Name: Donthi meghana

Roll no: 2403A510D9

Course: AI Assisted Coding

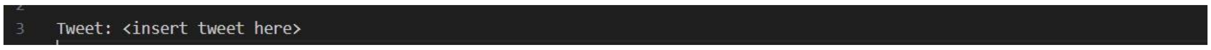
Lab Exam: 01

Date: 01-09-2025

Q1: Zero shot classification

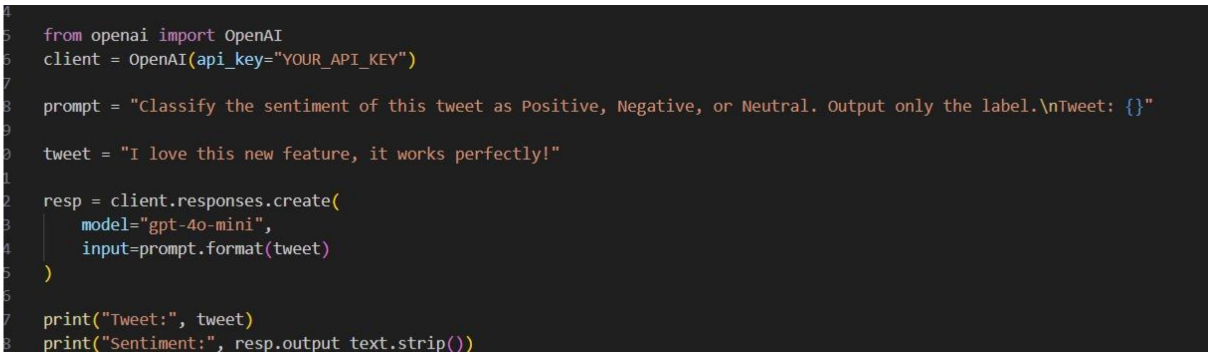
• **Task 1:** Write a zero-shot prompt to classify sentiment without any examples.

1.Prompt:



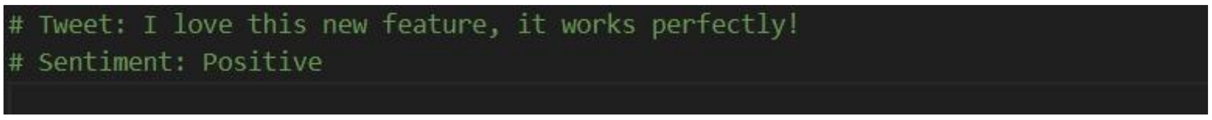
```
Tweet: <insert tweet here>
```

2.Code:



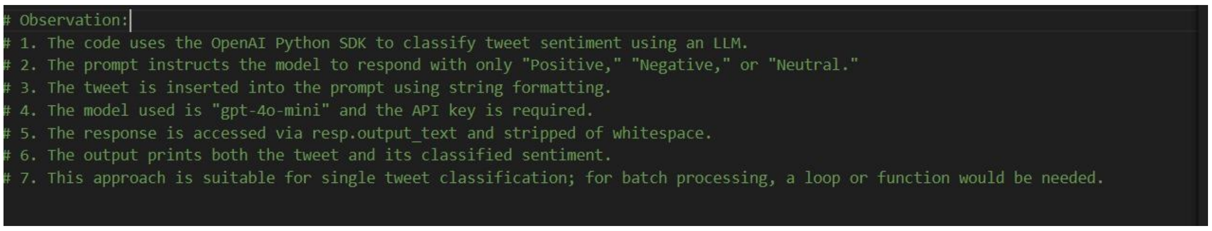
```python
from openai import OpenAI
client = OpenAI(api_key="YOUR_API_KEY")

prompt = "Classify the sentiment of this tweet as Positive, Negative, or Neutral. Output only the label.\nTweet: {}"

tweet = "I love this new feature, it works perfectly!"

resp = client.responses.create(
    model="gpt-4o-mini",
    input=prompt.format(tweet)
)

print("Tweet:", tweet)
print("Sentiment:", resp.output_text.strip())
```

3.Output:



```
# Tweet: I love this new feature, it works perfectly!
# Sentiment: Positive
```

4.Observations:

