Assignment 2: Large Language Models for Text Classification

CS 410/510 Large Language Models Fall 2024

Greg Witt

Load Dataset

Install Dependencies

In [ ]:
```
!pip install datasets
```

Split DataSets

In [1]:
```
from datasets import load_dataset
import random

# load the training set of tweets
ds_train = load_dataset("cardiffnlp/tweet_sentiment_multilingual", "english"
# the dataset labels {0 : negative, 1: neutral, 2: positive }
print(f"""tweet_sentiment_multilingual training dataset:
    ----------------------------------------
        {ds_train}
    """)
random_tweet_index = random.randint(0,1839)
print(f"""
    Random Tweet:
    {ds_train['text'][random_tweet_index]}

    Label:
    {ds_train['label'][random_tweet_index] }
""")
```

```
tweet_sentiment_multilingual training dataset:
    ----------------------------------------
        Dataset({
    features: ['text', 'label'],
    num_rows: 1839
})


    Random Tweet:
    Friends and Seinfeld (I didn't get cable until 3rd grade so I watched th
e shows on satellite)

    Label:
    1
```

In [2]:
```
# load the training set of tweets
ds_validation = load_dataset("cardiffnlp/tweet_sentiment_multilingual", "eng
# the dataset labels {0 : negative, 1: neutral, 2: positive }
print(f"""tweet_sentiment_multilingual validation set:
    ----------------------------------------
        {ds_validation}
    """)
random_tweet_index = random.randint(0,324)
print(f"""
    Random Tweet:
    {ds_validation['text'][random_tweet_index]}

    Label:
    {ds_validation['label'][random_tweet_index] }
""")
```

```
tweet_sentiment_multilingual validation set:
    ----------------------------------------
        Dataset({
    features: ['text', 'label'],
    num_rows: 324
})


    Random Tweet:
    I can\u2019t believe the 3 important people in my life are going to Hawa
ii Friday without me

    Label:
    0
```

Load Model Dependencies

In [ ]:
```
!pip install transformers torch
```

Experiments 1: Zero-Shot Inference

Create Prompt Iteration Function

Approach: Parse Single Token Response from Zero-Shot Inference

This approach prompts the model in question for a sentiment classification of a tweet, the **AutoModelForCausalLM's** `mode.generate()` call is invoked in order to ellicit a response to our provided prompt. the `generate()` method can be provided with several parameters. this approach modifies the **GenerationConfiguration** allowing for certain constraints. for this approach we ensure that the `max_new_tokens=1` and `pad_token_id` is set to ensure there is only one token generated from the model. This token is then used to represent our model's response to the sentiment prompt.

This approach can work but there are risks associated with this method, because these models haven't gone through any **fine-tuning**. Meaning their lack of training hasn't prepared them to for **sentiment classification**, meaning instead of receiving a "0" for neutral, "1" for negative, or "2" for positive there could be other undesirable tokens coming back. To handle this I created a `santize_response()` method that will take the generated tokens and determine if the response can be related to a sentiment classification.

In [404…
```
import re

def santize_response(response_token):
    if re.search(r'\d', response_token):
        return int(response_token)
    else:
        return 0
```

Llama 3.2 1B: Zero Shot Inference

In [4]:
```
from transformers import AutoModelForCausalLM, AutoTokenizer
import torch

llama_3_2_1B = "meta-llama/Llama-3.2-1B"

llama_3_2_1B_tokenizer = AutoTokenizer.from_pretrained(llama_3_2_1B)

llama_3_2_1B_model = AutoModelForCausalLM.from_pretrained(llama_3_2_1B)
```

In [7]:
```
prompt_template = "\"{}\" \n Sentiment (positive (2), negative (0), neutral
```

In [102…
```
import numpy as np

# hold the ground_truths
llama_3_2_1B_ground_truths = []
# hold the sentiment_predictions
llama_3_2_1B_sentiment_preds = []
```

```
# iterate through the validation set
for tweet, label in zip(ds_validation['text'], ds_validation['label']):
    prompt_tweet = tweet
    gt_label = label
    # combine the prompt
    prompt = prompt_template.format(prompt_tweet)

    # generate the response
    prompt_ids = llama_3_2_1B_tokenizer.encode(prompt, return_tensors="pt")

    outputs = llama_3_2_1B_model.generate(
                        prompt_ids,
                        temperature=0.86,
                        do_sample=True,
                        pad_token_id=llama_3_2_1B_tokenizer.eos_token_id,
                        max_new_tokens=1
                    )

    # get the response tokens from the model
    generated_tokens = outputs[-1]

    generated_response = llama_3_2_1B_tokenizer.decode(generated_tokens, ski

    # pass the generated_response to a sanitizer that determines if the resp
    sentiment_resp = santize_response(generated_response[-1])

    # Responses being compared to the Ground_Truth Labels
    # print(f"Generated Response: {sentiment_resp}")
    # print(f"Ground Truth: {gt_label}")

    llama_3_2_1B_ground_truths.append(gt_label)
    llama_3_2_1B_sentiment_preds.append(sentiment_resp)

llama_3_2_1B_ground_truths = np.asarray(llama_3_2_1B_ground_truths)
llama_3_2_1B_sentiment_preds = np.asarray(llama_3_2_1B_sentiment_preds)
```

Build Classification Report

Import Python Libraries

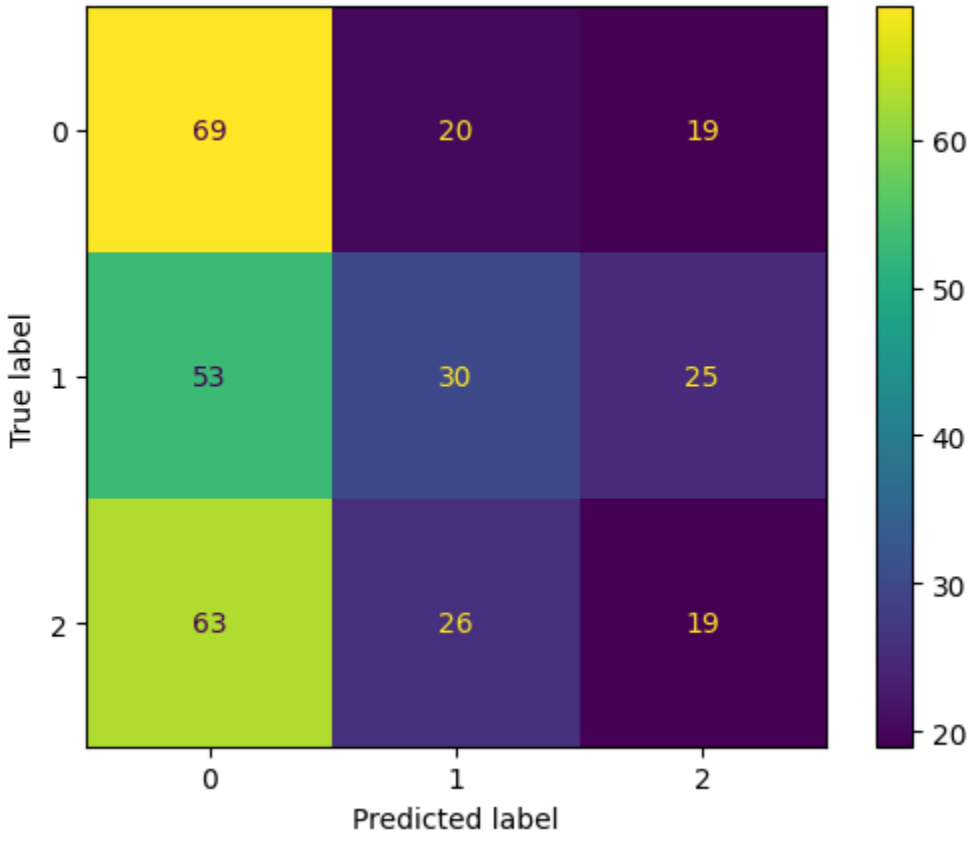
In [ ]:
```
pip install scikit-learn matplotlib seaborn
```

In [103…
```
from sklearn.metrics import classification_report

zero_shot_llama_3_2_1B_model_results = classification_report(llama_3_2_1B_gr
                                                             llama_3_2_1B_se
                                                             output_dict=Tru

print(classification_report(llama_3_2_1B_ground_truths, llama_3_2_1B_sentime
```

	precision	recall	f1-score	support
0	0.31	0.50	0.38	108
1	0.34	0.25	0.29	108
2	0.39	0.26	0.31	108
accuracy			0.34	324
macro avg	0.35	0.34	0.33	324
weighted avg	0.35	0.34	0.33	324

Build Confusion Matrix

In [93]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay


cm = confusion_matrix(llama_3_2_1B_ground_truths, llama_3_2_1B_sentiment_pre

disp = ConfusionMatrixDisplay(confusion_matrix=cm)

disp.plot()
plt.show()
```



Llama 3.2 3B: Zero Shot Inference

In [37]:
```
from transformers import AutoModelForCausalLM, AutoTokenizer
```

```
import torch

llama_3_2_3B = "meta-llama/Llama-3.2-3B"

llama_3_2_3B_tokenizer = AutoTokenizer.from_pretrained(llama_3_2_3B)

llama_3_2_3B_model = AutoModelForCausalLM.from_pretrained(llama_3_2_3B)
```

```
Loading checkpoint shards:   0%|          | 0/2 [00:00<?, ?it/s]
```

In [12]:
```
prompt_template = "\"{}\" \n Sentiment (positive (2), negative (0), neutral
```

In [52]:
```
import re

def santize_response(response_token):
    if re.search(r'\d', response_token):
       if int(response_token) > 2:
           return 0
       else:
           return int(response_token)

    else:
        return 0
```

In [107…
```
import numpy as np

# hold the ground_truths
llama_3_2_3B_ground_truths = []
# hold the sentiment_predictions
llama_3_2_3B_sentiment_preds = []

# iterate through the validation set
for tweet, label in zip(ds_validation['text'], ds_validation['label']):
    prompt_tweet = tweet
    gt_label = label
    # combine the prompt
    prompt = prompt_template.format(prompt_tweet)

    # generate the response
    prompt_ids = llama_3_2_3B_tokenizer.encode(prompt, return_tensors="pt")

    outputs = llama_3_2_3B_model.generate(
                        prompt_ids,
                        do_sample=True,
                        temperature=0.65,
                        pad_token_id=llama_3_2_3B_tokenizer.eos_token_id,
                        max_new_tokens=1
                    )

    # get the response tokens from the model
    generated_tokens = outputs[-1]

    generated_response = llama_3_2_1B_tokenizer.decode(generated_tokens, ski

    # pass the generated_response to a sanitizer that determines if the resp
    sentiment_resp = santize_response(generated_response[-1])
```

```
    # Responses being compared to the Ground_Truth Labels
    # print(f"Generated Response: {sentiment_resp}")
    # print(f"Ground Truth: {gt_label}")

    llama_3_2_3B_ground_truths.append(gt_label)
    llama_3_2_3B_sentiment_preds.append(sentiment_resp)

llama_3_2_3B_ground_truths = np.asarray(llama_3_2_3B_ground_truths)
llama_3_2_3B_sentiment_preds = np.asarray(llama_3_2_3B_sentiment_preds)
```

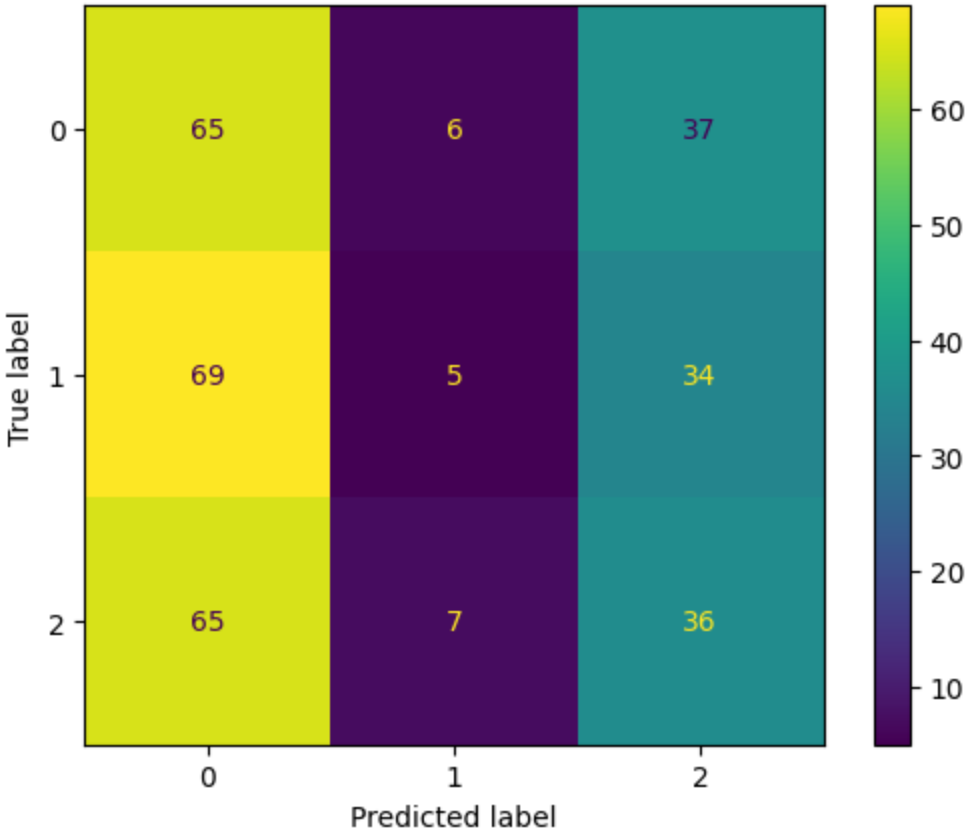
In [108…

