

Assignment-4.1

Name: J Sai Harini

Ht No: 2303A52426

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 a Jupyter Notebook environment with several open files. The main code cell contains the following Python code:

```
class EmailClassificationSystem:
    def __init__(self):
        self.prompts: Dict[str, Any] = {}
        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"
            }
        ]
```

Below the code, a sample email message is displayed:

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 10th 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-7000.

Thank you for your prompt assistance.

Best regards,
Sarah Johnson

The status bar at the bottom right indicates "Status: PRODUCTION READY" and "Quality: ★★★★★".

The screenshot shows a Jupyter Notebook environment with several open files. The main code cell contains the following Python code:

```
class EmailClassificationSystem:
    def prepare_sample_emails(self) -> None:
        Best regards,  
Sarah Johnson""",
```

Below the code, a sample email message is displayed:

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_chen@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 right indicates "Status: PRODUCTION READY" and "Quality: ★★★★★".

File Edit Selection View Go Run Terminal Help < > O AIC

EXPLORER

- AIMC
- BD_DASHBOARD_COMP1.html
- BD_START_HPRP.html
- COMPARISON_ANALYSIS.html
- COMPLETION_CHART.html
- DISPLAY_SUMMARY.html
- Email Classification Comparison.html
- email_classification_prompt.html
- email_classification_results.html
- email_classification_system.py
- EMAIL_SUMMARY.html
- physics_test.json
- PROJECT_INDEX.html
- RULEMAIL_email_classification.html
- reverse_string.html
- reverse_string.py
- string_removal_approach.py

VSCode Email Classification System

```
20 class EmailClassificationSystem:
38     def prepare_sample_emails(self) -> None:
71         }
72         {
73             "id": 3,
74             "category": EmailCategory.FEEDBACK.value,
75             "email": """Subject: Great Experience with Your Product!
76
77 Hi Team,
78
79 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.
80
81 I'd love to see some 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.
82
83 Keep up the excellent work!
```

REGARDS,
Jeniffer Martinez

OUTLINE > TIMELINE > PROJECTS

All files are ready for CI/CD.

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

File C:\Users\HP\PycharmProjects\EmailClassificationSystem\src\main\python\email_classification_system.py

File Edit Selection View Go Run Terminal Help < > O AIC

EXPLORER

- AIMC
- BD_DASHBOARD_COMP1.html
- BD_START_HPRP.html
- COMPARISON_ANALYSIS.html
- COMPLETION_CHART.html
- DISPLAY_SUMMARY.html
- Email Classification Comparison.html
- email_classification_prompt.html
- email_classification_results.html
- email_classification_system.py
- EMAIL_SUMMARY.html
- physics_test.json
- PROJECT_INDEX.html
- RULEMAIL_email_classification.html
- reverse_string.html
- reverse_string.py
- string_removal_approach.py

VSCode Email Classification System

```
20 class EmailClassificationSystem:
38     def prepare_sample_emails(self) -> None:
92         },
93         {
94             "id": 4,
95             "category": EmailCategory.OTHERS.value,
96             "email": """Subject: Partnership Inquiry
97
98 Dear Management,
99
100 My name is David Wong, and I represent TechVenture Solutions. We've been following
101 your company's growth in the market and would like to explore potential partnership
102 opportunities. We believe there could be mutual benefits in collaborating on
103 enterprise solutions.
104
105 Would you be available for a brief call next week to discuss this further?
106
107 Looking forward to hearing from you.
108
109 Best regards,
110 David Wong,
111 Business Development Manager
112 TechVenture Solutions
```

OUTLINE > TIMELINE > PROJECTS

All files are ready for CI/CD.

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

File C:\Users\HP\PycharmProjects\EmailClassificationSystem\src\main\python\email_classification_system.py

File Edit Selection View Go Run Terminal Help

VSCode Email Classification System

```

class EmailClassificationSystem:
    def prepare_sample_emails(self) -> None:
        pass
        id: 5,
        "category": EmailCategory.BILLING.value,
        "email": "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: SUB-2024-98765
Thank you,
Robert Clark

```

OUTLINE

PROJECTS

File Edit Selection View Go Run Terminal Help

VSCode Email Classification System

```

class EmailClassificationSystem:
    def prepare_test_emails(self) -> None:
        print(f"Prepared {len(self.test_emails)} test emails for classification")
        def create_zero_shot_prompt(self, email: str) -> str:
            """Create a zero-shot prompt (no examples provided)."""
            prompt = f"Classify the following customer email into one of these categories:
- Billing
- Technical Support

```

ZERO-SHOT 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.

Result:

Subject: App Crashes on Startup

Hi,

My app keeps crashing immediately after I open it. I've tried restarting my device and reinstallation the application, but the problem continues. The error message says "application fault" but I don't know what that means.

Can you help me fix this issue?

All files are ready for PRODUCTION

Status: PRODUCTION READY

Quality: ★★★★★

PROMPT REFINEMENT AND DEPLOYMENT

4. PROMPT REFINEMENT AND DEPLOYMENT

A. Read:

- Production deployment roadmap (4 steps)
- Decision framework (choose your framework)
- Customization guide (adapt to your needs)
- Quality verification (all tested)

B. Getting Started:

- READ → 00 START (FRONTEND) (2 min)
- RUN → system
- DECODE → Use # 00-0000_SUMMARY.txt
- DEPLOY → Follow

C. STATUS: Email Classification System (4.5 days)

D. OUT

E. PUBLISH

F. Python Dev

G. Python Test

H. Python Unit Tests

I. Python Linting

J. Python Style

K. Python Coverage

L. Python Profiling

M. Python Memory

N. Python Security

O. Python Performance

P. Python Logging

File Edit Selection View Go Run Terminal Help

VSCode Email Classification System

```

class EmailClassificationSystem:
    def prepare_sample_emails(self) -> None:
        pass
        id: 5,
        "category": EmailCategory.BILLING.value,
        "email": "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: SUB-2024-98765
Thank you,
Robert Clark

```

OUTLINE

PROJECTS

File Edit Selection View Go Run Terminal Help

VSCode Email Classification System

```

class EmailClassificationSystem:
    def prepare_test_emails(self) -> None:
        print(f"Prepared {len(self.test_emails)} test emails for classification")
        def create_zero_shot_prompt(self, email: str) -> str:
            """Create a zero-shot prompt (no examples provided)."""
            prompt = f"Classify the following customer email into one of these categories:
- Billing
- Technical Support

```

ZERO-SHOT 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.

Result:

Subject: App Crashes on Startup

Hi,

My app keeps crashing immediately after I open it. I've tried restarting my device and reinstallation the application, but the problem continues. The error message says "application fault" but I don't know what that means.

Can you help me fix this issue?

