Problem Statement

Business Context

The prices of the stocks of companies listed under a global exchange are influenced by a variety of factors, with the company's financial performance, innovations and collaborations, and market sentiment being factors that play a significant role. News and media reports can rapidly affect investor perceptions and, consequently, stock prices in the highly competitive financial industry. With the sheer volume of news and opinions from a wide variety of sources, investors and financial analysts often struggle to stay updated and accurately interpret its impact on the market. As a result, investment firms need sophisticated tools to analyze market sentiment and integrate this information into their investment strategies.

Problem Definition

With an ever-rising number of news articles and opinions, an investment startup aims to leverage artificial intelligence to address the challenge of interpreting stock-related news and its impact on stock prices. They have collected historical daily news for a specific company listed under NASDAQ, along with data on its daily stock price and trade volumes.

As a member of the Data Science and AI team in the startup, you have been tasked with analyzing the data, developing an AI-driven sentiment analysis system that will automatically process and analyze news articles to gauge market sentiment, and summarizing the news at a weekly level to enhance the accuracy of their stock price predictions and optimize investment strategies. This will empower their financial analysts with actionable insights, leading to more informed investment decisions and improved client outcomes.

Data Dictionary

- `Date` : The date the news was released
- `News` : The content of news articles that could potentially affect the company's stock price
- `Open` : The stock price (in $) at the beginning of the day
- `High` : The highest stock price (in $) reached during the day
- `Low` : The lowest stock price (in $) reached during the day
- `Close` : The adjusted stock price (in $) at the end of the day
- `Volume` : The number of shares traded during the day
- `Label` : The sentiment polarity of the news content
    - 1: positive
    - 0: neutral
    - -1: negative

Installing and Importing Necessary Libraries

In [ ]:
```
# installing the sentence-transformers and gensim libraries for word embeddings
!pip install -U sentence-transformers gensim transformers tqdm -q
```

```
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.0/44.0 kB 4.0 MB/s eta 0:00:00
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.0/61.0 kB 5.8 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 10.0/10.0 MB 89.5 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.3/18.3 MB 106.1 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 2.8 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 111.1 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 89.5 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 62.0 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 1.8 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 10.9 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 39.0 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 18.0 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 3.2 MB/s eta 0:00:00
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 94.3 MB/s eta 0:00:00
```

In [ ]:
```
# To manipulate and analyze data
import pandas as pd
import numpy as np

# To visualize data
import matplotlib.pyplot as plt
import seaborn as sns

# To used time-related functions
import time

# To parse JSON data
import json
```

```
# For text processing
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt_tab')

# To build, tune, and evaluate ML models
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score, precision_score, recall_score

# To load/create word embeddings
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
from gensim.scripts.glove2word2vec import glove2word2vec

# To work with transformer models
import torch
from sentence_transformers import SentenceTransformer

# To implement progress bar related functionalities
from tqdm import tqdm
tqdm.pandas()

# To ignore unnecessary warnings
import warnings
warnings.filterwarnings('ignore')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
```

Loading the dataset

In [ ]:
```
# Mounting google drive and initilizing path variable
from google.colab import drive
drive.mount("/content/drive")
path = '/content/drive/MyDrive/PGPAIML/Project-6/'
```

```
Mounted at /content/drive
```

In [ ]:
```
# loading data into a pandas dataframe
original_data = pd.read_csv(path+'stock_news.csv') #delimiter="\t", encoding='utf-8')
```

In [ ]:
```
# creating a copy of the data
data = original_data.copy()
```

Data Overview

In [ ]:
```
data.head()
```

Out[ ]:

	Date	News	Open	High	Low	Close	Volume	Label
0	2019-01-02	The tech sector experienced a significant dec...	41.740002	42.244999	41.482498	40.246914	130672400	-1
1	2019-01-02	Apple lowered its fiscal Q1 revenue guidance ...	41.740002	42.244999	41.482498	40.246914	130672400	-1
2	2019-01-02	Apple cut its fiscal first quarter revenue fo...	41.740002	42.244999	41.482498	40.246914	130672400	-1
3	2019-01-02	This news article reports that yields on long...	41.740002	42.244999	41.482498	40.246914	130672400	-1
4	2019-01-02	Apple's revenue warning led to a decline in U...	41.740002	42.244999	41.482498	40.246914	130672400	-1

In [ ]:
```
data.shape
```

Out[ ]: (349, 8)

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 349 entries, 0 to 348
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    349 non-null    object
 1   News    349 non-null    object
 2   Open    349 non-null    float64
 3   High    349 non-null    float64
 4   Low     349 non-null    float64
 5   Close   349 non-null    float64
 6   Volume  349 non-null    int64
 7   Label   349 non-null    int64
dtypes: float64(4), int64(2), object(2)
memory usage: 21.9+ KB
```

Observation

- No missing values – All columns have 349 non-null entries.
- Data Types:
  - Date & News are object types.
  - Stock prices & Volume are numeric (float64 & int64).
  - Label is an integer (expected, since sentiment labels are -1, 0, and 1).
- **We need to convert Date as Object to a Date-Time datatype.**

In [ ]:
```
# Converting 'Date' column from Onject to panda's datetime data type
data['Date'] = pd.to_datetime(data['Date'])
```

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 349 entries, 0 to 348
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Date    349 non-null    datetime64[ns]
 1   News    349 non-null    object
 2   Open    349 non-null    float64
 3   High    349 non-null    float64
 4   Low     349 non-null    float64
 5   Close   349 non-null    float64
 6   Volume  349 non-null    int64
 7   Label   349 non-null    int64
dtypes: datetime64[ns](1), float64(4), int64(2), object(1)
memory usage: 21.9+ KB
```

In [ ]:
```
data.describe(include='all').T
```

Out[ ]:

	count	unique	top	freq	mean	min	25%	50%	75%	max	
Date	349	NaN	NaN	NaN	2019-02-16 16:05:30.085959936	2019-01-02 00:00:00	2019-01-14 00:00:00	2019-02-05 00:00:00	2019-03-22 00:00:00	2019-04-30 00:00:00	
News	349	349	In the first quarter, South Korea's Samsung E...	1	NaN	NaN	NaN	NaN	NaN	NaN	
Open	349.0	NaN	NaN	NaN	46.229233	37.567501	41.740002	45.974998	50.7075	66.817497	
High	349.0	NaN	NaN	NaN	46.700458	37.817501	42.244999	46.025002	50.849998	67.0625	
Low	349.0	NaN	NaN	NaN	45.745394	37.305	41.482498	45.639999	49.7775	65.862503	
Close	349.0	NaN	NaN	NaN	44.926317	36.254131	40.246914	44.596924	49.11079	64.805229	
Volume	349.0	NaN	NaN	NaN	128948236.103152	45448000.0	103272000.0	115627200.0	151125200.0	244439200.0	4317
Label	349.0	NaN	NaN	NaN	-0.054441	-1.0	-1.0	0.0	0.0	1.0	

In [ ]:
```
# Creating new dataframe to calculate Price difference in a given data. This is only for analysis price differe
df = pd.DataFrame()
df["Price_Diff"] = data["High"] - data["Low"]
df.describe().T
```

Out[ ]:

	count	mean	std	min	25%	50%	75%	max
Price_Diff	349.0	0.955064	0.485368	0.357498	0.564998	0.837502	1.129997	2.822499

In [ ]:
```
# Checking for missimg values
data.isnull().sum()
```

Out[ ]:

	0
Date	0
News	0
Open	0
High	0
Low	0
Close	0
Volume	0
Label	0

dtype: int64

In [ ]:
```
# Checking for duplicate values
data.duplicated().sum()
```

Out[ ]: 0

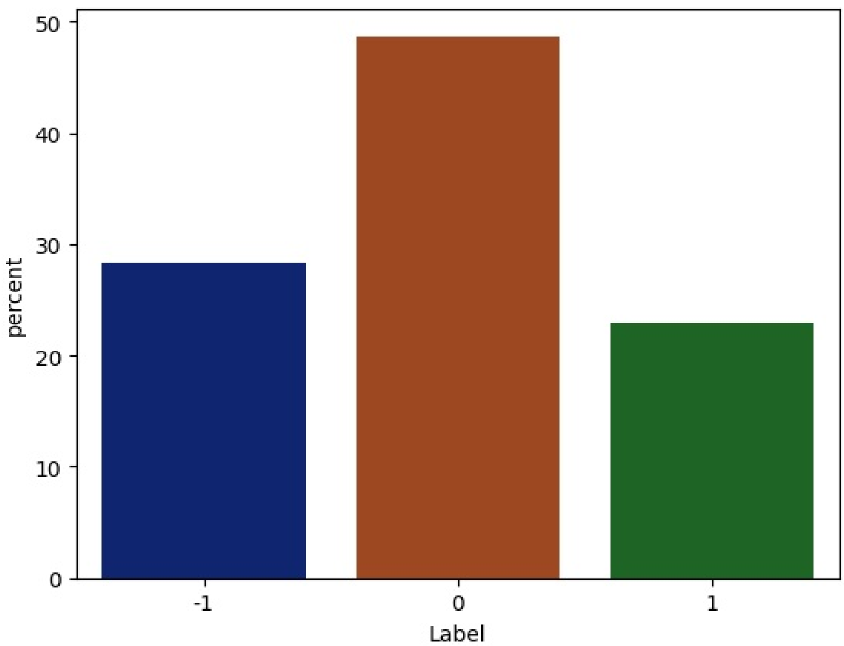
Observations

- Stock Price Trends
  - Opening Price - Mean $46.23, Min: $37.57, Max: $66.81.
  - Closing Price - Mean: $44.92, Min: $36.25, Max: $64.80.
- Variation in prices indicate volatility.
- Price Difference (High - Low): $46.70, Mean low: $45.74.
- Trade Volume: Mean 128.9M shares, Min 45.4M, Max 244.4M.
- High trading volume indication of market interest.
- Sentiment: Standard deviation 0.71, (mix of sentiments or positive sentiment?).
- No null or duplicate values.

Exploratory Data Analysis

Univariate Analysis

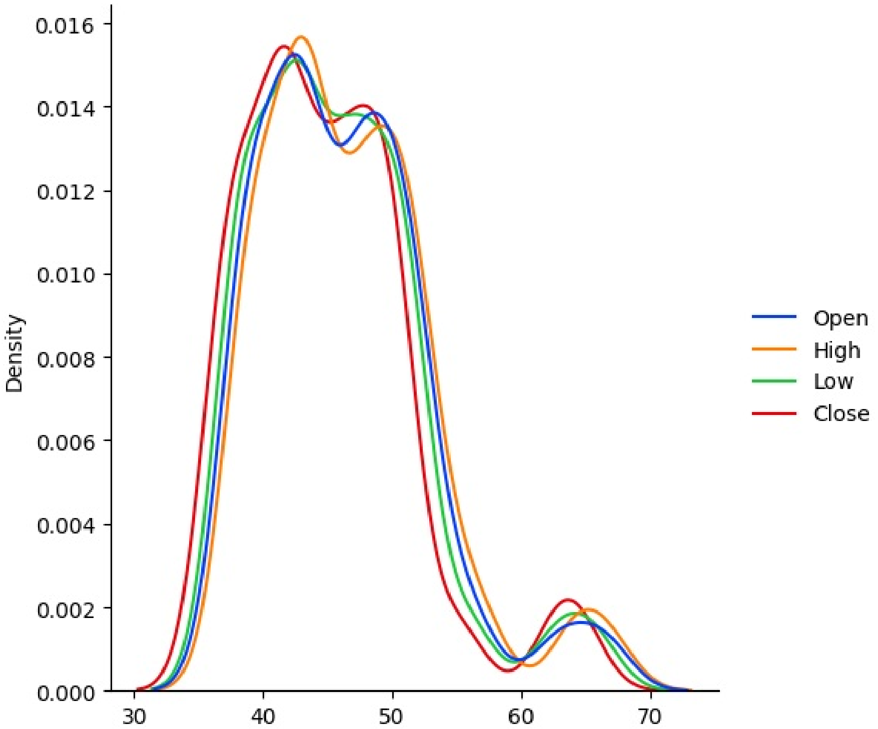
In [ ]:
```
sns.countplot(data=data, x='Label', stat='percent', palette='dark');
```