```
from sklearn.metrics import classification_report

zero_shot_llama_3_2_3B_results = classification_report(llama_3_2_3B_ground_t

print(classification_report(llama_3_2_3B_ground_truths, llama_3_2_3B_sentime
```

```
              precision    recall  f1-score   support

           0       0.33      0.60      0.42       108
           1       0.28      0.05      0.08       108
           2       0.34      0.33      0.33       108

    accuracy                           0.33       324
   macro avg       0.31      0.33      0.28       324
weighted avg       0.31      0.33      0.28       324
```

In [109…

```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay


cm = confusion_matrix(llama_3_2_3B_ground_truths, llama_3_2_3B_sentiment_pre

disp = ConfusionMatrixDisplay(confusion_matrix=cm)

disp.plot()
plt.show()
```



Phi-3.5-mini-instruct: Zero Shot Inference

Phi 3.5 Instruct

In [56]:
```
from transformers import AutoModelForCausalLM, AutoTokenizer
import torch

# model name
phi_model = "microsoft/Phi-3.5-mini-instruct"

phi_3_5_tokenizer = AutoTokenizer.from_pretrained(phi_model)

phi_3_5_model = AutoModelForCausalLM.from_pretrained(phi_model)
```

```
Loading checkpoint shards:   0%|          | 0/2 [00:00<?, ?it/s]
```

In [28]:
```
prompt_template = "\"{}\" \n Sentiment (positive (2), negative (0), neutral
```

In [57]:
```
import re

def santize_response(response_token):
    if re.search(r'\d', response_token):
       if int(response_token) > 2:
           return 0
       else:
           return int(response_token)

    else:
```

```
        return 0
```

In [110…

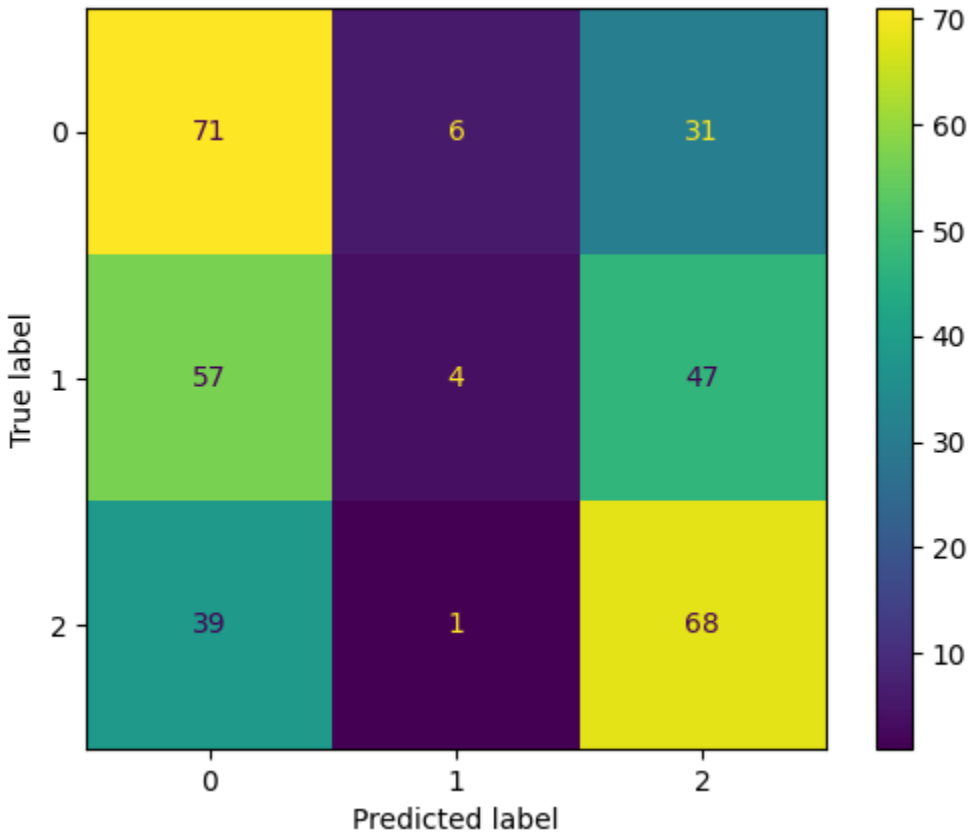
```
import numpy as np

# hold the ground_truths
phi_3_5_ground_truths = []
# hold the sentiment_predictions
phi_3_5_sentiment_preds = []

# iterate through the validation set
for tweet, label in zip(ds_validation['text'], ds_validation['label']):
    prompt_tweet = tweet
    gt_label = label
    # combine the prompt
    prompt = prompt_template.format(prompt_tweet)

    # generate the response
    prompt_ids = phi_3_5_tokenizer.encode(prompt, return_tensors="pt")

    outputs = phi_3_5_model.generate(
                        prompt_ids,
                        temperature=0.23,
                        do_sample=True,
                        pad_token_id=phi_3_5_tokenizer.eos_token_id,
                        max_new_tokens=1
                    )

    # get the response tokens from the model
    generated_tokens = outputs[-1]

    generated_response = phi_3_5_tokenizer.decode(generated_tokens, skip_spe

    # pass the generated_response to a sanitizer that determines if the resp
    sentiment_resp = santize_response(generated_response[-1])

    # Responses being compared to the Ground_Truth Labels
    # print(f"Generated Response: {sentiment_resp}")
    # print(f"Ground Truth: {gt_label}")

    phi_3_5_ground_truths.append(gt_label)
    phi_3_5_sentiment_preds.append(sentiment_resp)

phi_3_5_ground_truths = np.asarray(phi_3_5_ground_truths)
phi_3_5_sentiment_preds = np.asarray(phi_3_5_sentiment_preds)
```

In [113…

```
from sklearn.metrics import classification_report

zero_shot_phi_3_5_results = classification_report(phi_3_5_ground_truths, phi

print(classification_report(phi_3_5_ground_truths, phi_3_5_sentiment_preds))
```

	precision	recall	f1-score	support
0	0.46	0.76	0.57	108
1	0.83	0.05	0.09	108
2	0.50	0.65	0.56	108
accuracy			0.48	324
macro avg	0.60	0.48	0.41	324
weighted avg	0.60	0.48	0.41	324

In [84]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay


cm = confusion_matrix(phi_3_5_ground_truths, phi_3_5_sentiment_preds)

disp = ConfusionMatrixDisplay(confusion_matrix=cm)

disp.plot()
plt.show()
```



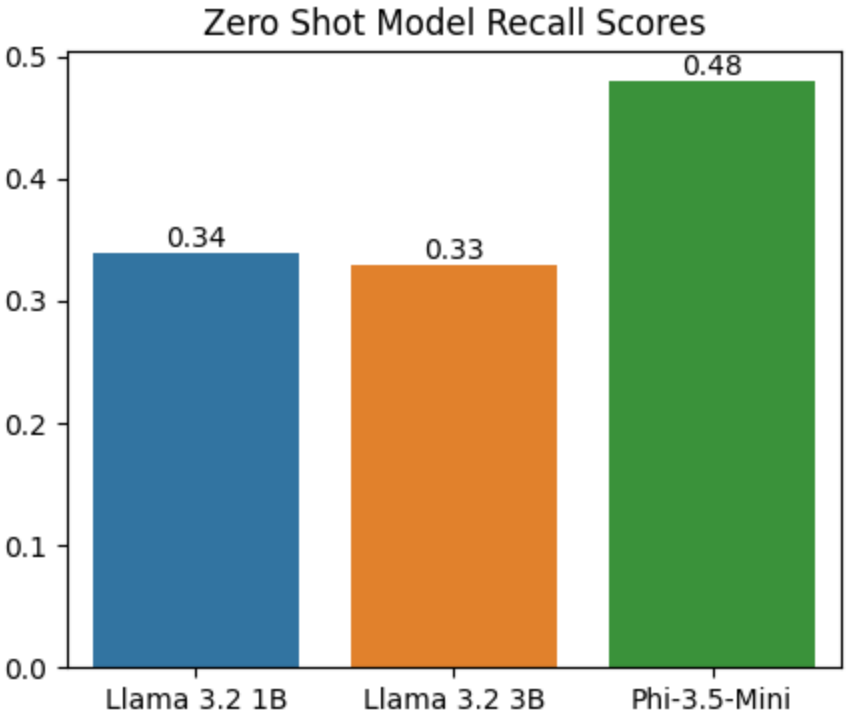
Experiment 1: Zero-Shot Results

Zero-Shot Model Recall Scores

In [403…
```
import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.color_palette("Set2")

models = ['Llama 3.2 1B','Llama 3.2 3B','Phi-3.5-Mini']

llama_32_1b = round(zero_shot_llama_3_2_1B_model_results['macro avg']['recal

llama_32_3b = round(zero_shot_llama_3_2_3B_results['macro avg']['recall'], 2

phi_35_mini = round(zero_shot_phi_3_5_results['macro avg']['recall'], 2)

precision_scores = [llama_32_1b, llama_32_3b, phi_35_mini]

plt.figure(figsize=(5,4), dpi=100)
plt.yticks()
ax = sns.barplot(x=models, y=precision_scores, hue=models, legend="auto" )
# add labels
ax.bar_label(ax.containers[0], fontsize=10)
ax.bar_label(ax.containers[1], fontsize=10)
ax.bar_label(ax.containers[2], fontsize=10)
plt.title('Zero Shot Model Recall Scores')
```

Out[403… Text(0.5, 1.0, 'Zero Shot Model Recall Scores')



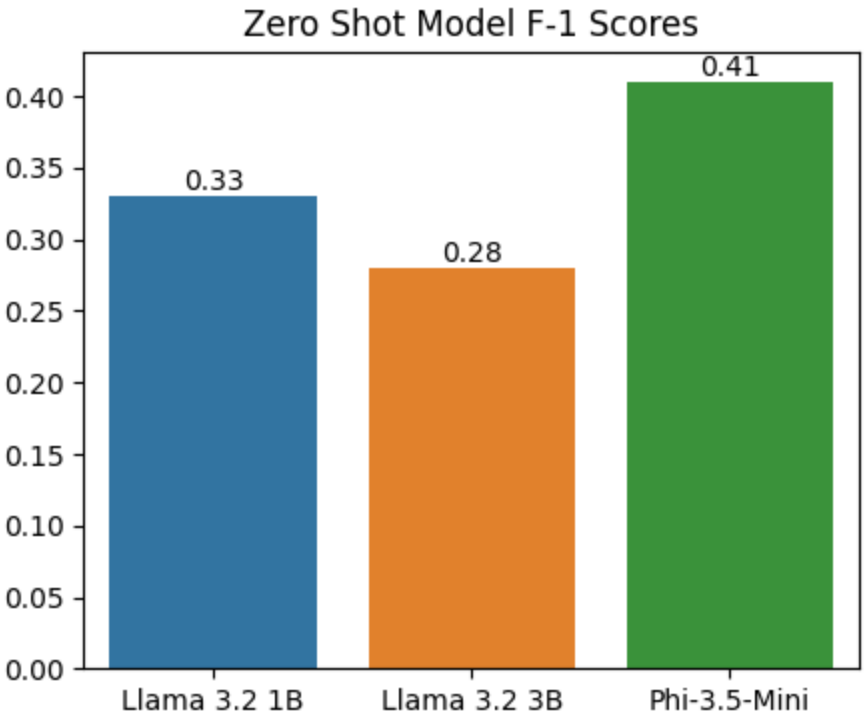
Zero-Shot Models F1-Scores

In [402…

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.color_palette("Set2")
```

```
models = ['Llama 3.2 1B','Llama 3.2 3B','Phi-3.5-Mini']

llama_32_1b = round(zero_shot_llama_3_2_1B_model_results['macro avg']['f1-sc

llama_32_3b = round(zero_shot_llama_3_2_3B_results['macro avg']['f1-score'],

phi_35_mini = round(zero_shot_phi_3_5_results['macro avg']['f1-score'], 2)

precision_scores = [llama_32_1b, llama_32_3b, phi_35_mini]