All files are ready for PRODUCTION

Status: PRODUCTION READY

Quality: ★★★★★

PROMPT REFINEMENT AND DEPLOYMENT

4. PROMPT REFINEMENT AND DEPLOYMENT

A. Read:

- Production deployment roadmap (4 steps)
- Decision framework (choose your framework)
- Customization guide (adapt to your needs)
- Quality verification (all tested)

B. Getting Started:

- READ → 00 START (FRONTEND) (2 min)
- RUN → system
- DECODE → Use # 00-0000_SUMMARY.txt
- DEPLOY → Follow

C. STATUS: Email Classification System (4.5 days)

D. OUT

E. PUBLISH

F. Python Dev

G. Python Test

H. Python Unit Tests

I. Python Linting

J. Python Style

K. Python Coverage

L. Python Profiling

M. Python Memory

N. Python Security

O. Python Performance

P. Python Logging

File Edit Selection View Go Run Terminal Help

VSCode Email Classification System

```

class EmailClassificationSystem:
    def prepare_sample_emails(self) -> None:
        pass
        id: 5,
        "category": EmailCategory.BILLING.value,
        "email": "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: SUB-2024-98765
Thank you,
Robert Clark

```

OUTLINE

PROJECTS

File Edit Selection View Go Run Terminal Help

VSCode Email Classification System

```

class EmailClassificationSystem:
    def prepare_sample_emails(self) -> None:
        pass
        id: 5,
        "category": EmailCategory.BILLING.value,
        "email": "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: SUB-2024-98765
Thank you,
Robert Clark

```

OUTLINE

PROJECTS

File Edit Selection View Go Run Terminal Help

VSCode Email Classification System

```

class EmailClassificationSystem:
    def prepare_test_emails(self) -> None:
        print(f"Prepared {len(self.test_emails)} test emails for classification")
        def create_zero_shot_prompt(self, email: str) -> str:
            """Create a zero-shot prompt (no examples provided)."""
            prompt = f"Classify the following customer email into one of these categories:
- Billing
- Technical Support

```

ZERO-SHOT 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.

Result:

Subject: App Crashes on Startup

Hi,

My app keeps crashing immediately after I open it. I've tried restarting my device and reinstallation the application, but the problem continues. The error message says "application fault" but I don't know what that means.

Can you help me fix this issue?

All files are ready for PRODUCTION

Status: PRODUCTION READY

Quality: ★★★★★

PROMPT REFINEMENT AND DEPLOYMENT

4. PROMPT REFINEMENT AND DEPLOYMENT

A. Read:

- Production deployment roadmap (4 steps)
- Decision framework (choose your framework)
- Customization guide (adapt to your needs)
- Quality verification (all tested)

B. Getting Started:

- READ → 00 START (FRONTEND) (2 min)
- RUN → system
- DECODE → Use # 00-0000_SUMMARY.txt
- DEPLOY → Follow

C. STATUS: Email Classification System (4.5 days)

D. OUT

E. PUBLISH

F. Python Dev

G. Python Test

H. Python Unit Tests

I. Python Linting

J. Python Style

K. Python Coverage

L. Python Profiling

M. Python Memory

N. Python Security

O. Python Performance

P. Python Logging

File Edit Selection View Go Run Terminal Help

VSCode Email Classification System

```

class EmailClassificationSystem:
    def prepare_sample_emails(self) -> None:
        pass
        id: 5,
        "category": EmailCategory.BILLING.value,
        "email": "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: SUB-2024-98765
Thank you,
Robert Clark

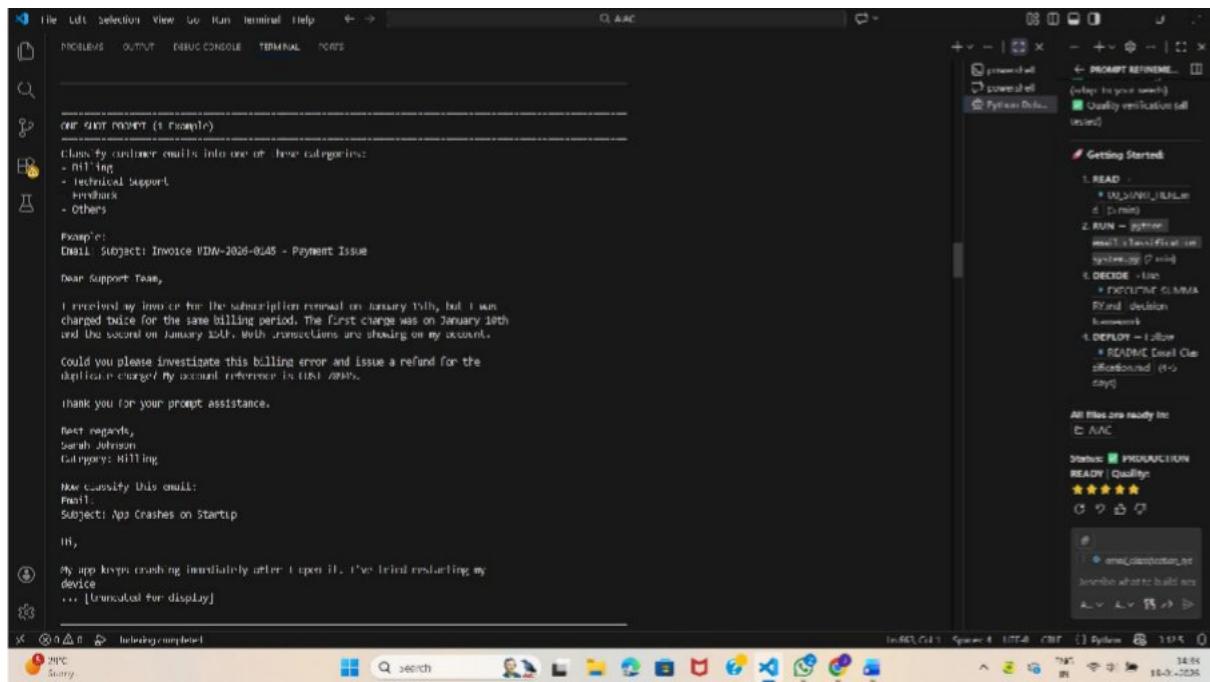
```

OUTLINE

PROJECTS

The screenshot shows a Jupyter Notebook interface with several tabs open at the top: 'File Edit Selection View Go Run Terminal Help' and 'ABC'. The main area contains the following Python code:

```
1 import string
2 import re
3 import random
4 import smtplib
5 from email.mime.multipart import MIMEMultipart
6 from email.mime.text import MIMEText
7
8 class EmailClassificationSystem:
9
10     def __init__(self):
11         self.categories = {
12             'Billing': [
13                 "Hi, I'm writing to you about my bill for last month.",
14                 "I have a question regarding my electricity bill for last month."
15             ],
16             'Technical Support': [
17                 "I am experiencing some issues with my account and need your assistance.",
18                 "I am having trouble with my account and would like to speak with someone about it."
19             ],
20             'Feedback': [
21                 "I wanted to provide some feedback on the service I received from your company.",
22                 "I would like to provide some feedback on the service I received from your company."
23             ],
24             'Others': [
25                 "I have a question about a product I purchased from your company.",
26                 "I have a question about a product I purchased from your company."
27             ]
28         }
29
30     def create_one_shot_prompt(self, email: str, example_email: str, example_category: str) -> str:
31         """Create a one-shot prompt (one labeled example provided)."""
32         prompt = f"""Classify customer emails into one of these categories:
33
34         - Billing
35         - Technical Support
36         - Feedback
37         - Others
38
39         Example:
40         Email: {example_email}
41         Category: {example_category}
42
43         Now classify this email:
44         Email:
45         {email}
46
47         Category:"""
48
49     def classify_email(self, email: str) -> str:
50         """Classify an email into one of the defined categories."""
51         # Implement classification logic here
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
517
518
519
519
520
521
522
523
524
525
526
527
527
528
529
529
530
531
532
533
534
535
536
537
537
538
539
539
540
541
542
543
544
545
545
546
547
547
548
549
549
550
551
552
553
554
555
555
556
557
557
558
559
559
559
560
561
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
1624
1624
1625
16
```



```
ANALYSIS | ④ using rebus approach.py | Email Classification Prompt System | Email Classification System X | ④ DELIVERY COMPLETED | ④ DELIVERY SUMMARY | COMP | ... |
④ email classification system? ...
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}:\nEmail: {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
245
```