- Distribution of individual variables

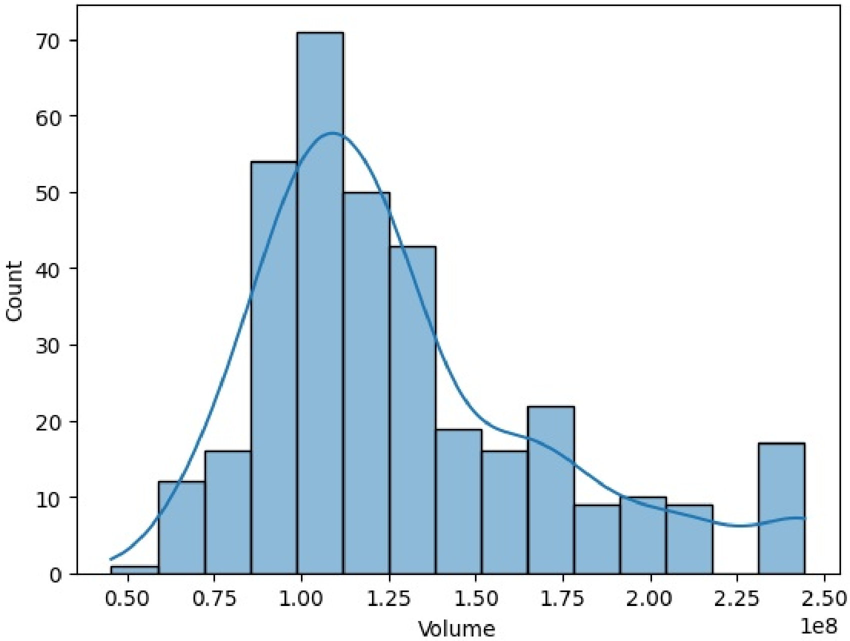
In [ ]:
```
sns.displot(data=data[['Open','High','Low','Close']], kind="kde", palette="bright")
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7ec120f822d0>

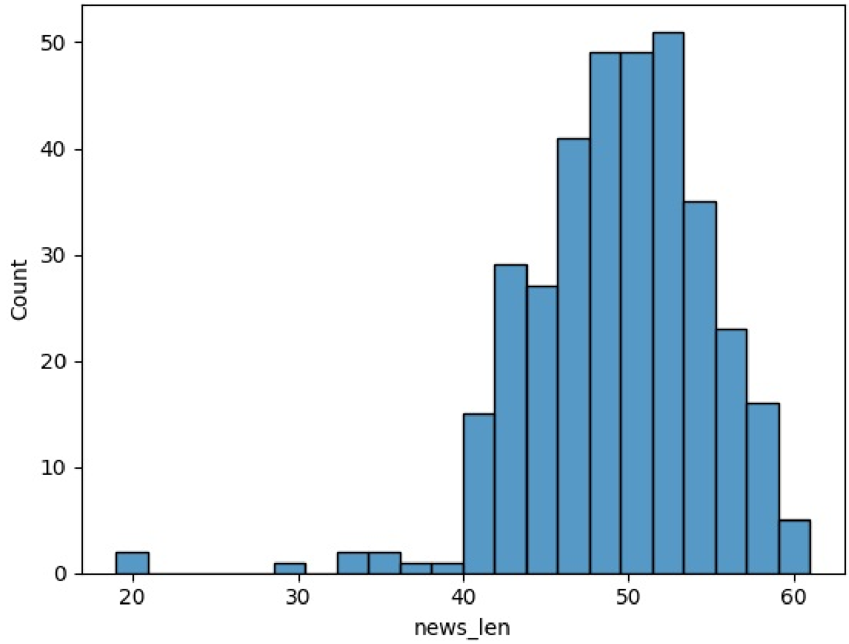


In [ ]:
```
sns.histplot(data, x='Volume', kde=True, palette='dark')
```

Out[ ]: <Axes: xlabel='Volume', ylabel='Count'>



In [ ]:
```
data['news_len'] = data['News'].apply(lambda x: len(x.split(' ')))
sns.histplot(data, x='news_len');
```



Observations

1. Sentiment Distribution

- Neutral sentiment (less than 50%) dominates, followed by negative (around 30%) and positive (around 20%).
- Indicates potential **class imbalance**, which might require handling.
- The lower proportion of **positive** news articles suggests fewer instances for learning positive sentiment.

2. Stock Price Distribution

- Open, High, Low, and Close prices follow a similar distribution.
- Prices peak around $40 - $50.

3. Trade Volume Distribution

- Trade volume follows a slightly right-skewed distribution.
- Most trading volumes are concentrated around 1.0 - 1.5.

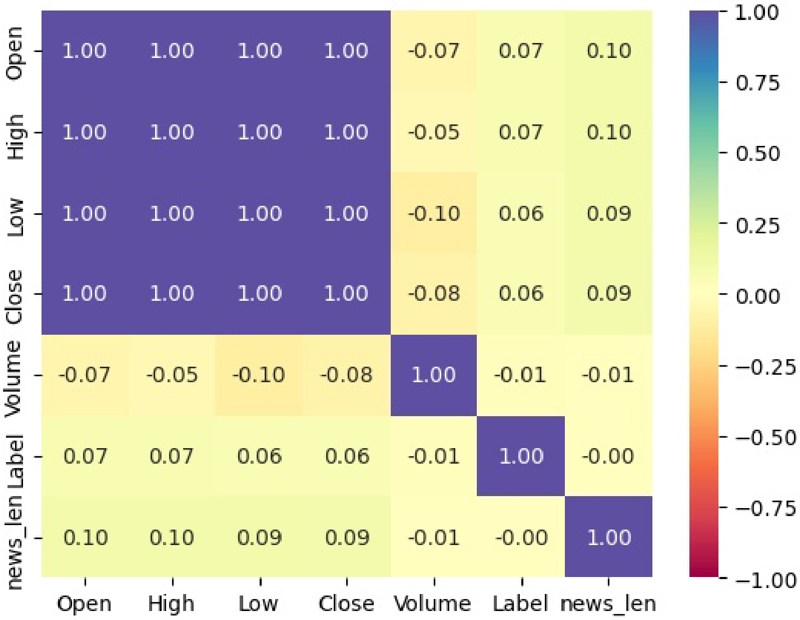
4. News Length Distribution

- The majority of news articles have 40-55 words.
- There are very few small news articles (~20 words).
- News length is relatively consistent.

Bivariate Analysis

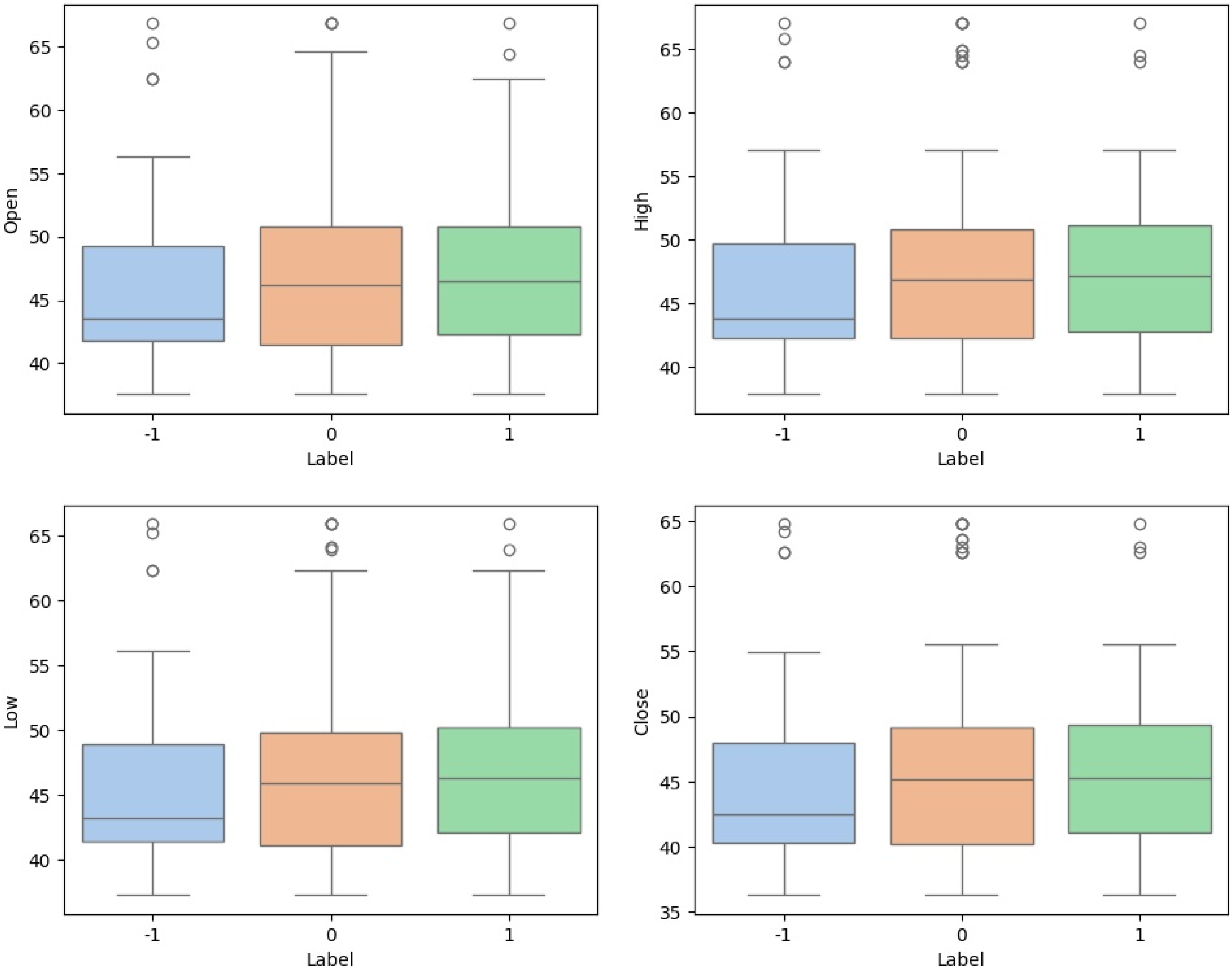
In [ ]:

```
# Heatmap for all numerical columns
sns.heatmap(
    data.select_dtypes(include=['number']).corr(), annot=True, vmin=-1, vmax=1, fmt=".2f", cmap="Spectral"
);
```



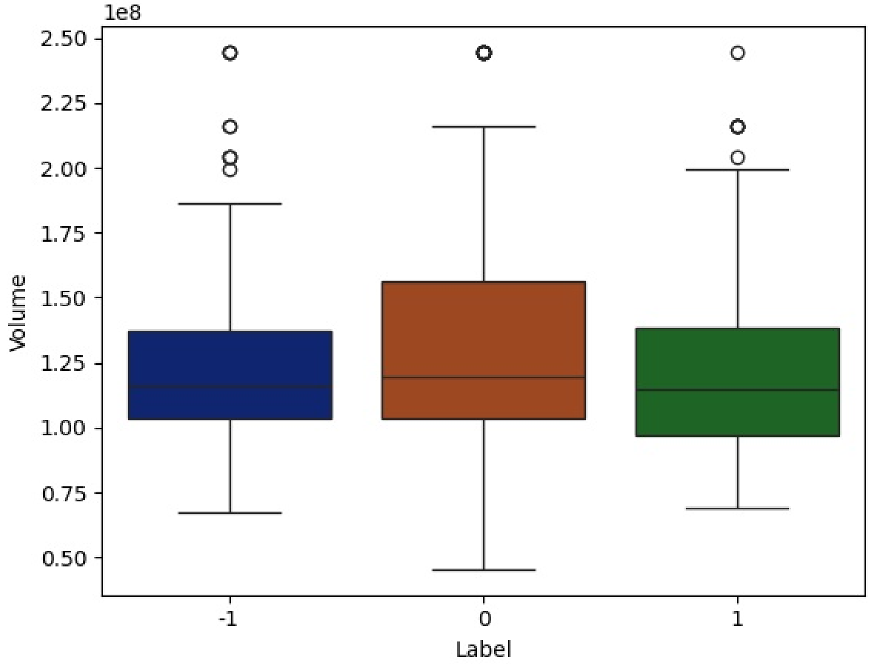
In [ ]:

```python
# Box plot for days stock prices by target sentiment
plt.figure(figsize=(10, 8))

for i, variable in enumerate(['Open', 'High', 'Low', 'Close']):
    plt.subplot(2, 2, i + 1)
    sns.boxplot(data=data, x="Label", y=variable, palette='pastel')
    plt.tight_layout(pad=2)

plt.show()
```



In [ ]:

