# Final Report:

# News Analysis for Potential Investment Strategies on Tesla Stock

---

## Why this project?

Investment strategy on stocks is an extremely challenging topic and of keen interest to a wide audience. The stock market, with its inherent complexity and unpredictability, presents a fascinating area for machine learning exploration. Investors and stock analysts dedicate considerable effort to conducting fundamental company analysis, identifying patterns in stock price movements, analyzing market and investor sentiment, studying economic indicators, and scrutinizing public news and events. Given the extensive nature of this analysis, employing Large Language Models (LLMs) to harness their strengths in natural language processing tasks appears to be a promising strategy.

LLMs enable the rapid processing and interpretation of vast amounts of news information. The goal in this project is to conduct a thorough analysis of news related to Tesla in a short period of time, aiming to uncover any potential correlations between public sentiment in the news and fluctuations in Tesla's stock price. This endeavor not only poses a significant challenge but also holds the promise of offering novel insights into the complicated dynamics between media sentiment and stock market movements.

Simply focusing on news related to Tesla is insufficient for accurately predicting the stock's price movements. To develop a precise forecasting tool for Tesla's stock, it's essential to consider a multitude of factors. However, this work can act as a preliminary component of the ultimate tool to formulate investment strategies for Tesla stock.

## Data

Two free Application Programming Interfaces (APIs) were used in this work to collect Tesla related news from multiple English sources and Tesla stock price movements in the past. APIs allow automatic and efficient data collection for individual developers and companies.

- NewsAPI
    - Limited to past 30 days
    - Limited to 1,000 requests per day
    - 345 news articles collected
- MarketStack
    - Limited to 1,000 requests per month
    - Stock market closes on weekends and holidays

In this work, analysis was performed for the time period from **2023-09-12** to **2023-10-11**.

## Method

### Stock Price

After collecting the data, the stock price movements were calculated based on the daily close price. Close price change compared to yesterday was used as the indicator for Tesla stock price movements:

$$Close\ Price\ Change\ compared\ to\ Yesterday\ =\ Close\ Today\ -\ Close\ Yesterday$$

Positive close price change values indicated a price raise compared to yesterday and negative ones indicated a price drop.

### News Analysis

After a preliminary manual study on the collected news, some news articles were found not directly related to Tesla but related to Elon Musk's personal life, his political opinions

and his other companies. This resulted from the keywords extraction method used by NewsAPI where ‘Tesla’ and ‘Elon Musk’ were used. Therefore, the news analysis required two tools:

1. A classification tool to tell if a news article was directly related to Tesla or not
2. A sentiment analysis tool to predict if a news article would have positive, negative or neutral impact on Tesla stock.

Pre-trained Large Language Models (LLMs) are highly effective and suitable for a range of natural language processing tasks, including text generation, Q&A chatbots, text summarization, text classification, and sentiment analysis. The two news analysis tasks above can be addressed by using pre-trained LLMs with tuning. Figure 1 showed how LLMs work with encoder and decoder systems.

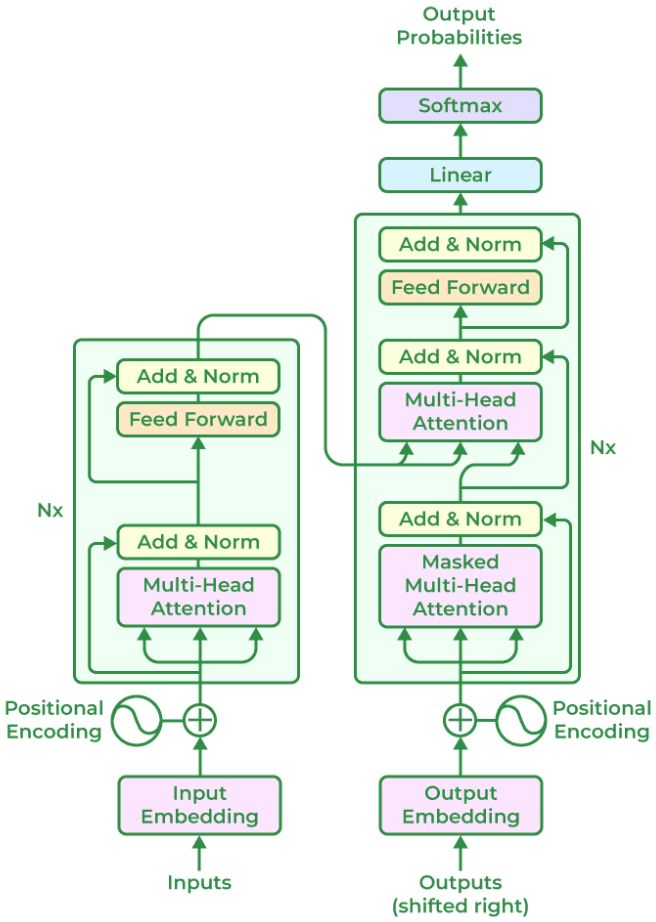



Figure 1. LLM General Architect with Encoder and Decoder Components.

Task 1. Text Classification

✨ Flan-T5 model, a LLM developed by Google to handle text2text generation tasks, such as translation and summarization, was chosen for this task. The model size is 248M parameters. In order to perform the required text classification task on news, tuning is needed for this model. Full fine-tuning is a very expensive approach and requires a sufficient amount of data. Therefore, the following cost-effective approaches were adopted for tuning Flan-T5 for the news classification task.

- Prompt Engineering / In-Context Learning (ICL)
  - worked well with one-shot learning
- Parameter-Efficient Fine-tuning (PEFT) with Low-Rank Adaptation (LoRA)
  - didn't work well due to limited number of data

Task 2. Sentiment Analysis

✨ FinBERT model, a LLM developed for financial sentiment analysis with BERT, was directly used for sentiment analysis of collected news. FinBERT was developed to achieve better financial domain sentiment analysis by taking the pre-trained BERT and then further training it on a purely financial corpus. That corpus is Reuters TRC2.

## News Classification

### Approach 1: Prompt Engineering / In-Context Learning

Prompt engineering is a technique that can improve the performance of LLMs on specific tasks. It involves carefully designing the input prompt to guide the LLM towards the desired output. Prompt engineering is also known as in-context learning (ICL) due to its nature to modify the prompt context to clarify task definition and to provide context provision.

## Data Pre-processing

60% of the total collected news (207 news articles) were randomly selected for manual labeling for news classification. News having a high potential impact on Tesla stock prices were labeled as directly related to Tesla. Other news such as on Elon Musk's personality were marked as not directly related to Tesla. Figure 2 showed the distribution of manual labels. The binary class was overall balanced.

