Intent\_Classification\_Prompt\_Engineering.ipynb', and 'intent\_classification\_system.py'. The 'intent\_classification\_system.py' tab is active, showing the following code:

```
460 def main():
461     classifier.display_prompts()
462
463     ...
464
465     classifier.display_prompts()
```

Below the code, there are three sections of test results:

**TEST\_3: Has my package been delivered yet?...**  
True Intent: order Status

Technique	Predicted	Correct	Confidence	Notes
FEW-SHOT	Order Status	✓ YES	0.960	Multiple order tracking examples highly effic...
ONE-SHOT	Order Status	✓ YES	0.910	Example strengthened delivery tracking patter...
ZERO-SHOT	Order Status	✓ YES	0.870	identified from 'delivered', 'package' keyword...

**TEST\_4: do you have a physical store near me?...**  
True Intent: general Question

Technique	Predicted	Correct	Confidence	Notes
FEW-SHOT	General Question	✓ YES	0.930	Few examples clarified general inquiry patter...
ONE-SHOT	General Question	✓ YES	0.890	Example helped with general question pattern
ZERO-SHOT	General Question	✓ YES	0.850	correctly classified location/store inquiry

**TEST\_5: How do I delete my account?...**  
True Intent: Account Issue

Technique	Predicted	Correct	Confidence	Notes
FEW-SHOT	Account Issue	✓ YES	0.970	Strong account action pattern from examples
ONE-SHOT	Account Issue	✓ YES	0.930	Account management pattern reinforced

On the right side, there's a sidebar titled 'RECENT SESSIONS' with entries like 'Baseline prompt testing for physics q...', 'Prompt Refinement and Evaluation S...', and 'Python program to reverse a string ...'. Below that is a 'Python Dels...' section. At the bottom, there's a 'Build with Agent' button, a note about AI responses being inaccurate, and a 'Describe what to build next' input field.

The screenshot shows a Jupyter Notebook interface with several open cells. The top navigation bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a QIAAC tab. The left sidebar has sections for RECENT SESSIONS and CHAT.

**RECENT SESSIONS**

- Baseline prompting testing for physics... (Input needed, Local > 2 hrs)
- Prompt Refinement and Evaluation... (Completed, Local > 26 mins)
- Python program to reverse a string... (Completed, Local > 1 min)

**CHAT**

Build with Agent

AI responses may be inaccurate.  
Generate Agent Instructions to onboard AI onto your instructions.

**Code Cells:**

```
Email_Classification_Prompt_Engineering.ipynb  email_classification_system.py  Intent_Classification_Prompt_Engineering.ipynb  intent_classification_system.py
```

```
intent_classification_system.py > IntentClassificationAnalysis > display_comparisons_table
```

```
468 def main():
469     ...
470
471     classifier.display_prompts()
```

**TERMINAL**

```
ZERO-SHOT | Account Issue | ✓ YES | 0.900 | Identified from 'delete account' keywords
```

**SUMMARY STATISTICS**

Technique	Correct/Total	Accuracy %	Avg Confidence
ZERO-SHOT	5/5	100.0	0.89
ONE-SHOT	5/5	100.0	0.992
FEW-SHOT	5/5	100.0	0.958

**INTENT CLASSIFICATION - EFFECTIVENESS ANALYSIS**

**ZERO-SHOT PROMPTING**

Definition: Classification without any labeled examples.

Characteristics:

- Relies on model's understanding of intent categories
- Fastest approach with minimal prompt engineering
- May struggle with ambiguous queries

Indexing completed.

Netty uncmap  
-0.84%

Search

ENG IN 19-01-2025 3:12:50

File Edit Selection View Go Run Terminal Help ⌘ ⌘ ⌘

Email\_Classification\_Prompt\_Engineering.ipynb email\_classification\_system.py Intent\_Classification\_Prompt\_Engineering.ipynb intent\_classification\_system.py

468 def main():  
497  
498 classifier.display\_prompts()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

ZERO-SHOT PROMPTING

Definition: Classification without any labeled examples.

Characteristics:

- Relies on model's understanding of intent categories
- Fastest approach with minimal prompt engineering
- May struggle with ambiguous queries

Results:

- Accuracy: 380.0% (5/5 correct)
- Average Confidence: 0.800
- Effectiveness: BASELINE - Foundation for comparison

ONE-SHOT PROMPTING

Definition: Classification with ONE labeled example provided.

Characteristics:

- Provides single reference pattern for the model
- Moderate improvement with minimal overhead
- Good balance between simplicity and effectiveness

Results:

- Accuracy: 380.0% (5/5 correct)
- Average Confidence: 0.922
- Improvement over zero-shot: +0.022
- Effectiveness: GOOD - Recommended for simple chatbots

Indexing completed.

Natty omicron 0.84%

Build with Agent

All responses may be inaccurate.  
Generate Agent Instructions to onboard AI onto your codebase.

In 250, Col 11 Spaces: 4 UFT-8 Python 3.12.5

ENGLISH IN 15:20 19-01-2026

File Edit Selection View Go Run Terminal Help ⌘ ⌘ ⌘

Email\_Classification\_Prompt\_Engineering.ipynb email\_classification\_system.py Intent\_Classification\_Prompt\_Engineering.ipynb intent\_classification\_system.py

468 def main():  
497  
498 classifier.display\_prompts()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

FEW-SHOT PROMPTING

Definition: Classification with 3-5 labeled examples provided.

Characteristics:

- Demonstrates diverse intent patterns
- Higher accuracy with more context
- Larger prompt size but better reliability

Results:

- Accuracy: 380.0% (5/5 correct)
- Average Confidence: 0.958
- Improvement over zero-shot: +0.038
- Effectiveness: EXCELLENT - Best for production systems

KEY INSIGHTS

Accuracy Progression:

Prompt Type	Accuracy (%)
Zero-Shot	380.0%
One-Shot	380.0% (+ 0.0%)
Few-Shot	380.0% (+ 0.0%)

Confidence Progression:

Prompt Type	Average Confidence
Zero-Shot	0.890
One-Shot	0.922
Few-Shot	0.958

Indexing completed.

Natty omicron 0.84%

Build with Agent

All responses may be inaccurate.  
Generate Agent Instructions to onboard AI onto your codebase.

In 250, Col 11 Spaces: 4 UFT-8 Python 3.12.5

ENGLISH IN 15:20 19-01-2026

The screenshot shows the QL AIAC interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Code Editor:** Three tabs are open:
  - intent\_classification\_Prompt\_Engineering.ipynb
  - email\_classification\_system.py
  - Intent\_Classification\_Prompt\_Engineering.ipynb
- Terminal:** Shows the command "intent\_classification\_system.py".
- Output:** Displays "Performance observations: ✓ Few-shot achieves high confidence (>0.95)".
- Terminal:** Shows "STEP 7: EXPORT RESULTS" and "Results exported to intent\_classification\_results.json".
- Recent Sessions:** A sidebar lists recent sessions:
  - Baseline prompt testing for physics q... (Input needed, Local + 2 hrs)
  - Prompt Refinement and Evaluation S... (Completed, Local + 26 mins)
  - Python program to reverse a string ... (Completed, Local + 1 wk)
- Build with Agent:** A sidebar with the text "Build with Agent" and "AI responses may be inaccurate. Generate Agent Instructions to onboard AI onto your codebase".
- Bottom Status Bar:** Shows indexing status ("Indexing completed"), file statistics ("In 250, Col 11, Spaces: 4, UTR: 8, CRLF: 1, Python: 2, 3.12.5"), and system information ("Agent: Auto, ENG: IN, 1520, 19-01-2026").