```python
# Boxplot of trade volume by setiment
sns.boxplot(
    data=data, x='Label', y='Volume', palette='dark'
);
```

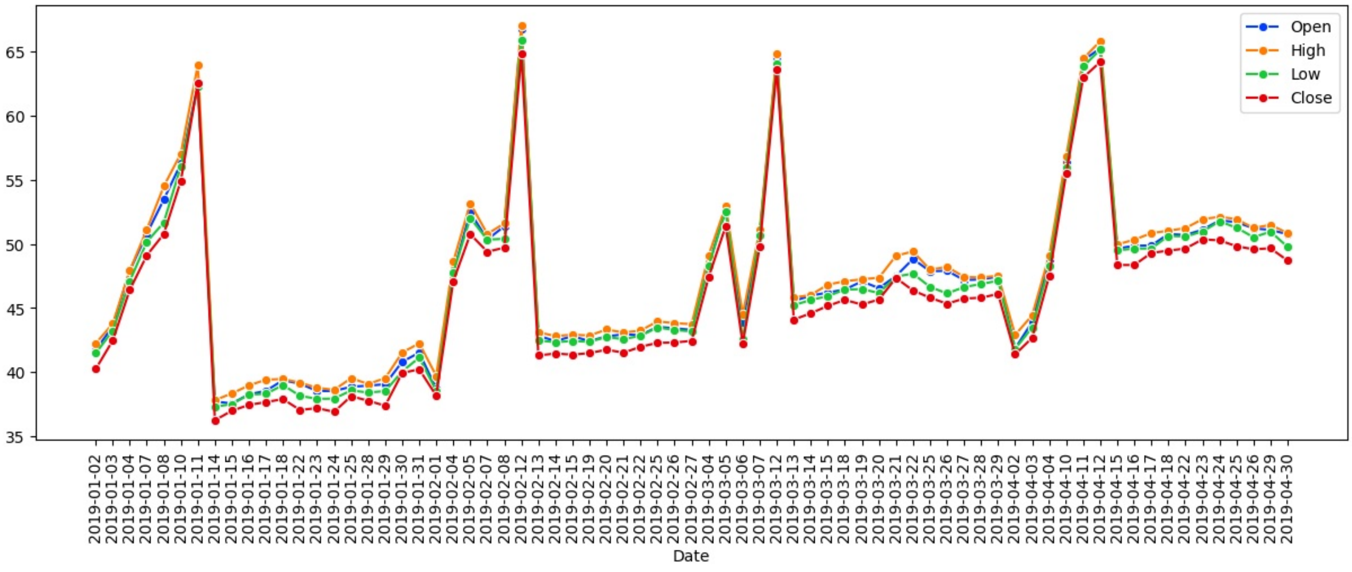


In [ ]:
```
# Creating new data-set for stock prices by date, taking mean value of Open, Close, High & Lonw. Also mean of Vo
stock_daily = data.groupby('Date').agg(
    {
        'Open': 'mean',
        'High': 'mean',
        'Low': 'mean',
        'Close': 'mean',
        'Volume': 'mean',
    }
).reset_index()

stock_daily.set_index('Date', inplace=True)
stock_daily.head()
```

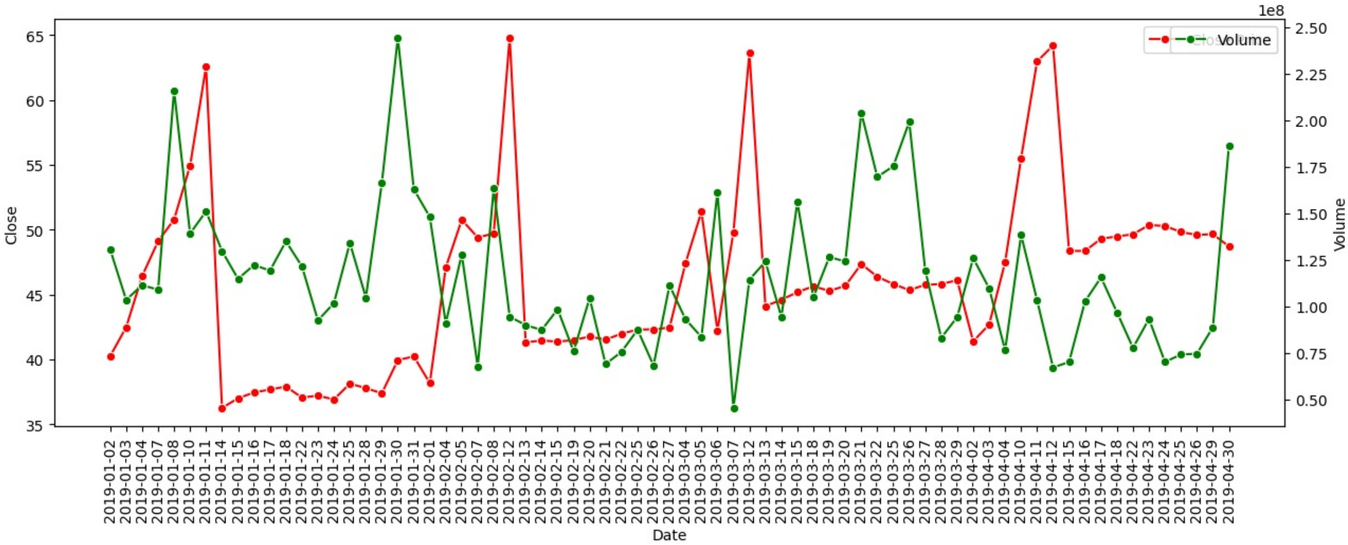
Out[ ]:

Date	Open	High	Low	Close	Volume
2019-01-02	41.740002	42.244999	41.482498	40.246914	130672400.0
2019-01-03	43.570000	43.787498	43.222500	42.470604	103544800.0
2019-01-04	47.910000	47.919998	47.095001	46.419842	111448000.0
2019-01-07	50.792500	51.122501	50.162498	49.110790	109012000.0
2019-01-08	53.474998	54.507500	51.685001	50.787209	216071600.0

In [ ]:
```
# Creating line plot for days mean / average prices for each given day.
plt.figure(figsize=(15,5))
sns.lineplot(stock_daily.drop('Volume', axis=1), palette='bright', dashes=False, marker='o');
plt.xticks(rotation=90)
plt.show()
```



In [ ]:
```
# Creating line plot for closing price & trade volum by day to visualize if they are related or not.
fig, ax1 = plt.subplots(figsize=(15,5))
sns.lineplot(data=stock_daily.reset_index(), x='Date', y='Close', ax=ax1, color='red', marker='o', label='Close
plt.xticks(rotation=90)
ax2 = ax1.twinx()
sns.lineplot(data=stock_daily.reset_index(), x='Date', y='Volume', ax=ax2, color='green', marker='o', label='Vo
plt.xticks(rotation=90)
ax1.legend(bbox_to_anchor=(1,1));
plt.show()
```



Observations

1. Correlation Heatmap

- Stock prices (Open, High, Low, Close) are highly correlated **(1.00)**, which is expected behaviour.
- Sentiment has weak correlations with stock prices (0.07 - -0.1). This seems to be indicative, that sentiment (alone) may not have a strong effect on price.
- News Length and Stock Prices havery very weak correlation (0.09). This could be a random effect.

2. Sentiment vs. Stock Prices

- There is no clear separation between different sentiment categories. The median values for all price points are quite similar across positive, neutral, and negative sentiment labels.
- Some **outliers** exist, this might be indication of other market factors.

3. Sentiment vs. Trade Volume

- Trade volume distribution is similar across sentiments.
- Outliers exist for all sentiment categories, suggests high trading activity on some specific days.

4. Stock Price Trends by Day

- The Open, High, Low, and Close prices move in parallel.
- There exists several sharp spikes and drops, days with high volatility.

5. Stock Close Price vs. Volume by Day

- Some high trading volume coincide with major price changes.
- There are periods where volume increases but price remains stable.

Data Preprocessing

In [ ]:
```
data['Date'].describe().T
```

Out[ ]:

	Date
count	349
unique	71
top	2019-01-03
freq	28

dtype: object

Observations

Since we are working with time-series data, we should split the dataset in a chronological order:

- Training Set (70%) → Used to train the model on historical data.
- Validation Set (15%) → Used to fine-tune hyperparameters and prevent overfitting.
- Test Set (15%) → Used to evaluate model performance on unseen future data.

Splitting the dataset

In [ ]:
```
# Calculating size of each data-set
df = data.sort_values(by="Date")

train_size = int(len(df) * 0.7)
val_size = int(len(df) * 0.15)

# Splitting the by datetime.
train_data = df.iloc[:train_size]
val_data = df.iloc[train_size:train_size + val_size]
test_data = df.iloc[train_size + val_size:]

print(f"Train size: {len(train_data)}, Validation size: {len(val_data)}, Test size: {len(test_data)}")
```

```
Train size: 244, Validation size: 52, Test size: 53
```

In [ ]:
```
# Define the feature columns (excluding 'Label' which is the target)
feature_columns = ['Open', 'High', 'Low', 'Close', 'Volume', 'News']  # Add 'News' if using NLP features

# Splitting Train Data
X_train = train_data[feature_columns]
y_train = train_data['Label']

# Splitting Validation Data
X_val = val_data[feature_columns]
y_val = val_data['Label']

# Splitting Test Data
X_test = test_data[feature_columns]
y_test = test_data['Label']

# Display shapes of the datasets
print(f"X_train: {X_train.shape}, y_train: {y_train.shape}")
print(f"X_val: {X_val.shape}, y_val: {y_val.shape}")
print(f"X_test: {X_test.shape}, y_test: {y_test.shape}")
```

```
X_train: (244, 6), y_train: (244,)
X_val: (52, 6), y_val: (52,)
X_test: (53, 6), y_test: (53,)
```

Serializig the data

In [ ]:
```
#Store Data
# Saving all the data sets so that we can read them back if required, later.
X_train.to_csv(path+'X_train')
X_val.to_csv(path+'X_val')
X_test.to_csv(path+'X_test')
y_train.to_csv(path+'y_train')
y_val.to_csv(path+'y_val')
y_test.to_csv(path+'y_test')
```

In [ ]:
```
#Load Data
# Read splitted datasets not needed when session is current.
X_train=pd.read_csv(path+'X_train', index_col=0)
X_val=pd.read_csv(path+'X_val', index_col=0)
X_test=pd.read_csv(path+'X_test', index_col=0)
y_train=pd.read_csv(path+'y_train', index_col=0)
y_val=pd.read_csv(path+'y_val', index_col=0)
y_test=pd.read_csv(path+'y_test', index_col=0)
```

In [ ]:

Word Embeddings

Utility functions

In [ ]:
```
def clean_text(text):
    # Convert to lowercase
    text = text.lower()
```

```python
    # Remove punctuation and special characters
    text = re.sub(r'[^\w\s]', '', text)
    # Tokenize words
    words = word_tokenize(text)
    # Remove stopwords and lemmatize words
    words = [lemmatizer.lemmatize(word) for word in words if word not in stop_words]
    return words
```

In [ ]:
```python
# Initialize Lemmatizer and Stopwords
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))
```

In [ ]:
```python
# Function to compute the vector representation for a sentence
def vectorize_sentence_wv(sentence, model, vector_size):
    feature_vector = np.zeros(vector_size)  # Step 1: Initialize feature vector
    valid_words = [word for word in sentence if word in model.wv]  # Step 2: Filter words in model vocabulary

    if len(valid_words) > 0:
        feature_vector = np.sum([model.wv[word] for word in valid_words], axis=0)  # Step 3: Sum vectors
        feature_vector /= len(valid_words)  # Step 4: Average the vectors

    return feature_vector
```

In [ ]:
```python
def vectorize_sentence_gv(sentence, model, vector_size):
    feature_vector = np.zeros(vector_size)  # Initialize feature vector
    valid_words = [word for word in sentence if word in model]  # Keep words in GloVe vocab

    if len(valid_words) > 0:
        feature_vector = np.sum([model[word] for word in valid_words], axis=0)  # Sum vectors
        feature_vector /= len(valid_words)  # Average the vectors

    return feature_vector
```

Text cleaning and creating a **Word2Vec**

In [ ]:
```python
# Apply cleaning function to the News column
df['cleaned_news'] = data['News'].astype(str).apply(clean_text)

# Flatten the list of words
word_list = [word for words in df['cleaned_news'] for word in words]
# Display the first 20 words for verification
print(word_list[:20])
```

```
['tech', 'sector', 'experienced', 'significant', 'decline', 'aftermarket', 'following', 'apple', 'q1', 'revenue'
, 'warning', 'notable', 'supplier', 'including', 'skyworks', 'broadcom', 'lumentum', 'qorvo', 'tsmc', 'saw']
```

In [ ]:
```python
X_train['cleaned_news'] = X_train['News'].astype(str).apply(clean_text)
X_val['cleaned_news'] = X_val['News'].astype(str).apply(clean_text)
X_test['cleaned_news'] = X_test['News'].astype(str).apply(clean_text)
```

In [ ]:
```python
# Define Word2Vec parameters
vec_size = 100  # using 100 as we have less than 1K news articles.