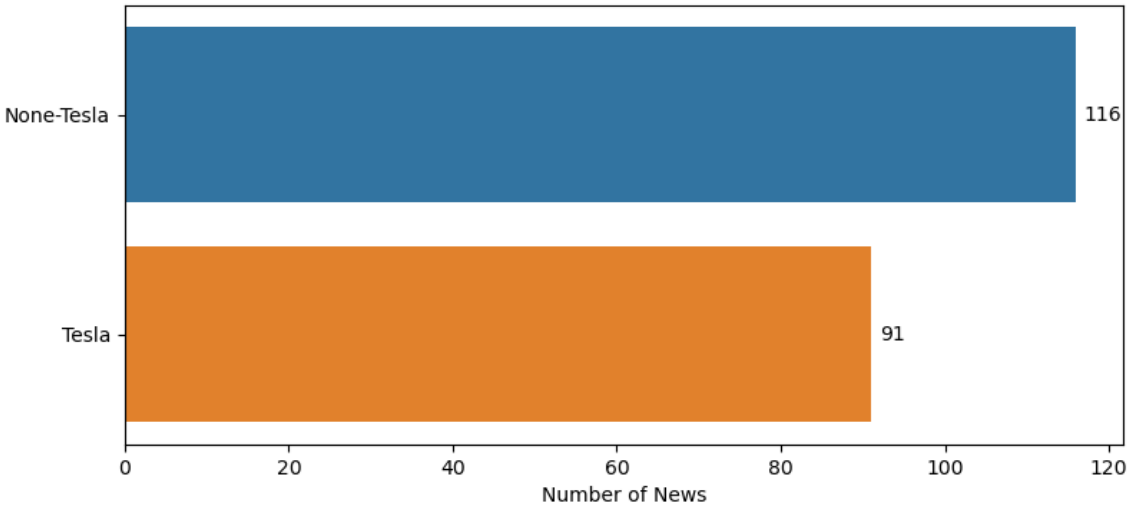



Figure 2. Tesla and None-Tesla News Distribution of Manual Labeling

## Prompt Engineering for Flan-T5

*The `transformers`* library was used to load the pre-trained Flan-T5 base model. Without any prompt engineering, the Flan-T5 base model tended to summarize the news content. In order to achieve the news classification task, prompt engineering was investigated to leverage the Flan-T5 model to classify if a news was directly related to Tesla or not. The BERT model was also evaluated but ICL failed to work for it. ICL worked for LLMs with generative capability, such as Flan-T5.

### *Zero-shot learning*

The prompt was modified as the code snippet shown below. The outputs from Flan-T5 with zero-shot learning were just "Not". Without giving any example, the model was confused with what the proper output should be for this specific classification task.

```
prompt = f"""
Classify if the news is related to Tesla or not.
{context}
Classification output:
"""
```

*One-shot learning*

A function called *"make_prompt"* was defined for one-shot and few-shot learning, shown in the code snippet below. For one shot learning, only one example (randomly selected and used for all) was provided in the prompt before the current classification task. With one-shot learning, Flan-T5 started to show reasonable classification outputs like the example given below.

```
News 2
This is the prompt:

        Context from news:
        AOC says she's looking to trade in her Tesla for a union-made EV after clash with Elon Musk AOC reiterated on Face the
Nation on Sunday that she wants to trade in her Tesla for a union-made EV.
        Is the context related to Tesla?
        Yes, it is related to Tesla.

    Context from news:
    Mercedes-Benz's $100,000 electric SUV is an awesome Tesla rival – but its blob-like looks aren't for everyone The Mercedes
EQE SUV is an awesome alternative to Tesla's Model X with controversial style.
    Is the context related to Tesla?

Model output: Yes
True label: 1
```

```
def make_prompt(examples, new):
    prompt = ''
    for i in range(examples.shape[0]):
        context = examples.iloc[i]['title_desc']
        true_label = examples.iloc[i]['tesla_related_num']
        if true_label == 0:
            label = 'No, it is not related to Tesla.'
        else:
            label = 'Yes, it is related to Tesla.'

        prompt += f"""
        Context from news:
        {context}
        Is the context related to Tesla?
        {label}
        """
    prompt += f"""
    Context from news:
    {new}
    Is the context related to Tesla?
    """
    return prompt
```

*Few-shot learning (2, 3, 4, 5 shots)*

For few-shot learning, more examples were given in the prompt before the current classification task. The same function *"make_prompt"* was used to generate few-shot prompts with 2, 3, 4 and 5 shots (randomly selected and used for all). From other people's experience, few-shot learning doesn't see much improvement with more than 5 shots. A two-shot learning example prompt was given below. Flan-T5 with few-shot learning also provided reasonable classification outputs.

```
News 3
This is the prompt:

        Context from news:
        AOC says she's looking to trade in her Tesla for a union-made EV after clash with Elon Musk AOC reiterated on Face the
Nation on Sunday that she wants to trade in her Tesla for a union-made EV.
        Is the context related to Tesla?
        Yes, it is related to Tesla.

        Context from news:
        Elon Musk sent a graphic mid-childbirth picture of Grimes to friends and family and was surprised when she got upset W
hen Musk shared the photo with Grimes' father and brothers, she said "It was his Asperger's coming out in full."
        Is the context related to Tesla?
        No, it is not related to Tesla.

    Context from news:
    Did Elon Musk Turn Off Starlink for Ukraine? What We Know Excerpt from new Elon Musk biography had said the billionaire ef
fectively thwarted a Ukrainian sub attack
    Is the context related to Tesla?

Model output: No
True label: 0
```

## Prompt Engineering Evaluation

*Hallucination Issues*

Hallucination issues were found for prompt engineering on the Flan-T5 model. The issues arose when increasing the shot examples in the prompt. With more example shots in the prompt, the model struggled to give correct classification results, like shown in Figure 3.

Simply providing more data or information in a prompt does not necessarily lead to better outputs. An overload of information, like the few-shot learning in this work, can sometimes confuse the model or dilute the focus, leading to less accurate responses. The choice of words in the prompt is more critical than the number of words. Finding the

proper words to improve model performance on a specific task is key to prompt engineering.

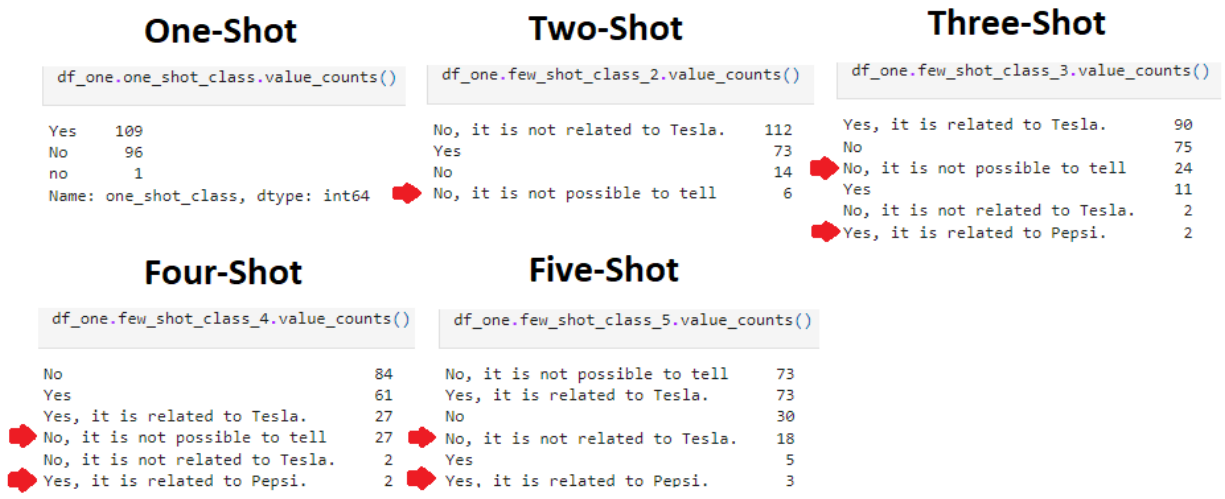



Figure 3. Hallucination Issues of Flan-T5 Prompt Engineering

*Classification Report and Confusion Matrix*

For this text classification task, metrics such as classification report and confusion matrix were utilized to evaluate Flan-T5 performance with different prompt engineering methods applied. Model outputs were compared with manual labels to calculate the metrics. One-shot learning provided the best performance with an average accuracy of 0.85, highest among all ICL tried. Including more examples in the prompt (few-shot learning) didn't improve the overall accuracy.