plt.figure(figsize=(5,4), dpi=100)
plt.yticks()
ax = sns.barplot(x=models, y=precision_scores, hue=models, legend="auto" )
# add labels
ax.bar_label(ax.containers[0], fontsize=10)
ax.bar_label(ax.containers[1], fontsize=10)
ax.bar_label(ax.containers[2], fontsize=10)
plt.title('Zero Shot Model F-1 Scores')
```

Out[402… Text(0.5, 1.0, 'Zero Shot Model F-1 Scores')



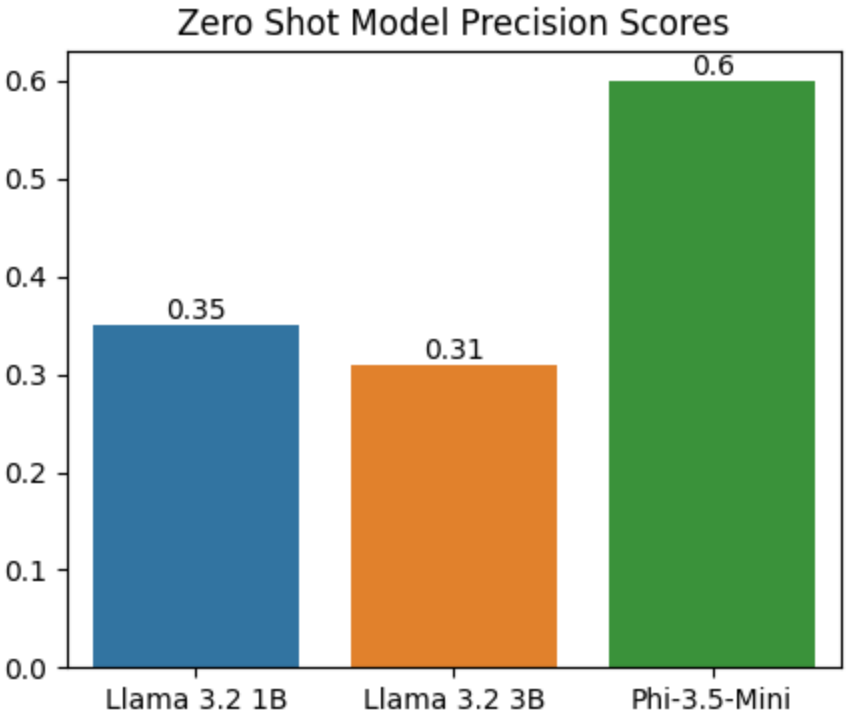
Zero-Shot Model Percisions

In [400…

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.color_palette("Set2")

models = ['Llama 3.2 1B','Llama 3.2 3B','Phi-3.5-Mini']

llama_32_1b = round(zero_shot_llama_3_2_1B_model_results['macro avg']['preci
```

```
llama_32_3b = round(zero_shot_llama_3_2_3B_results['macro avg']['precision']

phi_35_mini = round(zero_shot_phi_3_5_results['macro avg']['precision'], 2)

precision_scores = [llama_32_1b, llama_32_3b, phi_35_mini]

plt.figure(figsize=(5,4), dpi=100)
ax = sns.barplot(x=models, y=precision_scores, hue=models, legend="auto" )
# add labels
ax.bar_label(ax.containers[0], fontsize=10)
ax.bar_label(ax.containers[1], fontsize=10)
ax.bar_label(ax.containers[2], fontsize=10)
plt.title('Zero Shot Model Precision Scores')
```

Out[400… Text(0.5, 1.0, 'Zero Shot Model Precision Scores')



Analysis for Experiment 1: Zero-Shot Approach

the models during this experiement required some hyperparameter adjustment. **Llama** was specifically sensitive to higher ranges and lower ranges of `temperature` often the instinctive decisions to go for a range of `0.1–0.3` for more refined responses weren't possible the models suffered with terrible accuracies and precision. often struggling to decern between `positive` and `negative` but struggling to accurately determine `neutral` tweet sentiment. According to the hugging-face documentation the model temperature is defaulted to a high value of `1.0` this is much to creative for these models. **LLama** models as seen in the previous Assignment 1 do not fair well with prompts of this kind of instructional quality compared to a `instruction` model such as **Phi 3.5**. The results can be seen below. overall the models were actually close in

performance from both **Llama** varieties but, not suprised by the top performing modle for this experiment **Phi 3.5** but the second highest highest for metrics such as **precision**, **recall** and **f1-score Llama 3.2 1B** performed much better than **Llama 3.2 3B** on my system.

Experiment 2: Few-Shot In-Context Learning

Approach: Parse Single Token Response from Segmented Tweets for Balanced Few-Shot Learning

Leveraging the dataset package from hugging-face. I was able to filer each of the tweeets by their label of either `0`, `1`, `2`. and assigning them to their collections. I was able to ensure that each tweet sentiment flavor was divided by their label within the collection of english tweets. I then tested the `random` library as before from when I first pulled them. I used a `random.seed(23)` for reproducability. The random tweets of negative, neutral and positive were confirmed to infact match as expected.

next I build a `generate_few_shot_prompt()` method to take an incoming tweet `validation_tweet` this tweet was appended to a list of `random(0,612)` tweets. this allows for variety but performed twice for a `k` of `2`. Meaning two positive, two negative and finally two neutral tweets with the retrived label. these are `extend()` fed into a list and then formatted into the `few_shot_prompt` template for the model to see and take instruction from. like the previous **Zero-Shot** approach a `max_new_token` is set to `max_new_token=1` to ensure a single response could be returned as an answer for our model. Often times this approach of supplying multiple answers also got a bit tricky because the model would return tokens that weren't apart of the options. even with `temperature` modifications. to solve this I also implemented a `sanitize_response()` method to handle goofy or inappropriate answers using the same regex tool, I ensured all of the first digits would be accounted as the models response and anything that didn't make the maximum of `2` was established as a `0`. The entire validation set was applied to all three models as before for consistency amongst all few-shot in context learning experiments

Llama 3.2 1B: Few-Shot In-Context Learning

Generate a Prompt

In [184…

```
import random

# sets a for consistent repeatability amongst the tests
random.seed(23)

# filter the data for each type of tweet.
# filter is specific to hugging face datasets
```

```
# it will take a lambda that will return a boolean value to filter elements

negative_tweets = ds_train.filter(lambda x: x['label'] == 0)
neutral_tweets = ds_train.filter(lambda x: x['label'] == 1)
positive_tweets = ds_train.filter(lambda x: x['label'] == 2)

# Optional: Preview the sizes
print(f"Number of negative tweets: {len(negative_tweets)}")
print(f"Number of neutral tweets: {len(neutral_tweets)}")
print(f"Number of positive tweets: {len(positive_tweets)}")

print()
random_num = random.randint(0,613)

print(f"Random Negative Tweet: {negative_tweets['text'][random_num]}")

print(f"Random Neutral Tweet: {neutral_tweets['text'][random_num]}")

print(f"Random Positive Tweet: {positive_tweets['text'][random_num]}")
```

```
Number of negative tweets: 613
Number of neutral tweets: 613
Number of positive tweets: 613

Random Negative Tweet: "I'm sorry, but Randy Orton is the most plain wrestle
r on the roster. He does nothing for me in the main event spot. 2nd is Sheam
us."
Random Neutral Tweet: @user @user If it weren't for Blair there would have b
een no Good Friday Agreement, and the IRA would have won!"
Random Positive Tweet: Last night may have been my last Digi concert for a w
hile. But I had an amazing night. Thank you for everything Digi. @user
```

In [187…

```
def generate_few_shot_prompt(validation_tweet):

    # Format the few-shot prompt with placeholders
    few_shot_prompt = """
    {}

    Sentiment (positive (2), negative (0), neutral (1)): {}

    {}

    Sentiment (positive (2), negative (0), neutral (1)): {}

    {}

    Sentiment (positive (2), negative (0), neutral (1)): {}

    {}

    Sentiment (positive (2), negative (0), neutral (1)): {}

    {}

    Sentiment (positive (2), negative (0), neutral (1)): {}
```

```
    {}

    Sentiment (positive (2), negative (0), neutral (1)): {}
    """

    # Flatten the list of text-label pairs for easy insertion
    tweet_texts_and_labels = []

    for tweet in range(2):
        random_num = random.randint(0,612)
        tweet_texts_and_labels.extend([positive_tweets['text'][random_num],
        random_num = random.randint(0,612)
        tweet_texts_and_labels.extend([negative_tweets['text'][random_num],
        random_num = random.randint(0,612)
        tweet_texts_and_labels.extend([neutral_tweets['text'][random_num], n


    # print(tweet_texts_and_labels)

    # Use the format method to fill in the placeholders
    filled_prompt = few_shot_prompt.format(*tweet_texts_and_labels)

    filled_prompt += f"""
    {validation_tweet}

    Sentiment (positive (2), negative (0), netural (1)): """

    return filled_prompt
```

In [200…

```
import numpy as np

# hold the ground_truths
llama_3_2_1B_ground_truths = []
# hold the sentiment_predictions
llama_3_2_1B_sentiment_preds = []

# iterate through the validation set
for tweet, label in zip(ds_validation['text'], ds_validation['label']):
    prompt_tweet = tweet
    gt_label = label
    # combine the prompt
    prompt = generate_few_shot_prompt(prompt_tweet)

    # generate the response
    prompt_ids = llama_3_2_1B_tokenizer.encode(prompt, return_tensors="pt")

    outputs = llama_3_2_1B_model.generate(
                        prompt_ids,
                        temperature=0.18,
                        do_sample=True,
                        pad_token_id=llama_3_2_1B_tokenizer.eos_token_id,
                        max_new_tokens=1
                    )

    # get the response tokens from the model
    generated_tokens = outputs[-1]
```

```
    generated_response = llama_3_2_1B_tokenizer.decode(generated_tokens, ski

    # pass the generated_response to a sanitizer that determines if the resp
    sentiment_resp = santize_response(generated_response[-1])

    # Responses being compared to the Ground_Truth Labels
    # print(f"Generated Response: {sentiment_resp}")
    # print(f"Ground Truth: {gt_label}")

    llama_3_2_1B_ground_truths.append(gt_label)
    llama_3_2_1B_sentiment_preds.append(sentiment_resp)

llama_3_2_1B_ground_truths = np.asarray(llama_3_2_1B_ground_truths)
llama_3_2_1B_sentiment_preds = np.asarray(llama_3_2_1B_sentiment_preds)
```

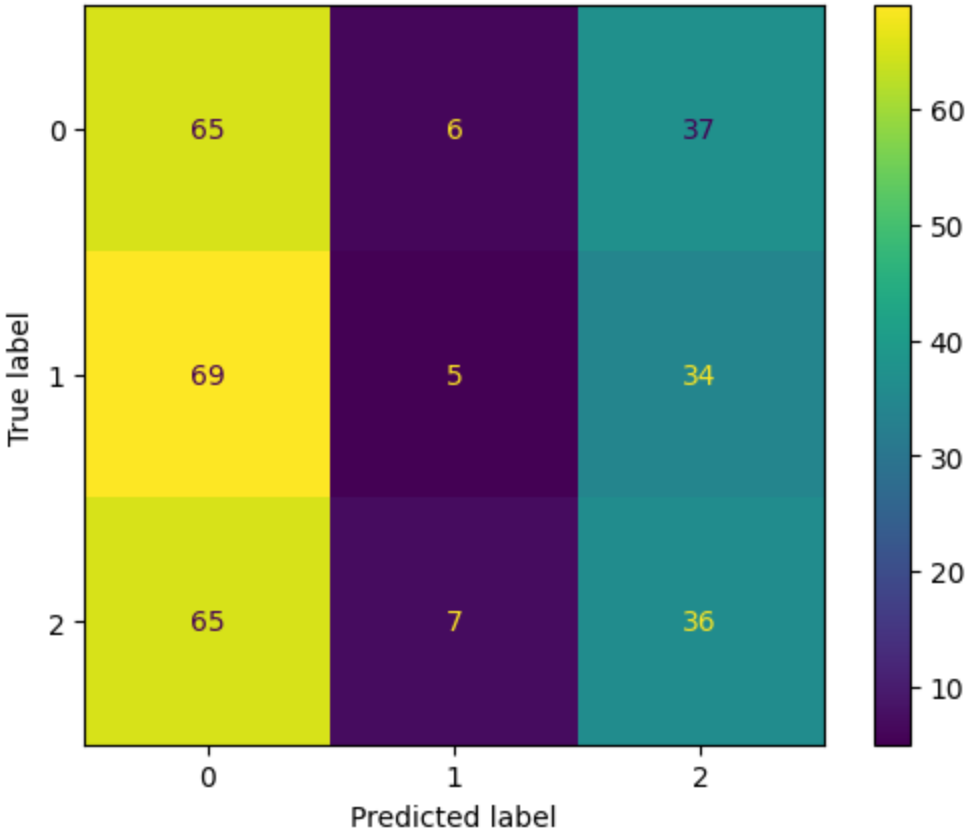
In [201…

```
from sklearn.metrics import classification_report

few_shot_in_context_llama_32_1b_results = classification_report(llama_3_2_1B
print(classification_report(llama_3_2_1B_ground_truths, llama_3_2_1B_sentime
```

```
              precision    recall  f1-score   support

           0       0.38      0.44      0.41       108
           1       0.36      0.59      0.44       108
           2       0.33      0.06      0.11       108

    accuracy                           0.36       324
   macro avg       0.36      0.36      0.32       324
weighted avg       0.36      0.36      0.32       324
```