# Train the Word2Vec model
w2v_model = Word2Vec(sentences=df['cleaned_news'], vector_size=vec_size, min_count=1, window=5, workers=6)

# Save the model
w2v_model.save("word2vec_model_1.model")

# Get vector representation of a word
word = "stock"
if word in w2v_model.wv:
    print(f"Vector for '{word}':\n", w2v_model.wv[word])

# Find similar words
print("\nMost similar words to 'stock':", w2v_model.wv.most_similar("stock", topn=5))
```

```
Vector for 'stock':
 [-1.26079628e-02  1.43854422e-02  9.87961702e-03  4.92452737e-03
  9.33055580e-03 -2.06732284e-02  7.80736096e-03  2.82788668e-02
 -8.65586568e-03 -1.24492273e-02 -6.19018590e-03 -2.54446436e-02
 -8.08156747e-03  1.25669008e-02  5.88114467e-03  3.25407455e-04
  7.19970465e-03 -4.60193167e-03 -3.73106741e-04 -2.01679748e-02
  9.67807043e-03  1.43489824e-03  1.60216354e-02 -1.40979541e-02
  2.42970791e-03  3.57994298e-03 -1.34489015e-02 -3.12900124e-03
 -1.04135545e-02  6.17367541e-03  2.14739610e-02 -1.31761585e-03
  1.95503840e-03 -1.31854331e-02  3.23697081e-04  1.47552611e-02
  7.10454443e-03 -4.37183306e-03  1.99492136e-03 -1.37420259e-02
  9.37174913e-03 -1.65933855e-02 -1.48067689e-02 -4.81401046e-04
  5.23545500e-03  3.83406482e-03 -5.48865180e-03 -3.28994379e-03
  6.72310311e-03  9.61985160e-03  1.16098858e-02 -1.89655107e-02
 -5.02463151e-03  2.17578514e-03 -6.75247330e-03  1.59547664e-02
  1.49933072e-02  5.19891316e-03 -1.16280476e-02  1.07989125e-02
 -6.54773274e-03  5.75393299e-03 -8.24477151e-03 -6.42121863e-03
 -1.13203004e-02  1.30376667e-02  9.70570836e-03  2.43100431e-03
 -5.97677613e-03  1.89243909e-02 -1.04655037e-02 -3.37071973e-03
  1.50268469e-02  5.61643718e-03  1.15899658e-02 -2.63036904e-03
 -5.02403593e-03 -5.65186935e-03 -5.33317542e-03  9.49247275e-04
 -1.42230308e-02  2.91042333e-03 -2.32726964e-03  9.21385083e-03
 -3.62238614e-03 -1.43983215e-03 -6.10329327e-04  1.82337314e-03
  1.86780952e-02 -6.76352211e-05  1.34246005e-02  2.78787152e-03
  2.47750105e-03 -5.89684444e-03  1.51031716e-02  1.34439366e-02
  1.01418523e-02 -1.10132983e-02 -6.36259420e-03  4.79586888e-03]

Most similar words to 'stock': [('apple', 0.7678140997886658), ('company', 0.7471034526824951), ('service', 0.71
54892086982727), ('new', 0.7135528326034546), ('concern', 0.710425615310669)]
```

In [ ]:
```python
# Compute sentence embeddings for each cleaned news article
x_train_wv = pd.DataFrame(X_train['cleaned_news'].apply(lambda x: vectorize_sentence_wv(x, w2v_model, vec_size)
x_val_wv = pd.DataFrame(X_val['cleaned_news'].apply(lambda x: vectorize_sentence_wv(x, w2v_model, vec_size)).to
x_test_wv = pd.DataFrame(X_test['cleaned_news'].apply(lambda x: vectorize_sentence_wv(x, w2v_model, vec_size)).

print(x_train_wv.shape, x_val_wv.shape, x_test_wv.shape)
```

```
(244, 100) (52, 100) (53, 100)
```

Creating the set of sentence vector using **GloVe**

In [ ]:
```python
# Load the Stanford GloVe model
filename = 'glove.6B.100d.word2vec.txt'
glove_model = KeyedVectors.load_word2vec_format(path+filename, binary=False)
print("Vocabulary size", len(glove_model.index_to_key))
```

```
Vocabulary size 400000
```

In [ ]:
```python
# Compute sentence embeddings for each cleaned news article
x_train_gv = pd.DataFrame(X_train['cleaned_news'].apply(lambda x: vectorize_sentence_gv(x, glove_model, vec_size
x_val_gv = pd.DataFrame(X_val['cleaned_news'].apply(lambda x: vectorize_sentence_gv(x, glove_model, vec_size)).
x_test_gv = pd.DataFrame(X_test['cleaned_news'].apply(lambda x: vectorize_sentence_gv(x, glove_model, vec_size)

print(x_train_gv.shape, x_val_gv.shape, x_test_gv.shape)
```

```
(244, 100) (52, 100) (53, 100)
```

Creating a set of sectence vector using **Sentence Transformer - ('sentence-transformers/all-MiniLM-L6-v2')**

In [ ]:
```python
model = SentenceTransformer('sentence-transformers/all-MiniLM-L6-v2')
```

```
modules.json:   0%|          | 0.00/349 [00:00<?, ?B/s]
config_sentence_transformers.json:   0%|          | 0.00/116 [00:00<?, ?B/s]
README.md:   0%|          | 0.00/10.7k [00:00<?, ?B/s]
sentence_bert_config.json:   0%|          | 0.00/53.0 [00:00<?, ?B/s]
config.json:   0%|          | 0.00/612 [00:00<?, ?B/s]
model.safetensors:   0%|          | 0.00/90.9M [00:00<?, ?B/s]
tokenizer_config.json:   0%|          | 0.00/350 [00:00<?, ?B/s]
vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]
tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]
special_tokens_map.json:   0%|          | 0.00/112 [00:00<?, ?B/s]
1_Pooling%2Fconfig.json:   0%|          | 0.00/190 [00:00<?, ?B/s]
```

In [ ]:
```python
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
x_train_st = model.encode(X_train['cleaned_news'].values, show_progress_bar=True, device=device)
x_val_st = model.encode(X_val['cleaned_news'].values, show_progress_bar=True, device=device)
x_test_st = model.encode(X_test['cleaned_news'].values, show_progress_bar=True, device=device)
print(x_train_st.shape, x_val_st.shape, x_test_st.shape)
```

```
Batches:   0%|          | 0/8 [00:00<?, ?it/s]
Batches:   0%|          | 0/2 [00:00<?, ?it/s]
Batches:   0%|          | 0/2 [00:00<?, ?it/s]
```

```
(244, 384) (52, 384) (53, 384)
```

Observations

- Each news content has been converted to a 100-dimensional vector using **Word2Vec**
- Each news content has been converted to a 100-dimensional vector using **GloVe**
- Each news content has been converted to a 384-dimensional vector using **Transformer - ('sentence-transformers/all-MiniLM-L6-v2')**

Sentiment Analysis

Model Evaluation Criteria

To assess the effectiveness of the sentiment analysis model, we should use the following key evaluation criteria:

Important Note: The dataset is not perfectly balanced because the Neutral class (0) dominates. However, it is not highly imbalanced either, since both Positive and Negative classes have a reasonable number of samples.

1. F1-score will be a better metric than Accuracy, since a model that predicts "Neutral" for most cases may still show high accuracy but perform poorly on minority classes.

2. As missing important signals can be costly → Recall is second most important score.

3. For deep insights → We need to analyze the confusion matrix

Functions to calculate and display metrics

In [ ]:
```python
def display_confusion_matrix(model, y, actual_target, title):

    pred = model.predict(y)  #predictions using the provided model

    cm = confusion_matrix(actual_target, pred)

    plt.figure(figsize=(5, 4))
    label_list = [0, 1,-1]
    sns.heatmap(cm, annot=True, fmt='.0f', cmap='Blues', xticklabels=label_list, yticklabels=label_list)
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
    plt.title('Confusion Matrix - ' + title)
    plt.show()
```

In [ ]:
```python
def print_evaluation_scores(model, y, actual_target, title):
    pred = model.predict(y)

    acc = accuracy_score(actual_target, pred)  #Accuracy.
    recall = recall_score(actual_target, pred,average='weighted')  #Recall.
    precision = precision_score(actual_target, pred,average='weighted')  #Precision.
    f1 = f1_score(actual_target, pred,average='weighted')  #F1-score.

    df_perf = pd.DataFrame(
        {
            "F1": [f1],
            "Recall": [recall],
            "Accuracy": [acc],
            "Precision": [precision],
        }
    )
    print(title)
    return df_perf
```

In [ ]:
```python
def display_confusion_matrix_pre_predicted(model, predicted_target, actual_target, title):

    cm = confusion_matrix(actual_target, predicted_target)

    plt.figure(figsize=(5, 4))
    label_list = [0, 1,-1]
    sns.heatmap(cm, annot=True, fmt='.0f', cmap='Blues', xticklabels=label_list, yticklabels=label_list)
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
    plt.title('Confusion Matrix - ' + title)
    plt.show()
```

In [ ]:
```python
def print_evaluation_scores_pre_predicted(model, predicted_target, actual_target, title):

    acc = accuracy_score(actual_target, predicted_target)  #Accuracy.
```

```
    recall = recall_score(actual_target, predicted_target,average='weighted')  #Recall.
    precision = precision_score(actual_target, predicted_target,average='weighted')  #Precision.
    f1 = f1_score(actual_target, predicted_target,average='weighted')  #F1-score.