Classification reports of one-shot learning Flan-T5 outputs and four-shot learning Flan-T5 outputs were compared in Figure 4. Few-shot learning showed high precision for 'yes' and high recall for 'no'.High precision for 'yes' indicates most predicted 'yes' are true 'yes', but there are possible true 'yes' predicted as 'no'. High recall for 'no' indicates most true 'no' are predicted as 'no', but there are possible true 'yes' predicted as 'no'. Therefore, all few-shot learning suffered from mislabeling true 'yes' as 'no', which was a big issue here.

Confusion matrix for one-shot prompt engineering Flan-T5 outputs was given in Figure 5, which was consistent with the classification report.

**One-Shot**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| no | 0.81 | 0.87 | 0.84 | 91 |
| yes | 0.89 | 0.84 | 0.87 | 115 |
| accuracy | | | 0.85 | 206 |
| macro avg | 0.85 | 0.86 | 0.85 | 206 |
| weighted avg | 0.86 | 0.85 | 0.85 | 206 |

**Four-Shot**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| no | 0.75 | 0.94 | 0.84 | 90 |
| yes | 0.94 | 0.75 | 0.84 | 113 |
| accuracy | | | 0.84 | 203 |
| macro avg | 0.85 | 0.85 | 0.84 | 203 |
| weighted avg | 0.86 | 0.84 | 0.84 | 203 |

Figure 4. Classification Report for One-shot Learning and Four-shot Learning Outputs

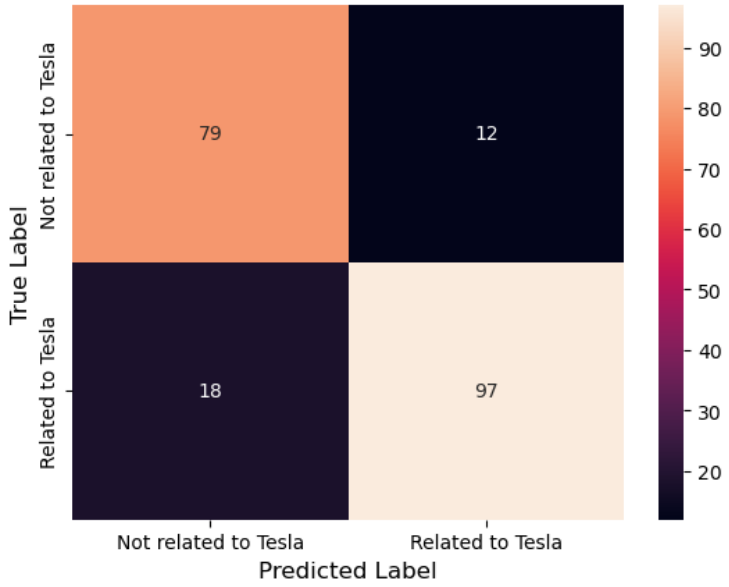



Figure 5. Confusion Matrix for One-shot Learning Flan-T5 Outputs

## Approach 2: PEFT with LoRA

Low-Rank Adaptation (LoRA) is a Parameter-Efficient Fine-Tuning (PEFT) method which has caught a lot of attention. Fine-tuning is the process of training a LLM (pre-trained on some large corpus) with some new data for a specific task of interest. Due to the large amount of parameters of LLMs, tens of billions of parameters, fine-tuning can be extremely computational expensive. PEFT techniques have been proposed to help reduce the cost when adapting LLMs to particular problems.

There are three major PEFT methods, selective, re-parameter and additive. Selective methods involve freezing the majority of the parameters of a LLM then only updating

selected layers, usually the outer layers. Additive methods tend to add more layers to embedding or more output layers to improve model performance on specific tasks. LoRA is one of the re-parameter methods. Instead of updating the whole weight matrices, LoRA freezes those large matrices and creates two smaller matrices for weight updating. As shown in Figure 6, matrix A and matrix B are the two rank decomposition matrices trained with new data during LoRA. The product of A and B is then added to the original weight matrix W in order to alter the model performance. Rank parameter r is an important hyperparameter to be tuned, the smaller, the lower the cost, but potentially the bigger drop in model performance.

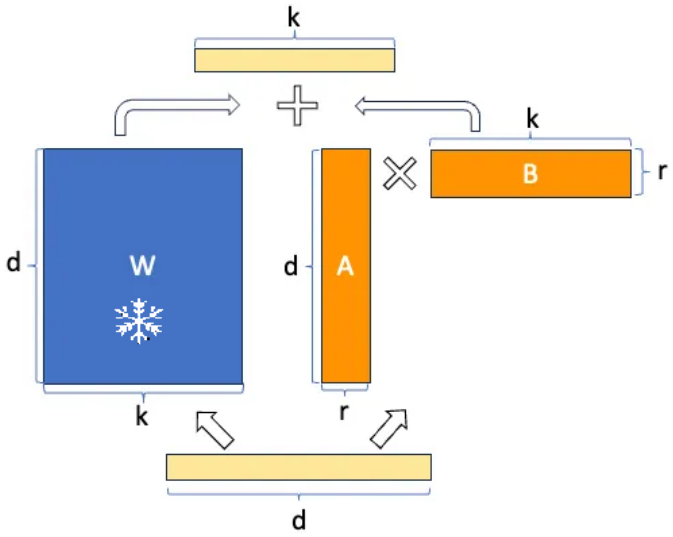



Figure 6. LoRA Weight Updating Algorithm

Data Pre-processing

207 news articles with manual classification labels were used for tuning the Flan-T5 base model through LoRA. In order to tune LLMs, embedding of text input and true labels as tensors of numbers is a prerequisite. `*torch.utils.data.Dataset*` was used to create a Pytorch dataset class to store the embeddings of data for modeling.

70% of the input data were randomly selected as the training data. The remaining 30% of input were evenly splitted as the validation data and the test data. The binary classes were balanced among all datasets.

Zero-shot prompt engineering was implemented as shown in the code snippet below to clarify the task for the model.