The screenshot shows a Jupyter Notebook environment with several tabs open. The active tab is titled 'EmailClassificationAnalysis' and contains Python code for generating a comparison table of classification techniques. The code uses a class named 'EmailClassificationAnalysis' which includes a method 'generate_comparison_table'. This method retrieves responses from a system and calculates accuracy. It then groups responses by test email and appends them to a table_lines list. Finally, it displays results per email and adds a header for testing emails.

```
class EmailClassificationAnalysis:
    def generate_comparison_table(self) -> str:
        """Generate a comparison table of all techniques."""
        responses = self.system.results.get("responses", [])
        accuracy = self.calculate_accuracy()

        table_lines = []
        table_lines.append("\n" + "="*140)
        table_lines.append("EMAIL CLASSIFICATION RESULTS - ZERO-SHOT vs ONE-SHOT")
        table_lines.append("="*140)

        # Group by test_email
        responses_by_email = {}
        for response in responses:
            email_id = response["test_email_id"]
            if email_id not in responses_by_email:
                responses_by_email[email_id] = []
            responses_by_email[email_id].append(response)

        # Display results per email
        for email_id in sorted(responses_by_email.keys()):
            email_responses = responses_by_email[email_id]
            true_category = email_responses[0]["true_category"]

            table_lines.append(f"\n{email_id} TEST EMAIL: {email_id}")
            for response in email_responses:
```

This screenshot shows a software development environment with multiple tabs open. The main tab displays Python code for an 'EmailClassificationAnalysis' class. The code includes methods for generating comparison tables and displaying analysis. The interface includes a sidebar with project files, a terminal at the bottom, and a status bar indicating the file is 'fully compiled'.

```
370 class EmailClassificationAnalysis:
380     def generate_comparison_table(self) -> str:
381         table_lines.append(f"  True Category: {true_category}")
382         table_lines.append("-" * 140)
383
384         header = f"{'Technique':<20} | {'Predicted':<20} | {'Correct':<10}"
385         table_lines.append(header)
386         table_lines.append("-" * 140)
387
388         for response in sorted(email_responses, key=lambda x: x["technique"]):
389             technique = response["technique"].replace("_", "-").upper()
390             predicted = response["predicted_category"]
391             correct = "✓ YES" if response["correct"] else "✗ NO"
392             confidence = f"{response['confidence']:.2f}"
393             notes = response["notes"][:55] + "... " if len(response["notes"]) > 55 else ""
394
395             row = f"{technique:<20} | {predicted:<20} | {correct:<10} | {confidence:<10} | {notes}"
396             table_lines.append(row)
397
398         # Summary statistics
399         table_lines.append("\n" + "="*140)
400         table_lines.append("SUMMARY STATISTICS")
401         table_lines.append("-" * 140)
402
403         summary_header = f"{'Technique':<20} | {'Correct/Total':<20} | {'Accuracy':<10}"
404         table_lines.append(summary_header)
405
406     def display_analysis(self) -> None:
407         """Display comprehensive analysis."""
408         print(self.generate_comparison_table())
409
410         accuracy = self.calculate_accuracy()
411
412         print("\n" + "="*140)
413         print("DETAILED EFFECTIVENESS ANALYSIS")
414         print("="*140)
415
416         print("\n🕒 ZERO-SHOT PROMPTING")
417         print("-" * 140)
```

This screenshot shows the same software development environment as the first one, but with a different section of the code visible. The 'generate_comparison_table' method is now fully visible, showing its implementation for generating a comparison table based on email responses. The interface remains consistent with the first screenshot.

```
370 class EmailClassificationAnalysis:
380     def generate_comparison_table(self) -> str:
381         table_lines.append("-" * 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['correct']}/{stats['total']}"
387                 table_lines.append(row)
388
389         table_lines.append("\n" + "="*140)
390
391     def display_analysis(self) -> None:
392         """Display comprehensive analysis."""
393         print(self.generate_comparison_table())
394
395         accuracy = self.calculate_accuracy()
396
397         print("\n" + "="*140)
398         print("DETAILED EFFECTIVENESS ANALYSIS")
399         print("="*140)
400
401         print("\n🕒 ZERO-SHOT PROMPTING")
402         print("-" * 140)
```


A screenshot of a code editor window titled "EmailClassificationAnalysis.py". The code is a Python script for email classification analysis. It includes sections for "WHEN TO USE EACH TECHNIQUE:", "PRACTICAL RECOMMENDATIONS:", and a "HYBRID APPROACH:". The code uses print statements to output classification logic and recommendations.

```
370 class EmailClassificationAnalysis:
371     def display_effectiveness(self):
372         print(f" - Effectiveness: HIGHEST - Best accuracy and confidence")
373
374     def display_recommendations(self):
375         """Display recommendations based on analysis."""
376         print("\n WHEN TO USE EACH TECHNIQUE:")
377         print(" ZERO-SHOT:")
378         print(" ✓ Quick classification for straightforward emails")
379         print(" ✓ When computational resources are limited")
380         print(" ✓ For high-level category detection")
381         print(" X Avoid for mission-critical classifications")
382         print(" X Not suitable when categories are similar or ambiguous")
383
384         print("\n ONE-SHOT:")
385         print(" ✓ Good balance between context and efficiency")
386         print(" ✓ When slight improvement in accuracy is needed")
387         print(" ✓ For moderately complex classification tasks")
388         print(" X May still miss edge cases")
389         print(" X Single example may not cover all category variations")
390
391         print("\n FEW-SHOT:")
392         print(" ✓ Highest accuracy and confidence")
```

A screenshot of a code editor window titled "EmailClassificationAnalysis.py", showing a modified version of the script. The "display_effectiveness" method has been removed, and the "display_recommendations" method has been updated to include practical recommendations and a hybrid approach. The code uses print statements to output classification logic and recommendations.

```
370 class EmailClassificationAnalysis:
371     def display_recommendations(self):
372         """Display recommendations based on analysis."""
373         print(" ✓ For critical classification (e.g., high-value customer issues)")
374         print(" ✓ When category boundaries are fuzzy")
375         print(" ✓ Most reliable for production systems")
376         print(" X Larger prompt size increases latency")
377         print(" X Higher computational cost per request")
378
379         print("\n PRACTICAL RECOMMENDATIONS:")
380         print(" 1. START with zero-shot for rapid prototyping and validation")
381         print(" 2. MEASURE accuracy and identify problematic email types")
382         print(" 3. USE one-shot when zero-shot shows 85-90% accuracy")
383         print(" 4. EMPLOY few-shot for production systems requiring >95% accuracy")
384         print(" 5. SELECT diverse, representative examples for few-shot prompts")
385         print(" 6. REGULARLY update examples as new email patterns emerge")
386         print(" 7. COMBINE with confidence scores to flag uncertain classifications")
387         print(" 8. CONSIDER human review for low confidence predictions")
388
389         print("\n HYBRID APPROACH:")
390         print(" - Use zero-shot as initial filter for obvious classifications")
391         print(" - Escalate ambiguous cases (low confidence) to one-shot or few-shot")
392         print(" - Maintain human review queue for edge cases")
393         print(" - Build confidence thresholds for automatic routing vs. manual review")
```


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

- File Edit Selection View Go Run Terminal Help** are the top navigation menu items.
- COLAB** is the active workspace indicator.
- EXPLORER** is the sidebar tab.
- AIM** is the current notebook tab.
- GO DELIVERY COMPLETION**, **GO START/TFP**, **COMPARISON/ANALYSIS**, **COMBINE/CHUNK/START**, **DELIVERY SUMMARY**, **Email Classification Comparison Results**, **EmailClassification_Prompt_Engine**, **EmailClassification_results.json**, and **EmailClassification_system** are listed as other notebooks.
- Q Search** is the search bar at the bottom.
- Projects** is the bottom navigation tab.