    df_perf = pd.DataFrame(
        {
            "F1": [f1],
            "Recall": [recall],
            "Accuracy": [acc],
            "Precision": [precision],
        }
    )
    print(title)
    return df_perf
```

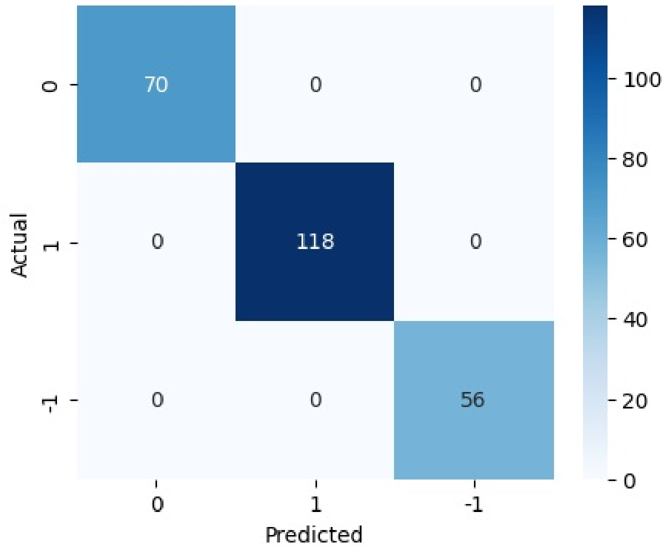
Model-1 Using Word2Vec & RandomForestClassifier

In [ ]:
```
model_wv_rfc = RandomForestClassifier(random_state=1)
model_wv_rfc.fit(x_train_wv, y_train)
```

Out[ ]:
```
RandomForestClassifier
RandomForestClassifier(random_state=1)
```

In [ ]:
```
display_confusion_matrix(model_wv_rfc,x_train_wv,y_train, 'Model-1 Using Word2Vec & RandomForestClassifier on T
```

Confusion Matrix - Model-1 Using Word2Vec & RandomForestClassifier on Train data



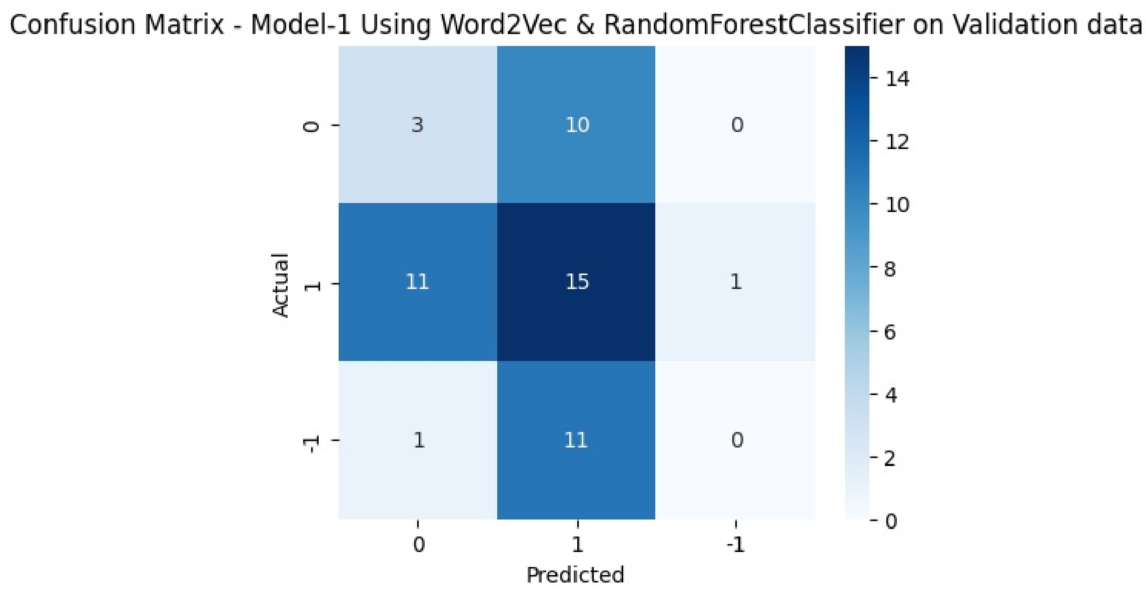
In [ ]:
```
model_wv_rfc_train_score = print_evaluation_scores(model_wv_rfc,x_train_wv,y_train, 'Model-1 Using Word2Vec & Ra
model_wv_rfc_train_score
```

Model-1 Using Word2Vec & RandomForestClassifier on Train data

Out[ ]:

	F1	Recall	Accuracy	Precision
0	1.0	1.0	1.0	1.0

In [ ]:
```
display_confusion_matrix(model_wv_rfc,x_val_wv,y_val, 'Model-1 Using Word2Vec & RandomForestClassifier on Valid
```



In [ ]:
```
model_wv_rfc_val_score = print_evaluation_scores(model_wv_rfc,x_val_wv,y_val, 'Model-1 Using Word2Vec & RandomF
model_wv_rfc_val_score
```

Model-1 Using Word2Vec & RandomForestClassifier on Validation data

Out[ ]:

	F1	Recall	Accuracy	Precision
0	0.300824	0.346154	0.346154	0.266346

Observations

Training Data Performance

- Confusion Matrix:
  - Perfect classification – The model predicts **all labels correctly** (0, 1, -1).
  - No misclassifications observed in training data.

Validation Data Performance

- Confusion Matrix:
  - The model struggles to generalize, misclassifying many samples.
  - Many Neutral (0) and Positive (1) labels are misclassified.
- Metrics:
  - F1-score = 0.30 (very low)
  - Accuracy = 0.346 (random guessing?)
  - Precision & Recall are, indicating poor generalization.
- **Overfitting**: Such high scores in training and low score on validation data indicate the model has likely memorized the training data.

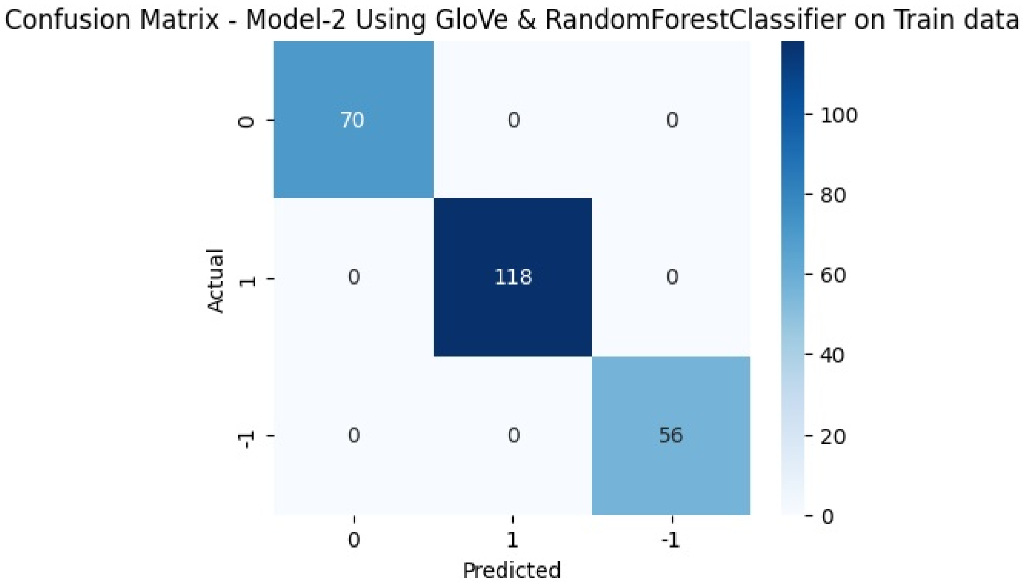
Model-2 Using GloVe & RandomForestClassifier

In [ ]:
```
model_gv_rfc = RandomForestClassifier(random_state=1)
model_gv_rfc.fit(x_train_gv, y_train)
```

Out[ ]:

```
▾ RandomForestClassifier
RandomForestClassifier(random_state=1)
```

In [ ]:
```
display_confusion_matrix(model_gv_rfc,x_train_gv,y_train, 'Model-2 Using GloVe & RandomForestClassifier on Trai
```



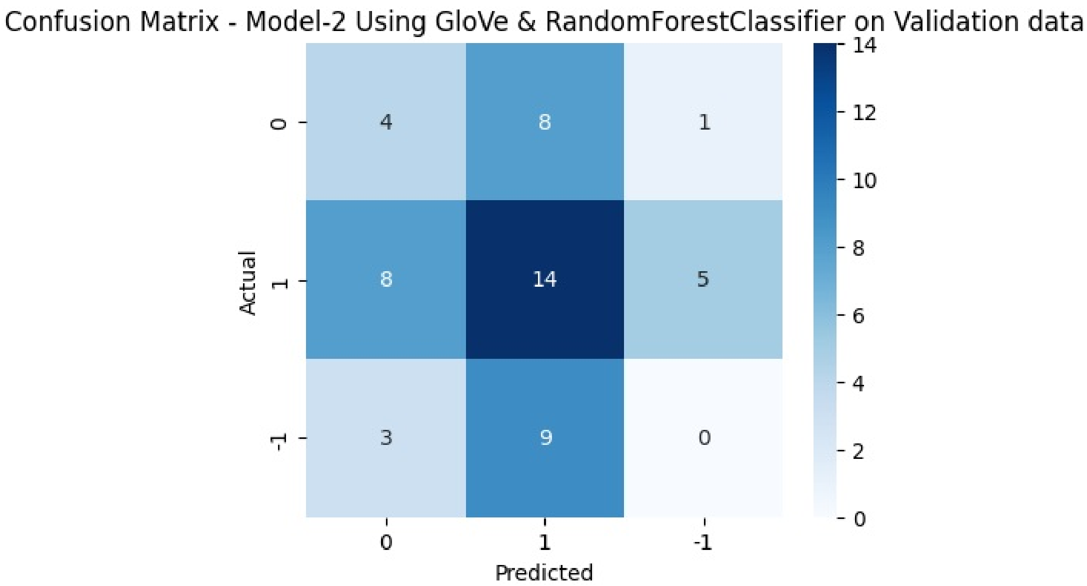

```
In [ ]: model_gv_rfc_train_score = print_evaluation_scores(model_gv_rfc,x_train_gv,y_train, 'Model-2 Using GloVe & Rand
        model_gv_rfc_train_score
```

Model-2 Using GloVe & RandomForestClassifier on Train data

Out[ ]:

	F1	Recall	Accuracy	Precision
0	1.0	1.0	1.0	1.0

```
In [ ]: display_confusion_matrix(model_gv_rfc,x_val_gv,y_val, 'Model-2 Using GloVe & RandomForestClassifier on Validati
```




```
In [ ]: model_gv_rfc_val_score = print_evaluation_scores(model_gv_rfc,x_val_gv,y_val, 'Model-2 Using GloVe & RandomFore
        model_gv_rfc_val_score
```

Model-2 Using GloVe & RandomForestClassifier on Validation data

Out[ ]:

	F1	Recall	Accuracy	Precision
0	0.322092	0.346154	0.346154	0.301158

Observations

Training Data Performance

- Confusion Matrix:
  - Perfect classification – The model predicts all labels correctly (0, 1, -1).

Validation Data Performance

- Confusion Matri:
  - The model misclassifies many Neutral (0) and Positive (1) instances.
  - Prediction distribution is scattered, meaning the model struggles to differentiate sentiments.
- Metrics:

- F1-score = 0.32 (poor generalization, but better than Word2Vec)
- Accuracy = 0.34 (random guess for a three-class problem)
- Precision & Recall are low, showing an inability to distinguish between sentiment classes.

- Overfitting: The score indicates memorization of training data rather than true learning.

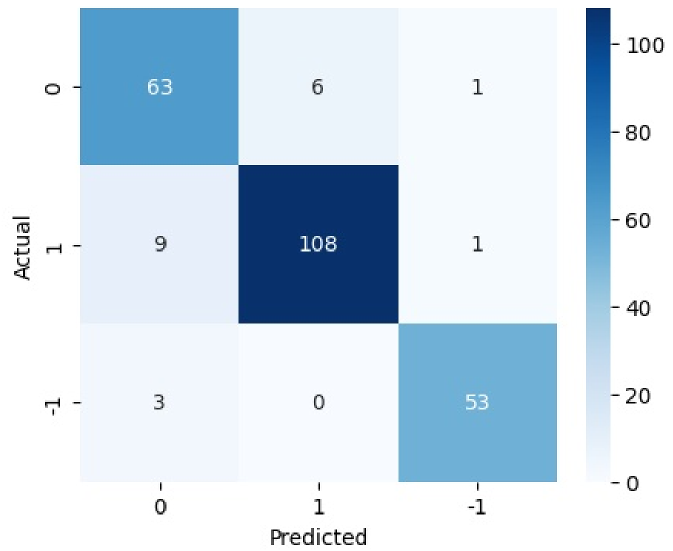
Model-3 Sentence Transformer & RandomForestClassifier

In [ ]:
```
model_st_rfc = RandomForestClassifier(random_state=1)
model_st_rfc.fit(x_train_st, y_train)
```

Out[ ]:
```
▾ RandomForestClassifier
RandomForestClassifier(random_state=1)
```

In [ ]:
```
display_confusion_matrix(model_st_rfc,x_train_st,y_train, 'Model-3 Sentence Transformer & RandomForestClassifie
```

Confusion Matrix - Model-3 Sentence Transformer & RandomForestClassifier on Train data



In [ ]:
```
model_st_rfc_train_score = print_evaluation_scores(model_st_rfc,x_train_st,y_train, 'Model-3 Sentence Transform
model_st_rfc_train_score
```

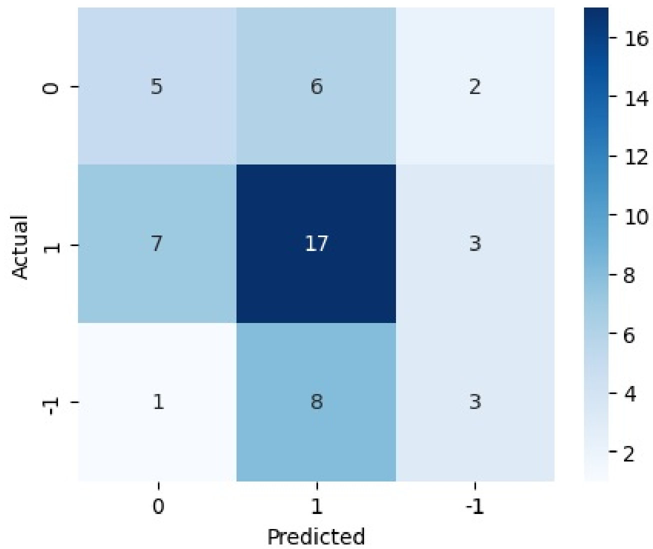
```
Model-3 Sentence Transformer & RandomForestClassifier on Train data
```

Out[ ]:

	F1	Recall	Accuracy	Precision
0	0.918718	0.918033	0.918033	0.9203

In [ ]:
```
display_confusion_matrix(model_st_rfc,x_val_st,y_val, 'Model-3 Sentence Transformer & RandomForestClassifier on
```

Confusion Matrix - Model-3 Sentence Transformer & RandomForestClassifier on Validation data



In [ ]:
```
model_st_rfc_val_score = print_evaluation_scores(model_st_rfc,x_val_st,y_val, 'Model-3 Sentence Transformer & Ra
model_st_rfc_val_score
```