In [202…

```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay


cm = confusion_matrix(llama_3_2_3B_ground_truths, llama_3_2_3B_sentiment_pre

disp = ConfusionMatrixDisplay(confusion_matrix=cm)

disp.plot()
plt.show()
```



Llama 3.2 3B: Few-Shot In-Context Learning

In [203…

```
import numpy as np

# hold the ground_truths
llama_3_2_3B_ground_truths = []
# hold the sentiment_predictions
llama_3_2_3B_sentiment_preds = []

# iterate through the validation set
for tweet, label in zip(ds_validation['text'], ds_validation['label']):
    prompt_tweet = tweet
    gt_label = label
    # combine the prompt
    prompt = generate_few_shot_prompt(prompt_tweet)

    # generate the response
    prompt_ids = llama_3_2_3B_tokenizer.encode(prompt, return_tensors="pt")

    outputs = llama_3_2_3B_model.generate(
                        prompt_ids,
                        temperature=0.08,
                        do_sample=True,
                        pad_token_id=llama_3_2_3B_tokenizer.eos_token_id,
                        max_new_tokens=1
                    )

    # get the response tokens from the model
```

```
    generated_tokens = outputs[-1]

    generated_response = llama_3_2_3B_tokenizer.decode(generated_tokens, ski

    # pass the generated_response to a sanitizer that determines if the resp
    sentiment_resp = santize_response(generated_response[-1])

    # Responses being compared to the Ground_Truth Labels
    # print(f"Generated Response: {sentiment_resp}")
    # print(f"Ground Truth: {gt_label}")

    llama_3_2_3B_ground_truths.append(gt_label)
    llama_3_2_3B_sentiment_preds.append(sentiment_resp)

llama_3_2_3B_ground_truths = np.asarray(llama_3_2_3B_ground_truths)
llama_3_2_3B_sentiment_preds = np.asarray(llama_3_2_3B_sentiment_preds)
```

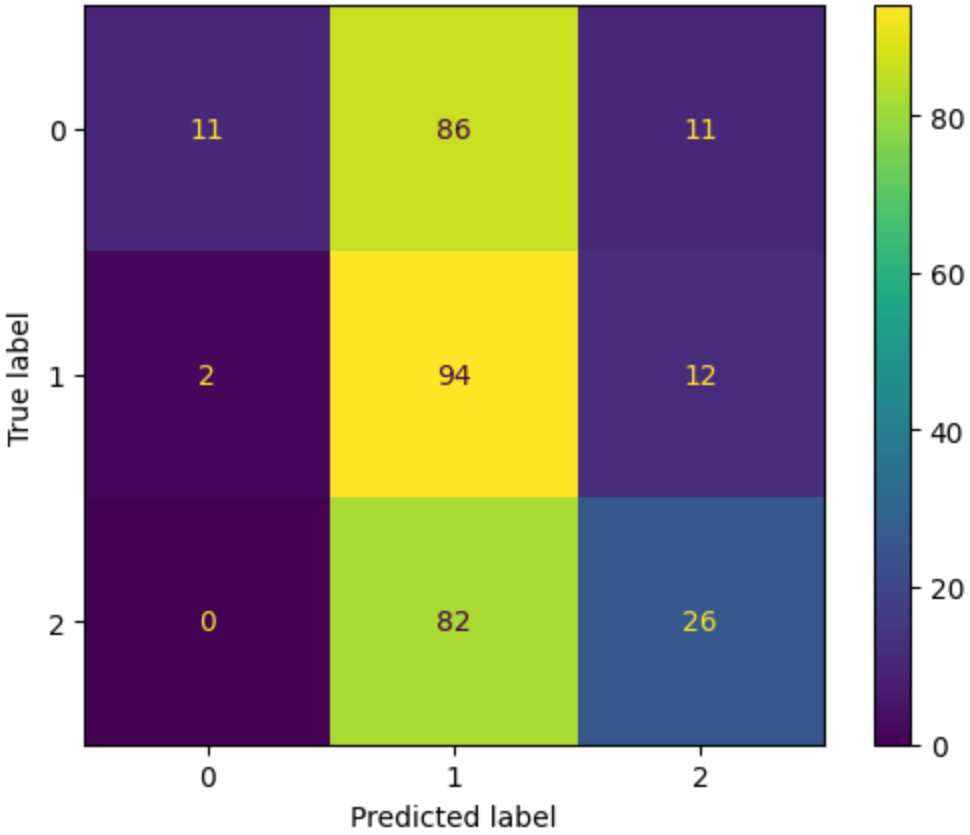
In [206…

```
from sklearn.metrics import classification_report

few_shot_prompting_llama_32_3b_results = classification_report(llama_3_2_3B_
print(classification_report(llama_3_2_3B_ground_truths, llama_3_2_3B_sentime
```

```
              precision    recall  f1-score   support

           0       0.85      0.10      0.18       108
           1       0.36      0.87      0.51       108
           2       0.53      0.24      0.33       108

    accuracy                           0.40       324
   macro avg       0.58      0.40      0.34       324
weighted avg       0.58      0.40      0.34       324
```

In [207…

```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay


cm = confusion_matrix(llama_3_2_3B_ground_truths, llama_3_2_3B_sentiment_pre

disp = ConfusionMatrixDisplay(confusion_matrix=cm)

disp.plot()
plt.show()
```



Phi 3.5-mini-instruct: Few Shot In-Context Learning

In [208…

```
import numpy as np

# hold the ground_truths
phi_3_5_ground_truths = []
# hold the sentiment_predictions
phi_3_5_sentiment_preds = []

# iterate through the validation set
for tweet, label in zip(ds_validation['text'], ds_validation['label']):
    prompt_tweet = tweet
    gt_label = label
    # combine the prompt
    prompt = generate_few_shot_prompt(prompt_tweet)

    # generate the response
    prompt_ids = phi_3_5_tokenizer.encode(prompt, return_tensors="pt")

    outputs = phi_3_5_model.generate(
                        prompt_ids,
                        temperature=0.07,
                        do_sample=True,
                        pad_token_id=phi_3_5_tokenizer.eos_token_id,
                        max_new_tokens=1
                    )

    # get the response tokens from the model
```

```
    generated_tokens = outputs[-1]

    generated_response = phi_3_5_tokenizer.decode(generated_tokens, skip_spe

    # pass the generated_response to a sanitizer that determines if the resp
    sentiment_resp = santize_response(generated_response[-1])

    # Responses being compared to the Ground_Truth Labels
    # print(f"Generated Response: {sentiment_resp}")
    # print(f"Ground Truth: {gt_label}")

    phi_3_5_ground_truths.append(gt_label)
    phi_3_5_sentiment_preds.append(sentiment_resp)

phi_3_5_ground_truths = np.asarray(phi_3_5_ground_truths)
phi_3_5_sentiment_preds = np.asarray(phi_3_5_sentiment_preds)
```

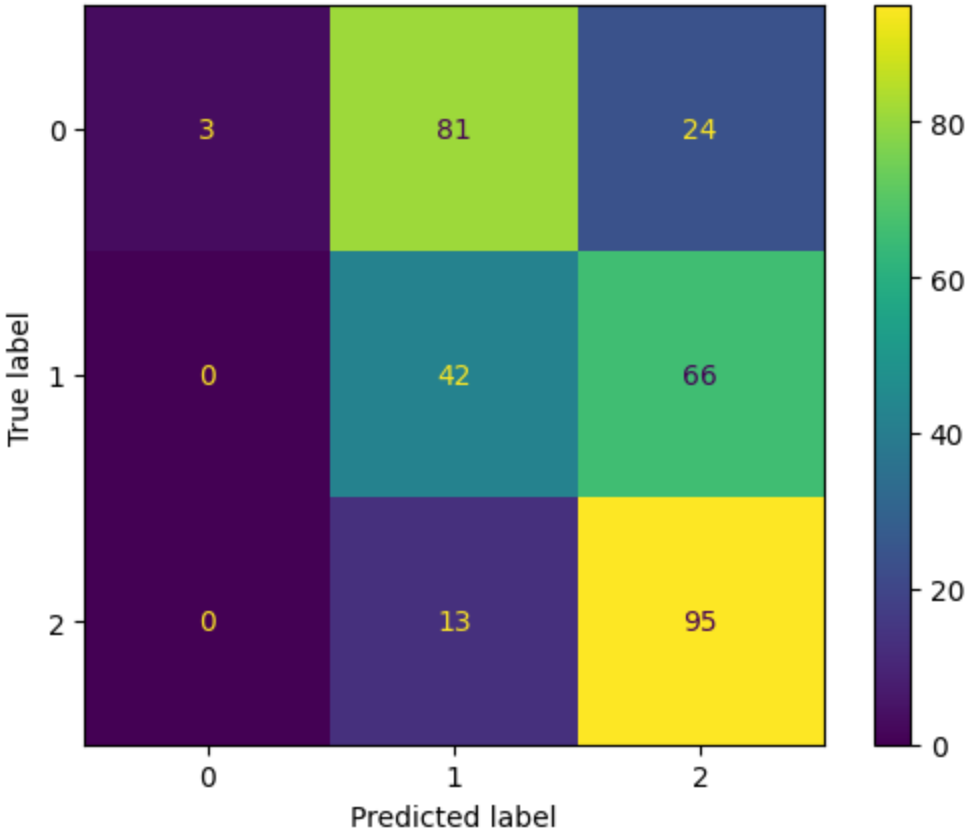
In [209…

```
from sklearn.metrics import classification_report

few_shot_prompt_phi_35_mini_results = classification_report(phi_3_5_ground_t
print(classification_report(phi_3_5_ground_truths, phi_3_5_sentiment_preds))
```

```
              precision    recall  f1-score   support

           0       1.00      0.03      0.05       108
           1       0.31      0.39      0.34       108
           2       0.51      0.88      0.65       108

    accuracy                           0.43       324
   macro avg       0.61      0.43      0.35       324
weighted avg       0.61      0.43      0.35       324
```

In [210…

```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay


cm = confusion_matrix(phi_3_5_ground_truths, phi_3_5_sentiment_preds)

disp = ConfusionMatrixDisplay(confusion_matrix=cm)

disp.plot()
plt.show()
```



Experiment 2: Few-Shot In-Context Learning Results

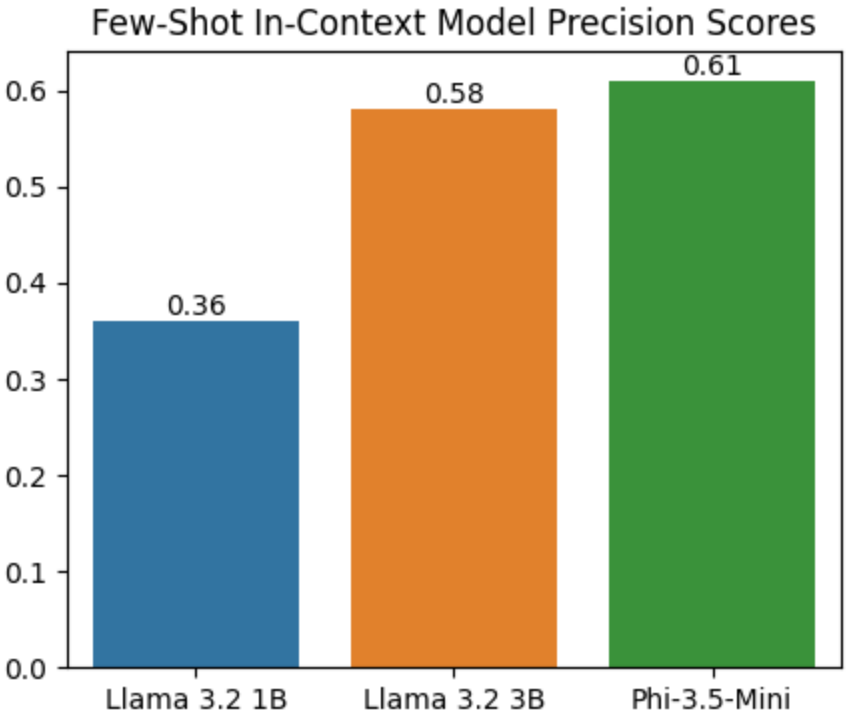
Few-Shot In-Context Model Precision Scores

In [407…

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.color_palette("Set2")

models = ['Llama 3.2 1B','Llama 3.2 3B','Phi-3.5-Mini']

few_shot_llama_32_1b = round(few_shot_in_context_llama_32_1b_results['macro

few_shot_llama_32_3b = round(few_shot_prompting_llama_32_3b_results['macro a

few_shot_phi_3_5_mini = round(few_shot_prompt_phi_35_mini_results['macro avg

# create precision plot list
precision_scores = [few_shot_llama_32_1b, few_shot_llama_32_3b, few_shot_phi

plt.figure(figsize=(5,4), dpi=100)
ax = sns.barplot(x=models, y=precision_scores, hue=models)
# add labels
ax.bar_label(ax.containers[0], fontsize=10)
ax.bar_label(ax.containers[1], fontsize=10)
ax.bar_label(ax.containers[2], fontsize=10)
```