In [5]:
```
# convert text to a prompt for classification task
def create_prompt(news):
  return f"""
  Classify if the following news paragraph is directly related to Tesla or not.\n\n
      {news}\n\n
  Is the news directly related to Tesla?
  """
```

PEFT with LoRA

The *`transformers.Trainer`* library from Hugging Face was used for fine-tuning the Flan-T5 base model with the prepared training data and validation data. The *`peft`* library was used for PEFT with LoRA.

The code snippet below showed how to generate a PEFT model using the base model and LoRA configurations. LoRA related parameters were defined by using the `*peft.lora_config*` library. The `*peft.peft_model*` library was used to combine the base model and the LoRA configurations to create a new PEFT model for tuning. With the current LoRA configurations, the percentage of trainable parameters was significantly reduced to only 1.41% of all model parameters.

In [84]:
```
# create LoRA training configuration
lora_config = LoraConfig(
    r = 32,  # rank
    target_modules = ["q", "v"],
    lora_alpha = 32,
    lora_dropout = 0.05,
    bias = 'none',
    task_type = TaskType.SEQ_2_SEQ_LM  # for Flan-T5
)
```

In [85]:
```
# add LoRA adapter to the original LLM
peft_model = get_peft_model(original_model, lora_config)
print(print_number_of_trainable_model_parameters(peft_model))
```

```
Trainable model paramters: 3538944
All model paramters: 251116800
Percentage of trainables: 1.41 %
```

Training related parameters such as learning rate and epoch numbers were defined using the *`transformers.TrainingArguments`* library, shown in the code snippet below. From the current setting, the best model, based on loss of validation data, was saved at

the last checkpoint. Relatively small learning rate and a large number of epochs were used allowing longer training and better models arrived.

In [86]:
```
# define training arguments
out_dir = f'./peft-tesla-classification-training-{str(int(time.time()))}'

peft_training_args = TrainingArguments(
    output_dir = out_dir,
    evaluation_strategy = 'epoch',
    logging_strategy = 'epoch',
    learning_rate = 1e-4,
    num_train_epochs = 20,
    save_strategy = "epoch",
    load_best_model_at_end = True,
    metric_for_best_model = 'eval_loss',
    greater_is_better = False,
    per_device_train_batch_size = 1,
    per_device_eval_batch_size = 1
)
```

The trainer object was defined as shown in the code snippet below.

In [87]:
```
# define trainer
peft_trainer = Trainer(
    model = peft_model,
    args = peft_training_args,
    train_dataset = traindata,
    eval_dataset = valdata
)
```

The training was started by calling the .train() method of the trainer object. The process is shown in the code snippet below together with the training logs. In the first few epochs, both training loss and validation loss dropped quickly. After 10 epochs, the reduction on those metrics became slow. A cuda GPU with ~15GB memory was used for this training process, which took around 14 minutes to finish.

```
In [88]: # start training
         peft_trainer.train()
```

[2880/2880 14:21, Epoch 20/20]

| Epoch | Training Loss | Validation Loss |
|---|---|---|
| 1 | 30.759100 | 4.750241 |
| 2 | 4.414000 | 1.810738 |
| 3 | 1.583900 | 0.268391 |
| 4 | 0.470500 | 0.105635 |
| 5 | 0.236800 | 0.061598 |
| 6 | 0.154500 | 0.044009 |
| 7 | 0.116400 | 0.035154 |
| 8 | 0.091400 | 0.027932 |
| 9 | 0.077900 | 0.026098 |
| 10 | 0.068900 | 0.023801 |
| 11 | 0.061100 | 0.021528 |
| 12 | 0.056600 | 0.020919 |
| 13 | 0.052700 | 0.019945 |
| 14 | 0.050600 | 0.019461 |
| 15 | 0.048000 | 0.018610 |
| 16 | 0.046900 | 0.018168 |
| 17 | 0.045100 | 0.018195 |
| 18 | 0.044300 | 0.017663 |
| 19 | 0.042800 | 0.017523 |
| 20 | 0.042200 | 0.017329 |

## Model Evaluation

The best model from LoRA was the last epoch output from the logs and there were 20 checkpoints saved. However, the saved checkpoints were adapters for the base Flan-T5 model. The following code snippet showed how to combine the best adapter with the base model.

The `*peft.PeftModel*` library was used to create the updated model using the saved adapter and the base Flan-T5 model. Then the new model (peft_1127ml) was used to generate classification outputs for all data sets, train, validation and test. Then compared to the manual labels.

```
In [89]:
# save the best model
peft_model_path = './peft-tesla-classification-checkpoint-1127'

peft_trainer.model.save_pretrained(peft_model_path)
tokenizer.save_pretrained(peft_model_path)
```

```
Out[89]: ('./peft-tesla-classification-checkpoint-1127/tokenizer_config.json',
 './peft-tesla-classification-checkpoint-1127/special_tokens_map.json',
 './peft-tesla-classification-checkpoint-1127/tokenizer.json')
```

```
In [90]:
# reload the original model
model_base = AutoModelForSeq2SeqLM.from_pretrained(model_name, torch_dtype = torch.bfloat16)
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

```
In [92]:
# get the trained model
peft_1127ml = PeftModel.from_pretrained(
    peft_model_base,
    './peft-tesla-classification-checkpoint-1127/',
    torch_dtype = torch.bfloat16,
    is_trainable = False
)
```

Unfortunately, the fine-tuned model didn't provide reasonable outputs for this classification task. The majority of the outputs from the best model were all positives, Figure 7. The reason could be related to the very limited number of data provided for training. The Flan-T5 model was developed for text generative purposes. In order to make it work for classification tasks, more data is required for training. Further hyperparameter tuning might not be very helpful for this case.

```
df_train['manual_baseline'].value_counts()

Yes, it is directly related to Tesla.      82
No, it is not directly related to Tesla    62
Name: manual_baseline, dtype: int64

df_train['peft_1127ml_output'].value_counts()

Yes, it is directly related to Tesla.      143
No, it is not directly related to Tesla      1
Name: peft_1127ml_output, dtype: int64
```

```
df_val['manual_baseline'].value_counts()

Yes, it is directly related to Tesla.      17
No, it is not directly related to Tesla    15
Name: manual_baseline, dtype: int64

df_val['peft_1127ml_output'].value_counts()

Yes, it is directly related to Tesla.      31
No, it is not directly related to Tesla     1
Name: peft_1127ml_output, dtype: int64
```

```
df_test['manual_baseline'].value_counts()

Yes, it is directly related to Tesla.      17
No, it is not directly related to Tesla    14
Name: manual_baseline, dtype: int64

df_test['peft_1127ml_output'].value_counts()

