AI Assistant Coding

Assignment 4.1.2

Name : Shaestha Kownain **HT. No :** 2303A52503 **Batch:** 50

Objective: To explore and compare Zero-shot, One-shot, and Few-shot prompting techniques for classifying emails into predefined categories using a large language model (LLM).

1. Suppose that you work for a company that receives hundreds of customer emails daily. Management wants to automatically classify emails into categories like "Billing", "Technical Support", "Feedback", and "Others" before assigning them to appropriate departments. Instead of training a new model, your task is to use prompt engineering techniques with an existing LLM to handle the classification.

Tasks to be completed are as below

a. Prepare Sample Data:

• Create or collect 10 short email samples, each belonging to one of the 4 categories.

b. Zero-shot Prompting:

• Design a prompt that asks the LLM to classify a single email without providing any examples.

• Example prompt:

"Classify the following email into one of the following categories: Billing, Technical Support, Feedback, Others. Email: 'I have not received my invoice for last month.'"

c. One-shot Prompting:

• **Add one labeled example before asking the model to classify a new email.**

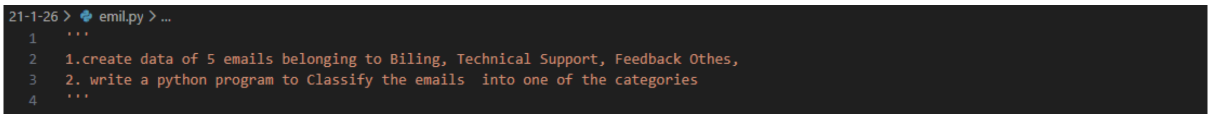
d. Few-shot Prompting:

• **Use 3–5 labeled examples in your prompt before asking the model to classify a new email.**

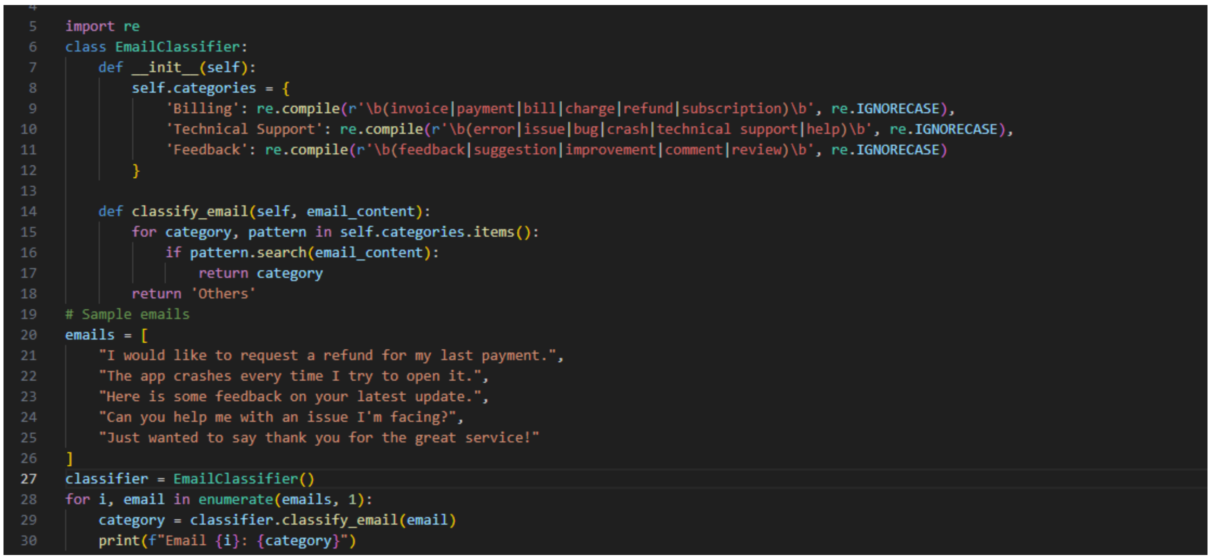
e. Evaluation:

• **Run all three techniques on the same set of 5 test emails.**

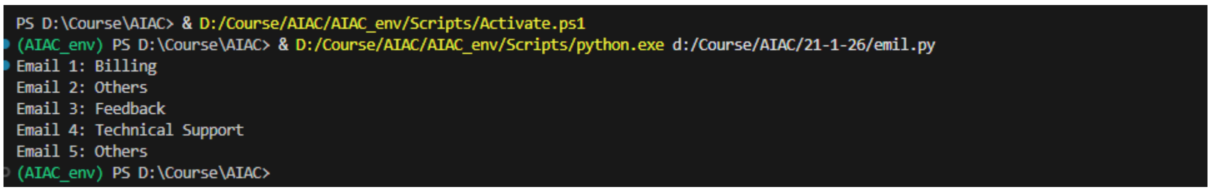
• **Compare and document the accuracy and clarity of responses.**



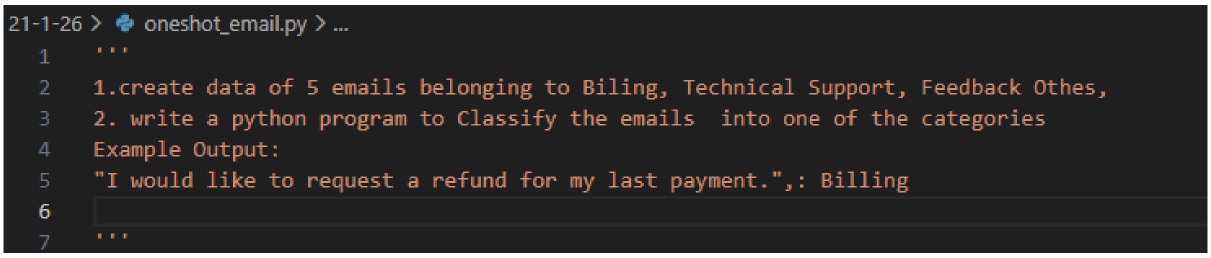
```python
'''
1.create data of 5 emails belonging to Biling, Technical Support, Feedback Othes,
2. write a python program to Classify the emails  into one of the categories
'''
```



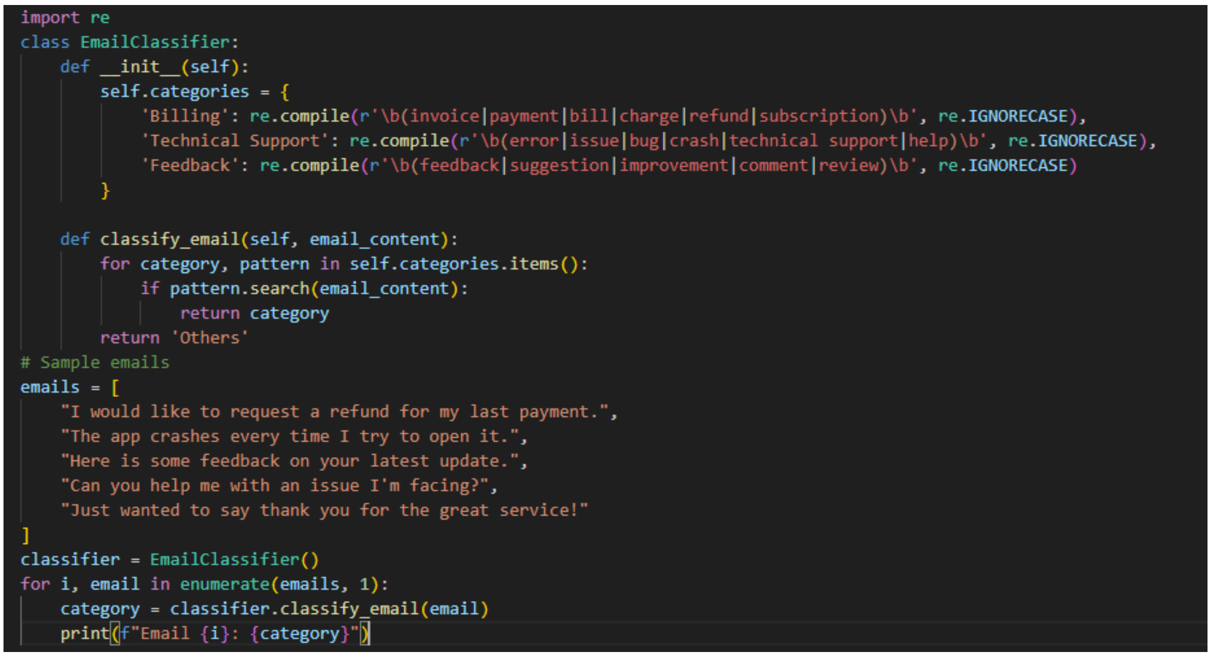
```python
import re
class EmailClassifier:
    def __init__(self):
        self.categories = {
            'Billing': re.compile(r'\b(invoice|payment|bill|charge|refund|subscription)\b', re.IGNORECASE),
            'Technical Support': re.compile(r'\b(error|issue|bug|crash|technical support|help)\b', re.IGNORECASE),
            'Feedback': re.compile(r'\b(feedback|suggestion|improvement|comment|review)\b', re.IGNORECASE)
        }

    def classify_email(self, email_content):
        for category, pattern in self.categories.items():
            if pattern.search(email_content):
                return category
        return 'Others'
# Sample emails
emails = [
    "I would like to request a refund for my last payment.",
    "The app crashes every time I try to open it.",
    "Here is some feedback on your latest update.",
    "Can you help me with an issue I'm facing?",
    "Just wanted to say thank you for the great service!"
]
classifier = EmailClassifier()
for i, email in enumerate(emails, 1):
    category = classifier.classify_email(email)
    print(f"Email {i}: {category}")
```