```
plt.title('Few-Shot In-Context Model Precision Scores')
```

Out[407… Text(0.5, 1.0, 'Few-Shot In-Context Model Precision Scores')

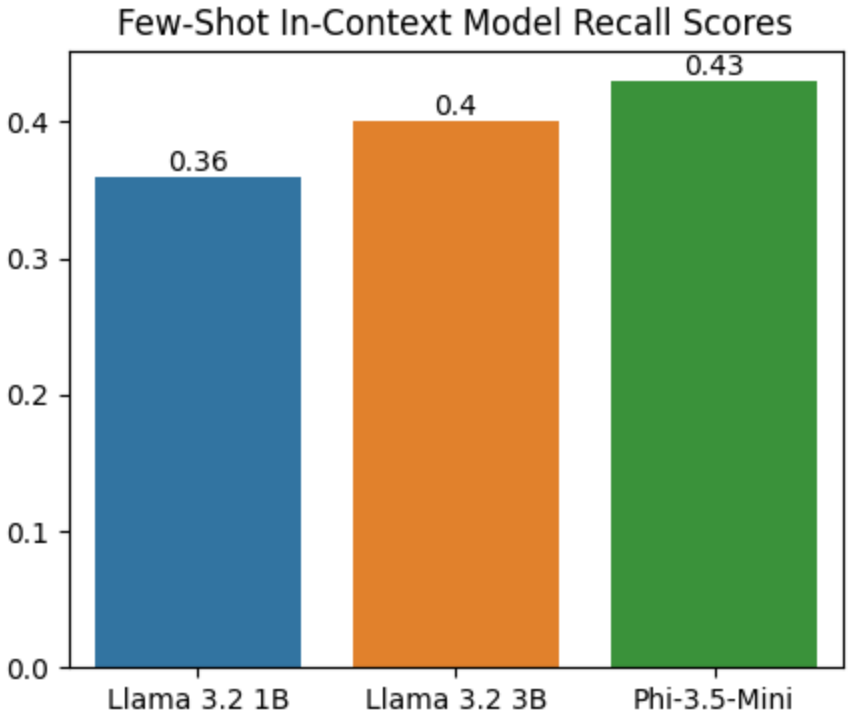


Few-Shot In-Context Model Recall Scores

In [399…

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.color_palette("Set2")

models = ['Llama 3.2 1B','Llama 3.2 3B','Phi-3.5-Mini']

few_shot_llama_32_1b = round(few_shot_in_context_llama_32_1b_results['macro

few_shot_llama_32_3b = round(few_shot_prompting_llama_32_3b_results['macro a

few_shot_phi_3_5_mini = round(few_shot_prompt_phi_35_mini_results['macro avg

# create precision plot list
recall_scores = [few_shot_llama_32_1b, few_shot_llama_32_3b, few_shot_phi_3_

plt.figure(figsize=(5,4), dpi=100)
ax = sns.barplot(x=models, y=recall_scores, hue=models)
# add labels
ax.bar_label(ax.containers[0], fontsize=10)
ax.bar_label(ax.containers[1], fontsize=10)
ax.bar_label(ax.containers[2], fontsize=10)
plt.title('Few-Shot In-Context Model Recall Scores')
```

Out[399… Text(0.5, 1.0, 'Few-Shot In-Context Model Recall Scores')

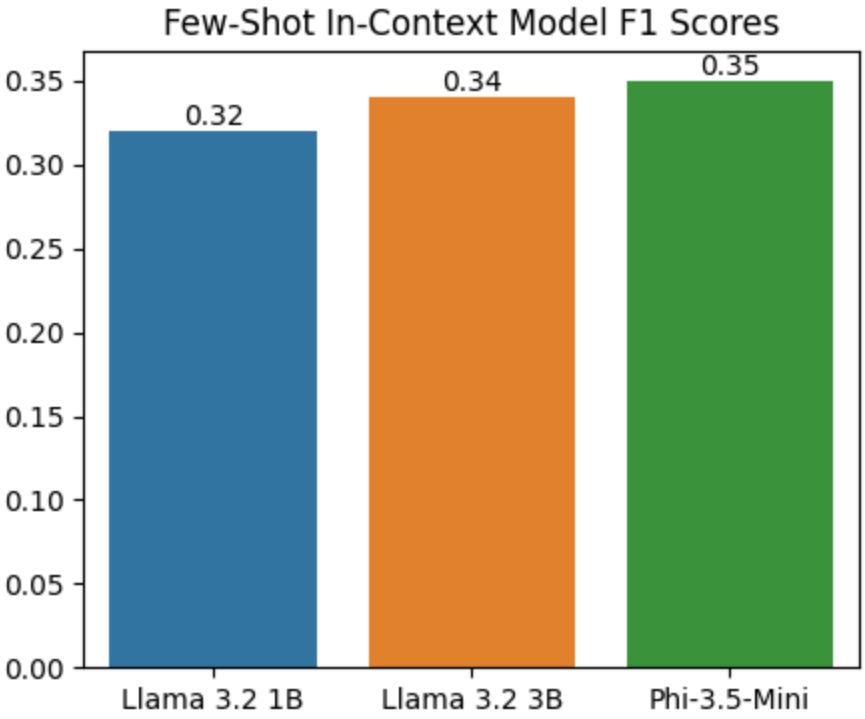


Few-Shot In-Context Model F1-Scores

In [398…

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.color_palette("Set2")

models = ['Llama 3.2 1B','Llama 3.2 3B','Phi-3.5-Mini']

few_shot_llama_32_1b = round(few_shot_in_context_llama_32_1b_results['macro

few_shot_llama_32_3b = round(few_shot_prompting_llama_32_3b_results['macro a

few_shot_phi_3_5_mini = round(few_shot_prompt_phi_35_mini_results['macro avg

# create precision plot list
f1_scores = [few_shot_llama_32_1b, few_shot_llama_32_3b, few_shot_phi_3_5_mi

plt.figure(figsize=(5,4), dpi=100)
ax = sns.barplot(x=models, y=f1_scores, hue=models)
# add labels
ax.bar_label(ax.containers[0], fontsize=10)
ax.bar_label(ax.containers[1], fontsize=10)
ax.bar_label(ax.containers[2], fontsize=10)
plt.title('Few-Shot In-Context Model F1 Scores')
```

Out[398… `Text(0.5, 1.0, 'Few-Shot In-Context Model F1 Scores')`



Analysis for Experiment 2: Few-Shot In Context Learning

Overall this experiment offered many prompts for our models to glance over and analyise before making a sentiment decision. It help formulate a moderate exposure to various sentiment flavors using a `k` value of `2`, meaning there were `6` total training prompts `2` of each sentiment variety. The highest performing `precision` and `recall` models were the **Llama 3.2 3B** and **Phi 3.5** models. There was little improvement within the `f1-scores` of either models to declare a large lead, ~33% averaged micro average between all three models. which is quite frankly terrible and quite possibly so due to the `single token approach` implementation.

Experiment 3: Advanced Prompting Technique (Zero-Shot Inference)

Approach: Parse Single Token Response from Emotion Based Chain of Thought (Pleaing for Homework help)

the strategy for this experiement was much like the approach taken for the Zero-Shot Inference. The Prompt was assigned and with a format string template and the `sanitize_response()` method was called to handle any additional responses that weren't specified from the format of the model's `generate()` call. the `new_max_token` was set equal to `1` as in the other responses that were gathered from previous experiments up to this point. I tried several prompts but I noticed a sort of

combination between `chain of thought` asking the model to `think about the sentiment` and an emotional plea of `I need this for my homework` was just as serious as asking `I need this for my job`. I supplied the response in the form of a provided numeric range just like the other prompts using `2` for `positive`, `0` for `negative` and `1` for `neutral`. I also applied the use of hyper parameters using `do_sample=True` to ensure the `temperature` is applied to all of the models **Llama 3.2 1B** and **Llama 3.2 3B** as well as **Phi 3.5**. I tired with larger temperatures and found that they weren't able to produce results much better than the previous runs.

Llama 3.2 1 B: Advanced Prompting Technique

In [240…
```
chain_of_thought_prompt = """
Read the following tweet:

{}

think about the sentiment, I need this for my homework,
respond with a single number that is your numeric choice of either (positive
```

In [262…
```
import numpy as np

# hold the ground_truths
llama_3_2_1B_ground_truths = []
# hold the sentiment_prediction
llama_3_2_1B_sentiment_preds = []

# iterate through the validation set
for tweet, label in zip(ds_validation['text'], ds_validation['label']):
    prompt_tweet = tweet
    gt_label = label
    # combine the prompt
    prompt = chain_of_thought_prompt.format(prompt_tweet)

    # generate the response
    prompt_ids = llama_3_2_1B_tokenizer.encode(prompt, return_tensors="pt")

    outputs = llama_3_2_1B_model.generate(
                        prompt_ids,
                        do_sample=True,
                        temperature=0.23,
                        pad_token_id=llama_3_2_1B_tokenizer.eos_token_id,
                        max_new_tokens=5
                    )

    # get the response tokens from the model
    generated_tokens = outputs[-1]

    generated_response = llama_3_2_1B_tokenizer.decode(generated_tokens, ski

    # pass the generated_response to a sanitizer that determines if the resp
    # print(generated_response)
```

```
    # print(generated_response[-1])
    sentiment_resp = santize_response(generated_response[-1])

    # Responses being compared to the Ground_Truth Labels
    # print(f"Generated Response: {sentiment_resp}")
    # print(f"Ground Truth: {gt_label}")

    llama_3_2_1B_ground_truths.append(gt_label)
    llama_3_2_1B_sentiment_preds.append(sentiment_resp)

llama_3_2_1B_ground_truths = np.asarray(llama_3_2_1B_ground_truths)
llama_3_2_1B_sentiment_preds = np.asarray(llama_3_2_1B_sentiment_preds)
```

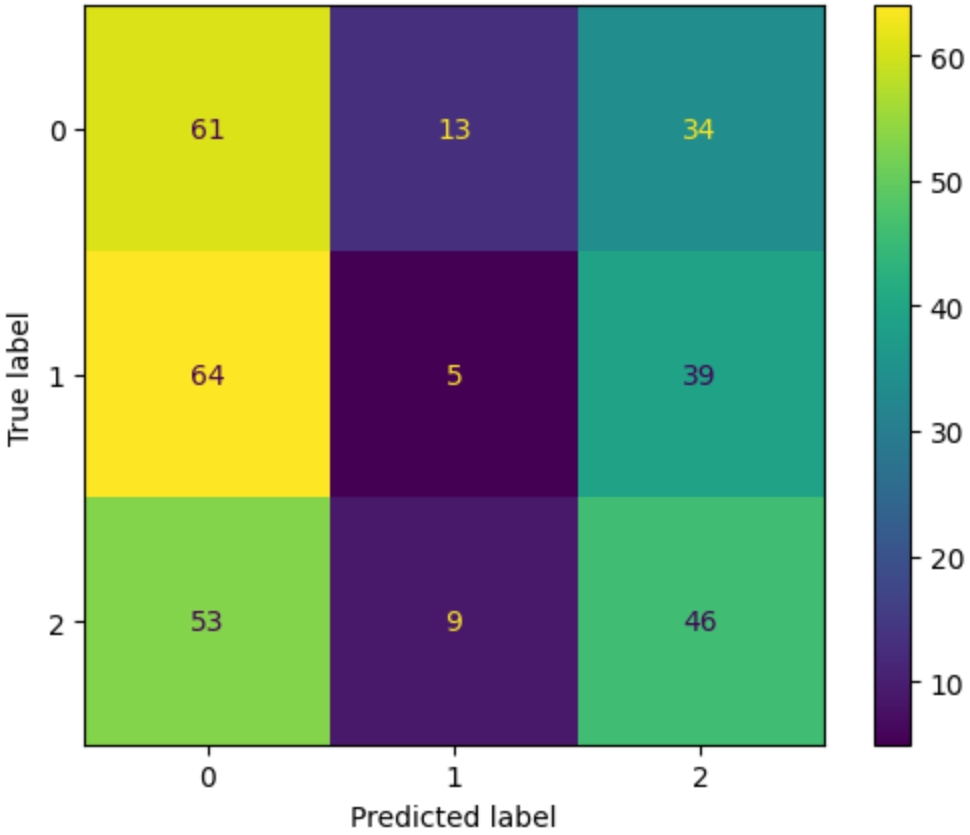
In [264…

```
from sklearn.metrics import classification_report

adv_prompting_llama_3_2_1B_results = classification_report(llama_3_2_1B_grou
print(classification_report(llama_3_2_1B_ground_truths, llama_3_2_1B_sentime
```

```
              precision    recall  f1-score   support

           0       0.34      0.56      0.43       108
           1       0.19      0.05      0.07       108
           2       0.39      0.43      0.41       108

    accuracy                           0.35       324
   macro avg       0.30      0.35      0.30       324
weighted avg       0.30      0.35      0.30       324
```

In [265…

```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay


cm = confusion_matrix(llama_3_2_1B_ground_truths, llama_3_2_1B_sentiment_pre

disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```



Llama 3.2 3B: Advanced Prompting Technique

In [266…
```
chain_of_thought_prompt = """
Read the following tweet:

{}

think about the sentiment, I need this for my homework,
respond with a single number that is your numeric choice of either (positive
```