Yes, it is directly related to Tesla.    31
Name: peft_1127ml_output, dtype: int64
```

Figure 7. Comparison between Manual Labels and PEFT (LoRA) Model Outputs

## Summary

With a limited number of available data, prompt engineering with one-shot learning was the best choice for this news classification task with the Flan-T5 model. FEFT with LoRA didn't work well for Flan-T5 to perform this text classification task due to the limited number of data for training. Prompt engineering was more promising with an average 85% accuracy when compared to manual labels. Hallucination issues were

observed when the number of examples in the prompt was increased. The model started to confuse with the task and generated irrelevant outputs.

One-shot learning with Flan-T5 was chosen to classify all the 304 news collected and used for further analysis. Figure 8 showed the one-shot learning predictions.

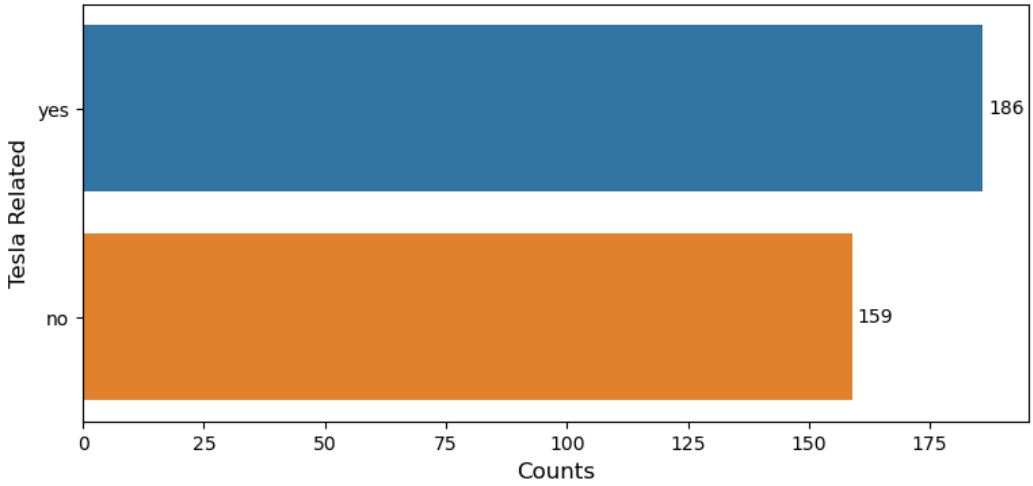



Figure 8. All News Classification Distribution

## Sentiment Analysis

### Data Pre-processing

No additional prompt engineering was applied for this sentiment analysis task. The news title, description, content and title+description were used for sentiment analysis, respectively.

### Modeling

Pre-trained FinBERT model was used to perform the sentiment analysis on all news collected. The `*transformers*` library was used to load the model and its corresponding tokenizer. The code snippet below showed how to load them and to create a pipeline for sentiment analysis. An example of the model output was also provided. The FinBERT

model directly gave the sentiment label for the provided text content together with a confidence score associated with the sentiment prediction.

```
# choose finbert model which is a pre-trained LL model to analyze sentiment of financial text
model_name = "ProsusAI/finbert"

tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForSequenceClassification.from_pretrained(model_name)

# create a pipeline for sentiment analysis
sent_ana = pipeline(task = 'sentiment-analysis', model = model, tokenizer = tokenizer)