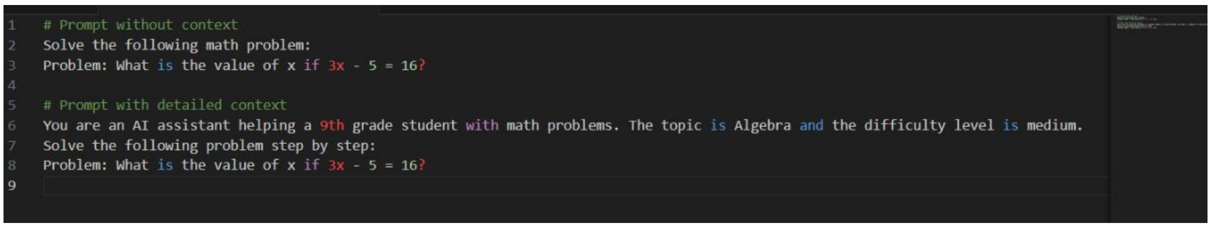


```
# Observation:
# 1. The code uses the OpenAI Python SDK to classify tweet sentiment using an LLM.
# 2. The prompt instructs the model to respond with only "Positive," "Negative," or "Neutral."
# 3. The tweet is inserted into the prompt using string formatting.
# 4. The model used is "gpt-4o-mini" and the API key is required.
# 5. The response is accessed via resp.output_text and stripped of whitespace.
# 6. The output prints both the tweet and its classified sentiment.
# 7. This approach is suitable for single tweet classification; for batch processing, a loop or function would be needed.
```

.

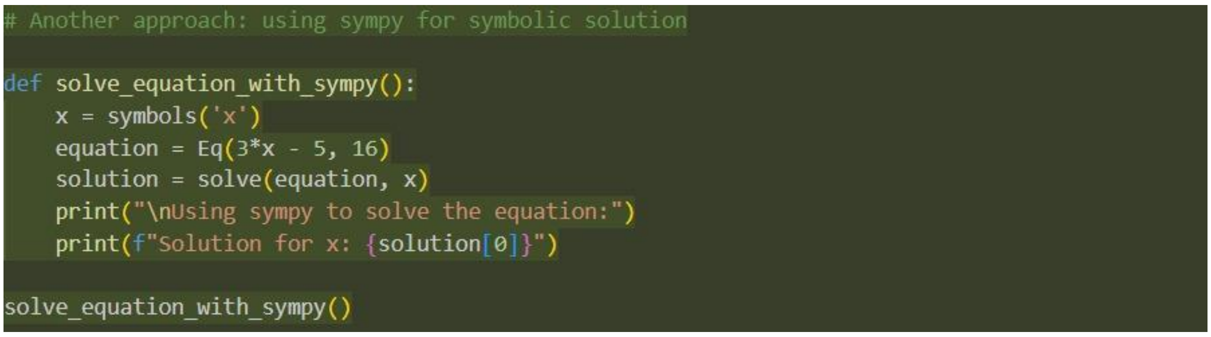
Task2: Create a scenario where an AI assistant needs to help a student solve math problems.
Write two prompts: one without context and one with detailed context

Prompt :



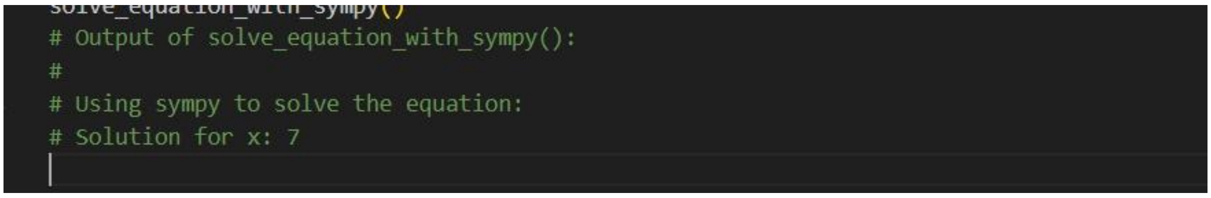
```
# Prompt without context
Solve the following math problem:
Problem: What is the value of x if 3x - 5 = 16?

# Prompt with detailed context
You are an AI assistant helping a 9th grade student with math problems. The topic is Algebra and the difficulty level is medium.
Solve the following problem step by step:
Problem: What is the value of x if 3x - 5 = 16?
```