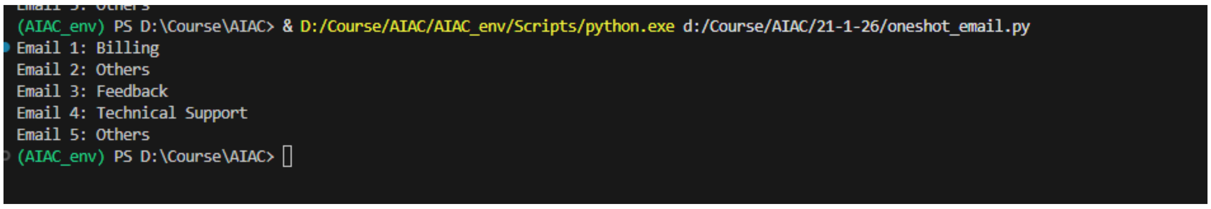
```
PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/Activate.ps1
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/21-1-26/emil.py
Email 1: Billing
Email 2: Others
Email 3: Feedback
Email 4: Technical Support
Email 5: Others
(AIAC_env) PS D:\Course\AIAC>
```



```python
'''
1.create data of 5 emails belonging to Biling, Technical Support, Feedback Othes,
2. write a python program to Classify the emails  into one of the categories
Example Output:
"I would like to request a refund for my last payment.",: Billing

'''
```



```python
import re
class EmailClassifier:
    def __init__(self):
        self.categories = {
            'Billing': re.compile(r'\b(invoice|payment|bill|charge|refund|subscription)\b', re.IGNORECASE),
            'Technical Support': re.compile(r'\b(error|issue|bug|crash|technical support|help)\b', re.IGNORECASE),
            'Feedback': re.compile(r'\b(feedback|suggestion|improvement|comment|review)\b', re.IGNORECASE)
        }

    def classify_email(self, email_content):
        for category, pattern in self.categories.items():
            if pattern.search(email_content):
                return category
        return 'Others'
# Sample emails
emails = [
    "I would like to request a refund for my last payment.",
    "The app crashes every time I try to open it.",
    "Here is some feedback on your latest update.",
    "Can you help me with an issue I'm facing?",
    "Just wanted to say thank you for the great service!"
]
classifier = EmailClassifier()
for i, email in enumerate(emails, 1):
    category = classifier.classify_email(email)
    print(f"Email {i}: {category}")
```



```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/21-1-26/oneshot_email.py
Email 1: Billing
Email 2: Others
Email 3: Feedback
Email 4: Technical Support
Email 5: Others
(AIAC_env) PS D:\Course\AIAC> 
```

```python
import re
class EmailClassifier:
    def __init__(self):
        self.categories = {
            'Billing': re.compile(r'\b(invoice|payment|bill|charge|refund|subscription)\b', re.IGNORECASE),
            'Technical Support': re.compile(r'\b(error|issue|bug|crash|technical support|help)\b', re.IGNORECASE),
            'Feedback': re.compile(r'\b(feedback|suggestion|improvement|comment|review)\b', re.IGNORECASE)
        }

    def classify_email(self, email_content):
        for category, pattern in self.categories.items():
            if pattern.search(email_content):
                return category
        return 'Others'
# Sample emails
emails = [
    "I would like to request a refund for my last payment.",
    "The app crashes every time I try to open it.",
    "Here is some feedback on your latest update.",
    "Can you help me with an issue I'm facing?",
    "Just wanted to say thank you for the great service!"
]
classifier = EmailClassifier()
for i, email in enumerate(emails, 1):
    category = classifier.classify_email(email)
    print(f"Email {i}: {category}")
```

```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/21-1-26/oneshot_email.py
Email 1: Billing
Email 2: Others
Email 3: Feedback
Email 4: Technical Support
Email 5: Others
(AIAC_env) PS D:\Course\AIAC> 
```

Q3.

3. Programming Question Type Identification

Scenario:

A coding help chatbot must classify queries into Syntax Error, Logic Error, Optimization, or Conceptual Question.

Tasks:

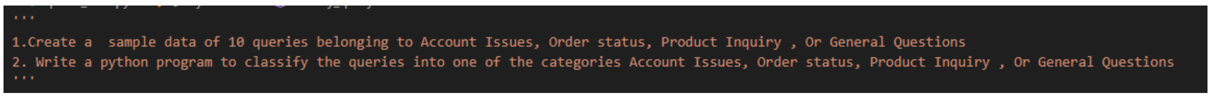
a. Prepare coding-related user queries.

b. Perform Zero-shot classification.

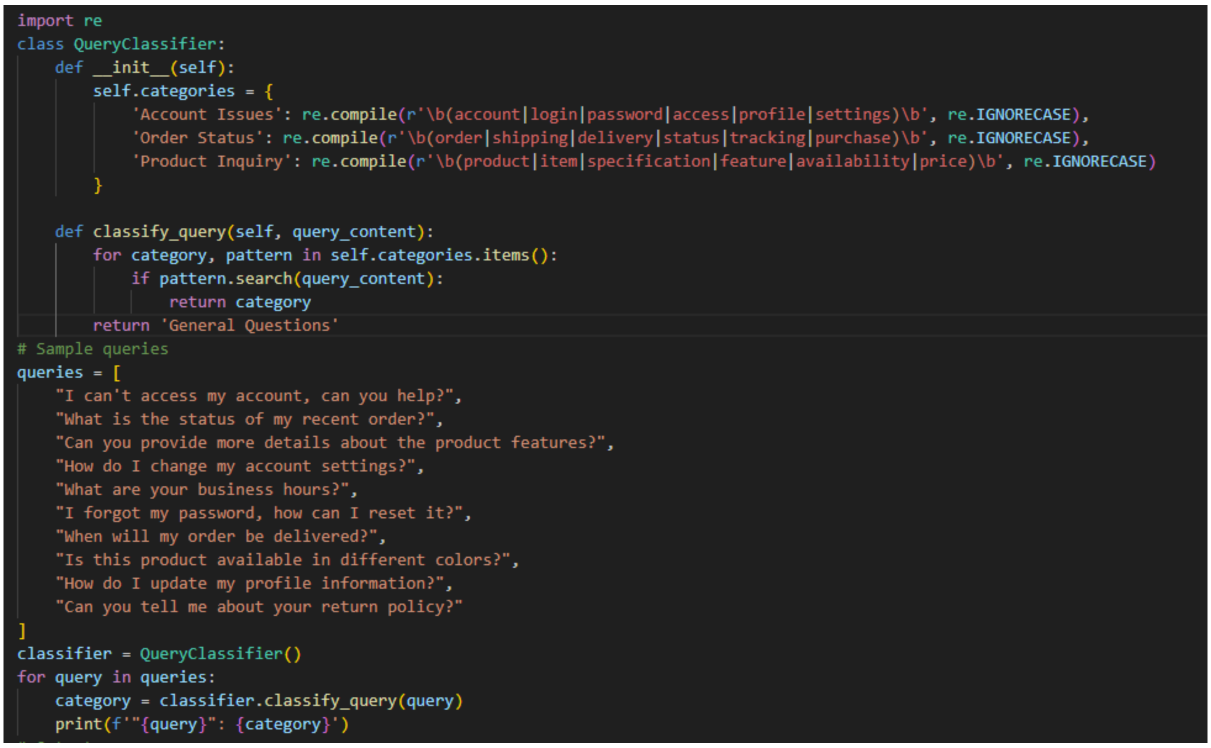
c. Perform One-shot classification.

d. Perform Few-shot classification.

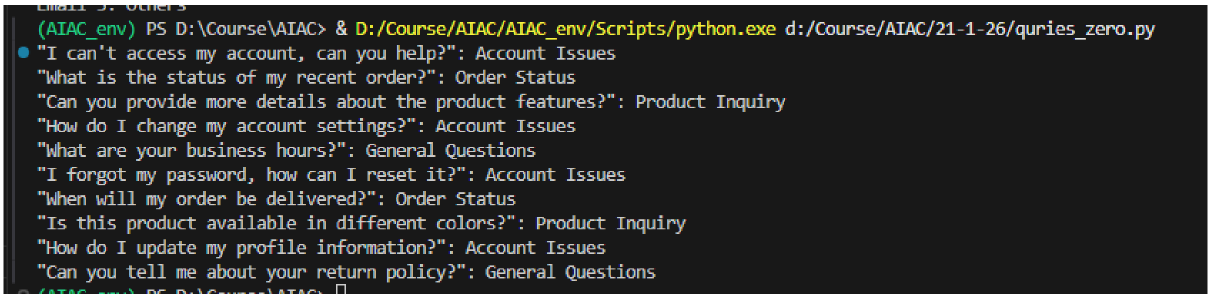
e. Analyze improvements in technical accuracy.