# test 1
text = 'Tesla just launched its first self-driving car!'

result = sent_ana(text)
result
```

```
[{'label': 'neutral', 'score': 0.8160327672958374}]
```

## Evaluation

In order to evaluate the sentiment analysis performance, manual sentiment labels were generated for the randomly selected 60% of the news. Sentiment outputs from the FinBERT model were compared to manual labels and the confusion matrix was computed for evaluation.

Multiple sets of sentiment outputs were generated based on different contexts, title, description, content and title+description. Another set of sentiment labels were generated by selecting the one with the highest score from the four above. After comparing the confusion matrix for the 5 different sets of outputs with manual labels, the sentiment labels predicted with description matched the best with the manual labels.

Figure 9 showed the confusion matrices computed between manual labels and the FinBERT predictions based on news description, for all news and for news directly related to Tesla, respectively. The FinBERT model tended to label positive news as neutral news for all cases. FinBERT was more on the conservative side when labeling a news as positive news.

After checking some of the examples for which the FinBERT labeled as neutral but whose manual labels were positive, the FinBERT model performed the sentiment

analysis purely on words used in the content. Potential inference used during manual labeling was not possible for FinBERT. For example, there was news about other EV makers adopting the charging system from Tesla. The FinBERT model marked those as neutral news. However, such news indicated a large market share of Tesla in the EV charging industry and had potential positive benefits on Tesla. This news was marked as positive in the manual labels.

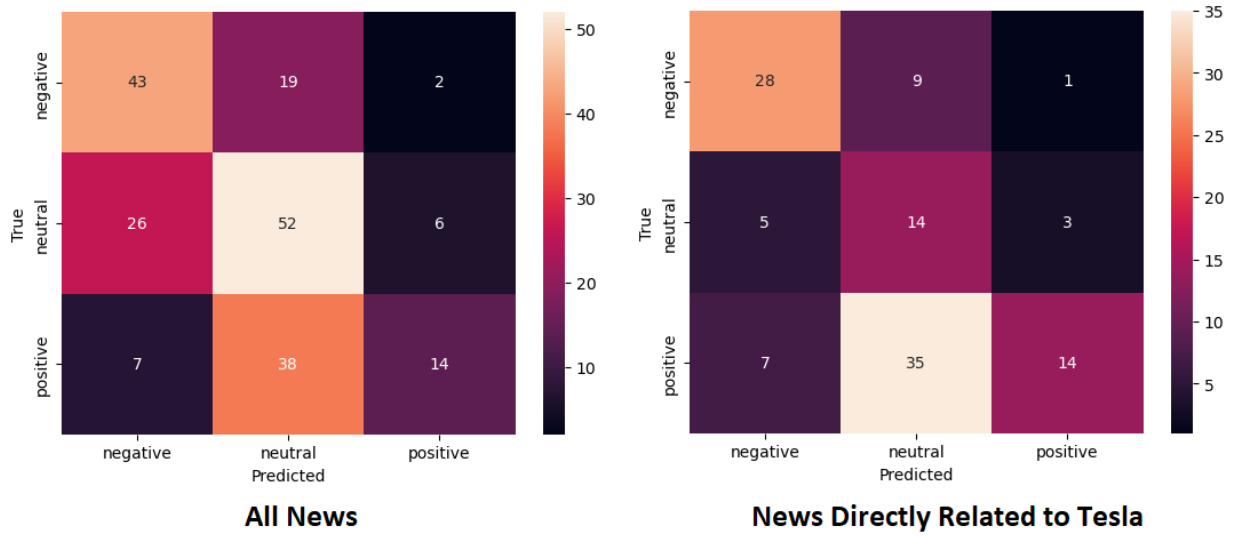



Figure 9: Confusion Matrix of FinBERT Sentiments from Description and Manual Sentiments for All News (left) and News Directly Related to Tesla (right).

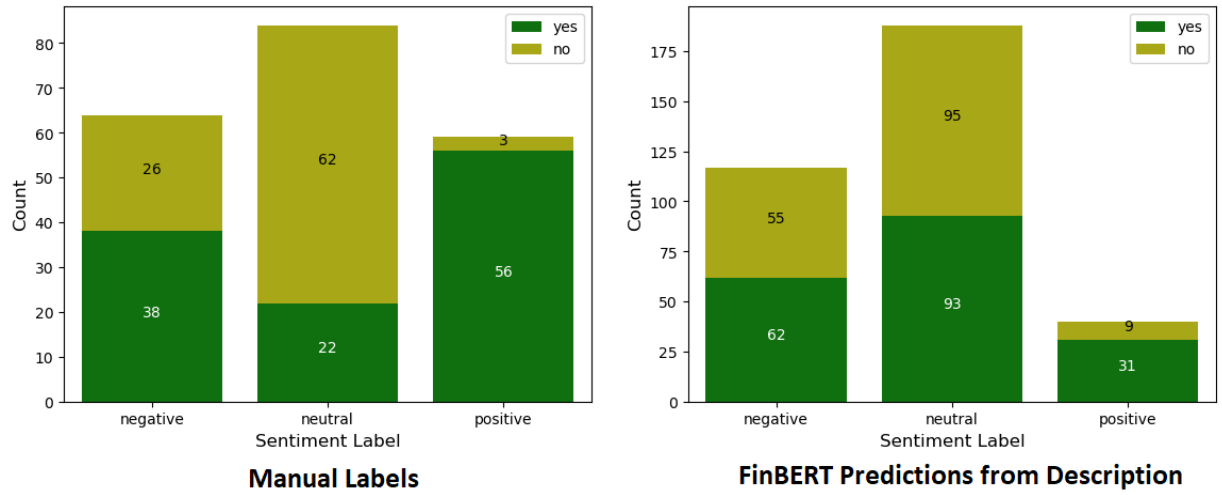



Figure 10. Sentiments Distributions of Manual Labels (left) and FinBERT Labels (right)

The FinBERT sentiment output was chosen for the next step analysis, correlation between Tesla news sentiments and its stock price movements. Though the FinBERT model might not be perfect compared to human intelligence on this type of complex NLP task. To further improve the model performance or even develop a better tool for this particular task was not investigated due to data and time limit of the current work.

Figure 11 showed the sentiment polarity (positive news rate - negative news rate) of different publishers from the FinBERT predictions. Note that the neutral news (dominant in the data) was not counted for this sentiment polarity.

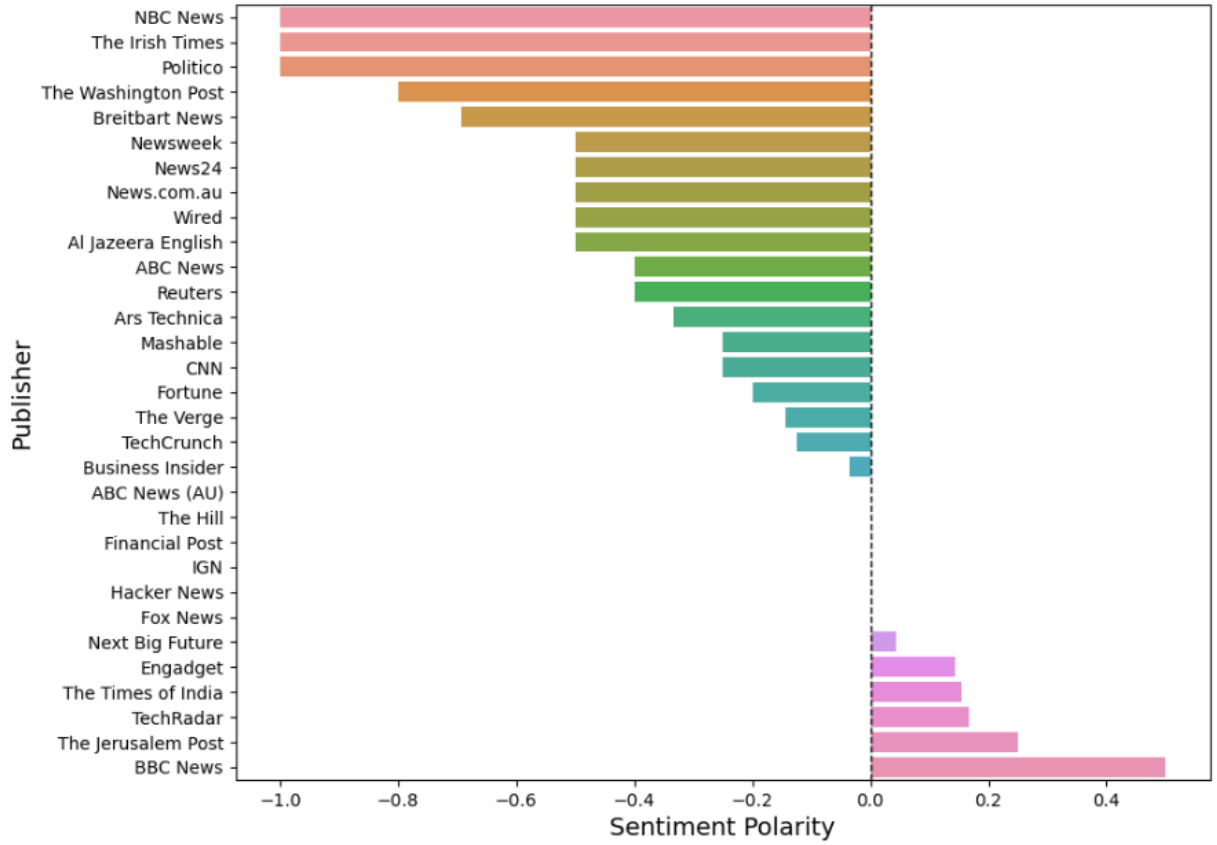



Figure 11. Sentiment Polarity of Publishers from the FinBERT Predictions

Figure 12 showed the positive news rate and the negative news rate of different publishers. The majority of publishers were more on the neutral side. There were some publishers that tended to report more negative Tesla news, such as NBC News, The Irish Times, Politico, The Washington Post and Breitbart News. Some of the publishers above are big names, such as NBC News and The Washington Post that could have a