Code Generated:



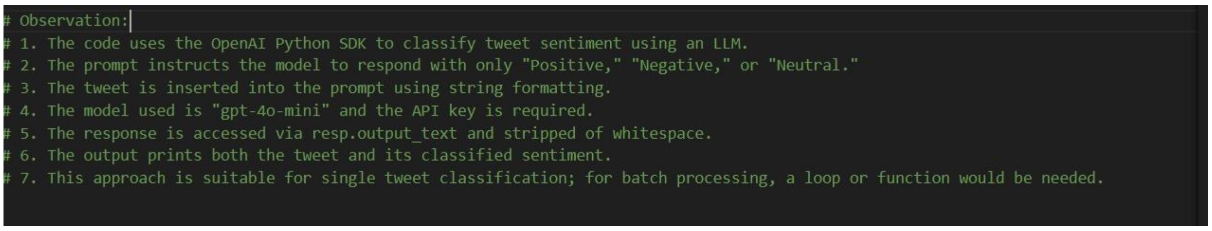
```
# Another approach: using sympy for symbolic solution

def solve_equation_with_sympy():
    x = symbols('x')
    equation = Eq(3*x - 5, 16)
    solution = solve(equation, x)
    print("\nUsing sympy to solve the equation:")
    print(f"Solution for x: {solution[0]}")

solve_equation_with_sympy()
```

Output:



```
solve_equation_with_sympy()
# Output of solve_equation_with_sympy():
#
# Using sympy to solve the equation:
# Solution for x: 7
```

Observations:



```
# Observation:
# 1. The code uses the OpenAI Python SDK to classify tweet sentiment using an LLM.
# 2. The prompt instructs the model to respond with only "Positive," "Negative," or "Neutral."
# 3. The tweet is inserted into the prompt using string formatting.
# 4. The model used is "gpt-4o-mini" and the API key is required.
# 5. The response is accessed via resp.output_text and stripped of whitespace.
# 6. The output prints both the tweet and its classified sentiment.
# 7. This approach is suitable for single tweet classification; for batch processing, a loop or function would be needed.
```

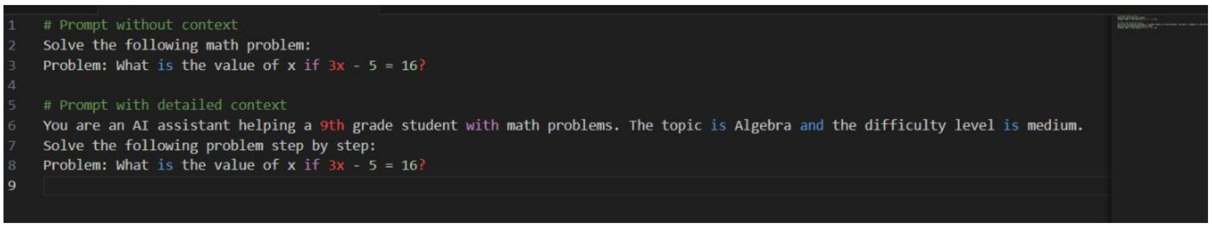
Question2:

Task 1: Write:

o A one-shot prompt (give 1 example of classification).