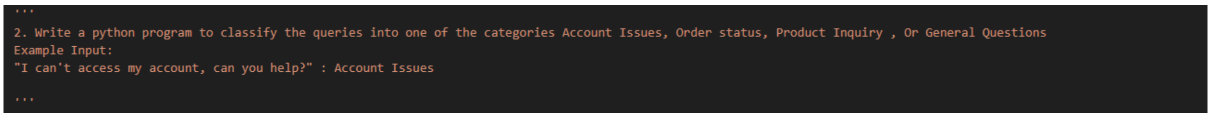
```
'''
1.Create a  sample data of 10 queries belonging to Account Issues, Order status, Product Inquiry , Or General Questions
2. Write a python program to classify the queries into one of the categories Account Issues, Order status, Product Inquiry , Or General Questions
'''
```



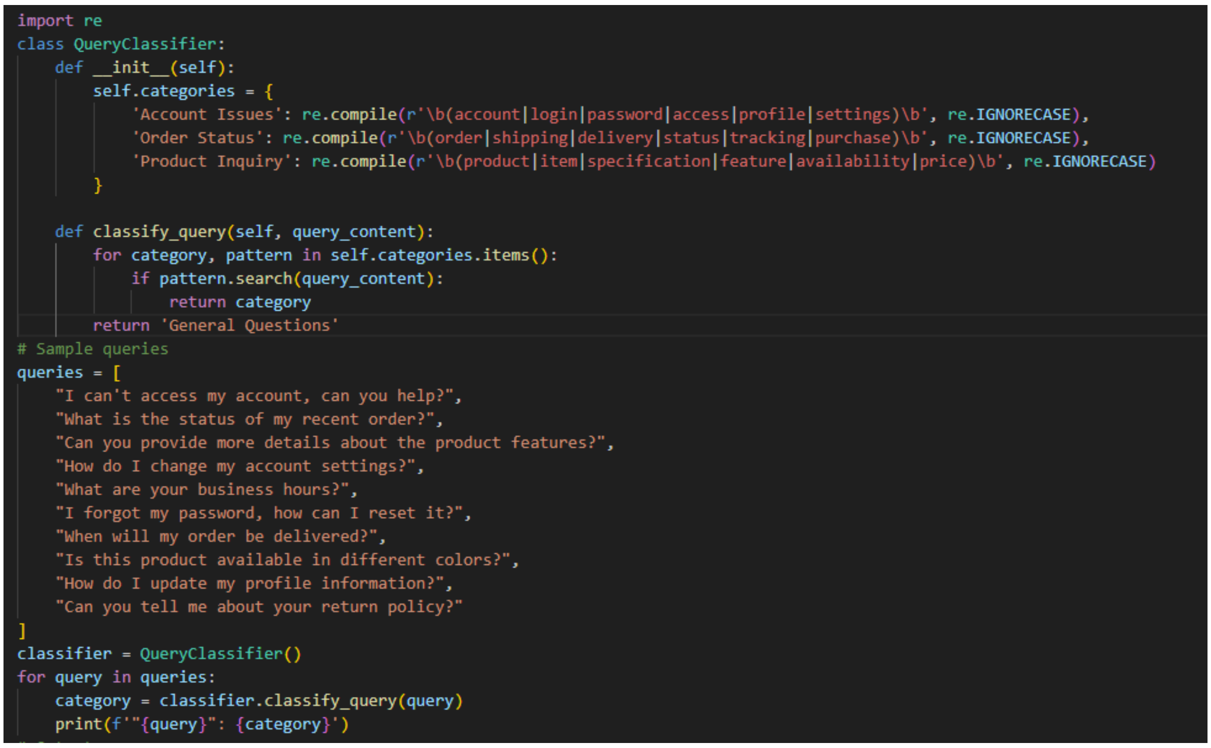
```python
import re
class QueryClassifier:
    def __init__(self):
        self.categories = {
            'Account Issues': re.compile(r'\b(account|login|password|access|profile|settings)\b', re.IGNORECASE),
            'Order Status': re.compile(r'\b(order|shipping|delivery|status|tracking|purchase)\b', re.IGNORECASE),
            'Product Inquiry': re.compile(r'\b(product|item|specification|feature|availability|price)\b', re.IGNORECASE)
        }

    def classify_query(self, query_content):
        for category, pattern in self.categories.items():
            if pattern.search(query_content):
                return category
        return 'General Questions'
# Sample queries
queries = [
    "I can't access my account, can you help?",
    "What is the status of my recent order?",
    "Can you provide more details about the product features?",
    "How do I change my account settings?",
    "What are your business hours?",
    "I forgot my password, how can I reset it?",
    "When will my order be delivered?",
    "Is this product available in different colors?",
    "How do I update my profile information?",
    "Can you tell me about your return policy?"
]
classifier = QueryClassifier()
for query in queries:
    category = classifier.classify_query(query)
    print(f'"{query}": {category}')
```



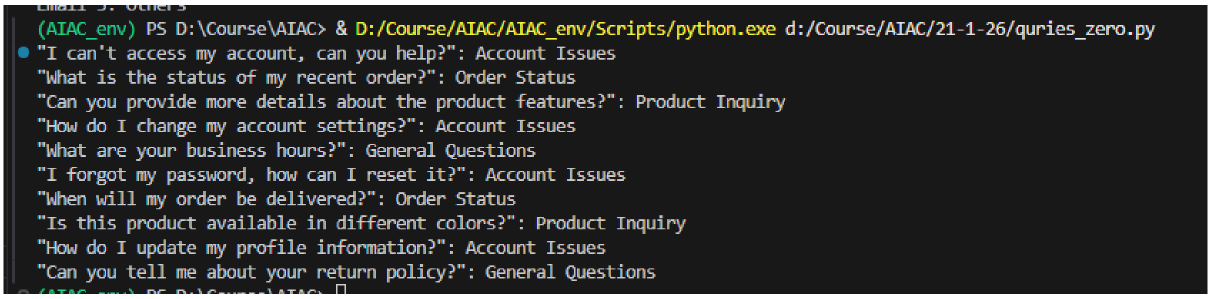
```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/21-1-26/quries_zero.py
"I can't access my account, can you help?": Account Issues
"What is the status of my recent order?": Order Status
"Can you provide more details about the product features?": Product Inquiry
"How do I change my account settings?": Account Issues
"What are your business hours?": General Questions
"I forgot my password, how can I reset it?": Account Issues
"When will my order be delivered?": Order Status
"Is this product available in different colors?": Product Inquiry
"How do I update my profile information?": Account Issues
"Can you tell me about your return policy?": General Questions
```



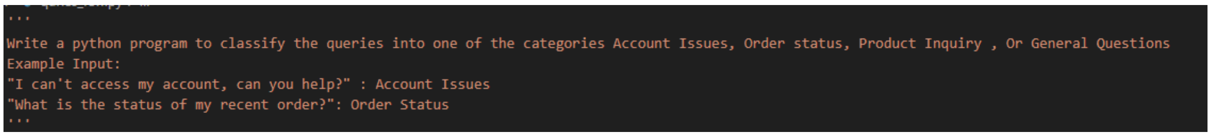
```
'''
2. Write a python program to classify the queries into one of the categories Account Issues, Order status, Product Inquiry , Or General Questions
Example Input:
"I can't access my account, can you help?" : Account Issues

'''
```



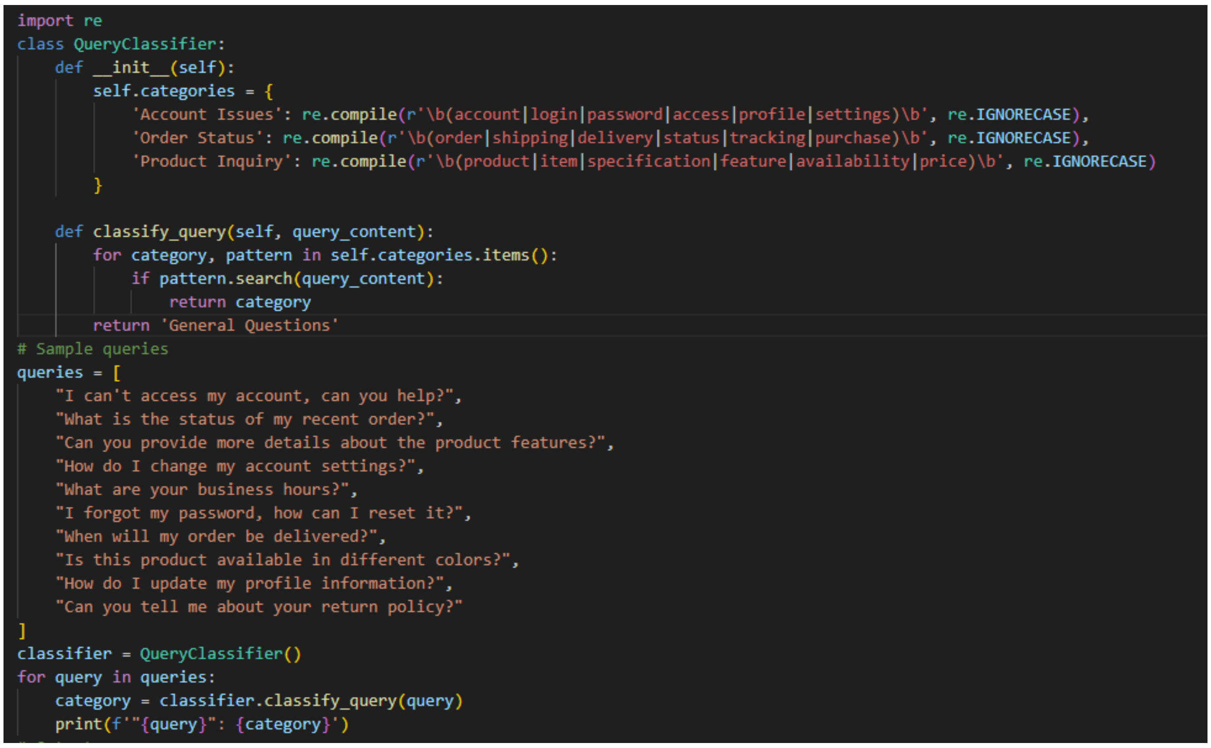
```python
import re
class QueryClassifier:
    def __init__(self):
        self.categories = {
            'Account Issues': re.compile(r'\b(account|login|password|access|profile|settings)\b', re.IGNORECASE),
            'Order Status': re.compile(r'\b(order|shipping|delivery|status|tracking|purchase)\b', re.IGNORECASE),
            'Product Inquiry': re.compile(r'\b(product|item|specification|feature|availability|price)\b', re.IGNORECASE)
        }

    def classify_query(self, query_content):
        for category, pattern in self.categories.items():
            if pattern.search(query_content):
                return category
        return 'General Questions'
# Sample queries
queries = [
    "I can't access my account, can you help?",
    "What is the status of my recent order?",
    "Can you provide more details about the product features?",
    "How do I change my account settings?",
    "What are your business hours?",
    "I forgot my password, how can I reset it?",
    "When will my order be delivered?",
    "Is this product available in different colors?",
    "How do I update my profile information?",
    "Can you tell me about your return policy?"
]
classifier = QueryClassifier()
for query in queries:
    category = classifier.classify_query(query)
    print(f'"{query}": {category}')
```