larger impact on public opinion on how Tesla is doing. There were publishers that tended to report more positive Tesla news, such as Next Big Future, Engadget, The Times of India, TechRadar, The Jerusalem Post and BBC News. This trend suggests that technology-focused publishers generally maintain a positive stance towards Tesla. Major publishers from specific countries appeared to be more positive in their coverage of Tesla, which could indicate a favorable disposition towards providing Tesla with better opportunities.

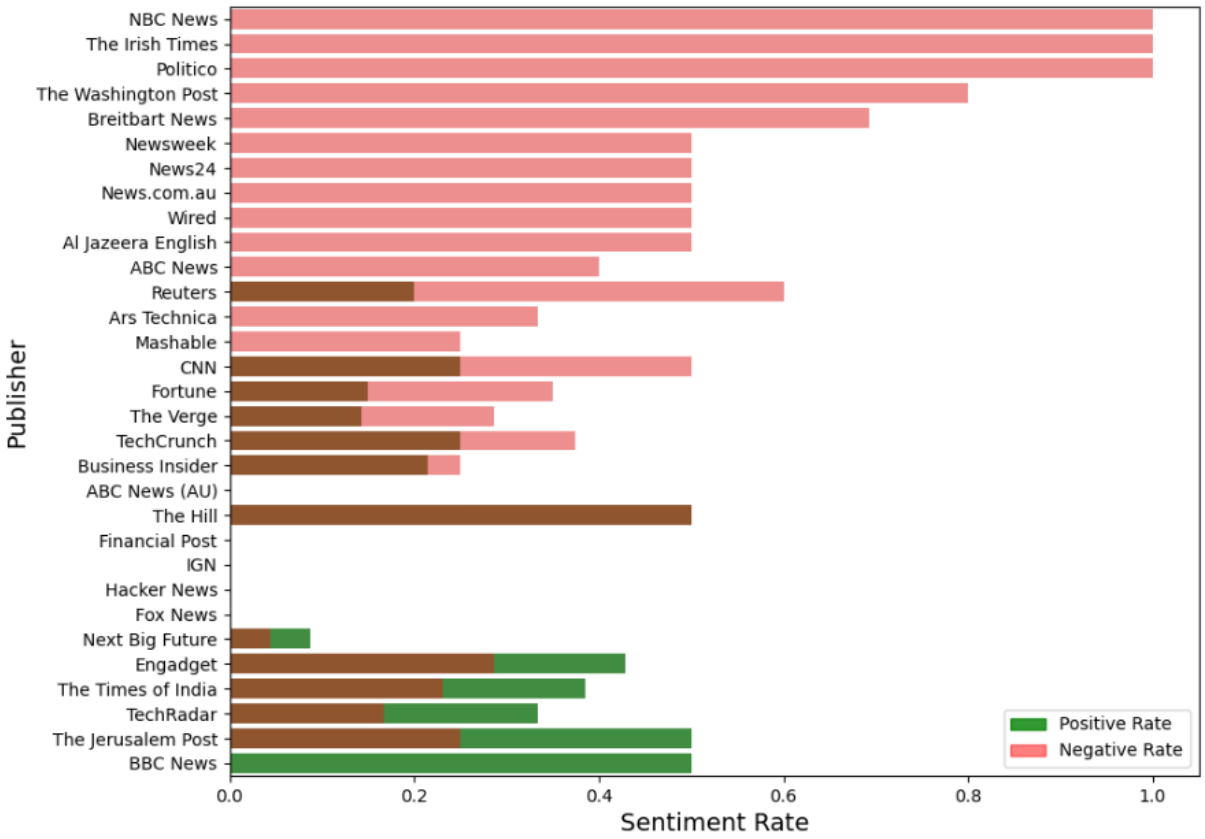



Figure 12. Positive and Negative Rates of Publishers from the FinBERT Predictions

## Correlation with Stock Price

Tesla's stock price data were collected from the API and its close price change compared to yesterday was computed. The stock related data were time-series data.

The sentiment analysis results required to be modified to be a time-series data as well in order to investigate its correlation with the stock price data.

The number of positive, neutral and negative news and their corresponding rate were calculated for each date. Then they were merged with the daily open price, close price, close - open, close price change compared to yesterday, 2 example rows shown in Table 1.

| | date | positive_num | negative_num | nuetral_num | positive_rate | negative_rate | neutral_rate | open | close | inday_move | yesterday_move |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2023-09-12 | 3 | 2 | 16 | 0.142857 | 0.095238 | 0.761905 | 270.76 | 267.48 | -3.28 | -6.10 |
| **1** | 2023-09-13 | 2 | 5 | 21 | 0.071429 | 0.178571 | 0.750000 | 270.07 | 271.30 | 1.23 | 3.82 |

Table 1. First 2 Rows of Daily Sentiment and Stock Data

### Open Price *VS* Number of Positive News

Correlation map was generated for sentiment features and stock features. A high correlation coefficient was found between the open price and the number of positive news. Those two features were plotted on Figure 13. Figure 13 only included news sentiments directly related to Tesla. No obvious correlation was found from the plot.

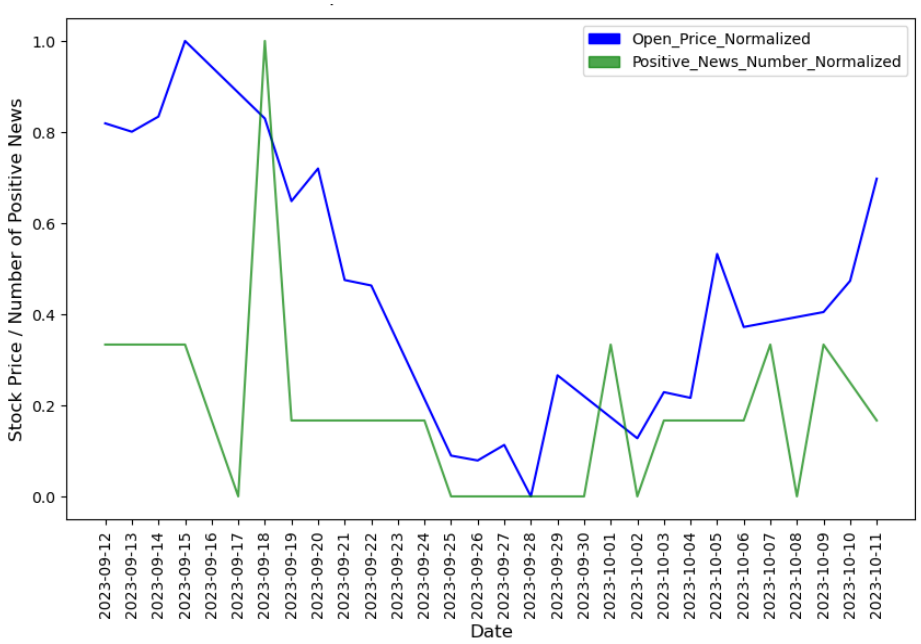



Figure 13. Tesla Open Price VS Number of Positive News Directly Related to Tesla

Dominant News Sentiments (Neutral Ignored) VS Stock Price Move

Figure 14 showed the comparison of close price change of yesterday and dominant news sentiments (positive - negative) from yesterday for news directly related to Tesla. If the green bar had the same polarity with the gray bar, then the two correlated well with each other, indicating that the positive dominant news sentiments had a positive impact on the Tesla stock price of the next day and vice versa.

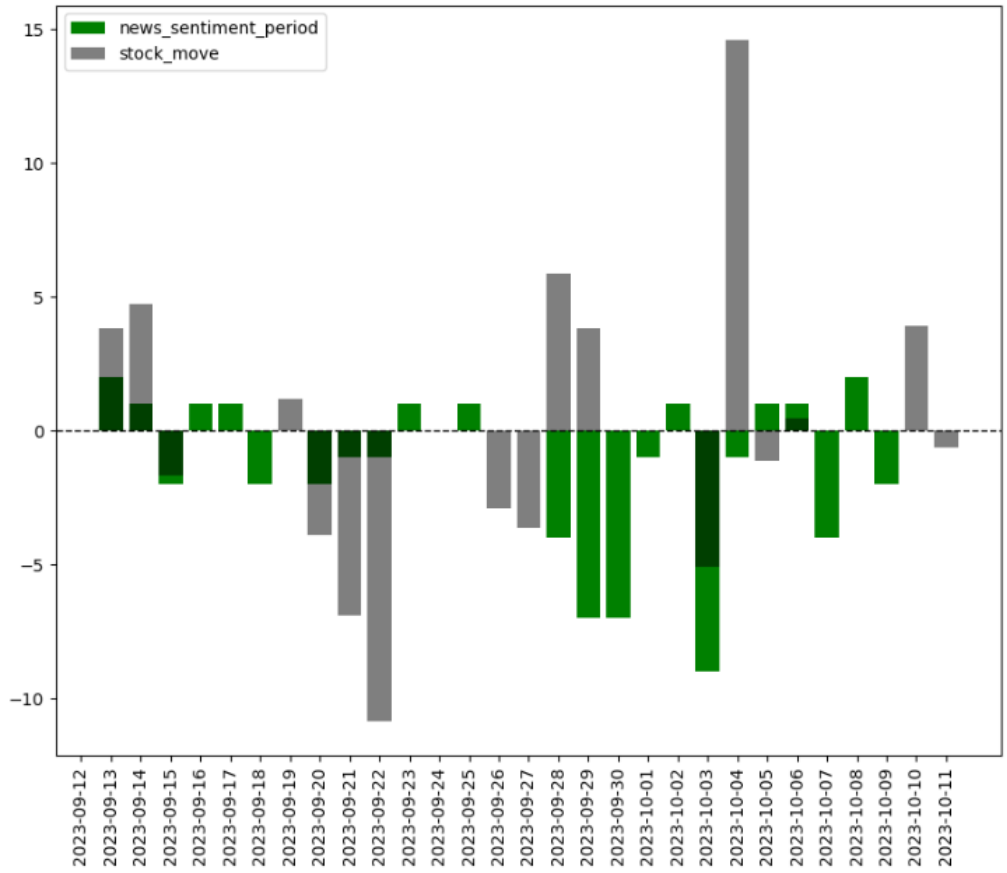



Figure 14. Tesla Stock Close Price Change from Yesterday VS Positive News - Negative News from Yesterday

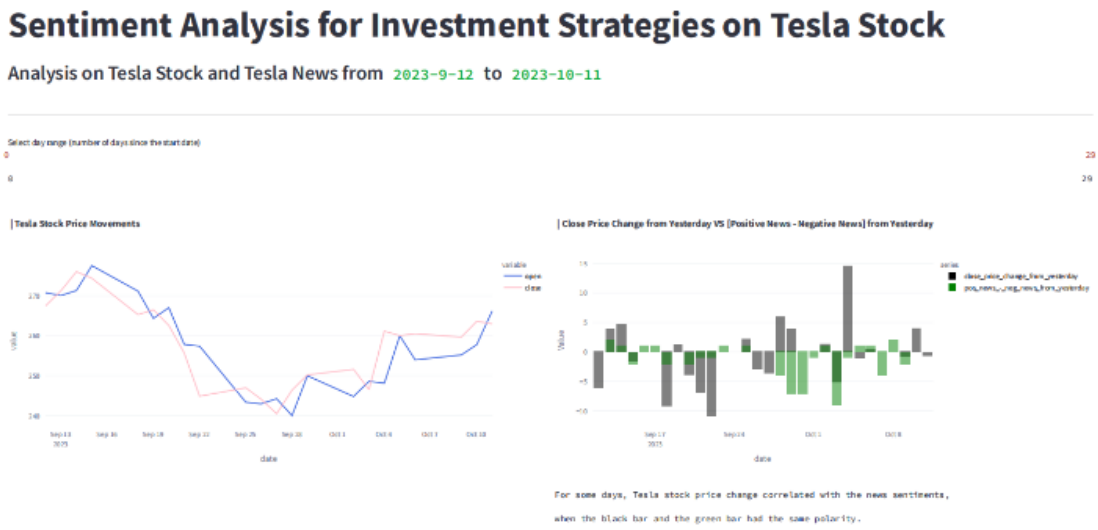
Besides studying the impact of sentiments from yesterday on the close price change, sentiments from a longer time were also investigated. The negative sentiments were exaggerated by averaging impact from more previous days due to the fact that there was more negative news than positive ones in the past 30 days.

The stock price movements correlates with the previous-day news sentiment for some days. However, there was no clear correlation between the stock price change with the dominant news sentiments. There were multiple reasons why this happened. The sentiment results were generated by using the FinBERT model which might not reflect the true sentiments. It labeled positive news as neutral ones, which led to more negative news dominated days in the analysis. The time period was only 30 days which was very limited. In general, stock price movement predictions are difficult. The market is volatile due to a wide variety of reasons. News sentiments are only one part of all different factors that play a role here.