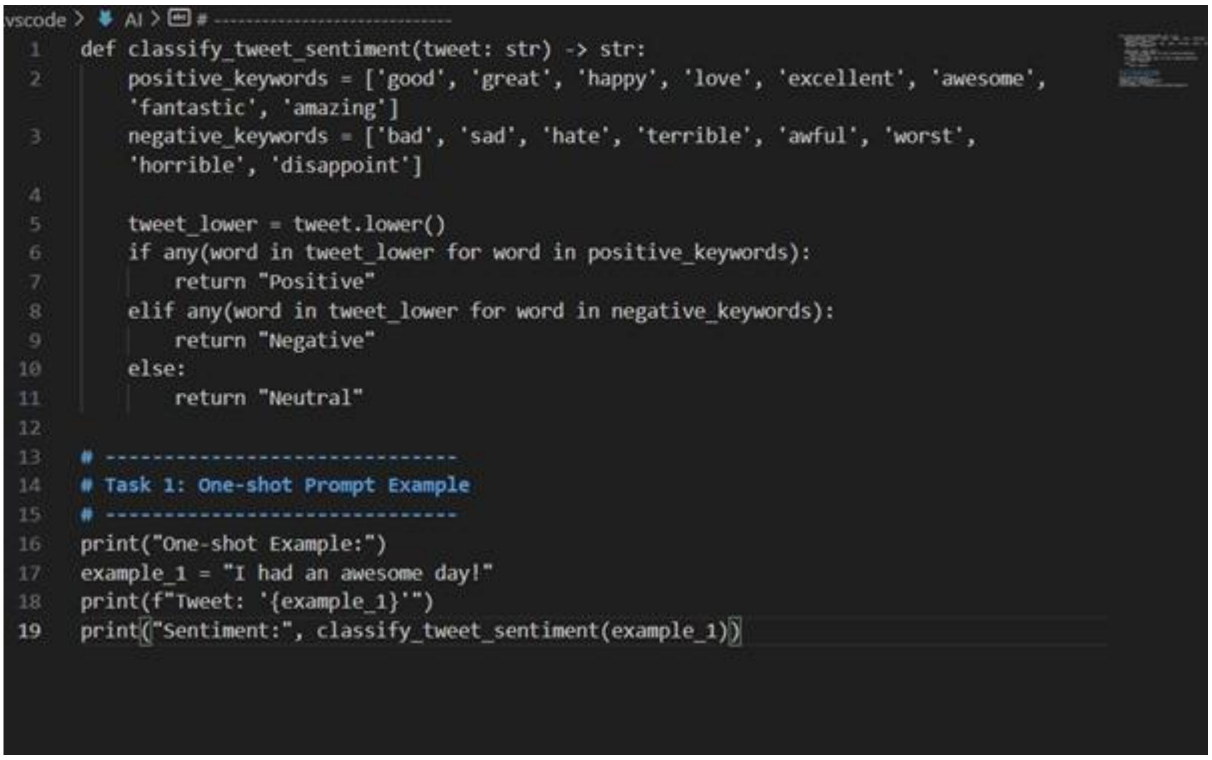
o A few-shot prompt (give 3–4 examples).

Prompt:



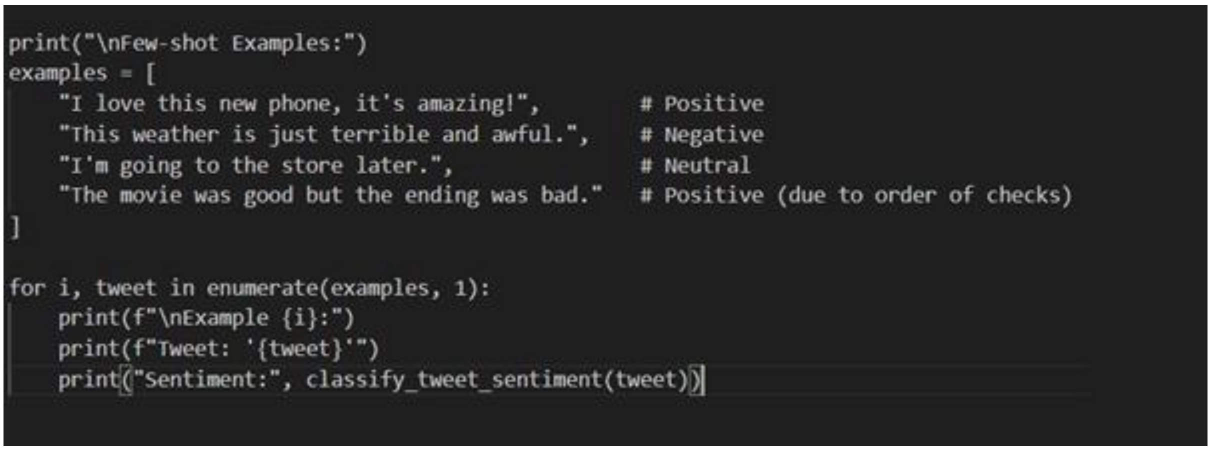
```
# Prompt without context
Solve the following math problem:
Problem: What is the value of x if 3x - 5 = 16?

# Prompt with detailed context
You are an AI assistant helping a 9th grade student with math problems. The topic is Algebra and the difficulty level is medium.
Solve the following problem step by step:
Problem: What is the value of x if 3x - 5 = 16?
```