In [267…
```
import numpy as np

# hold the ground_truths
llama_3_2_3B_ground_truths = []
# hold the sentiment_prediction
llama_3_2_3B_sentiment_preds = []

# iterate through the validation set
for tweet, label in zip(ds_validation['text'], ds_validation['label']):
    prompt_tweet = tweet
    gt_label = label
    # combine the prompt
    prompt = chain_of_thought_prompt.format(prompt_tweet)

    # generate the response
    prompt_ids = llama_3_2_3B_tokenizer.encode(prompt, return_tensors="pt")

    outputs = llama_3_2_1B_model.generate(
```

```
                        prompt_ids,
                        do_sample=True,
                        temperature=0.23,
                        pad_token_id=llama_3_2_3B_tokenizer.eos_token_id,
                        max_new_tokens=5
                    )

    # get the response tokens from the model
    generated_tokens = outputs[-1]

    generated_response = llama_3_2_1B_tokenizer.decode(generated_tokens, ski

    # pass the generated_response to a sanitizer that determines if the resp
    # print(generated_response)

    # print(generated_response[-1])
    sentiment_resp = santize_response(generated_response[-1])

    # Responses being compared to the Ground_Truth Labels
    # print(f"Generated Response: {sentiment_resp}")
    # print(f"Ground Truth: {gt_label}")

    llama_3_2_3B_ground_truths.append(gt_label)
    llama_3_2_3B_sentiment_preds.append(sentiment_resp)

llama_3_2_3B_ground_truths = np.asarray(llama_3_2_3B_ground_truths)
llama_3_2_3B_sentiment_preds = np.asarray(llama_3_2_3B_sentiment_preds)
```

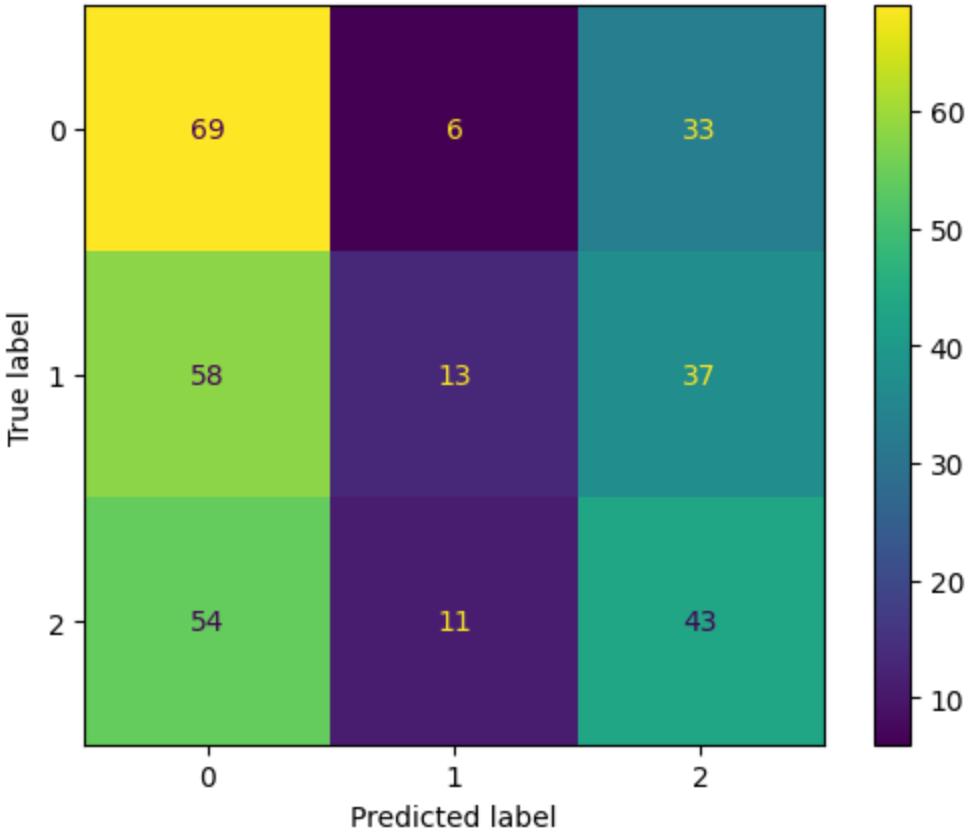
In [268…

```
from sklearn.metrics import classification_report

adv_prompting_lllama_3_2_3B_results = classification_report(llama_3_2_3B_gro
print(classification_report(llama_3_2_3B_ground_truths, llama_3_2_3B_sentime
```

```
              precision    recall  f1-score   support

           0       0.38      0.64      0.48       108
           1       0.43      0.12      0.19       108
           2       0.38      0.40      0.39       108

    accuracy                           0.39       324
   macro avg       0.40      0.39      0.35       324
weighted avg       0.40      0.39      0.35       324
```

In [269…

```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay


cm = confusion_matrix(llama_3_2_3B_ground_truths, llama_3_2_3B_sentiment_pre

disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```



Phi 3.5-mini-instruct

In [270…
```
chain_of_thought_prompt = """
Read the following tweet:

{}

think about the sentiment, I need this for my homework,
respond with a single number that is your numeric choice of either (positive
```

In [274…
```
import numpy as np

# hold the ground_truths
phi_3_5_ground_truths = []
# hold the sentiment_predictions
phi_3_5_sentiment_preds = []

# iterate through the validation set
for tweet, label in zip(ds_validation['text'], ds_validation['label']):
    prompt_tweet = tweet
    gt_label = label
    # combine the prompt
    prompt = generate_few_shot_prompt(prompt_tweet)

    # generate the response
    prompt_ids = phi_3_5_tokenizer.encode(prompt, return_tensors="pt")

    outputs = phi_3_5_model.generate(
```

```
                        prompt_ids,
                        temperature=0.1,
                        do_sample=True,
                        pad_token_id=phi_3_5_tokenizer.eos_token_id,
                        max_new_tokens=1
                    )

    # get the response tokens from the model
    generated_tokens = outputs[-1]

    generated_response = phi_3_5_tokenizer.decode(generated_tokens, skip_spe

    # pass the generated_response to a sanitizer that determines if the resp
    sentiment_resp = santize_response(generated_response[-1])

    # Responses being compared to the Ground_Truth Labels
    # print(f"Generated Response: {sentiment_resp}")
    # print(f"Ground Truth: {gt_label}")

    phi_3_5_ground_truths.append(gt_label)
    phi_3_5_sentiment_preds.append(sentiment_resp)

phi_3_5_ground_truths = np.asarray(phi_3_5_ground_truths)
phi_3_5_sentiment_preds = np.asarray(phi_3_5_sentiment_preds)
```

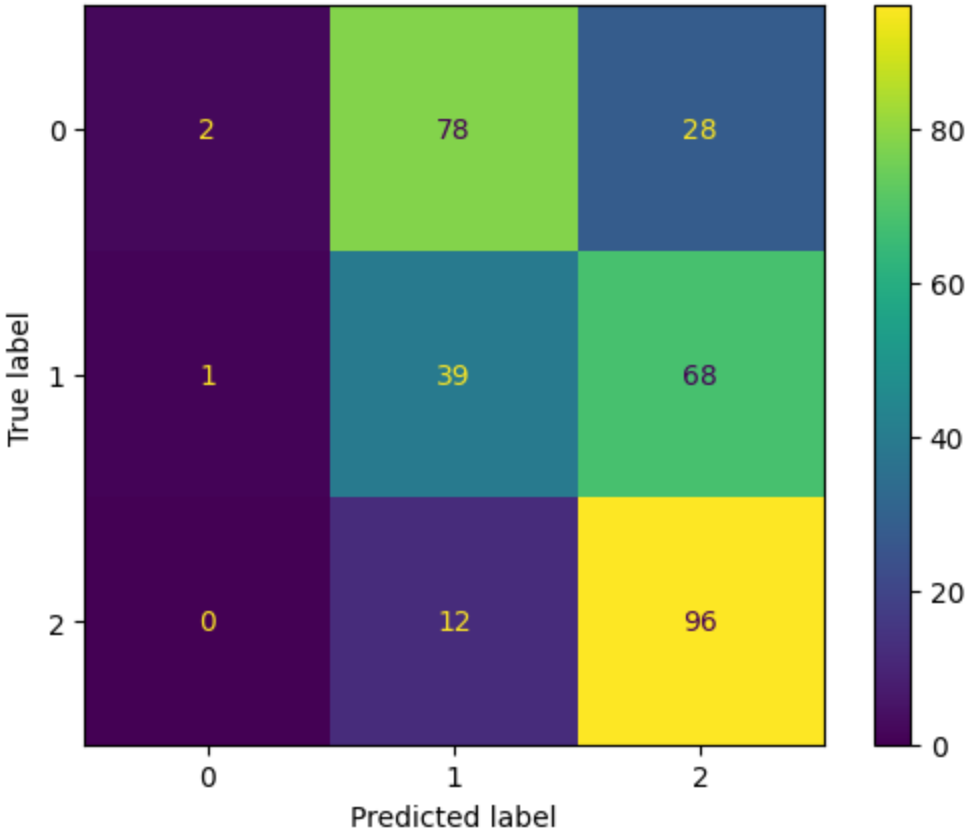
In [275…

```
from sklearn.metrics import classification_report

adv_prompt_phi_35_mini_results = classification_report(phi_3_5_ground_truths
print(classification_report(phi_3_5_ground_truths, phi_3_5_sentiment_preds))
```

```
              precision    recall  f1-score   support

           0       0.67      0.02      0.04       108
           1       0.30      0.36      0.33       108
           2       0.50      0.89      0.64       108

    accuracy                           0.42       324
   macro avg       0.49      0.42      0.34       324
weighted avg       0.49      0.42      0.34       324
```

In [276…

```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay


cm = confusion_matrix(phi_3_5_ground_truths, phi_3_5_sentiment_preds)

disp = ConfusionMatrixDisplay(confusion_matrix=cm)

disp.plot()
plt.show()
```



Experiment 3: Advanced Prompting Technique (Zero-Shot Innference) Results

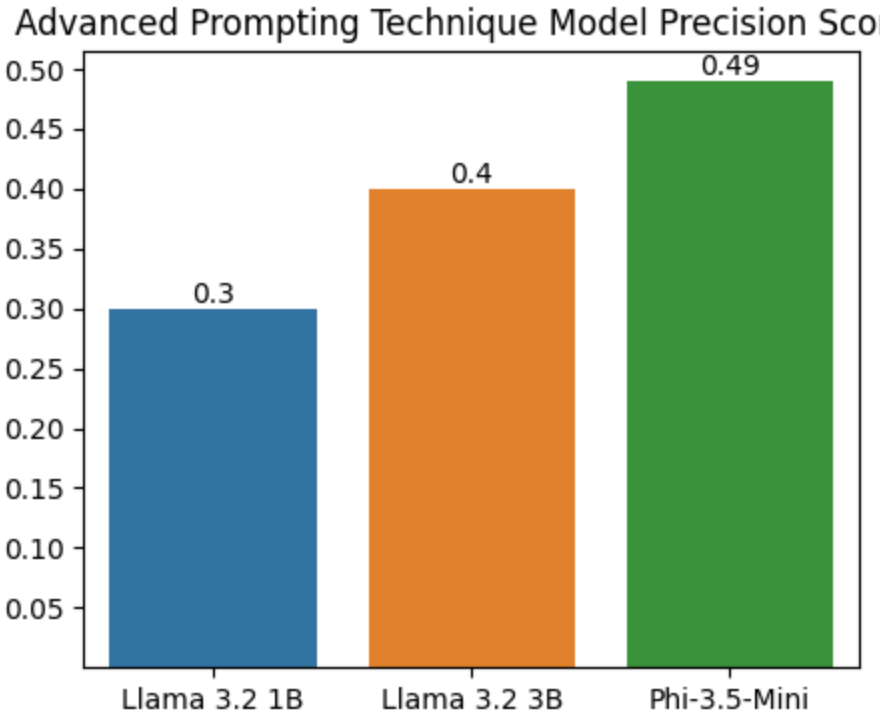
Advanced Prompting Technique Model Precision Scores

In [367…

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.color_palette("Set2")

models = ['Llama 3.2 1B','Llama 3.2 3B','Phi-3.5-Mini']

adv_prompting_llama_32_1b = round(adv_prompting_llama_3_2_1B_results['macro

adv_prompting_llama_32_3b = round(adv_prompting_lllama_3_2_3B_results['macro

adv_prompting_phi_35_mini = round(adv_prompt_phi_35_mini_results['macro avg'

precision_scores = [adv_prompting_llama_32_1b, adv_prompting_llama_32_3b, ad

plt.figure(figsize=(5,4), dpi=100)
ticks = np.arange(0.05,0.65,0.05)

plt.yticks(ticks=ticks)
ax = sns.barplot(x=models, y=precision_scores, hue=models )
# add labels
```

```
ax.bar_label(ax.containers[0], fontsize=10)
ax.bar_label(ax.containers[1], fontsize=10)
ax.bar_label(ax.containers[2], fontsize=10)
plt.title('Advanced Prompting Technique Model Precision Scores')
```