## Deploy to Streamlit

The sentiment analysis and the stock price data analysis were deployed to `*streamlit.sharing*` via interactive dashboards. The dashboards are open to the public.

The script for deployment can be found in this GitHub repo. Screen shots of the dashboards are shown below.




## Takeaways

Here are the key takeaways for this work:

- With help of prompt engineering and fine-tuning, we can facilitate open-source LLMs to work on a particular task of interest.
- Prompt engineering works better than fine-tuning when data is limited.
- Hallucination issues can happen when including more examples in the prompt.
- The choice of words are more critical than the number of words in prompt engineering.
- PEFT with LoRA significantly reduces the cost of fine-tuning a LLM.
- The majority of news related to Tesla were neutral ones.
- Tech publishers were more on the positive side when reporting on Tesla.
- No clear correlation was found between news sentiments and Tesla stock price movements.
- The analysis results were deployed to streamlit.sharing via interactive dashboards.

## Limitations

### LLM Performance

The news classification was not perfect. The accuracy was around 0.85 for the Flan-T5 model with in-context learning technique applied. Further improvements can be achieved by collecting more data and investigating other possible LLMs for this task.

The sentiment analysis from the FinBERT model didn't match the manual labels well. The FinBERT model tended to label positive news as neutral ones. It was pre-trained using a financial corpus for sentiment analysis based on BERT. The manual labeling paid more attention to potential impact on Tesla stock price, which is challenging to the FinBERT model.

The NLP task here is no longer simple sentiment analysis on a piece of context. It requires a tool to predict the potential impact of news on Tesla's stock price with help from LLMs. More data is required to generate such a tool through training existing LLMs.

### Stock Market Prediction Challenge

There are multiple and complex factors that will impact the stock price of a company. Market-related factors such as GDP, interest rates, inflation, employment rate, indicate how well the whole market is performing and they will have a significant impact on individual stocks. Company reports showing how the company performed in the past period of time is important to its stock price. Earnings, dividends, debts, management qualities etc. are key components to drive the stock price movement. Other factors such as global economy, regulations and government policies, industry-specific changes and natural disasters and pandemics all play a role here. Without considering all the potential impact factors, it is hard to accurately predict the stock movements.

## Next Steps

### Data

Collecting more data for a longer period of time will help the followings:

- Fine-tuning LLMs for classification and sentiment analysis
- Checking if there is any long-term trend between the news sentiments and the stock price movement.

### Modeling

Other LLMs are worth investigating to see if there is any better performance compared to the current model:

- Llama-2 from Meta (open-source)
- GPT4 from OpenAI