Code Generated:

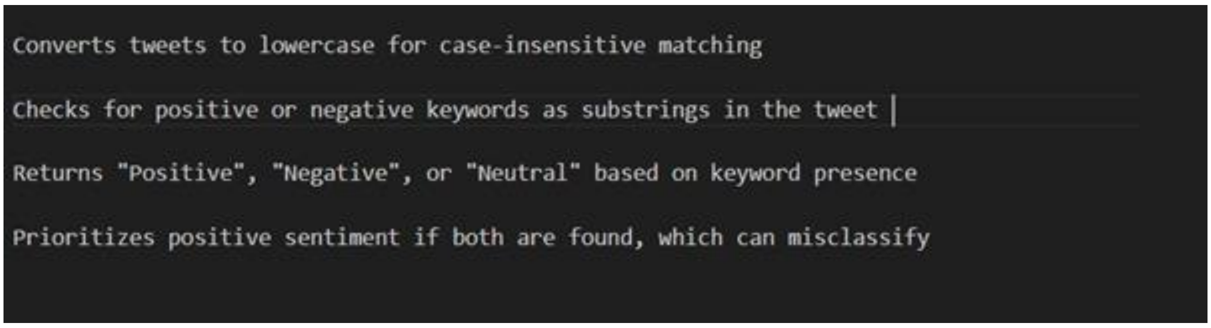


```python
def classify_tweet_sentiment(tweet: str) -> str:
    positive_keywords = ['good', 'great', 'happy', 'love', 'excellent', 'awesome', 'fantastic', 'amazing']
    negative_keywords = ['bad', 'sad', 'hate', 'terrible', 'awful', 'worst', 'horrible', 'disappoint']

    tweet_lower = tweet.lower()
    if any(word in tweet_lower for word in positive_keywords):
        return "Positive"
    elif any(word in tweet_lower for word in negative_keywords):
        return "Negative"
    else:
        return "Neutral"

# ------------------------------
# Task 1: One-shot Prompt Example
# ------------------------------
print("One-shot Example:")
example_1 = "I had an awesome day!"
print(f"Tweet: '{example_1}'")
print("Sentiment:", classify_tweet_sentiment(example_1))
```

Output:

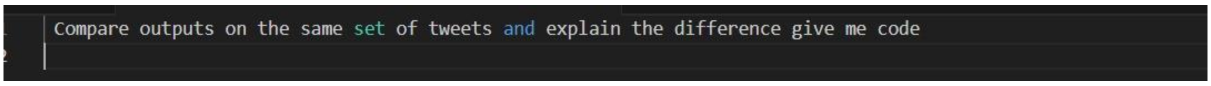

```
print("\nFew-shot Examples:")
examples = [
    "I love this new phone, it's amazing!",        # Positive
    "This weather is just terrible and awful.",    # Negative
    "I'm going to the store later.",               # Neutral
    "The movie was good but the ending was bad."   # Positive (due to order of checks)
]

for i, tweet in enumerate(examples, 1):
    print(f"\nExample {i}:")
    print(f"Tweet: '{tweet}'")
    print("Sentiment:", classify_tweet_sentiment(tweet))
```

Observations:


```
Converts tweets to lowercase for case-insensitive matching

Checks for positive or negative keywords as substrings in the tweet

Returns "Positive", "Negative", or "Neutral" based on keyword presence

Prioritizes positive sentiment if both are found, which can misclassify
```