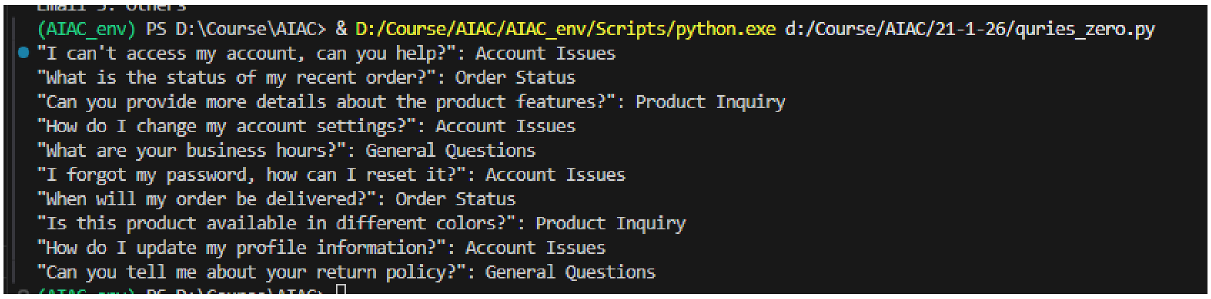
```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/21-1-26/quries_zero.py
"I can't access my account, can you help?": Account Issues
"What is the status of my recent order?": Order Status
"Can you provide more details about the product features?": Product Inquiry
"How do I change my account settings?": Account Issues
"What are your business hours?": General Questions
"I forgot my password, how can I reset it?": Account Issues
"When will my order be delivered?": Order Status
"Is this product available in different colors?": Product Inquiry
"How do I update my profile information?": Account Issues
"Can you tell me about your return policy?": General Questions
```



```
'''
Write a python program to classify the queries into one of the categories Account Issues, Order status, Product Inquiry , Or General Questions
Example Input:
"I can't access my account, can you help?" : Account Issues
"What is the status of my recent order?": Order Status
'''
```



```python
import re
class QueryClassifier:
    def __init__(self):
        self.categories = {
            'Account Issues': re.compile(r'\b(account|login|password|access|profile|settings)\b', re.IGNORECASE),
            'Order Status': re.compile(r'\b(order|shipping|delivery|status|tracking|purchase)\b', re.IGNORECASE),
            'Product Inquiry': re.compile(r'\b(product|item|specification|feature|availability|price)\b', re.IGNORECASE)
        }

    def classify_query(self, query_content):
        for category, pattern in self.categories.items():
            if pattern.search(query_content):
                return category
        return 'General Questions'
# Sample queries
queries = [
    "I can't access my account, can you help?",
    "What is the status of my recent order?",
    "Can you provide more details about the product features?",
    "How do I change my account settings?",
    "What are your business hours?",
    "I forgot my password, how can I reset it?",
    "When will my order be delivered?",
    "Is this product available in different colors?",
    "How do I update my profile information?",
    "Can you tell me about your return policy?"
]
classifier = QueryClassifier()
for query in queries:
    category = classifier.classify_query(query)
    print(f'"{query}": {category}')
```



```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/21-1-26/quries_zero.py
"I can't access my account, can you help?": Account Issues
"What is the status of my recent order?": Order Status
"Can you provide more details about the product features?": Product Inquiry
"How do I change my account settings?": Account Issues
"What are your business hours?": General Questions
"I forgot my password, how can I reset it?": Account Issues
"When will my order be delivered?": Order Status
"Is this product available in different colors?": Product Inquiry
"How do I update my profile information?": Account Issues
"Can you tell me about your return policy?": General Questions
```

---

2. Travel Query Classification

Scenario:

A travel assistant must classify queries into Flight Booking, Hotel Booking, Cancellation, or General Travel Info.

Tasks:

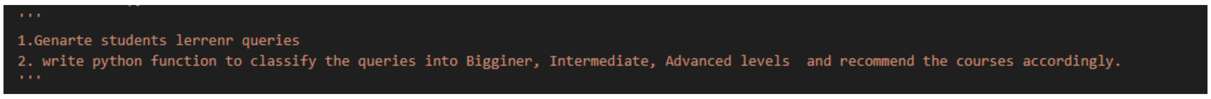
a. Prepare labeled travel queries.

b. Apply Zero-shot prompting.

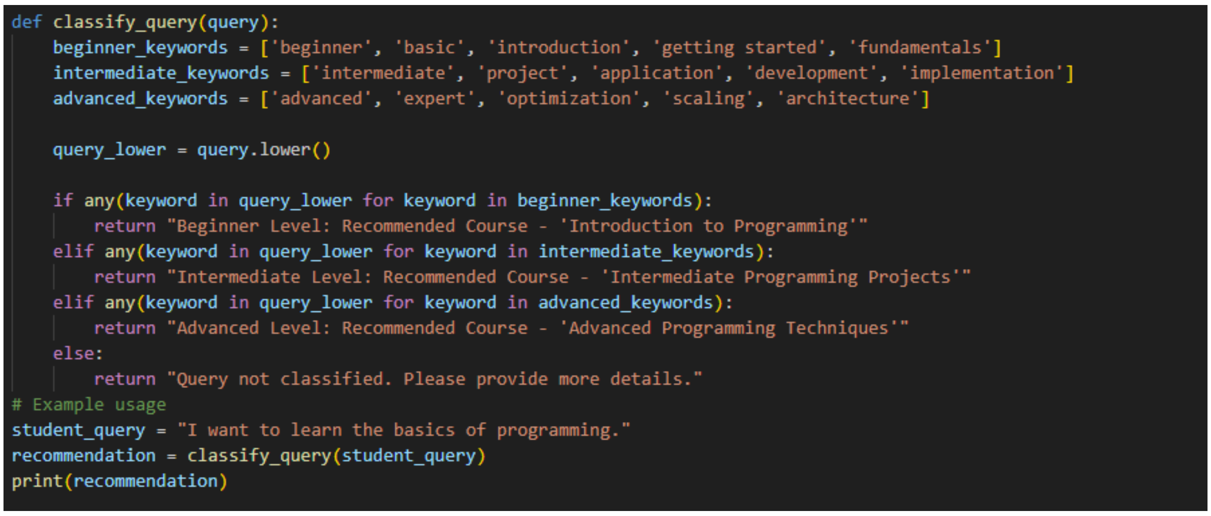
c. Apply One-shot prompting.

d. Apply Few-shot prompting.

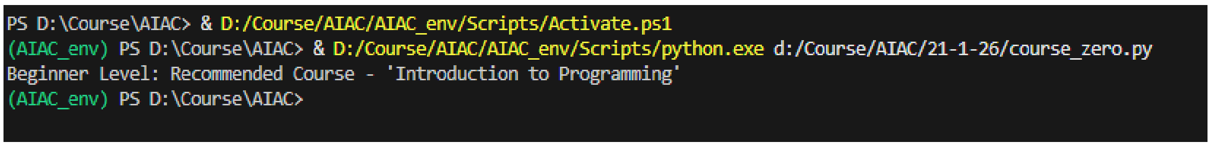
e. Compare response consistency.