The main area displays the following Python code:

```
def main():
    with open("email_classification_results.json", 'w', encoding='utf-8') as f:
        json.dump(export_data, f, indent=2, ensure_ascii=False)

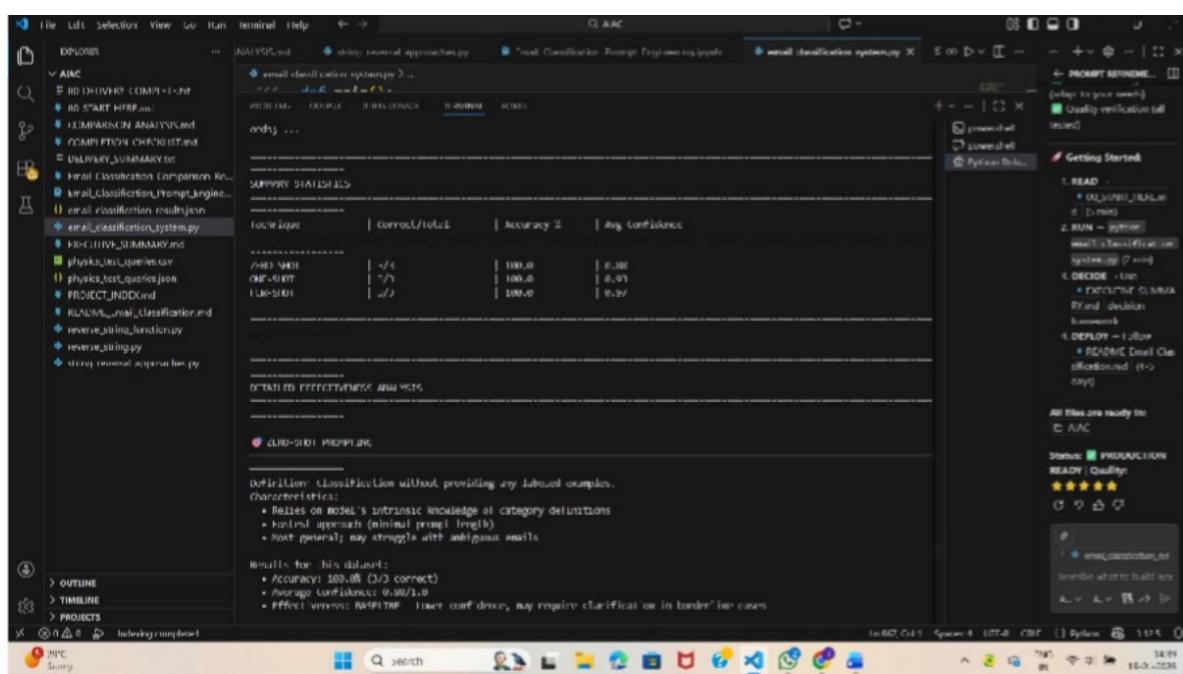
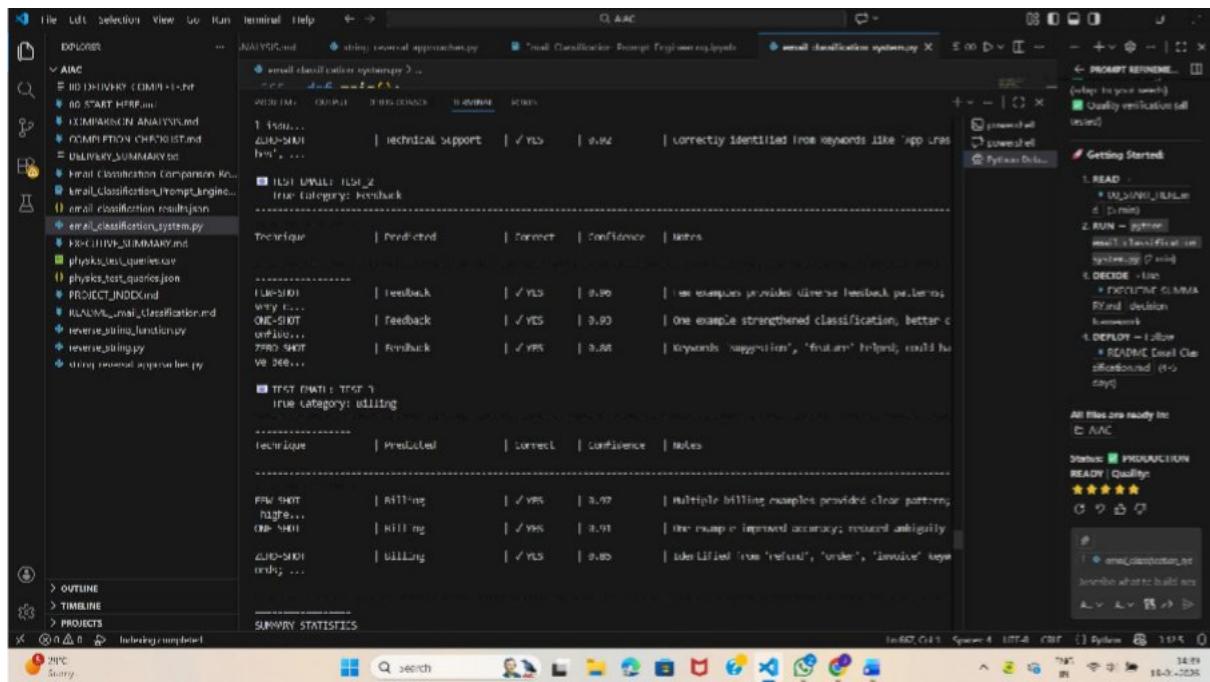
    print("\n\N{checkmark} Results exported to email_classification_results.json")

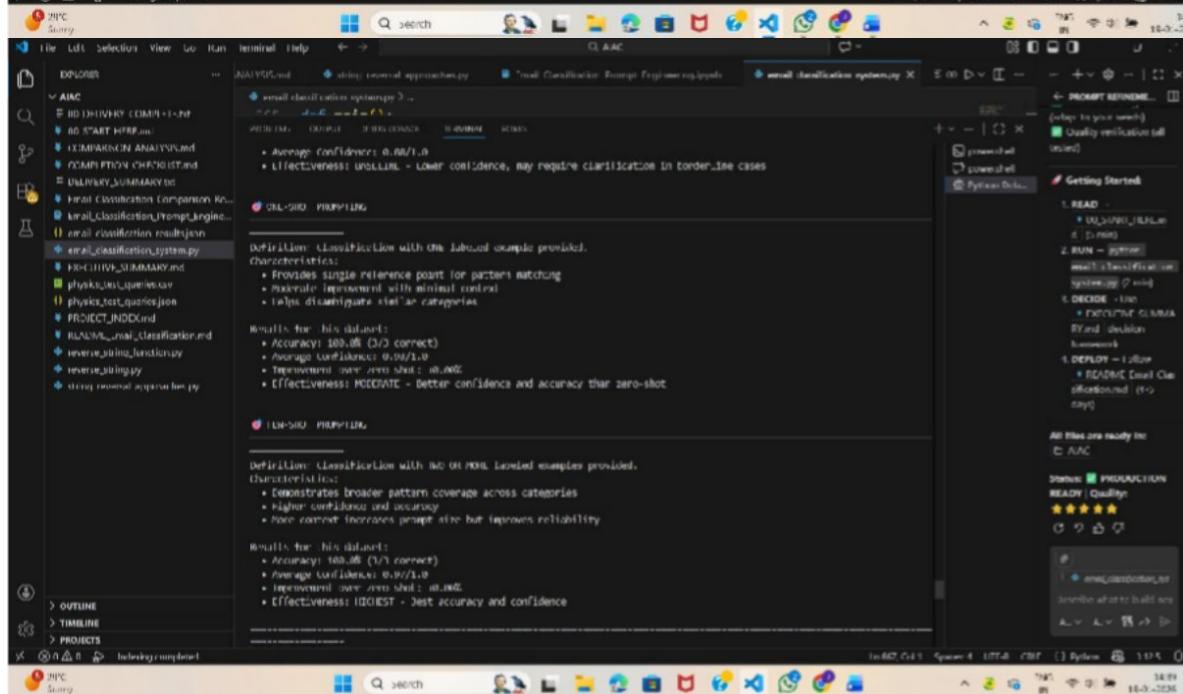
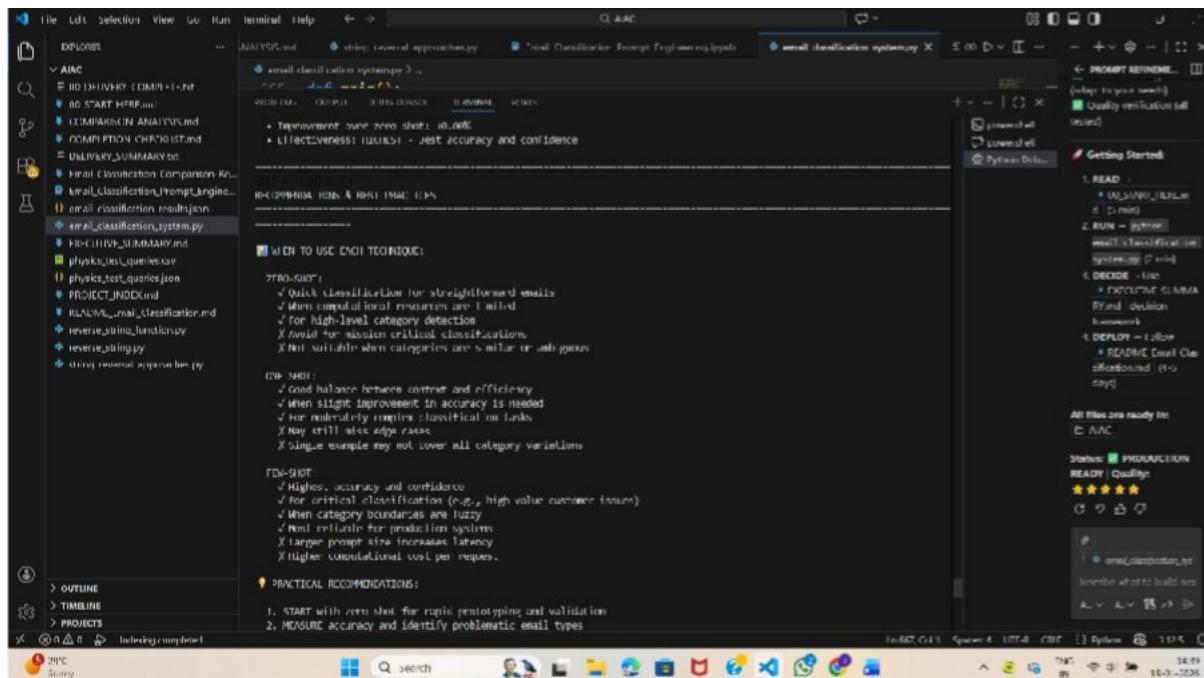
    # Final summary
    print("\n" + "="*100)
    print("EXECUTION COMPLETE")
    print("="*100)
    print("\n\N{checkmark} Summary:")
    print(" - Prepared 5 sample emails across all 4 categories")
    print(" - Generated 3 test emails for classification")
    print(" - Created zero-shot, one-shot, and few-shot prompts")
    print(" - Recorded and analyzed AI responses")
    print(" - Compared effectiveness of each technique")
    print("\n\N{checkmark} Generated Files:")
    print(" - email_classification_results.json - All results and data")
    print("\n\N{checkmark} Next Steps:")
    print(" 1. Copy the generated prompts and use them with your LLM")
    print(" 2. Record actual AI responses in the system")
    print(" 3. Update scores in the evaluation template")
    print(" 4. Re-run analysis for final comparison")
    print("\n" + "="*100 + "\n")
```