Out[367… `Text(0.5, 1.0, 'Advanced Prompting Technique Model Precision Scores')`



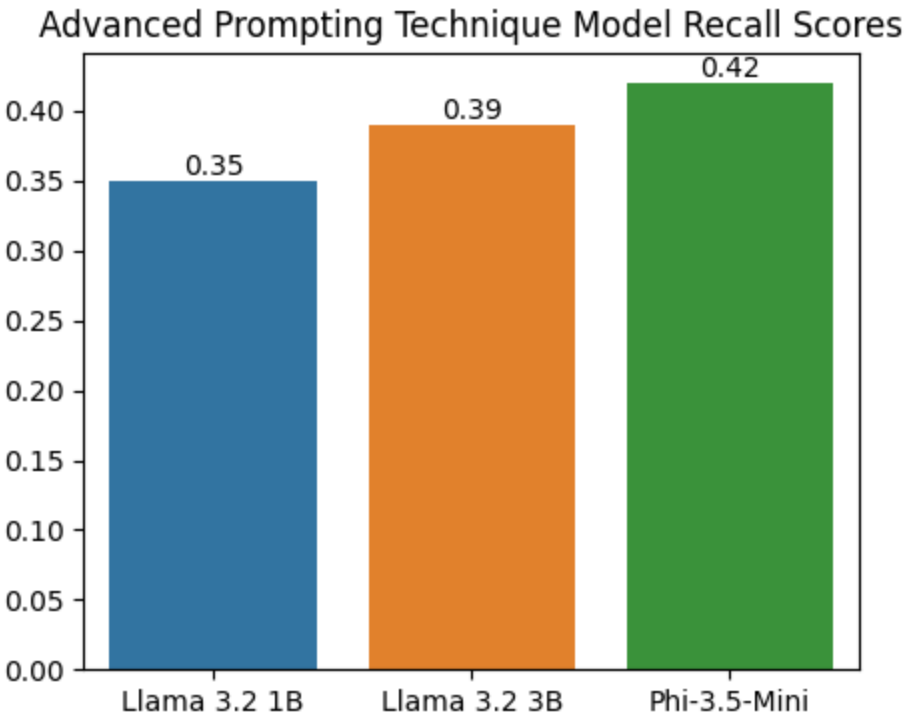
Advanced Prompting Technique Model Recall Scores

In [378…

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.color_palette("Set2")

models = ['Llama 3.2 1B','Llama 3.2 3B','Phi-3.5-Mini']

adv_prompting_llama_32_1b = round(adv_prompting_llama_3_2_1B_results['macro

adv_prompting_llama_32_3b = round(adv_prompting_lllama_3_2_3B_results['macro

adv_prompting_phi_35_mini = round(adv_prompt_phi_35_mini_results['macro avg'

recall_scores = [adv_prompting_llama_32_1b, adv_prompting_llama_32_3b, adv_p

plt.figure(figsize=(5,4), dpi=100)


ax = sns.barplot(x=models, y=recall_scores, hue=models )

# add labels
ax.bar_label(ax.containers[0], fontsize=10)
ax.bar_label(ax.containers[1], fontsize=10)
```

```
ax.bar_label(ax.containers[2], fontsize=10)

plt.title('Advanced Prompting Technique Model Recall Scores')
```

Out[378… Text(0.5, 1.0, 'Advanced Prompting Technique Model Recall Scores')

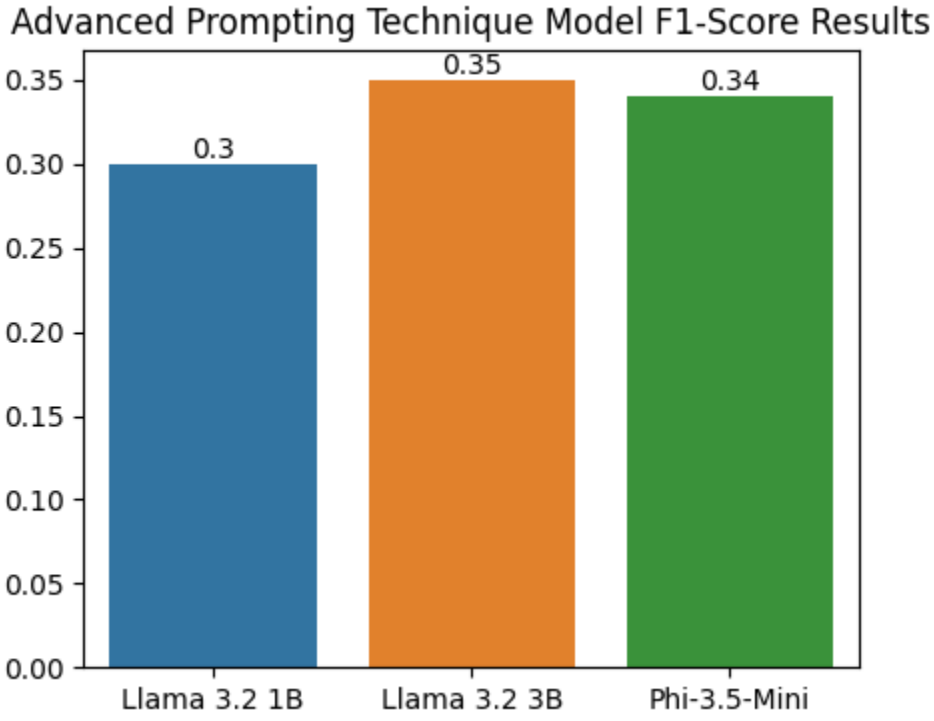


Advanced Prompting Technique Model F2 Scores

In [377…

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

sns.color_palette("Set2")

models = ['Llama 3.2 1B','Llama 3.2 3B','Phi-3.5-Mini']

adv_prompting_llama_32_1b = round(adv_prompting_llama_3_2_1B_results['macro

adv_prompting_llama_32_3b = round(adv_prompting_lllama_3_2_3B_results['macro

adv_prompting_phi_35_mini = round(adv_prompt_phi_35_mini_results['macro avg'

f1_scores = [adv_prompting_llama_32_1b, adv_prompting_llama_32_3b, adv_promp

plt.figure(figsize=(5,4), dpi=100)

ax = sns.barplot(x=models, y=f1_scores, hue=models )
# add labels
ax.bar_label(ax.containers[0], fontsize=10)
ax.bar_label(ax.containers[1], fontsize=10)
ax.bar_label(ax.containers[2], fontsize=10)

plt.title('Advanced Prompting Technique Model F1-Score Results')
```

Out[377… `Text(0.5, 1.0, 'Advanced Prompting Technique Model F1-Score Results')`



Analysis for Experiment 3: Advanced Prompting Technique

Over all the models behaved closer to the results I would expect. **Phi 3.5** performed Higher in `precision`, and `recall` metrics. There was a slight lead by **Llama 3.2 3B** of 1% in `F-1 Score` but this isn't significant enough to note as most of the `f1-scores` for these models during this experiment were ~.5% apart. Overall I was suprised to see a moderate performance increase for **Llama 3.2 3B** overall I would say this model performed well overall. It had a noticably improved ability to decern false-negatives that the other methods tended to fail to assist with. **Phi 3.5** seemed to miss identify `neutral` sentiment tweets. I am not sure if it was adjusting for my prompt or not. In the previous `few-shot` experiment **Phi 3.5** exhibited this behavior and also failed to correctly label `negative` sentiments. But the previous experiment saw higher `precision` from all models and this approach seemed to have reduced the `precision` and `recall` but ~20% for **Lama 3.2 3B** and **Phi 3.5**. So pleading for homework help with **One-Shot** Prompt will likely not be as helpful as providing a **few-shot** strategy for an untrained **Llama 3.2** model.

Combined Experiment Results

Precision Scores of Models Per Experiment

Build DataFrame for Analysis

In [384…

```
import pandas as pd


# experiment 1 precision results
exp_1_llama_32_1b_precision = round(zero_shot_llama_3_2_1B_model_results['ma

exp_1_llama_32_3b_precision = round(zero_shot_llama_3_2_3B_results['macro av

exp_1_phi_35_mini_precision = round(zero_shot_phi_3_5_results['macro avg']['

# experiment 2 precision results
exp_2_llama_32_1b_precision = round(few_shot_in_context_llama_32_1b_results[

exp_2_llama_32_3b_precision = round(few_shot_prompting_llama_32_3b_results['

exp_2_phi_35_mini_precision = round(few_shot_prompt_phi_35_mini_results['mac

# experiment 3 precision results
exp_3_llama_32_1b_precision = round(adv_prompting_llama_3_2_1B_results['macr

exp_3_llama_32_3b_precision = round(adv_prompting_lllama_3_2_3B_results['mac

exp_3_phi_35_mini_precision = round(adv_prompt_phi_35_mini_results['macro av


precision_data = {'Experiment': ['Zero-Shot Experiment 1','Few-Shot Experiem
```

```
                    'Model': [
                            ['Llama 3.2 1B', 'Llama 3.2 3B', 'Phi-3.5-Mini'],
                            ['Llama 3.2 1B', 'Llama 3.2 3B', 'Phi-3.5-Mini'],
                            ['Llama 3.2 1B', 'Llama 3.2 3B', 'Phi-3.5-Mini']
                    ],
                    'Precision': [
                                 [exp_1_llama_32_1b_precision, exp_1_llama_32
                                 [exp_2_llama_32_1b_precision, exp_2_llama_32
                                 [exp_3_llama_32_1b_precision, exp_3_llama_32
                                ]
                }

df_precision = pd.DataFrame(data=precision_data)

# expands each model and precision value to a new row per experiment
df_precision = df_precision.explode(['Model', 'Precision'], ignore_index=Tru

# dataframe to represent the model's and their precision scores
df_precision
```

Out[384…

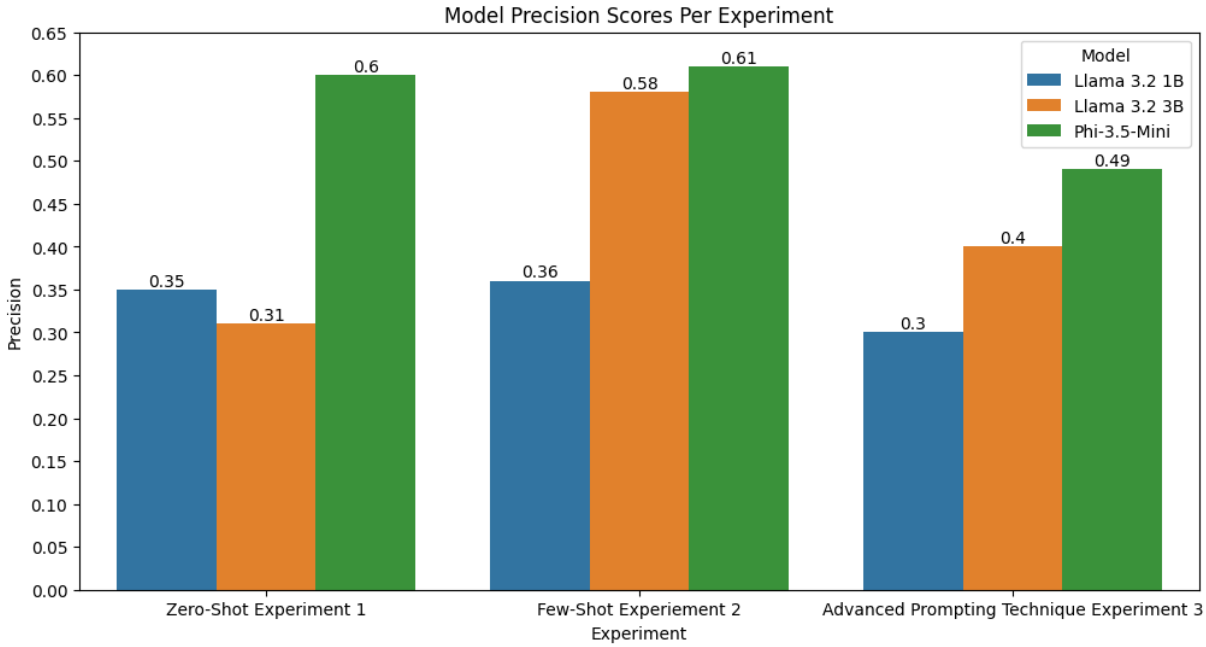
	Experiment	Model	Precision
0	Zero-Shot Experiment 1	Llama 3.2 1B	0.35
1	Zero-Shot Experiment 1	Llama 3.2 3B	0.31
2	Zero-Shot Experiment 1	Phi-3.5-Mini	0.6
3	Few-Shot Experiement 2	Llama 3.2 1B	0.36
4	Few-Shot Experiement 2	Llama 3.2 3B	0.58
5	Few-Shot Experiement 2	Phi-3.5-Mini	0.61
6	Advanced Prompting Technique Experiment 3	Llama 3.2 1B	0.3
7	Advanced Prompting Technique Experiment 3	Llama 3.2 3B	0.4
8	Advanced Prompting Technique Experiment 3	Phi-3.5-Mini	0.49