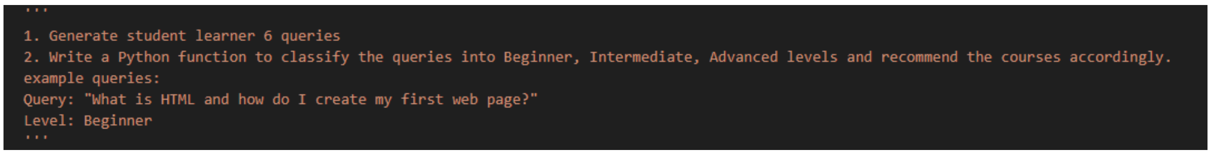
```
'''
1.Genarte students lerrenr queries
2. write python function to classify the queries into Bigginer, Intermediate, Advanced levels  and recommend the courses accordingly.
'''
```



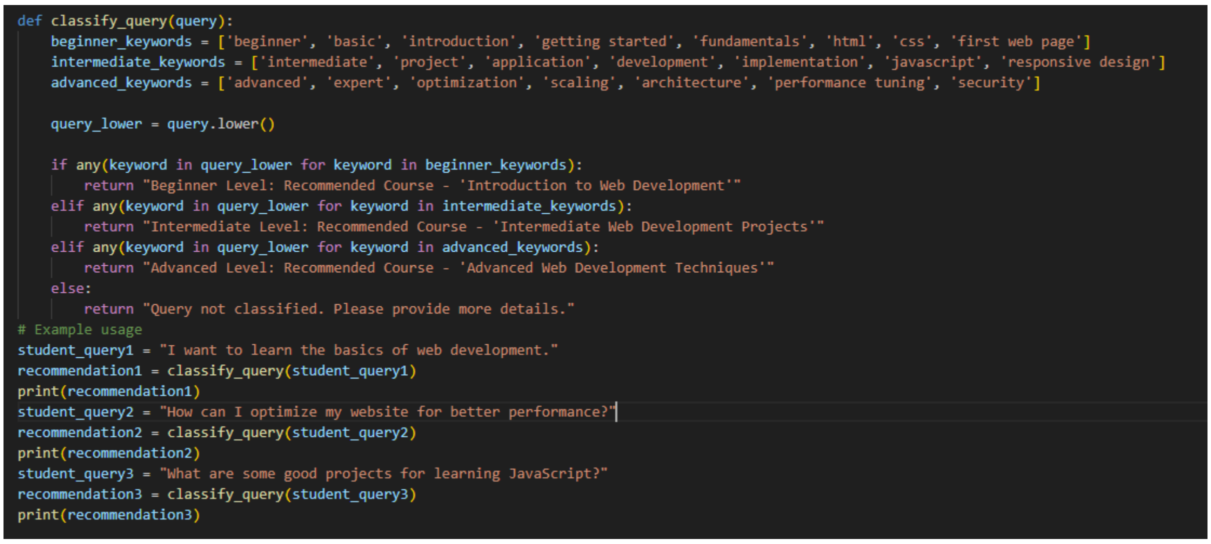
```python
def classify_query(query):
    beginner_keywords = ['beginner', 'basic', 'introduction', 'getting started', 'fundamentals']
    intermediate_keywords = ['intermediate', 'project', 'application', 'development', 'implementation']
    advanced_keywords = ['advanced', 'expert', 'optimization', 'scaling', 'architecture']

    query_lower = query.lower()

    if any(keyword in query_lower for keyword in beginner_keywords):
        return "Beginner Level: Recommended Course - 'Introduction to Programming'"
    elif any(keyword in query_lower for keyword in intermediate_keywords):
        return "Intermediate Level: Recommended Course - 'Intermediate Programming Projects'"
    elif any(keyword in query_lower for keyword in advanced_keywords):
        return "Advanced Level: Recommended Course - 'Advanced Programming Techniques'"
    else:
        return "Query not classified. Please provide more details."
# Example usage
student_query = "I want to learn the basics of programming."
recommendation = classify_query(student_query)
print(recommendation)
```



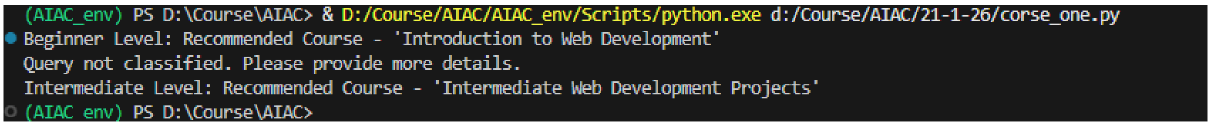
```
PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/Activate.ps1
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/21-1-26/course_zero.py
Beginner Level: Recommended Course - 'Introduction to Programming'
(AIAC_env) PS D:\Course\AIAC>
```



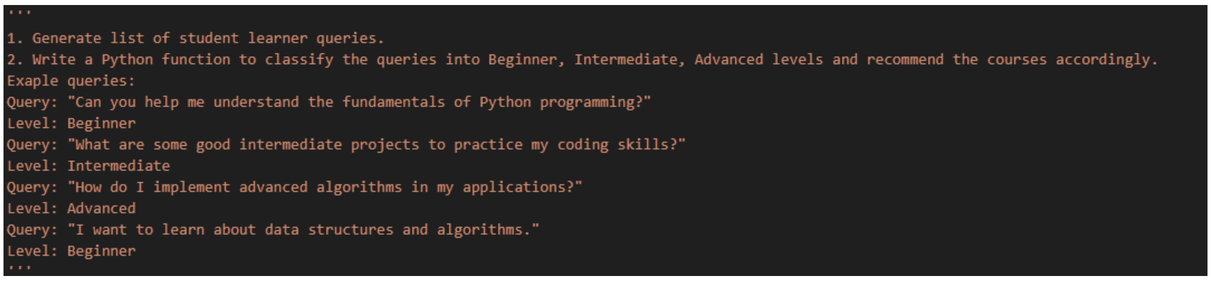
```
'''
1. Generate student learner 6 queries
2. Write a Python function to classify the queries into Beginner, Intermediate, Advanced levels and recommend the courses accordingly.
example queries:
Query: "What is HTML and how do I create my first web page?"
Level: Beginner
'''
```



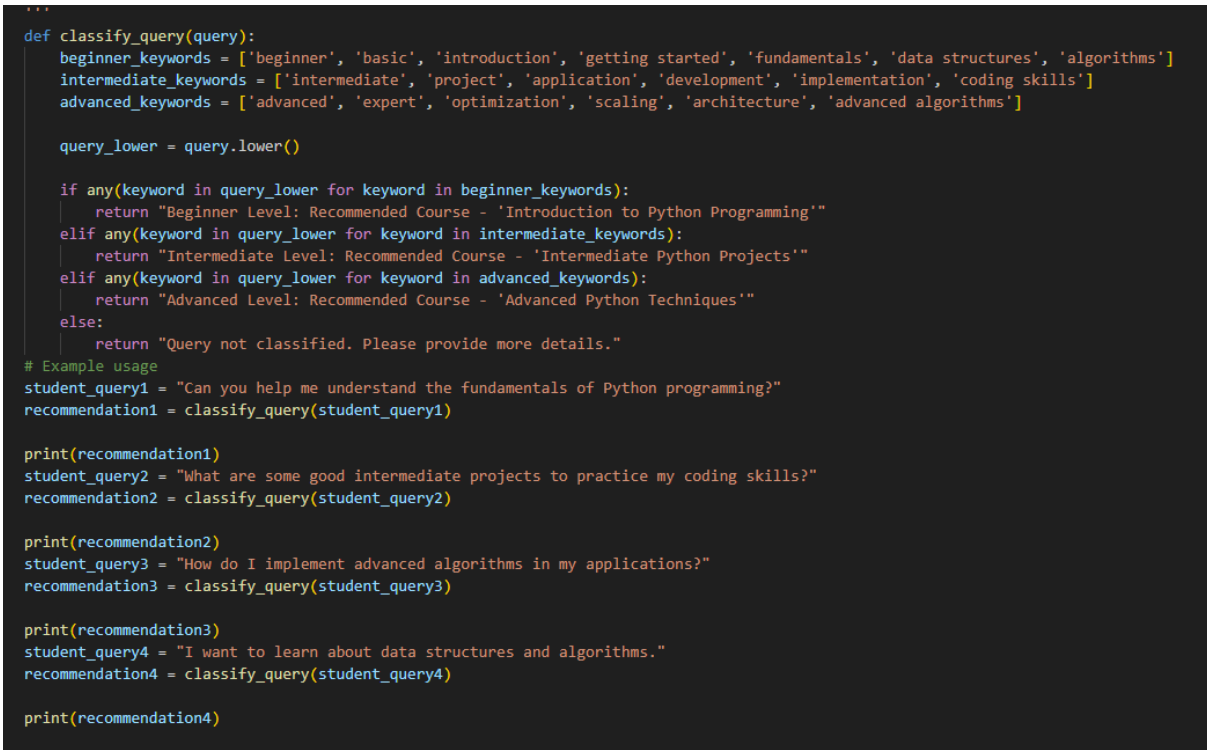
```python
def classify_query(query):
    beginner_keywords = ['beginner', 'basic', 'introduction', 'getting started', 'fundamentals', 'html', 'css', 'first web page']
    intermediate_keywords = ['intermediate', 'project', 'application', 'development', 'implementation', 'javascript', 'responsive design']
    advanced_keywords = ['advanced', 'expert', 'optimization', 'scaling', 'architecture', 'performance tuning', 'security']

    query_lower = query.lower()

    if any(keyword in query_lower for keyword in beginner_keywords):
        return "Beginner Level: Recommended Course - 'Introduction to Web Development'"
    elif any(keyword in query_lower for keyword in intermediate_keywords):
        return "Intermediate Level: Recommended Course - 'Intermediate Web Development Projects'"
    elif any(keyword in query_lower for keyword in advanced_keywords):
        return "Advanced Level: Recommended Course - 'Advanced Web Development Techniques'"
    else:
        return "Query not classified. Please provide more details."
# Example usage
student_query1 = "I want to learn the basics of web development."
recommendation1 = classify_query(student_query1)
print(recommendation1)
student_query2 = "How can I optimize my website for better performance?"
recommendation2 = classify_query(student_query2)
print(recommendation2)
student_query3 = "What are some good projects for learning JavaScript?"
recommendation3 = classify_query(student_query3)
print(recommendation3)
```