The screenshot shows a code editor with a dark theme. The left sidebar contains a file tree with various files listed, including `email_classification_system.py`, `reverse_string.py`, and `README_email_classification.md`. The main pane displays the `email_classification_system.py` file, which contains the following Python code:

```
def main():
    print(" 1. Copy the generated prompts and use them with your LLM")
    print(" 2. Record actual AI responses in the system")
    print(" 3. Update scores in the evaluation template")
    print(" 4. Re-run analysis for final comparison")
    print("\n" + "="*100 + "\n")

if __name__ == "__main__":
    main()
```





The screenshot shows the AATAC application interface. The left sidebar displays project navigation with sections like OUTLINE, TIMELINE, and PROJECTS. The main workspace shows a detailed view of a project named 'EMAIL CLASSIFICATION SYSTEM'. It includes tabs for 'EXPLORER', 'AATAC', and 'PROBLEMS'. The 'AATAC' tab is active, showing a summary of the system's performance across four categories. It lists generated prompts (zero-shot, one-shot, two-shot) and provides a JSON file for results. Below this, a 'Next Steps' section outlines tasks such as using generated prompts, recording user responses, updating scores, and running final comparisons. The bottom status bar indicates the application is ready for production.

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 code editor with several tabs open, including "Intent Classification Example (Python3).py" and "Intent Classification Script (Python3).py". The main pane displays a Python class named ChatbotIntentClassifier with methods for creating zero-shot prompts and classifying user queries. The sidebar features an "Agent" section with various AI tools, such as "Build with Agent", which is currently active, and other options like "Generate Agent Instructions" and "Deploy AI onto your codebase".

```
class ChatbotIntentClassifier:
    def create_zero_shot_prompt(self, query: str) -> str:
        """Create a zero-shot prompt (no examples provided)."""
        prompt = f"""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
```

```
File Edit Selection View Go Run Terminal Help < - > ABC
File Edit Selection View Go Run Terminal Help < - > ABC
File Edit Selection View Go Run Terminal Help < - > ABC
Email Classification System > ChatbotIntentClassifier > Create one shot prompt
Email Classification System > ChatbotIntentClassifier > Create one shot prompt
Email Classification System > ChatbotIntentClassifier > Create one shot prompt
20 class ChatbotIntentClassifier:
114     def create_one_shot_prompt(self, query: str, example_query: str, example_intent: str) -> str:
115         """Create a one shot prompt (one labeled example provided)."""
116         prompt = f"""Classify chatbot user queries into one of these intents:
117 - Account Issue
118 - Order Status
119 - Product Inquiry
120 - General Question
121 Example:
122 Query: {example_query}
123 Intent: {example_intent}
124 Now classify this query:
125 Query: {query}
126 Intent:"""
127         return prompt
128
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL POSIX
Build with Agent
All responses may be incomplete.
Generate Agent instructions or embedd AI onto your codebase.
Build with Agent
All responses may be incomplete.
Generate Agent instructions or embedd AI onto your codebase.
Build with Agent
All responses may be incomplete.
Generate Agent instructions or embedd AI onto your codebase.
```

ONE-SHOT PROMPT (1 Example)

```
-----
```

Classify chatbot user queries into one of these intents:

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

Example:

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

Intent: Account Issue

File Edit Selection View Go Run Terminal Help < - > ABC
File Edit Selection View Go Run Terminal Help < - > ABC
File Edit Selection View Go Run Terminal Help < - > ABC
Email Classification System > ChatbotIntentClassifier > Create one shot prompt
Email Classification System > ChatbotIntentClassifier > Create one shot prompt
Email Classification System > ChatbotIntentClassifier > Create one shot prompt
20 class ChatbotIntentClassifier:
114 def create_one_shot_prompt(self, query: str, example_query: str, example_intent: str) -> str:
115 """Create a one-shot prompt (one labeled example provided)."""
116 prompt = f"""Classify chatbot user queries into one of these intents:
117 - Account Issue
118 - Order Status
119 - Product Inquiry
120 - General Question
121 Example:
122 Query: {example_query}
123 Intent: {example_intent}
124 Now classify this query:
125 Query: {query}
126 Intent:"""
127 return prompt
128
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL POSIX
Build with Agent
All responses may be incomplete.
Generate Agent instructions or embedd AI onto your codebase.
Build with Agent
All responses may be incomplete.
Generate Agent instructions or embedd AI onto your codebase.
Build with Agent
All responses may be incomplete.
Generate Agent instructions or embedd AI onto your codebase.

ONE-SHOT PROMPT (1 Example)

```
-----
```

Classify chatbot user queries into one of these intents:

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

Example:

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

Intent: Account Issue

How classify this query:

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

Intent: ... [truncated for display]

File Edit Selection View Go Run Terminal Help < - > ABC
File Edit Selection View Go Run Terminal Help < - > ABC
File Edit Selection View Go Run Terminal Help < - > ABC
Email Classification System > ChatbotIntentClassifier > Create one shot prompt
Email Classification System > ChatbotIntentClassifier > Create one shot prompt
Email Classification System > ChatbotIntentClassifier > Create one shot prompt
20 class ChatbotIntentClassifier:
114 def create_one_shot_prompt(self, query: str, example_query: str, example_intent: str) -> str:
115 """Create a one-shot prompt (one labeled example provided)."""
116 prompt = f"""Classify chatbot user queries into one of these intents:
117 - Account Issue
118 - Order Status
119 - Product Inquiry
120 - General Question
121 Example:
122 Query: {example_query}
123 Intent: {example_intent}
124 Now classify this query:
125 Query: {query}
126 Intent:"""
127 return prompt
128
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL POSIX
Build with Agent
All responses may be incomplete.
Generate Agent instructions or embedd AI onto your codebase.
Build with Agent
All responses may be incomplete.
Generate Agent instructions or embedd AI onto your codebase.
Build with Agent
All responses may be incomplete.
Generate Agent instructions or embedd AI onto your codebase.

```
File Edit Selection View Go Run Terminal Help < - > ABC
File Edit Selection View Go Run Terminal Help < - > ABC
File Edit Selection View Go Run Terminal Help < - > ABC
Email Classification System > ChatbotIntentClassifier > Create one shot prompt
Email Classification System > ChatbotIntentClassifier > Create one shot prompt
Email Classification System > ChatbotIntentClassifier > Create one shot prompt
20 class ChatbotIntentClassifier:
114     def create_one_shot_prompt(self, query: str, example_query: str, example_intent: str) -> str:
115         """Create a one-shot prompt (one labeled example provided)."""
116         prompt = f"""Classify chatbot user queries into one of these intents:
117 - Account Issue
118 - Order Status
119 - Product Inquiry
120 - General Question
121 Example:
122 Query: {example_query}
123 Intent: {example_intent}
124 Now classify this query:
125 Query: {query}
126 Intent:"""
127         return prompt
128
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL POSIX
Build with Agent
All responses may be incomplete.
Generate Agent instructions or embedd AI onto your codebase.
Build with Agent
All responses may be incomplete.
Generate Agent instructions or embedd AI onto your codebase.
Build with Agent
All responses may be incomplete.
Generate Agent instructions or embedd AI onto your codebase.
```

ONE-SHOT PROMPT (1 Example)

```
-----
```

Classify chatbot user queries into one of these intents:

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

Example:

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

Intent: Account Issue

How classify this query:

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

Intent: Suspicious Login

File Edit Selection View Go Run Terminal Help < - > ABC
File Edit Selection View Go Run Terminal Help < - > ABC
File Edit Selection View Go Run Terminal Help < - > ABC
Email Classification System > ChatbotIntentClassifier > Create one shot prompt
Email Classification System > ChatbotIntentClassifier > Create one shot prompt
Email Classification System > ChatbotIntentClassifier > Create one shot prompt
20 class ChatbotIntentClassifier:
114 def create_one_shot_prompt(self, query: str, example_query: str, example_intent: str) -> str:
115 """Create a one-shot prompt (one labeled example provided)."""
116 prompt = f"""Classify chatbot user queries into one of these intents:
117 - Account Issue
118 - Order Status
119 - Product Inquiry
120 - General Question
121 Example:
122 Query: {example_query}
123 Intent: {example_intent}
124 Now classify this query:
125 Query: {query}
126 Intent:"""
127 return prompt
128
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL POSIX
Build with Agent
All responses may be incomplete.
Generate Agent instructions or embedd AI onto your codebase.
Build with Agent
All responses may be incomplete.
Generate Agent instructions or embedd AI onto your codebase.
Build with Agent
All responses may be incomplete.
Generate Agent instructions or embedd AI onto your codebase.

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

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** Intent Classification System > ChatbotIntentClassifier > Create one shot prompt
- Code Editor:** Python code for `ChatbotIntentClassifier`. The code defines a method `create_few_shot_prompt` which generates a few-shot prompt from a list of examples. It includes a list of intents: Account Issue, Order Status, Product Inquiry, and General Question. A placeholder `{examples_text}` is provided for the examples.
- Output Panel:** Shows the results of running the code, listing the four intents.
- Terminal:** Shows the command `Indexing completed.`
- Build with Agent:** A sidebar with the message "All responses may be incomplete. Generate Agent instructions or embedd AI onto your codebase".
- Bottom:** Taskbar with icons for File Explorer, Search, and others.