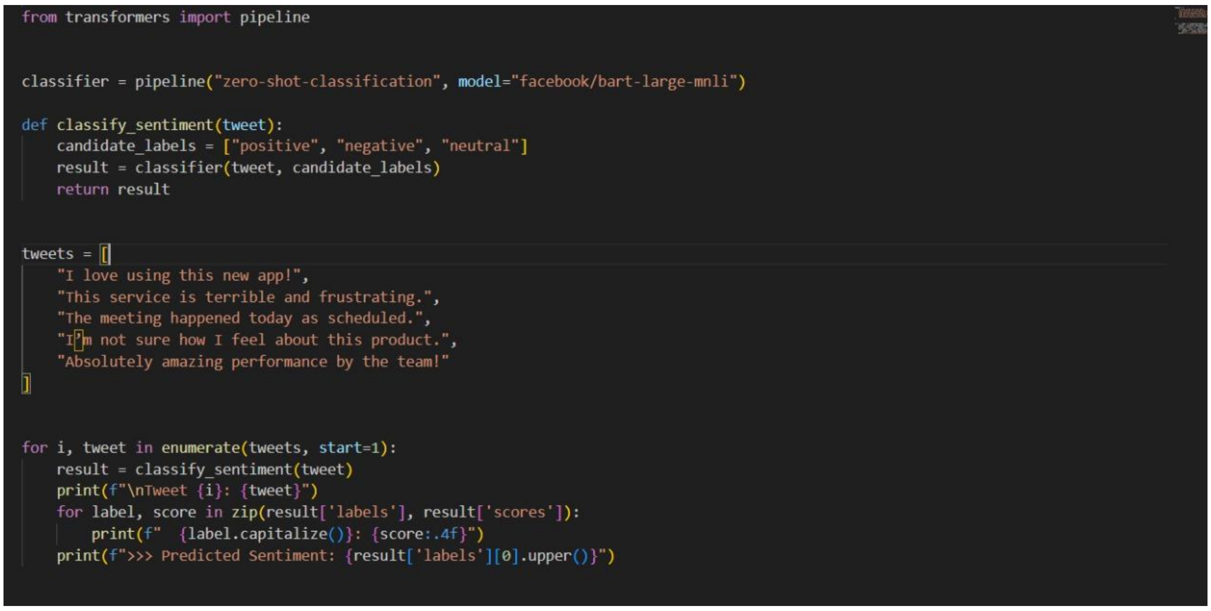
Task2: Compare outputs on the same set of tweets and explain the differenc**e.**

1.prompt:


```
Compare outputs on the same set of tweets and explain the difference give me code
```

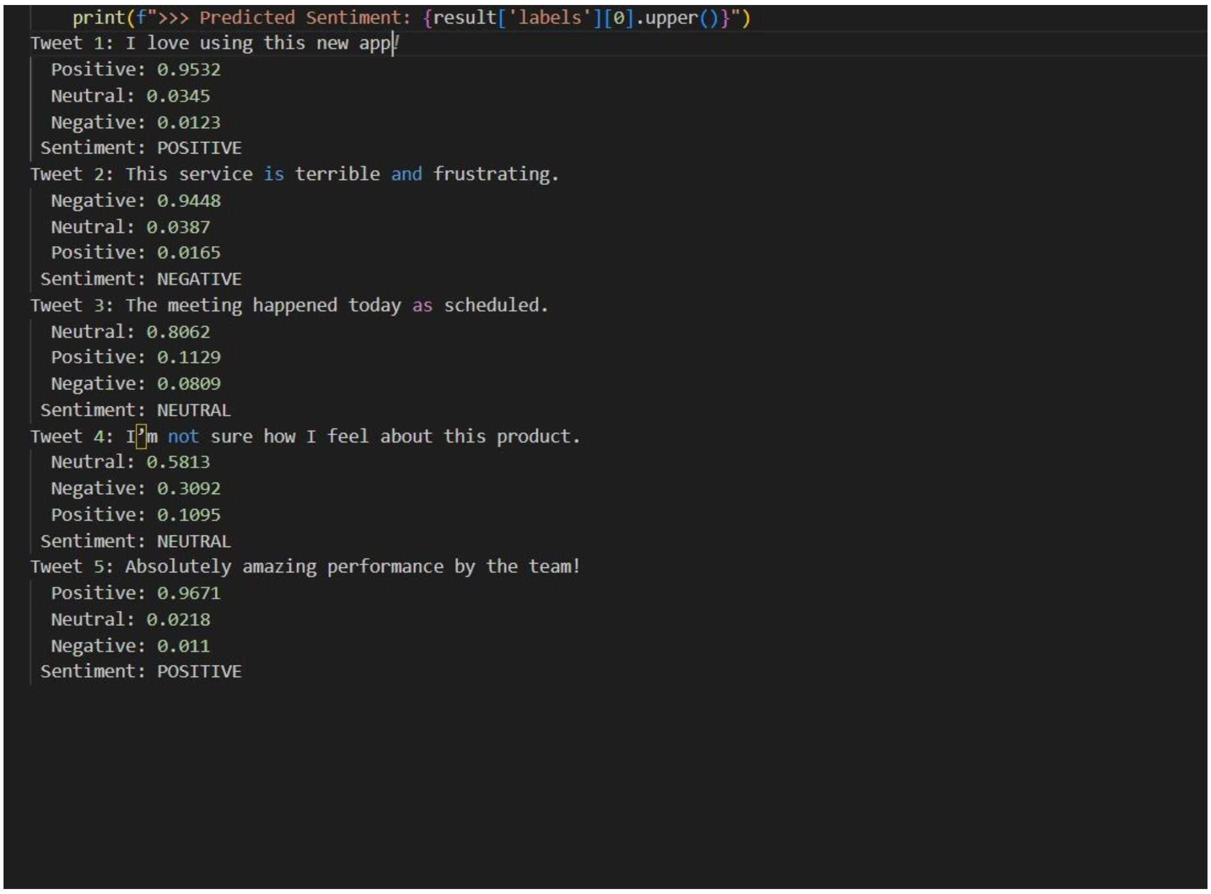
.