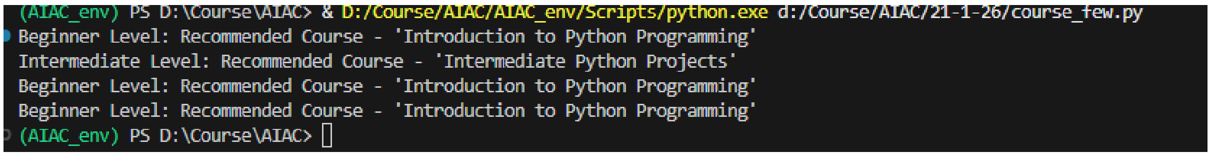
```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/21-1-26/corse_one.py
Beginner Level: Recommended Course - 'Introduction to Web Development'
Query not classified. Please provide more details.
Intermediate Level: Recommended Course - 'Intermediate Web Development Projects'
(AIAC_env) PS D:\Course\AIAC>
```



```
'''
1. Generate list of student learner queries.
2. Write a Python function to classify the queries into Beginner, Intermediate, Advanced levels and recommend the courses accordingly.
Exaple queries:
Query: "Can you help me understand the fundamentals of Python programming?"
Level: Beginner
Query: "What are some good intermediate projects to practice my coding skills?"
Level: Intermediate
Query: "How do I implement advanced algorithms in my applications?"
Level: Advanced
Query: "I want to learn about data structures and algorithms."
Level: Beginner
'''
```



```python
'''
def classify_query(query):
    beginner_keywords = ['beginner', 'basic', 'introduction', 'getting started', 'fundamentals', 'data structures', 'algorithms']
    intermediate_keywords = ['intermediate', 'project', 'application', 'development', 'implementation', 'coding skills']
    advanced_keywords = ['advanced', 'expert', 'optimization', 'scaling', 'architecture', 'advanced algorithms']

    query_lower = query.lower()

    if any(keyword in query_lower for keyword in beginner_keywords):
        return "Beginner Level: Recommended Course - 'Introduction to Python Programming'"
    elif any(keyword in query_lower for keyword in intermediate_keywords):
        return "Intermediate Level: Recommended Course - 'Intermediate Python Projects'"
    elif any(keyword in query_lower for keyword in advanced_keywords):
        return "Advanced Level: Recommended Course - 'Advanced Python Techniques'"
    else:
        return "Query not classified. Please provide more details."
# Example usage
student_query1 = "Can you help me understand the fundamentals of Python programming?"
recommendation1 = classify_query(student_query1)

print(recommendation1)
student_query2 = "What are some good intermediate projects to practice my coding skills?"
recommendation2 = classify_query(student_query2)

print(recommendation2)
student_query3 = "How do I implement advanced algorithms in my applications?"
recommendation3 = classify_query(student_query3)

print(recommendation3)
student_query4 = "I want to learn about data structures and algorithms."
recommendation4 = classify_query(student_query4)

print(recommendation4)
```



```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/21-1-26/course_few.py
Beginner Level: Recommended Course - 'Introduction to Python Programming'
Intermediate Level: Recommended Course - 'Intermediate Python Projects'
Beginner Level: Recommended Course - 'Introduction to Python Programming'
Beginner Level: Recommended Course - 'Introduction to Python Programming'
(AIAC_env) PS D:\Course\AIAC> 
```

Q4. 4. Social Media Post Categorization

Scenario:

A social media analytics tool must classify posts into Promotion,

Complaint, Appreciation, or Inquiry.

Tasks:

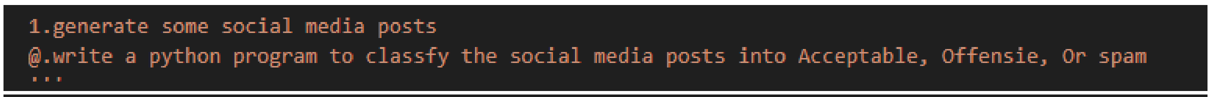
1. Prepare sample social media posts.

2. Use Zero-shot prompting.

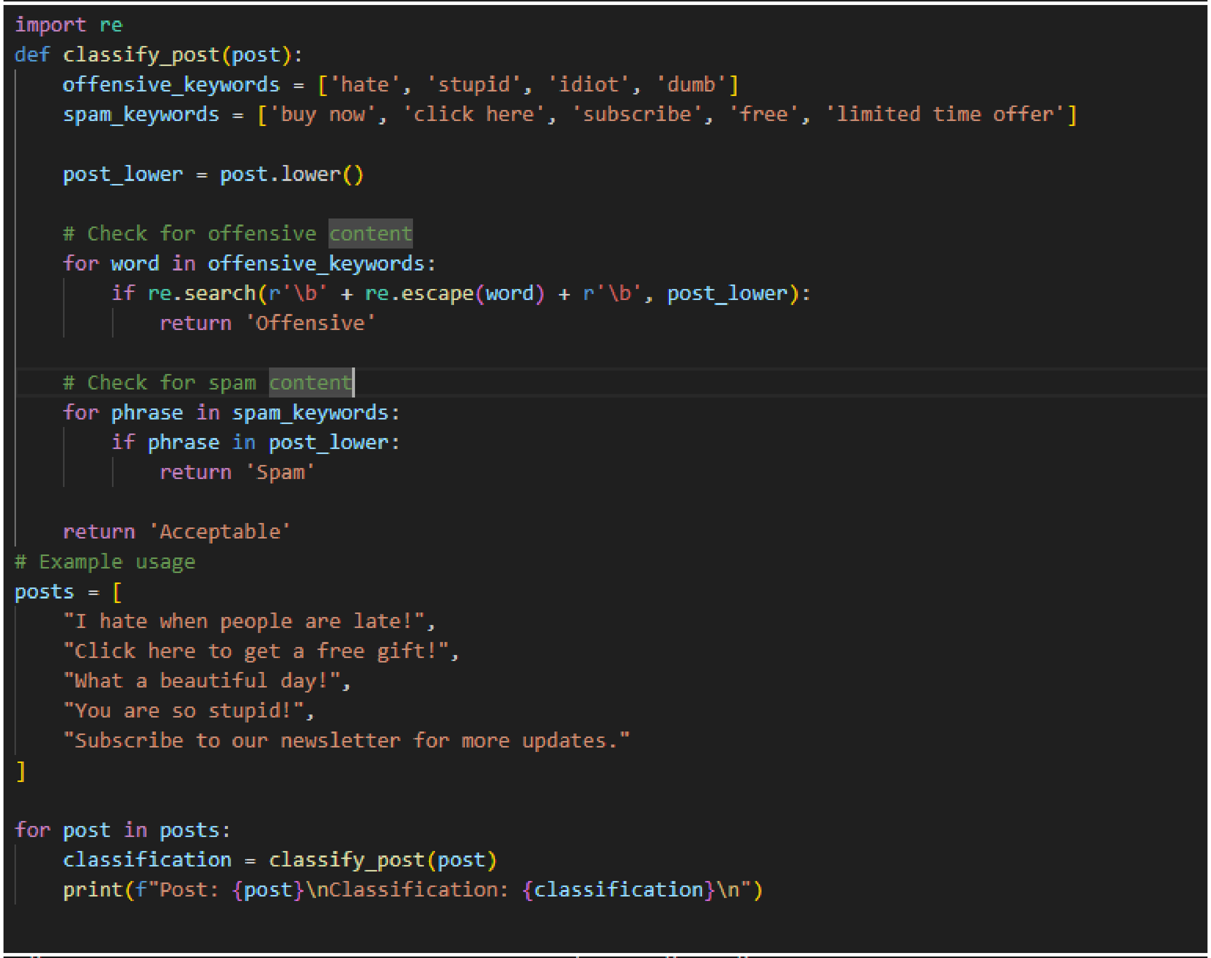
3. Use One-shot prompting.

4. Use Few-shot prompting.

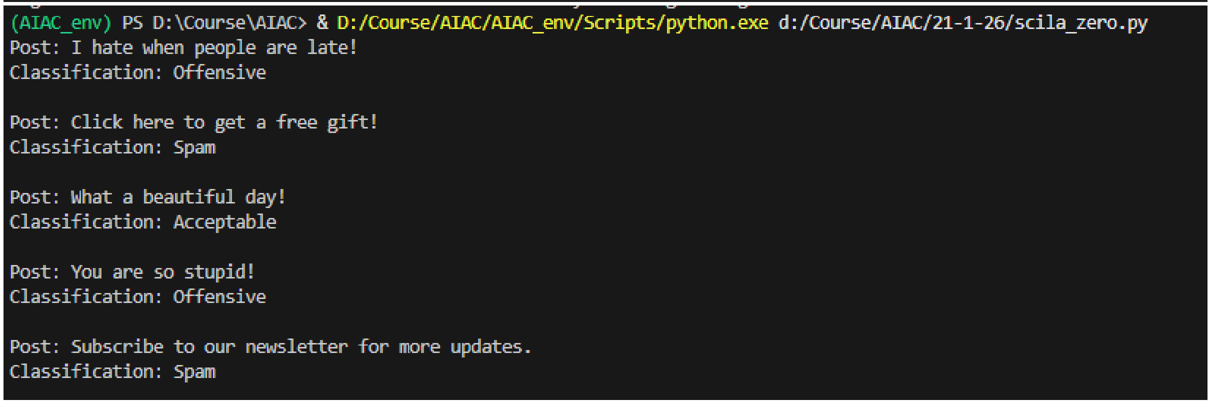
5. Analyze informal language handling.