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

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** Intent Classification System > ChatbotIntentClassifier > Create one shot prompt
- Code Editor:** The same Python code as the first screenshot, but now it displays the generated `Few shot prompt (4 Examples)`.
- Output Panel:** Shows the results of running the code, listing the four intents.
- Terminal:** Shows the command `Indexing completed.`
- Build with Agent:** A sidebar with the message "All responses may be incomplete. Generate Agent instructions or embedd AI onto your codebase".
- Bottom:** Taskbar with icons for File Explorer, Search, and others.

The screenshot shows a Jupyter Notebook environment with several tabs open at the top:

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

The main area displays Python code for a `ChatbotIntentClassifier` class. The code includes methods for adding sample classifications and record classifications for various categories like Product Inquiry, Order Status, and General Question.

```
class ChatbotIntentClassifier:  
    def add_sample_classifications(self) -> None:  
        self.record_classification("one_shot", "TEST_1", "Product Inquiry", 0.92,  
                                    "One example improved pattern recognition")  
        self.record_classification("few_shot", "TEST_2", "Product Inquiry", 0.95,  
                                    "Few examples provided comprehensive product inquiry patterns")  
        # Sample classifications for TEST_3 (Order Status)  
        self.record_classification("zero_shot", "TEST_3", "Order Status", 0.87,  
                                    "Identified from 'delivered', 'package' keywords")  
        self.record_classification("one_shot", "TEST_3", "Order Status", 0.91,  
                                    "Example strengthened delivery tracking pattern")  
        self.record_classification("few_shot", "TEST_3", "Order Status", 0.96,  
                                    "Multiple order tracking examples highly effective")  
        # Sample classifications for TEST_4 (General Question)  
        self.record_classification("zero_shot", "TEST_4", "General Question", 0.85,  
                                    "Correctly classified location/store inquiry")
```

Below the code, a section titled "INTENT CLASSIFICATION RESULTS - COMPARISON TABLE" contains a table with two rows of data. The first row is highlighted in red, indicating a suspicious login attempt.

Category	Result	Details
TEST_3	Suspicious	My account shows a suspicious login from another IP... TrueTelltale Account Threat
TEST_4	Normal	No suspicious activity detected.

A status bar at the bottom right indicates "Build with Agent" and shows a progress bar for "intent classification system" which is 100% complete.

File Edit Selection View Go Run Terminal Help

RECENT SESSIONS

- intent classification system.py (Local) 1 hr
- Intent Classification Project (Engineering) (Local) 74 mins
- Python program to reverse a string... (Local) 1 hr

Build with Agent

All responses may be incomplete. Generate Agent instructions or embed AI into your codebase.

```
20 class ChatbotIntentClassifier:  
21     def add_sample_classifications(self) -> None:  
22         self.record_classification("zero_shot", "TEST_4", "General Question", 0.85,  
23                                     "Correctly classified location/store inquiry")  
24         self.record_classification("one_shot", "TEST_4", "General Question", 0.89,  
25                                     "Example helped with general question pattern")  
26         self.record_classification("few_shot", "TEST_4", "General Question", 0.93,  
27                                     "Few examples clarified general inquiry patterns")  
28  
29         # Sample classifications for TEST_5 (Account Issue)  
30         self.record_classification("zero_shot", "TEST_5", "Account Issue", 0.99,  
31                                     "Identified from 'delete account' keywords")  
32         self.record_classification("one_shot", "TEST_5", "Account Issue", 0.93,  
33                                     "Account management pattern reinforced")  
34         self.record_classification("few_shot", "TEST_5", "Account Issue", 0.97,  
35                                     "Strong account action pattern from examples")  
36  
37     print("\n✓ Added sample classification results")  
38
```

INTENT CLASSIFICATION RESULTS - COMPARISON TABLE

User Query	Actual Intent	System Intent	Confidence
* User: my account shows a suspicious login from another l...	True Intent: Account Issue	True Intent: Account Issue	0.99

Filter metrics: 0.84%

File Edit Selection View Go Run Terminal Help

RECENT SESSIONS

- intent classification system.py (Local) 1 hr
- Intent Classification Project (Engineering) (Local) 74 mins
- Python program to reverse a string... (Local) 1 hr

Build with Agent

All responses may be incomplete. Generate Agent instructions or embed AI into your codebase.

```
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_classifications)  
307  
308         return {  
309             "accuracy": accuracy,  
310             "avg_confidence": avg_confidence,  
311             "metrics_by_technique": metrics_by_technique  
312         }
```

INTENT CLASSIFICATION RESULTS - COMPARISON TABLE

User Query	Actual Intent	System Intent	Confidence
* User: my account shows a suspicious login from another l...	True Intent: Account Issue	True Intent: Account Issue	0.99

Filter metrics: 0.84%

The screenshot shows a developer's workspace with the following details:

- File Explorer:** Shows several projects and files, including "Email Classification System", "Intent Classification System", and "Intent Classification Analysis".
- Code Editor:** The main area displays Python code for "IntentClassificationAnalysis". The code includes methods for calculating metrics and displaying comparison tables.
- Terminal:** A terminal window at the bottom shows the command "git status" and indicates the repository is up-to-date.
- Bottom Bar:** Includes icons for file operations like Open, Save, and Print, as well as links to "Help", "About", and "Feedback".

The screenshot shows the PyCharm IDE interface with several windows open. The main editor window displays Python code for 'IntentClassificationAnalysis'. A floating 'Build with Agent' window is visible on the right, showing progress for 'intent classification system' and 'IntentClassificationAnalysis'. The bottom status bar indicates 'File integrity checked'.

```
File Edit Selection View Go Run Terminal Help ← → C:\ABC
```

Email Classification Prompt Engineering.ipynb email classification system Intent Classification Prompt Engineering.ipynb Intent classification system

RECENT SESSIONS

- intent classification system > IntentClassificationAnalysis > display_comparison_table
- 290 class IntentClassificationAnalysis:
- 318 def display_comparison_table(self) -> None:
- 320 query_clfs = by_query[query_id]
- 321 true_intent = query_clfs[0][“true_intent”]
- 322
- 323 print(F"\n★ {query_id}: {query_data[‘query’]}...")
- 324 print(F" True Intent: {true_intent}")
- 325 print("-" * 160)
- 326
- 327 header = F'{‘Technique’:<20} | {‘Predicted’:<25} | {‘Correct’:<10} | {‘Confidence’:<10}'
- 328 print(header)
- 329 print("-" * 160)
- 330
- 331 for clf in sorted(query_clfs, key=lambda x: x[“technique”]):
- 332 technique = clf[“technique”].replace(“_”, “-”).upper()
- 333 predicted = clf[“predicted_intent”]
- 334 correct = “✓ YES” if clf[“correct”] else “✗ NO”
- 335 confidence = F'{clf[‘confidence’]:.3f}'

INTENT CLASSIFICATION RESULTS - COMPARISON TABLE

TEST_3: my account shows a suspicious login from another I...