```
Model-3 Sentence Transformer & RandomForestClassifier on Validation data
```

Out[ ]:

	F1	Recall	Accuracy	Precision
0	0.469761	0.480769	0.480769	0.467432

Observations

Training Data Performance:

- The confusion matrix shows that the model has relatively good classification ability on training data.
- Some misclassifications occur, but the model does not show the extreme overfitting observed with Word2Vec and GloVe.
- F1-score, recall, accuracy, and precision are all around 0.92, indicating strong training performance.

Validation Data Performance:

- The model shows improved generalization compared to Word2Vec and GloVe, but misclassification is still present.
- The confusion matrix indicates that the model struggles to separate neutral from both positive and negative sentiments.
- The F1-score, recall, and accuracy on validation data are around 0.47, which is an improvement over previous models but still not ideal.

Important Note

- The model generalizes better than the previous ones but still has difficulty distinguishing between sentiment classes.
- Transformer embeddings provide richer contextual information than static embeddings like Word2Vec and GloVe.
- Random Forest may still not be the best model for text classification, as it does not consider the sequential nature of language.

Model-4 Improving classification model (XGBoost) using Word2Vec

In [ ]:

```
model_wv_xgb_tuned = XGBClassifier(tree_method='gpu_hist', predictor='gpu_predictor')

# Define hyperparameter space
parameters = {
    'n_estimators': np.arange(50, 300, 50),
    'max_depth': np.arange(1, 10, 2),
    'learning_rate': [0.01, 0.1, 0.2, 0.5],
    'subsample': [0.7, 0.8, 1.0],
    'colsample_bytree': [0.7, 0.8, 1.0],
    'gamma': [0, 0.1, 0.2],
    'reg_lambda': [1, 5, 10],
    'reg_alpha': [0, 1, 5]
}

# Using RandomizedSearchCV
random_search = RandomizedSearchCV(
    model_wv_xgb_tuned,
    parameters,
    n_iter=10,  # test 10 random parameter sets
    scoring='f1_weighted',
    cv=5,  # cross-validation folds
    n_jobs=-1,
    verbose=2,
    random_state=1
)

label_map = {-1: 0, 0: 1, 1: 2}

# Apply mapping to NumPy arrays
y_train_xgb = np.vectorize(label_map.get)(y_train)
y_val_xgb = np.vectorize(label_map.get)(y_val)
y_test_xgb = np.vectorize(label_map.get)(y_test)


start_time = time.time()

random_search.fit(x_train_wv, y_train_xgb)

end_time = time.time()

print(f"Training completed in {round((end_time - start_time) / 60, 2)} minutes")
print("Best Hyperparameters:", random_search.best_params_)
# Best model
model_wv_xgb_tuned = random_search.best_estimator_
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
Training completed in 0.75 minutes
Best Hyperparameters: {'subsample': 1.0, 'reg_lambda': 1, 'reg_alpha': 5, 'n_estimators': 50, 'max_depth': 3, 'l
earning_rate': 0.01, 'gamma': 0.1, 'colsample_bytree': 0.8}
```

In [ ]:

```
model_wv_xgb_tuned.fit(x_train_wv, y_train_xgb)
```

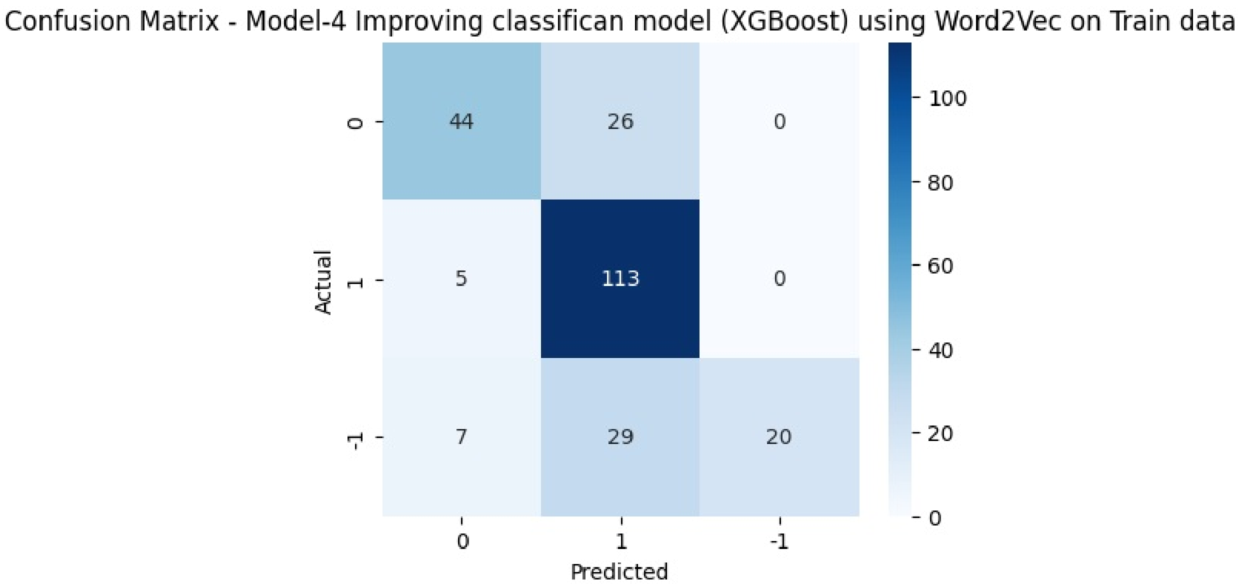
Out[ ]:

XGBClassifier

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.8, device=None, early_stopping_rounds=None
,
              enable_categorical=False, eval_metric=None, feature_types=Non
e,
              gamma=0.1, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.01, max_bin=Non
e,
```

In [ ]:

```
y_train_pred = model_wv_xgb_tuned.predict(x_train_wv)
reverse_label_map = {0: -1, 1: 0, 2: 1}
y_train_pred = pd.Series(y_train_pred).map(reverse_label_map)
display_confusion_matrix_pre_predicted(model_wv_xgb_tuned,y_train_pred,y_train, 'Model-4 Improving classifican r
```



In [ ]:

```
model_wv_xgb_tuned_train_score = print_evaluation_scores_pre_predicted(model_wv_xgb_tuned, y_train_pred,y_train
model_wv_xgb_tuned_train_score
```

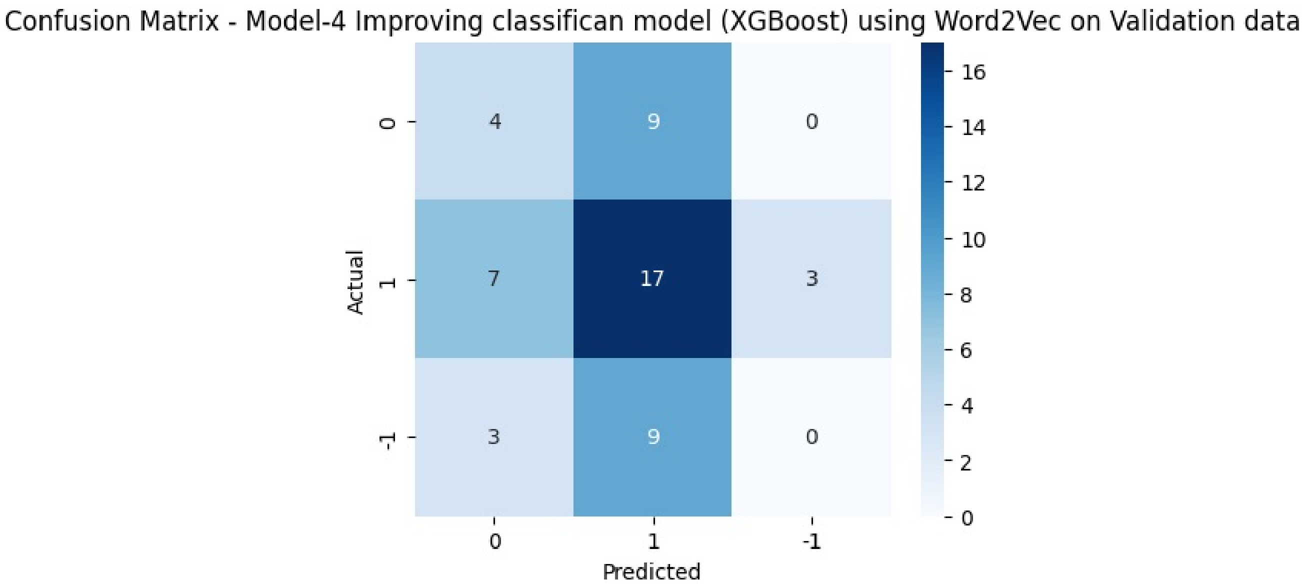
Model-4 Improving classifican model (XGBoost) using Word2Vec on Train data

Out[ ]:

	F1	Recall	Accuracy	Precision
0	0.703309	0.72541	0.72541	0.780201

In [ ]:

```
y_val_pred = model_wv_xgb_tuned.predict(x_val_wv)
reverse_label_map = {0: -1, 1: 0, 2: 1}
y_val_pred = pd.Series(y_val_pred).map(reverse_label_map)
display_confusion_matrix_pre_predicted(model_wv_xgb_tuned,y_val_pred,y_val, 'Model-4 Improving classifican mode
```



In [ ]:

```
model_wv_xgb_tuned_val_score = print_evaluation_scores_pre_predicted(model_wv_xgb_tuned,y_val_pred,y_val, 'Mode
```

```
model_wv_xgb_tuned_val_score
```

Model-4 Improving classifican model (XGBoost) using Word2Vec on Validation data

Out[ ]:

	F1	Recall	Accuracy	Precision
0	0.358814	0.403846	0.403846	0.323626

Observations

Training Data Performance:

- The confusion matrix shows that the model correctly classifies many samples but struggles with differentiating between neutral and other sentiment classes.
- Some misclassification is present, especially in the neutral class where many instances are classified incorrectly.
- The F1-score is 0.70, with an accuracy of 72.5 percent, which is significantly better than the previous Random Forest model with Word2Vec but still not ideal.

Validation Data Performance:

- The model struggles more on the validation set, with lower classification performance.
- The confusion matrix shows that many neutral samples are misclassified as positive or negative.
- The F1-score drops to 0.35, and the accuracy is 40.4 percent, indicating poor generalization to unseen data.
- The precision and recall values suggest that the model struggles to differentiate between sentiment classes effectively.

Notes:

- The model improves over Random Forest in handling sentiment classification but still faces generalization issues.
- Word2Vec embeddings may not be fully capturing the sentiment nuances in text data, leading to misclassification.
- The drop in performance from training to validation suggests some degree of overfitting.
- XGBoost provides a better balance between training accuracy and interpretability compared to previous models.