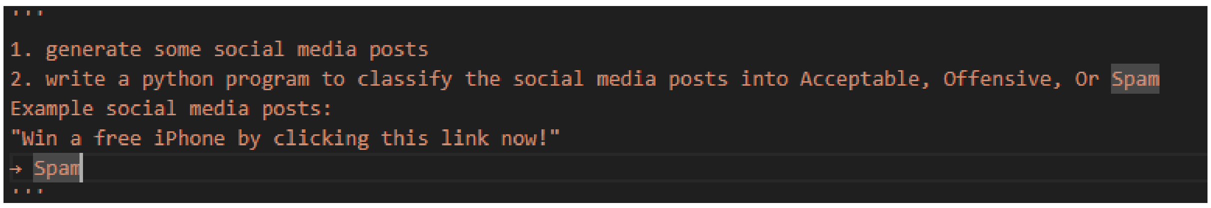
```
1.generate some social media posts
@.write a python program to classfy the social media posts into Acceptable, Offensie, Or spam
'''
```



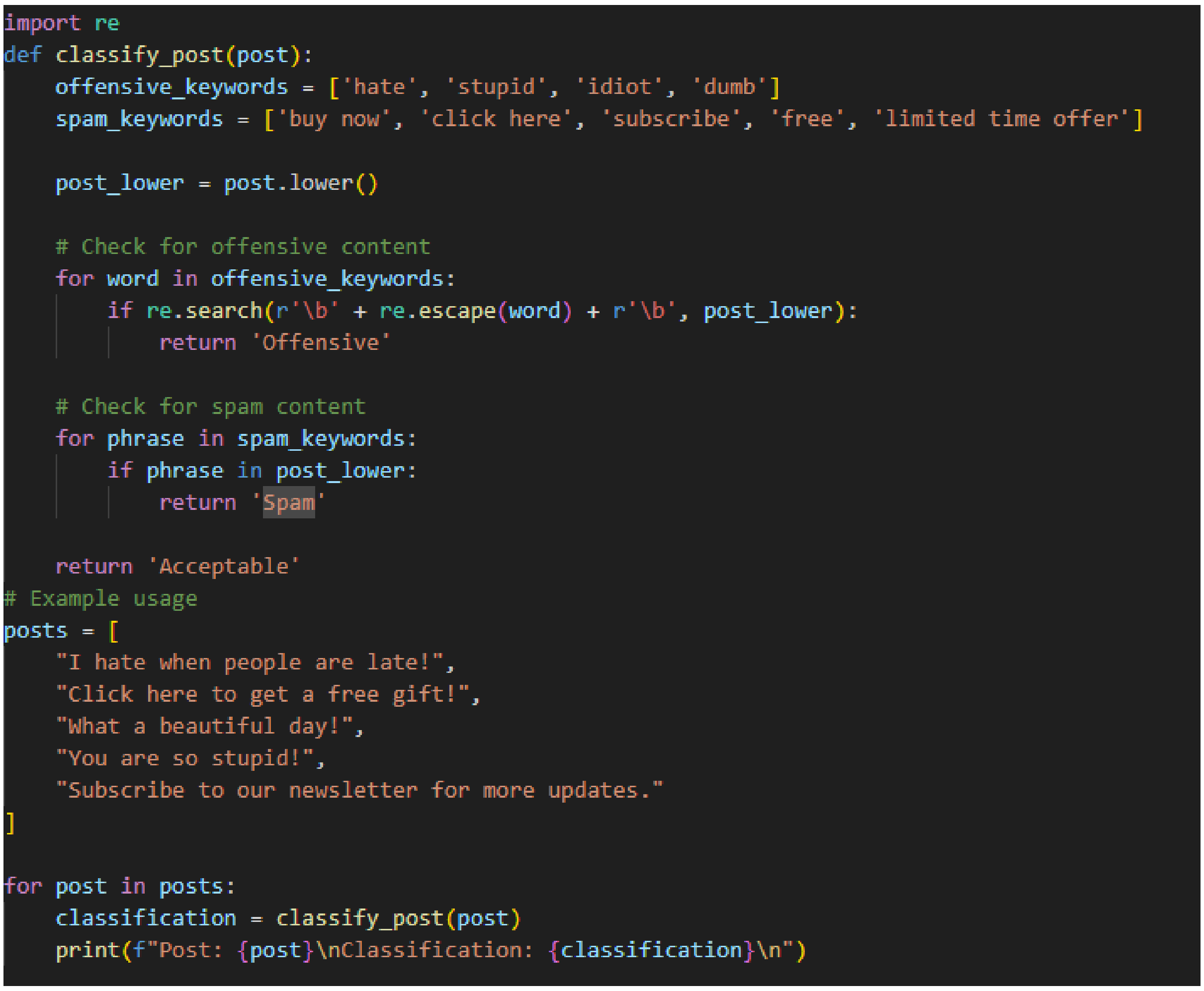
```python
import re
def classify_post(post):
    offensive_keywords = ['hate', 'stupid', 'idiot', 'dumb']
    spam_keywords = ['buy now', 'click here', 'subscribe', 'free', 'limited time offer']

    post_lower = post.lower()

    # Check for offensive content
    for word in offensive_keywords:
        if re.search(r'\b' + re.escape(word) + r'\b', post_lower):
            return 'Offensive'

    # Check for spam content
    for phrase in spam_keywords:
        if phrase in post_lower:
            return 'Spam'

    return 'Acceptable'
# Example usage
posts = [
    "I hate when people are late!",
    "Click here to get a free gift!",
    "What a beautiful day!",
    "You are so stupid!",
    "Subscribe to our newsletter for more updates."
]

for post in posts:
    classification = classify_post(post)
    print(f"Post: {post}\nClassification: {classification}\n")
```



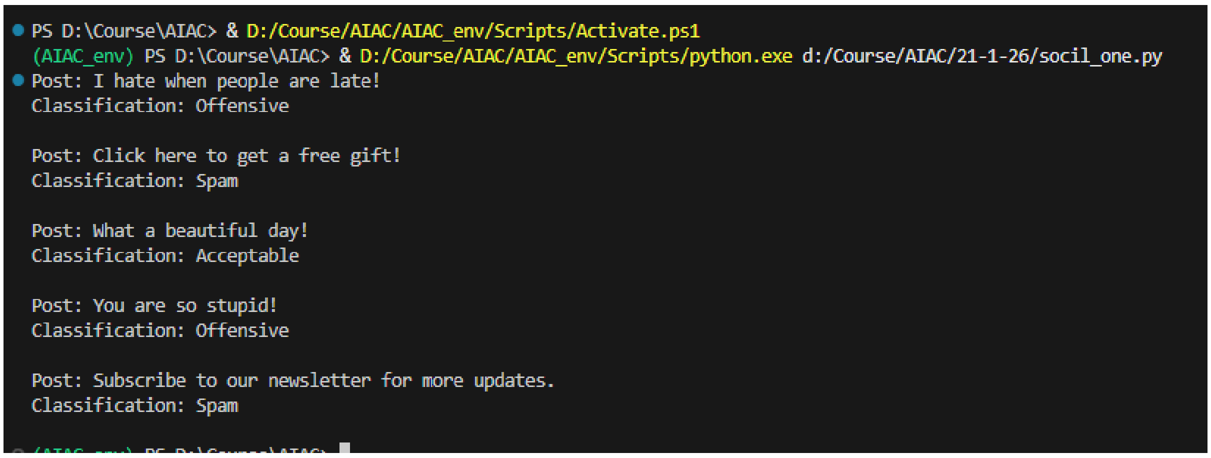
```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/21-1-26/scila_zero.py
Post: I hate when people are late!
Classification: Offensive

Post: Click here to get a free gift!
Classification: Spam

Post: What a beautiful day!
Classification: Acceptable

Post: You are so stupid!
Classification: Offensive

Post: Subscribe to our newsletter for more updates.
Classification: Spam
```