Graph Precision of Models Over All Experiments

In [386…

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12,6), dpi=100)
y_range = np.arange(0.0,0.7,0.05)

ax = sns.barplot(data=df_precision, x=df_precision['Experiment'], y=df_preci
ax.bar_label(ax.containers[0], fontsize=10);
ax.bar_label(ax.containers[1], fontsize=10);
plt.yticks(ticks=y_range)
ax.bar_label(ax.containers[2], fontsize=10);
plt.title('Model Precision Scores Per Experiment')
```

Out[386… Text(0.5, 1.0, 'Model Precision Scores Per Experiment')



Recall Scores of Models Per Experiment

Build DataFrame for Analysis

In [391…

```
import pandas as pd


# experiment 1 recall results
exp_1_llama_32_1b_recall = round(zero_shot_llama_3_2_1B_model_results['macro

exp_1_llama_32_3b_recall = round(zero_shot_llama_3_2_3B_results['macro avg']

exp_1_phi_35_mini_recall = round(zero_shot_phi_3_5_results['macro avg']['rec

# experiment 2 recall results
exp_2_llama_32_1b_recall = round(few_shot_in_context_llama_32_1b_results['ma

exp_2_llama_32_3b_recall = round(few_shot_prompting_llama_32_3b_results['mac

exp_2_phi_35_mini_recall = round(few_shot_prompt_phi_35_mini_results['macro

# experiment 3 recall results
exp_3_llama_32_1b_recall = round(adv_prompting_llama_3_2_1B_results['macro a

exp_3_llama_32_3b_recall = round(adv_prompting_lllama_3_2_3B_results['macro

exp_3_phi_35_mini_recall = round(adv_prompt_phi_35_mini_results['macro avg']


recall_data = {'Experiment': ['Zero-Shot Experiment 1','Few-Shot Experiement
                  'Model': [
                         ['Llama 3.2 1B', 'Llama 3.2 3B', 'Phi-3.5-Mini'],
```

```
                        ['Llama 3.2 1B', 'Llama 3.2 3B', 'Phi-3.5-Mini'],
                        ['Llama 3.2 1B', 'Llama 3.2 3B', 'Phi-3.5-Mini']
                    ],
                    'Recall': [
                                [exp_1_llama_32_1b_recall, exp_1_llama_32_3b
                                [exp_2_llama_32_1b_recall, exp_2_llama_32_3b
                                [exp_3_llama_32_1b_recall, exp_3_llama_32_3b
                               ]
                }

df_recall = pd.DataFrame(data=recall_data)

# expands each model and recall value to a new row per model
df_recall = df_recall.explode(['Model', 'Recall'], ignore_index=True)

# dataframe to represent the model's and their recall scores
df_recall
```

Out[391…

	Experiment	Model	Recall
0	Zero-Shot Experiment 1	Llama 3.2 1B	0.34
1	Zero-Shot Experiment 1	Llama 3.2 3B	0.33
2	Zero-Shot Experiment 1	Phi-3.5-Mini	0.48
3	Few-Shot Experiement 2	Llama 3.2 1B	0.36
4	Few-Shot Experiement 2	Llama 3.2 3B	0.4
5	Few-Shot Experiement 2	Phi-3.5-Mini	0.43
6	Advanced Prompting Technique Experiment 3	Llama 3.2 1B	0.35
7	Advanced Prompting Technique Experiment 3	Llama 3.2 3B	0.39
8	Advanced Prompting Technique Experiment 3	Phi-3.5-Mini	0.42

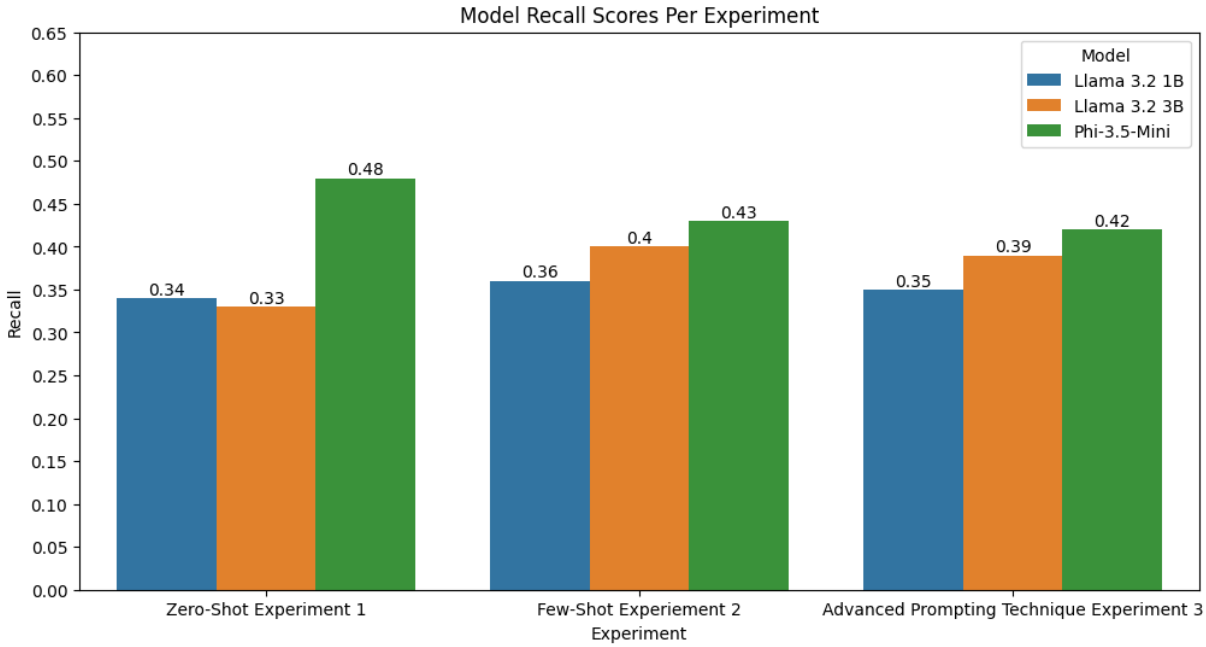
Graph Recall of Models Over all Experiments

In [392…

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12,6), dpi=100)
y_range = np.arange(0.0,0.7,0.05)

ax = sns.barplot(data=df_recall, x=df_recall['Experiment'], y=df_recall['Rec
ax.bar_label(ax.containers[0], fontsize=10);
ax.bar_label(ax.containers[1], fontsize=10);
plt.yticks(ticks=y_range)
ax.bar_label(ax.containers[2], fontsize=10);
plt.title('Model Recall Scores Per Experiment')
```

Out[392…

```
Text(0.5, 1.0, 'Model Recall Scores Per Experiment')
```



F-1 Scores of Models Per Experiment

Build Dataframe For Analysis

In [394…

```
import pandas as pd


# experiment 1 recall results
exp_1_llama_32_1b_f1_score = round(zero_shot_llama_3_2_1B_model_results['mac

exp_1_llama_32_3b_f1_score = round(zero_shot_llama_3_2_3B_results['macro avg

exp_1_phi_35_mini_f1_score = round(zero_shot_phi_3_5_results['macro avg']['f

# experiment 2 recall results
exp_2_llama_32_1b_f1_score = round(few_shot_in_context_llama_32_1b_results['

exp_2_llama_32_3b_f1_score = round(few_shot_prompting_llama_32_3b_results['m

exp_2_phi_35_mini_f1_score = round(few_shot_prompt_phi_35_mini_results['macr

# experiment 3 recall results
exp_3_llama_32_1b_f1_score = round(adv_prompting_llama_3_2_1B_results['macro

exp_3_llama_32_3b_f1_score = round(adv_prompting_lllama_3_2_3B_results['macr

exp_3_phi_35_mini_f1_score = round(adv_prompt_phi_35_mini_results['macro avg


f1_score_data = {'Experiment': ['Zero-Shot Experiment 1','Few-Shot Experieme
                 'Model': [
                         ['Llama 3.2 1B', 'Llama 3.2 3B', 'Phi-3.5-Mini'],
                         ['Llama 3.2 1B', 'Llama 3.2 3B', 'Phi-3.5-Mini'],
                         ['Llama 3.2 1B', 'Llama 3.2 3B', 'Phi-3.5-Mini']
```

```
                ],
                'F1-Score': [
                                [exp_1_llama_32_1b_f1_score, exp_1_llama_32_
                                [exp_2_llama_32_1b_f1_score, exp_2_llama_32_
                                [exp_3_llama_32_1b_f1_score, exp_3_llama_32_
                               ]
              }

df_f1_score = pd.DataFrame(data=f1_score_data)

# expands each model and f1 value to a new row per experiment
df_f1_score = df_f1_score.explode(['Model', 'F1-Score'], ignore_index=True)

# dataframe to represent the model's and their f1 scores
df_f1_score
```

Out[394…

	Experiment	Model	F1-Score
0	Zero-Shot Experiment 1	Llama 3.2 1B	0.33
1	Zero-Shot Experiment 1	Llama 3.2 3B	0.28
2	Zero-Shot Experiment 1	Phi-3.5-Mini	0.41
3	Few-Shot Experiement 2	Llama 3.2 1B	0.32
4	Few-Shot Experiement 2	Llama 3.2 3B	0.34
5	Few-Shot Experiement 2	Phi-3.5-Mini	0.35
6	Advanced Prompting Technique Experiment 3	Llama 3.2 1B	0.3
7	Advanced Prompting Technique Experiment 3	Llama 3.2 3B	0.35
8	Advanced Prompting Technique Experiment 3	Phi-3.5-Mini	0.34

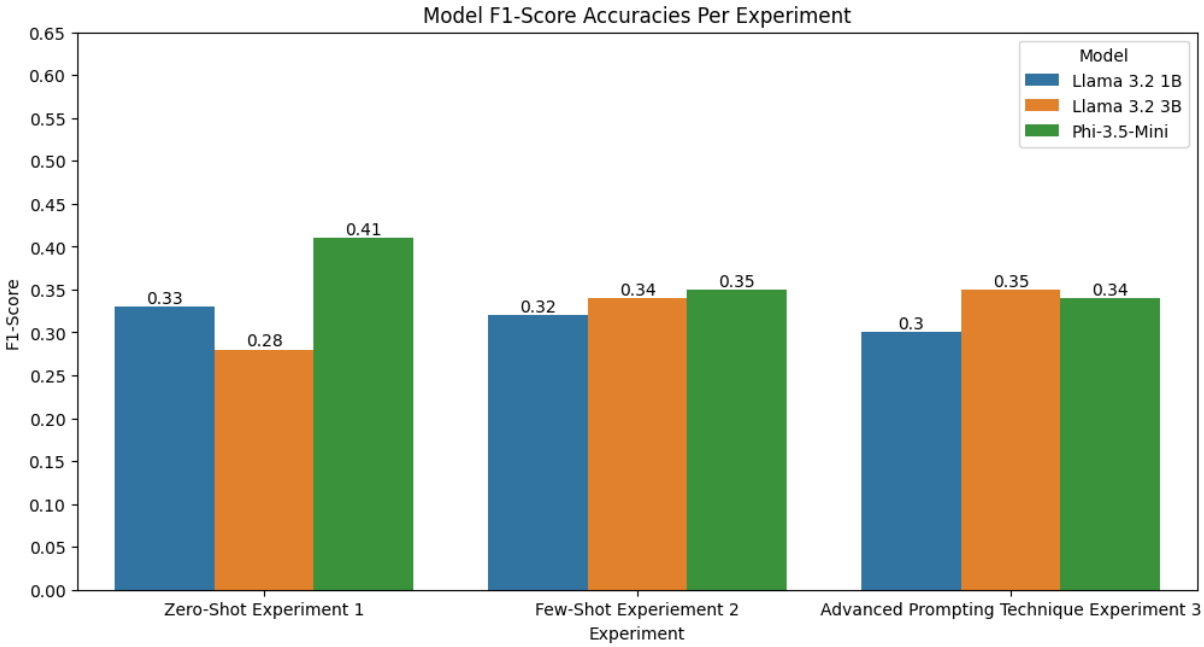
Graph F1-Score of Models Over all Experiments

In [396…

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(12,6), dpi=100)
y_range = np.arange(0.0,0.7,0.05)

ax = sns.barplot(data=df_f1_score, x=df_f1_score['Experiment'], y=df_f1_scor
ax.bar_label(ax.containers[0], fontsize=10);
ax.bar_label(ax.containers[1], fontsize=10);
plt.yticks(ticks=y_range)
ax.bar_label(ax.containers[2], fontsize=10);
plt.title('Model F1-Score Accuracies Per Experiment')
```

Out[396…

```
Text(0.5, 1.0, 'Model F1-Score Accuracies Per Experiment')
```


Model F1-Score Accuracies Per Experiment
Model
Llama 3.2 1B
Llama 3.2 3B
Phi-3.5-Mini
F1-Score
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0.33
0.28
0.41
0.32
0.34
0.35
0.3
0.35
0.34
Zero-Shot Experiment 1
Few-Shot Experiement 2
Advanced Prompting Technique Experiment 3
Experiment