Model-5 Improving classification model (XGBoost) using GloVec

In [ ]:

```python
model_gv_xgb_tuned = XGBClassifier(tree_method='gpu_hist', predictor='gpu_predictor')

# Define hyperparameter space
parameters = {
    'n_estimators': np.arange(50, 300, 50),
    'max_depth': np.arange(1, 10, 2),
    'learning_rate': [0.01, 0.1, 0.2, 0.5],
    'subsample': [0.7, 0.8, 1.0],
    'colsample_bytree': [0.7, 0.8, 1.0],
    'gamma': [0, 0.1, 0.2],
    'reg_lambda': [1, 5, 10],
    'reg_alpha': [0, 1, 5]
}

# Using RandomizedSearchCV
random_search = RandomizedSearchCV(
    model_gv_xgb_tuned,
    parameters,
    n_iter=10,  # test 10 random parameter sets
    scoring='f1_weighted',
    cv=5,  # cross-validation folds
    n_jobs=-1,
    verbose=2,
    random_state=1
)

label_map = {-1: 0, 0: 1, 1: 2}

# Apply mapping to NumPy arrays
y_train_xgb = np.vectorize(label_map.get)(y_train)
y_val_xgb = np.vectorize(label_map.get)(y_val)
y_test_xgb = np.vectorize(label_map.get)(y_test)


start_time = time.time()

random_search.fit(x_train_gv, y_train_xgb)

end_time = time.time()

print(f"Training completed in {round((end_time - start_time) / 60, 2)} minutes")
print("Best Hyperparameters:", random_search.best_params_)
# Best model
model_gv_xgb_tuned = random_search.best_estimator_
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
Training completed in 0.77 minutes
Best Hyperparameters: {'subsample': 0.8, 'reg_lambda': 1, 'reg_alpha': 0, 'n_estimators': 250, 'max_depth': 5, '
learning_rate': 0.01, 'gamma': 0.1, 'colsample_bytree': 1.0}
```

In [ ]:
```
model_gv_xgb_tuned.fit(x_train_gv, y_train_xgb)
```

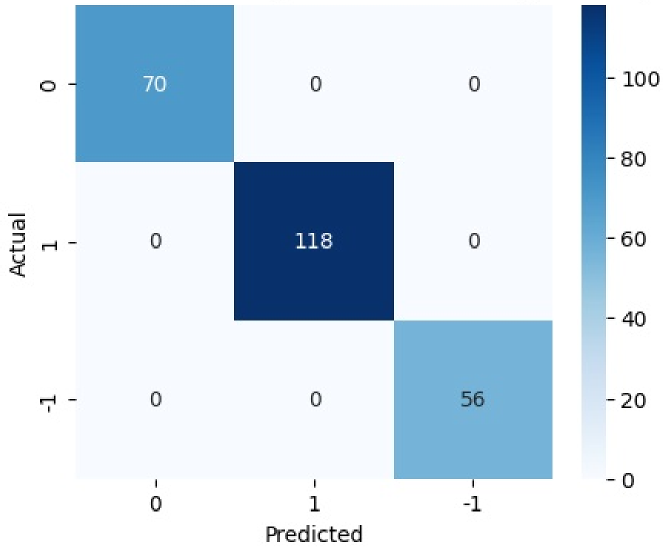
Out[ ]:

XGBClassifier

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=1.0, device=None, early_stopping_rounds=None
,
              enable_categorical=False, eval_metric=None, feature_types=Non
e,
              gamma=0.1, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.01, max_bin=Non
e,
```

In [ ]:
```
y_train_pred = model_gv_xgb_tuned.predict(x_train_gv)
reverse_label_map = {0: -1, 1: 0, 2: 1}
y_train_pred = pd.Series(y_train_pred).map(reverse_label_map)
display_confusion_matrix_pre_predicted(model_gv_xgb_tuned,y_train_pred,y_train, 'Model-5 Improving classifican r
```

Confusion Matrix - Model-5 Improving classifican model (XGBoost) using GloVec



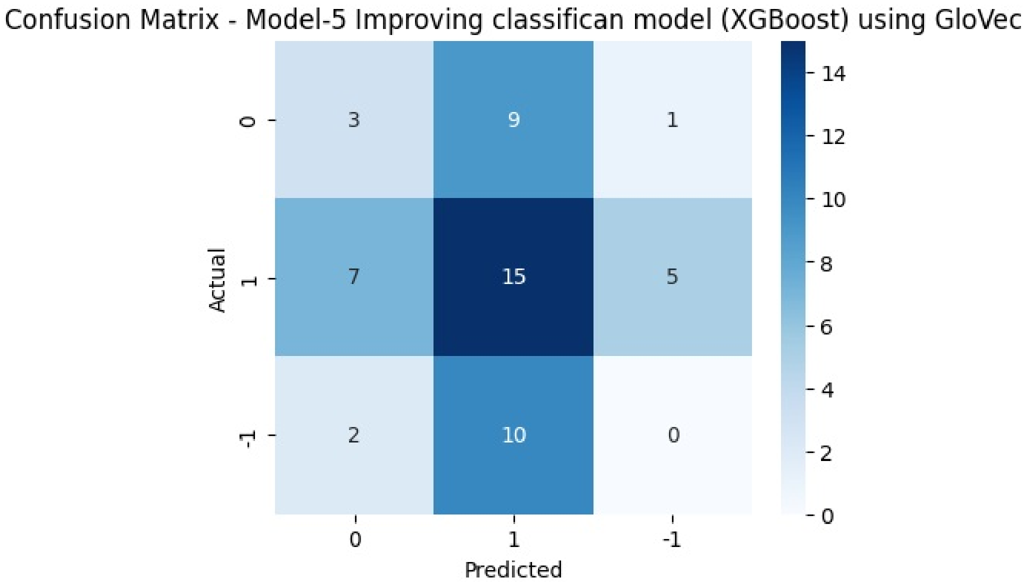
In [ ]:
```
model_gv_xgb_tuned_train_score = print_evaluation_scores_pre_predicted(model_gv_xgb_tuned, y_train_pred,y_train
model_gv_xgb_tuned_train_score
```

```
Model-5 Improving classifican model (XGBoost) using GloVec
```

Out[ ]:

	F1	Recall	Accuracy	Precision
0	1.0	1.0	1.0	1.0

In [ ]:
```
y_val_pred = model_gv_xgb_tuned.predict(x_val_gv)
reverse_label_map = {0: -1, 1: 0, 2: 1}
y_val_pred = pd.Series(y_val_pred).map(reverse_label_map)
display_confusion_matrix_pre_predicted(model_gv_xgb_tuned,y_val_pred,y_val, 'Model-5 Improving classifican mode
```



In [ ]:
```
model_gv_xgb_tuned_val_score = print_evaluation_scores_pre_predicted(model_gv_xgb_tuned,y_val_pred,y_val, 'Mode
model_gv_xgb_tuned_val_score
```

Model-5 Improving classifican model (XGBoost) using GloVec

Out[ ]:

	F1	Recall	Accuracy	Precision
0	0.315359	0.346154	0.346154	0.291572

Observations

Training Data Performance:

- The confusion matrix shows that the model achieves perfect classification on the training set.
- All instances are classified correctly, resulting in an F1-score, recall, accuracy, and precision of 1.0.
- This suggests that the model has memorized the training data completely, indicating overfitting.

Validation Data Performance:

- The confusion matrix on the validation set indicates significant misclassification, particularly among neutral and negative classes.
- The F1-score is 0.31, and accuracy is 34.6 percent, showing that the model does not generalize well to unseen data.
- Precision and recall values are low, reinforcing the model's struggle to correctly differentiate between sentiment classes.

Points to Note:

- The model is overfitting, as evident from the stark contrast between training and validation performance.
- GloVe embeddings may not be capturing enough contextual sentiment information, leading to misclassifications.
- The validation performance is similar to the results seen with Word2Vec, suggesting that static word embeddings may not be the best approach for sentiment classification.
- XGBoost, while effective in structured data problems, might not be the ideal choice for text classification without additional feature engineering.

Model-6 Improving classification model (XGBoost) using Sentence Transformer

In [ ]:
```
model_st_xgb_tuned = XGBClassifier(tree_method='gpu_hist', predictor='gpu_predictor')

# Define hyperparameter space
parameters = {
    'n_estimators': np.arange(50, 300, 50),
    'max_depth': np.arange(1, 10, 2),
    'learning_rate': [0.01, 0.1, 0.2, 0.5],
    'subsample': [0.7, 0.8, 1.0],
    'colsample_bytree': [0.7, 0.8, 1.0],
    'gamma': [0, 0.1, 0.2],
    'reg_lambda': [1, 5, 10],
    'reg_alpha': [0, 1, 5]
}

# Using RandomizedSearchCV
random_search = RandomizedSearchCV(
    model_st_xgb_tuned,
    parameters,
    n_iter=10,  # test 10 random parameter sets
    scoring='f1_weighted',
```

```
    cv=5,  # cross-validation folds
    n_jobs=-1,
    verbose=2,
    random_state=1
)

label_map = {-1: 0, 0: 1, 1: 2}

# Apply mapping to NumPy arrays
y_train_xgb = np.vectorize(label_map.get)(y_train)
y_val_xgb = np.vectorize(label_map.get)(y_val)
y_test_xgb = np.vectorize(label_map.get)(y_test)


start_time = time.time()

random_search.fit(x_train_st, y_train_xgb)

end_time = time.time()

print(f"Training completed in {round((end_time - start_time) / 60, 2)} minutes")
print("Best Hyperparameters:", random_search.best_params_)
# Best model
model_st_xgb_tuned = random_search.best_estimator_
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
Training completed in 0.97 minutes
Best Hyperparameters: {'subsample': 0.7, 'reg_lambda': 1, 'reg_alpha': 1, 'n_estimators': 50, 'max_depth': 7, 'l
earning_rate': 0.1, 'gamma': 0, 'colsample_bytree': 1.0}
```

In [ ]:
```
model_st_xgb_tuned.fit(x_train_st, y_train_xgb)
```

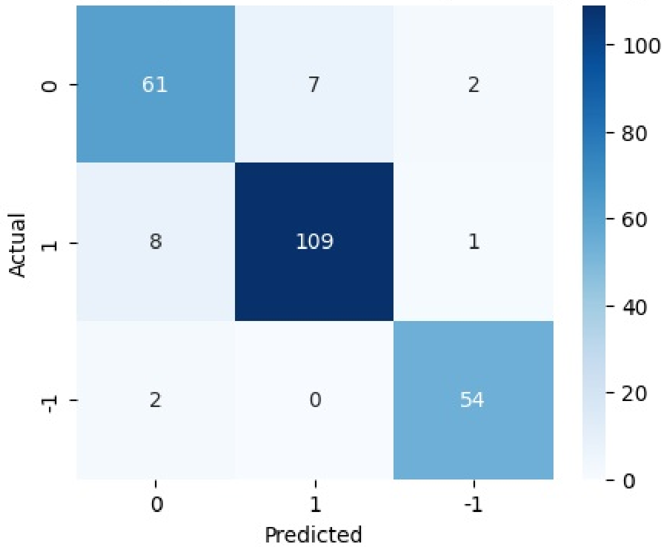
Out[ ]:

XGBClassifier

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=1.0, device=None, early_stopping_rounds=None
,
              enable_categorical=False, eval_metric=None, feature_types=Non
e,
              gamma=0, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.1, max_bin=None
,
```

In [ ]:
```
y_train_pred = model_st_xgb_tuned.predict(x_train_st)
reverse_label_map = {0: -1, 1: 0, 2: 1}
y_train_pred = pd.Series(y_train_pred).map(reverse_label_map)
display_confusion_matrix_pre_predicted(model_st_xgb_tuned,y_train_pred,y_train, 'Model-6 Improving classificati
```

Confusion Matrix - Model-6 Improving classification model (XGBoost) using Sentence Transformer



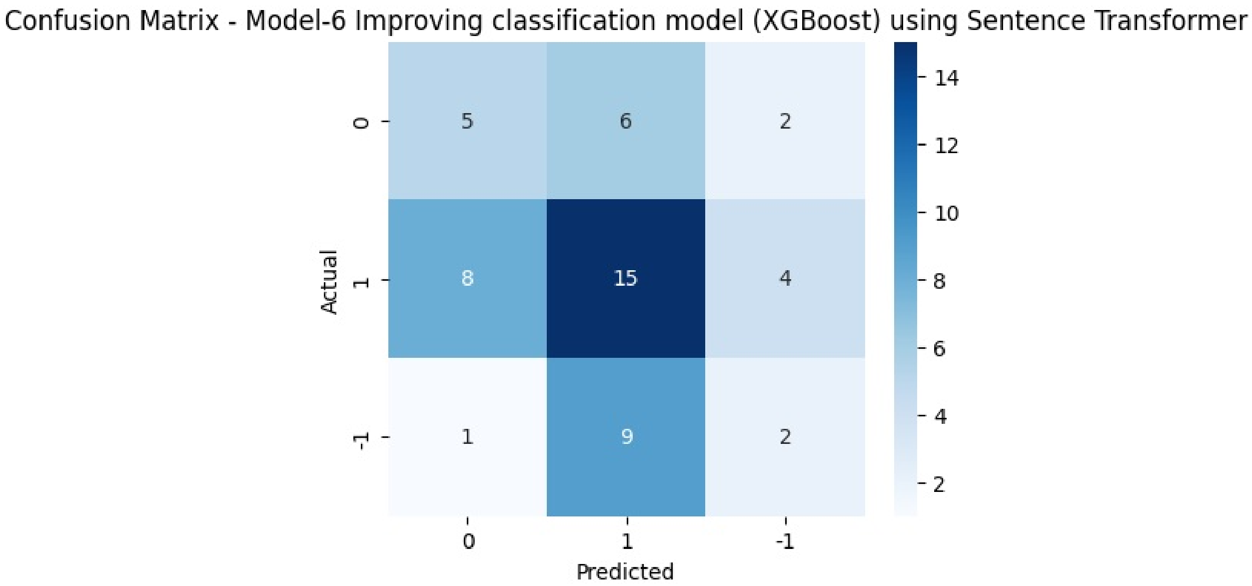
In [ ]:
```
model_st_xgb_tuned_train_score = print_evaluation_scores_pre_predicted(model_st_xgb_tuned, y_train_pred,y_train
model_st_xgb_tuned_train_score
```

```
Model-6 Improving classification model (XGBoost) using Sentence Transformer
```