No Problems | IntentClassificationSystem

File integrity checked

Search

100% 100% 100% 100% 100% 100% 100% 100% 100% 100%

The screenshot shows a Microsoft Edge browser window with several tabs open. The main content area displays a Python script for a classification system, with three test cases listed below it:

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

Technique	Predicted	Correct	Confidence	Notes
HIT-SHOT	Account Issue	✓ YES	0.980	Mallinckrodt provided strong visual pass...
ONE-HOT	Account Issue	✓ YES	0.930	#Mallin improved confidence; security review...
ZERO-SHOT	Account Issue	✓ YES	0.980	Correctly identified from 'suspicious login' ...
- TEST 2: Is this product compatible with older devices?** Intent: Product Inquiry.

Technique	Predicted	Correct	Confidence	Notes
HIT-SHOT	Product Inquiry	✓ YES	0.950	Merck examples provided comprehensive product i...
ONE-HOT	Product Inquiry	✓ YES	0.920	One example - improved pattern recognition
ZERO-SHOT	Product Inquiry	✓ YES	0.980	Keywords: 'compatible', 'devices' identified; ...
- TEST 3: Is my package here delivered yet?** Intent: Order Status.

Technique	Predicted	Correct	Confidence	Notes
HIT-SHOT	Order Status	✓ YES	0.980	Merck examples provided strong visual pass...
ONE-HOT	Order Status	✓ YES	0.950	#Merck improved confidence; security review...
ZERO-SHOT	Order Status	✓ YES	0.980	Correctly identified from 'package here delivered ...

On the right side of the screen, there is a sidebar titled "Build with Agent" which includes a "Generate Agent instructions" button and a "Details" section. The bottom of the screen shows the Windows taskbar with various pinned icons and the system tray.

File Edit Selection View Go Run Terminal Help ⌘ ⌘ ⌘

Small Classification Engine (Python).pyb Intent classification system... Intent Classification: Intent Engineering (Python) Intent classification system... Intent classification system...

```
450 def main():
451     ...
452
453     classifier.display_prompts()
```

TEST 3: Is my package been delivered yet?...
Is my package been delivered yet?...

Technique	Predicted	Actual	Confidence	Notes
FW SHOT	Order Status	✓ Yes	0.950	Multi-line order tracking query on highly rhythmic pattern
ONE-SHOT	Order Status	✓ YES	0.910	Example strengthened delivery tracking pattern
ZERO-SHOT	Order Status	✓ Yes	0.870	Identified from "delivered", "package" keyword

TEST 4: Do you have a physical store near me?...
Do you have a physical store near me?...

Technique	Predicted	Actual	Confidence	Notes
FW SHOT	General Question	✓ YES	0.940	New examples clarified generic inquiry pattern
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?...
How do I delete my account?...

Technique	Predicted	Actual	Confidence	Notes
FW SHOT	Account Issue	✓ YES	0.970	Some account action pattern from examples
ONE-SHOT	Account Issue	✓ YES	0.900	Account management pattern reinforced

Nifty prompt -0.34%

File Edit Selection View Go Run Terminal Help ⌘ ⌘ ⌘

Small Classification Engine (Python).pyb Intent classification system... Intent Classification: Intent Engineering (Python) Intent classification system... Intent classification system...

```
450 def main():
451     ...
452
453     classifier.display_prompts()
```

TEST 3: Is my package been delivered yet?...
Is my package been delivered yet?...

Technique	Predicted	Actual	Confidence	Notes
ZERO-SHOT	Account Issue	✓ YES	0.990	Identified from "delete account" keywords

SUMMARY STATISTICS

Technique	Correct/Total	Accuracy %	Avg Confidence
ZERO-SHOT	2/2	100.0	0.920
ONE-SHOT	2/2	100.0	0.950
FW-SHOT	3/3	100.0	0.950

DRIVE TECHNICAL OR PREDICTIVENESS ANALYSIS

ZERO-SHOT PROMPTING

Definition: Classification without any labeled examples.

Characteristics:

- Relyes on model's understanding of intent categories
- Bottom-up approach with minimal prompt engineering
- May struggle with ambiguous queries

Nifty prompt -0.34%

File Edit Selection View Go Run Terminal Help ⌘ ⌘ ⌘

Small Classification Engine (Python).pyb Intent classification system... Intent Classification: Intent Engineering (Python) Intent classification system... Intent classification system...

```
450 def main():
451     ...
452
453     classifier.display_prompts()
```

TEST 3: Is my package been delivered yet?...
Is my package been delivered yet?...

Technique	Predicted	Actual	Confidence	Notes
ZERO-SHOT	Account Issue	✓ YES	0.990	Identified from "delete account" keywords

SUMMARY STATISTICS

Technique	Correct/Total	Accuracy %	Avg Confidence
ZERO-SHOT	2/2	100.0	0.920
ONE-SHOT	2/2	100.0	0.950
FW-SHOT	3/3	100.0	0.950

DRIVE TECHNICAL OR PREDICTIVENESS ANALYSIS

ZERO-SHOT PROMPTING

Definition: Classification without any labeled examples.

Characteristics:

- Relyes on model's understanding of intent categories
- Bottom-up approach with minimal prompt engineering
- May struggle with ambiguous queries

Nifty prompt -0.34%

The screenshot shows a Jupyter Notebook interface with several tabs open. The active tab displays a Python script with two main sections: 'ZERO-SHOT PROMPT' and 'ONE-SHOT PROMPT'. The 'ZERO-SHOT PROMPT' section includes a detailed description of the classifier's behavior when no labeled examples are provided, mentioning its strengths (understanding intent categories) and weaknesses (struggling with ambiguous queries). It also lists results: Accuracy: 100.0% (5/5 correct), Average Confidence: 0.988, and Effectiveness: DRAFTY - Foundation for comparison. The 'ONE-SHOT PROMPT' section provides a definition, characteristics (single reference pattern), and results: Accuracy: 100.0% (5/5 correct), Average Confidence: 0.922, Improvement over zero shot: 0.000, and Effectiveness: GOOD - Recommended for simple classifiers. A 'Build with Agent' sidebar is visible on the right.

This screenshot is nearly identical to the one above, showing the same Jupyter Notebook interface with the same tabs and code. The results for the 'ONE-SHOT PROMPT' section have changed slightly: Accuracy: 100.0% (5/5 correct), Average Confidence: 0.998, Improvement over zero-shot: 40.0%, and Effectiveness: FINE-TUNED - Best for production systems. The 'Build with Agent' sidebar remains present.

The screenshot shows the Microsoft Visual Studio Code (VS Code) interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar. There are four tabs open in the editor area:

- IntentClassificationEngineEng.py
- intent_classification_system.py
- IntentClassification_PythonEng.py
- intent_classification_system.py

The main code editor displays the following Python code:

```
459 def main():
460     ...
461
462     classifier.display_prompts()
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498     classifier.display_prompts()
```

Below the code editor, the status bar shows "Performance: Observers: ✓ Few-shot achieves high confidence (>0.55)".

A sidebar on the right lists "RECENT SESSIONS" with items like "fewshot prompt writing for phisheng", "IntentReferent and Evaluation NLU", "Graphical", and "Python program to reverse a string...".

The bottom right corner features a "Build with Agent" button with the text "All requests may be incomplete. Generate Agent instructions to validate AI units in your codebase." and a "Build" progress bar.

The taskbar at the bottom shows various pinned icons, including Niftytrips, a search bar, and system icons for battery, volume, and network.