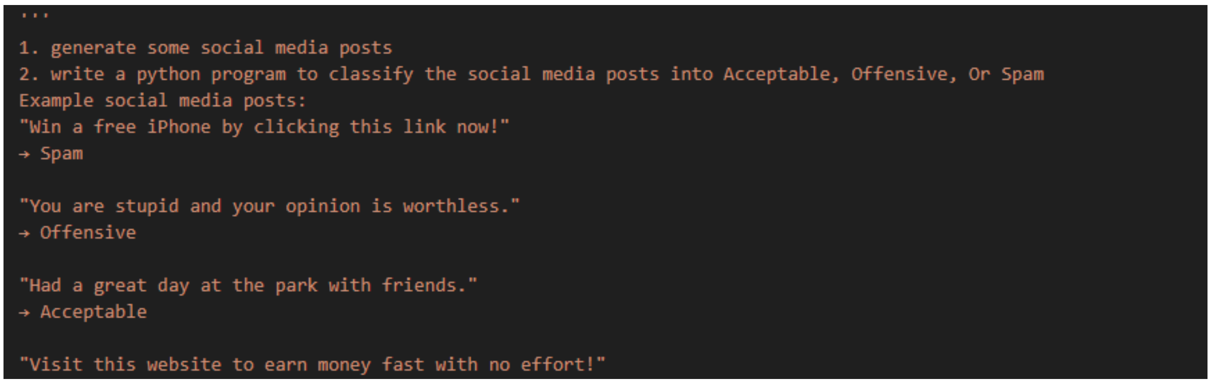
```
'''
1. generate some social media posts
2. write a python program to classify the social media posts into Acceptable, Offensive, Or Spam
Example social media posts:
"Win a free iPhone by clicking this link now!"
→ Spam
'''
```



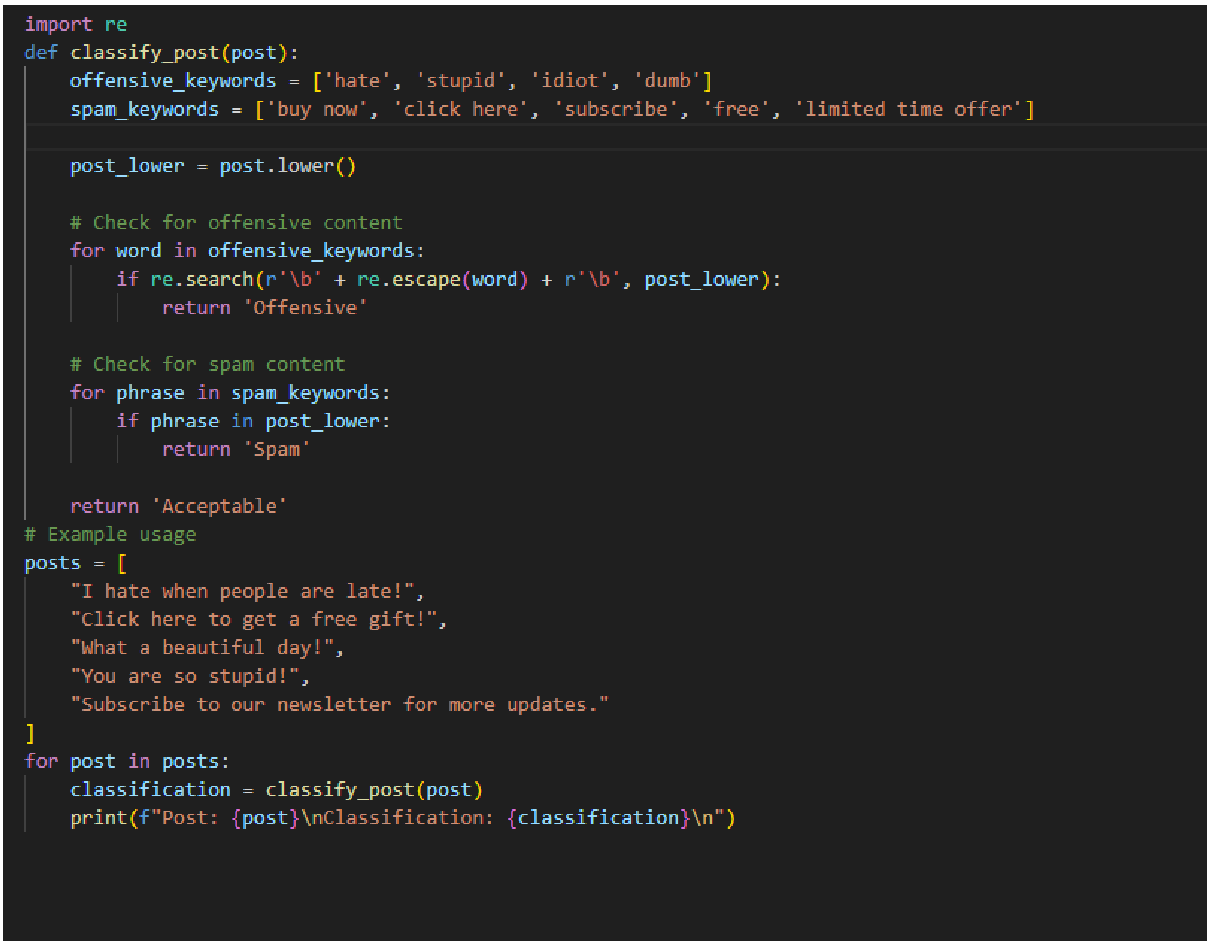
```python
import re
def classify_post(post):
    offensive_keywords = ['hate', 'stupid', 'idiot', 'dumb']
    spam_keywords = ['buy now', 'click here', 'subscribe', 'free', 'limited time offer']

    post_lower = post.lower()

    # Check for offensive content
    for word in offensive_keywords:
        if re.search(r'\b' + re.escape(word) + r'\b', post_lower):
            return 'Offensive'

    # Check for spam content
    for phrase in spam_keywords:
        if phrase in post_lower:
            return 'Spam'

    return 'Acceptable'
# Example usage
posts = [
    "I hate when people are late!",
    "Click here to get a free gift!",
    "What a beautiful day!",
    "You are so stupid!",
    "Subscribe to our newsletter for more updates."
]

for post in posts:
    classification = classify_post(post)
    print(f"Post: {post}\nClassification: {classification}\n")
```



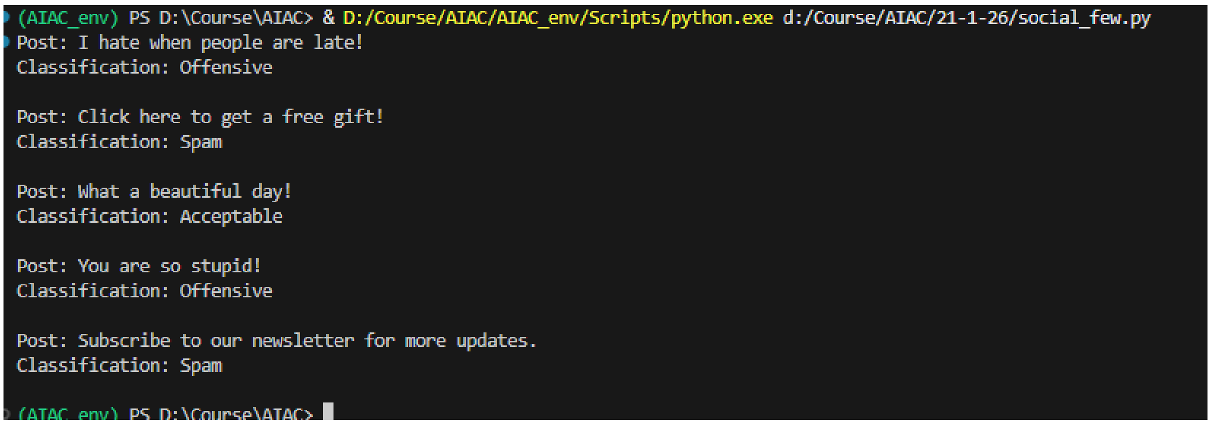
```
PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/Activate.ps1
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/21-1-26/socil_one.py
Post: I hate when people are late!
Classification: Offensive

Post: Click here to get a free gift!
Classification: Spam

Post: What a beautiful day!
Classification: Acceptable

Post: You are so stupid!
Classification: Offensive

Post: Subscribe to our newsletter for more updates.
Classification: Spam
```



```
'''
1. generate some social media posts
2. write a python program to classify the social media posts into Acceptable, Offensive, Or Spam
Example social media posts:
"Win a free iPhone by clicking this link now!"
→ Spam

"You are stupid and your opinion is worthless."
→ Offensive

"Had a great day at the park with friends."
→ Acceptable

"Visit this website to earn money fast with no effort!"
```



```python
import re
def classify_post(post):
    offensive_keywords = ['hate', 'stupid', 'idiot', 'dumb']
    spam_keywords = ['buy now', 'click here', 'subscribe', 'free', 'limited time offer']

    post_lower = post.lower()

    # Check for offensive content
    for word in offensive_keywords:
        if re.search(r'\b' + re.escape(word) + r'\b', post_lower):
            return 'Offensive'

    # Check for spam content
    for phrase in spam_keywords:
        if phrase in post_lower:
            return 'Spam'

    return 'Acceptable'
# Example usage
posts = [
    "I hate when people are late!",
    "Click here to get a free gift!",
    "What a beautiful day!",
    "You are so stupid!",
    "Subscribe to our newsletter for more updates."
]
for post in posts:
    classification = classify_post(post)
    print(f"Post: {post}\nClassification: {classification}\n")
```



```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/21-1-26/social_few.py
Post: I hate when people are late!
Classification: Offensive

Post: Click here to get a free gift!
Classification: Spam

Post: What a beautiful day!
Classification: Acceptable

Post: You are so stupid!
Classification: Offensive

Post: Subscribe to our newsletter for more updates.
Classification: Spam

(AIAC_env) PS D:\Course\AIAC>
```