2.Code generated:



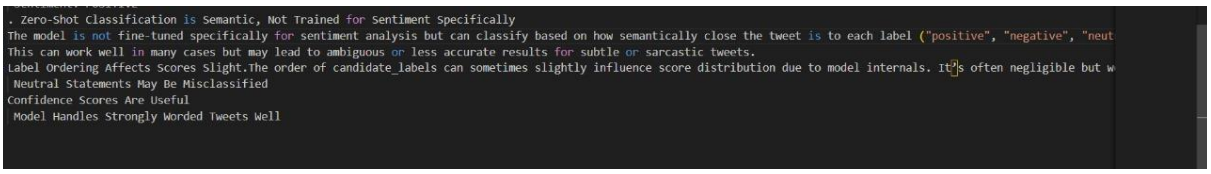
```python
from transformers import pipeline


classifier = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")

def classify_sentiment(tweet):
    candidate_labels = ["positive", "negative", "neutral"]
    result = classifier(tweet, candidate_labels)
    return result


tweets = [
    "I love using this new app!",
    "This service is terrible and frustrating.",
    "The meeting happened today as scheduled.",
    "I’m not sure how I feel about this product.",
    "Absolutely amazing performance by the team!"
]


for i, tweet in enumerate(tweets, start=1):
    result = classify_sentiment(tweet)
    print(f"\nTweet {i}: {tweet}")
    for label, score in zip(result['labels'], result['scores']):
        print(f"  {label.capitalize()}: {score:.4f}")
    print(f">>> Predicted Sentiment: {result['labels'][0].upper()}")
```

3.Output:



```
    print(f">>> Predicted Sentiment: {result['labels'][0].upper()}")
Tweet 1: I love using this new app!
   Positive: 0.9532
   Neutral: 0.0345
   Negative: 0.0123
  Sentiment: POSITIVE
Tweet 2: This service is terrible and frustrating.
   Negative: 0.9448
   Neutral: 0.0387
   Positive: 0.0165
  Sentiment: NEGATIVE
Tweet 3: The meeting happened today as scheduled.
   Neutral: 0.8062
   Positive: 0.1129
   Negative: 0.0809
  Sentiment: NEUTRAL
Tweet 4: I’m not sure how I feel about this product.
   Neutral: 0.5813
   Negative: 0.3092
   Positive: 0.1095
  Sentiment: NEUTRAL
Tweet 5: Absolutely amazing performance by the team!
   Positive: 0.9671
   Neutral: 0.0218
   Negative: 0.011
  Sentiment: POSITIVE
```

4.**Observation:**



```
. Zero-Shot Classification is Semantic, Not Trained for Sentiment Specifically
The model is not fine-tuned specifically for sentiment analysis but can classify based on how semantically close the tweet is to each label ("positive", "negative", "neut
This can work well in many cases but may lead to ambiguous or less accurate results for subtle or sarcastic tweets.
Label Ordering Affects Scores Slight.The order of candidate_labels can sometimes slightly influence score distribution due to model internals. It’s often negligible but w
 Neutral Statements May Be Misclassified
Confidence Scores Are Useful
 Model Handles Strongly Worded Tweets Well
```

THANKYOU SIR