Out[ ]:

	F1	Recall	Accuracy	Precision
0	0.918119	0.918033	0.918033	0.918331

In [ ]:
```
y_val_pred = model_st_xgb_tuned.predict(x_val_st)
reverse_label_map = {0: -1, 1: 0, 2: 1}
y_val_pred = pd.Series(y_val_pred).map(reverse_label_map)
display_confusion_matrix_pre_predicted(model_st_xgb_tuned,y_val_pred,y_val, 'Model-6 Improving classification mo
```



In [ ]:
```
model_st_xgb_tuned_val_score = print_evaluation_scores_pre_predicted(model_st_xgb_tuned,y_val_pred,y_val, 'Mode
model_st_xgb_tuned_val_score
```

```
Model-6 Improving classification model (XGBoost) using Sentence Transformer
```

Out[ ]:

	F1	Recall	Accuracy	Precision
0	0.412026	0.423077	0.423077	0.406593

Observations

Training Data Performance:

- The confusion matrix shows that the model performs well on the training set, with some misclassifications but overall strong predictions.
- The model achieves an F1-score, recall, accuracy, and precision of approximately 91.8 percent, indicating high performance on the training data.
- Compared to previous models using Word2Vec and GloVe, the Sentence Transformer embeddings lead to a better classification ability while reducing overfitting.

Validation Data Performance:

- The confusion matrix on the validation set shows misclassifications primarily between the neutral and positive/negative classes.
- The F1-score is 0.41, with an accuracy of 42.3 percent, which is an improvement over the previous Word2Vec and GloVe models.
- The precision and recall values suggest that the model generalizes better than the previous embeddings but still struggles with class separation.

Points to Note:

- The model performs significantly better than the previous models using Word2Vec and GloVe, particularly in training, but there is still a gap in validation performance.
- The improved results suggest that contextual embeddings from the Sentence Transformer are more effective in capturing sentiment nuances compared to static embeddings.
- The overfitting issue is still present, but it is less severe than in earlier experiments.
- Misclassifications in the validation set indicate that further tuning or additional features may be needed to improve differentiation between sentiment classes.

Model-7 Improving classification model (DecisionTree) using Sentence Transformer

In [ ]:

In [ ]:
```
model_st_sdt_tuned = DecisionTreeClassifier(random_state=1)
```

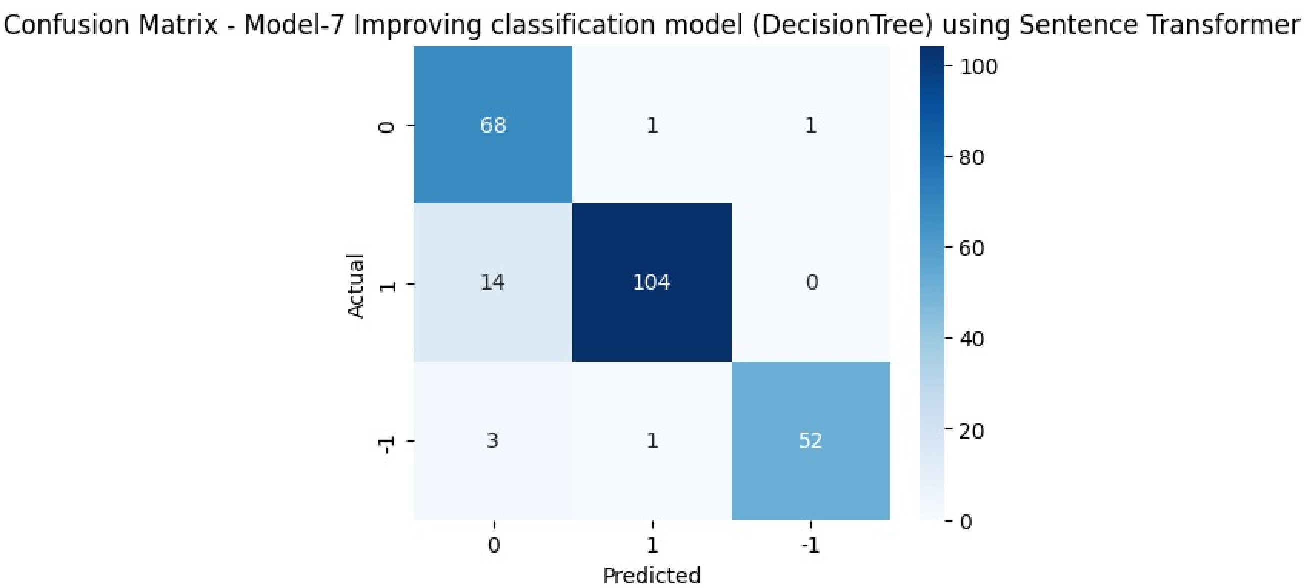
```python
# Define hyperparameter space

parameters = {
    'criterion': ['gini', 'entropy'],  # Adding 'log_loss' for newer sklearn versions if applicable
    'splitter': ['best', 'random'],  # Determines how nodes are split
    'max_depth': [3, 5, 10, 20, 30, None],  # Increased depth range
    'min_samples_split': [2, 5, 10, 20, 50],  # Minimum samples required to split
    'min_samples_leaf': [1, 2, 5, 10, 20],  # Minimum samples required in a leaf
    'max_features': [None, 'sqrt', 'log2'],  # Number of features considered for split
    'class_weight': [None, 'balanced'],  # Handles class imbalance
    'min_impurity_decrease': [0.0, 0.01, 0.05],  # Regularization to reduce overfitting
}


# Using GridSearchCV
grid_search = GridSearchCV(
    model_st_xgb_tuned,
    parameters,
    scoring='f1_weighted',
    cv=5,  # 5-fold cross-validation
    n_jobs=-1,  # Use all CPU cores for faster execution
    verbose=2  # Display progress
    )

label_map = {-1: 0, 0: 1, 1: 2}

# Apply mapping to NumPy arrays
y_train_sdt = np.vectorize(label_map.get)(y_train)
y_val_sdt = np.vectorize(label_map.get)(y_val)
y_test_sdt = np.vectorize(label_map.get)(y_test)


start_time = time.time()

random_search.fit(x_train_st, y_train_sdt)

end_time = time.time()

print(f"Training completed in {round((end_time - start_time) / 60, 2)} minutes")
print("Best Hyperparameters:", random_search.best_params_)
# Best model
model_st_xgb_tuned = random_search.best_estimator_
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
Training completed in 1.01 minutes
Best Hyperparameters: {'subsample': 0.7, 'reg_lambda': 1, 'reg_alpha': 1, 'n_estimators': 50, 'max_depth': 7, 'l
earning_rate': 0.1, 'gamma': 0, 'colsample_bytree': 1.0}
```

In [ ]:
```python
model_st_sdt_tuned.fit(x_train_st, y_train_sdt)
```

Out[ ]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(random_state=1)
```

In [ ]:
```python
y_train_pred = model_st_sdt_tuned.predict(x_train_st)
reverse_label_map = {0: -1, 1: 0, 2: 1}
y_train_pred = pd.Series(y_train_pred).map(reverse_label_map)
display_confusion_matrix_pre_predicted(model_st_sdt_tuned,y_train_pred,y_train, 'Model-7 Improving classificati
```



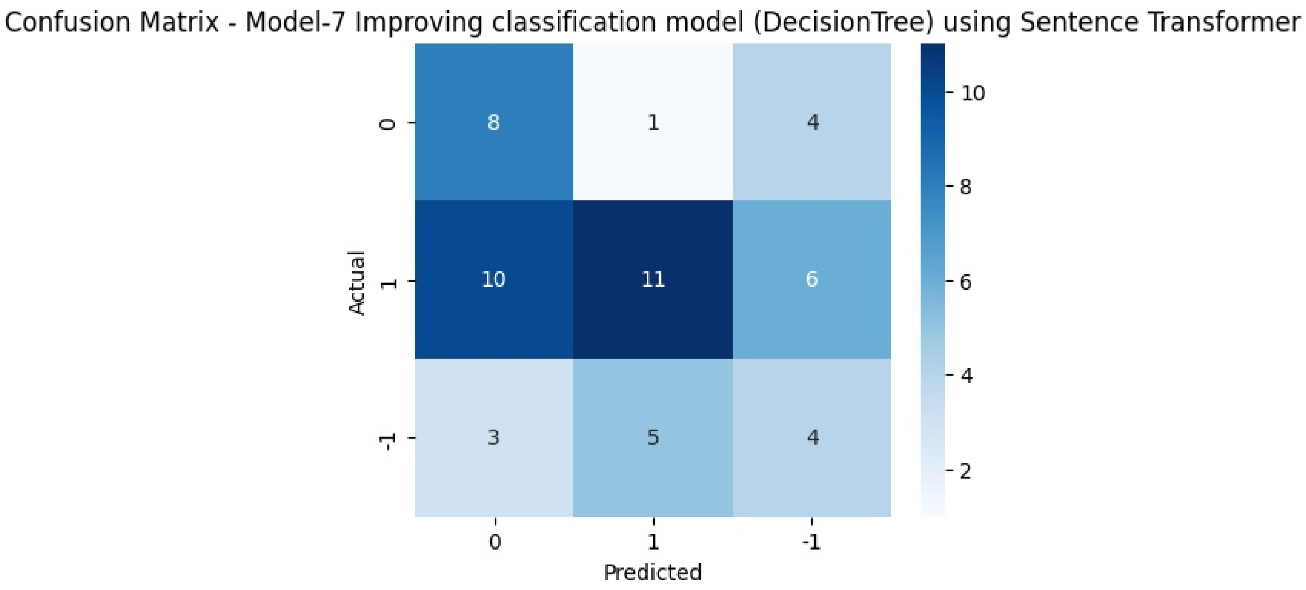
```
In [ ]: model_st_sdt_tuned_train_score = print_evaluation_scores_pre_predicted(model_st_sdt_tuned, y_train_pred,y_train
        model_st_sdt_tuned_train_score
```

Model-7 Improving classification model (DecisionTree) using Sentence Transformer

Out[ ]:

	F1	Recall	Accuracy	Precision
0	0.919762	0.918033	0.918033	0.929168

```
In [ ]: y_val_pred = model_st_sdt_tuned.predict(x_val_st)
        reverse_label_map = {0: -1, 1: 0, 2: 1}
        y_val_pred = pd.Series(y_val_pred).map(reverse_label_map)
        display_confusion_matrix_pre_predicted(model_st_sdt_tuned,y_val_pred,y_val, 'Model-7 Improving classification mo
```




```
In [ ]: model_st_sdt_tuned_val_score = print_evaluation_scores_pre_predicted(model_st_sdt_tuned,y_val_pred,y_val, 'Mode
        model_st_sdt_tuned_val_score
```

Model-7 Improving classification model (DecisionTree) using Sentence Transformer

Out[ ]:

	F1	Recall	Accuracy	Precision
0	0.448268	0.442308	0.442308	0.497145

Observations

Training Data Performance:

- The model performs well on the training data, with an F1-score, recall, and accuracy of approximately 91.8 percent.
- Some misclassifications occur, particularly with the neutral class, but the overall performance suggests strong learning from the data.
- The model does not exhibit complete overfitting like some previous models, but it still performs better on training than on validation.

Validation Data Performance:

- The confusion matrix shows that the model struggles with classifying neutral and positive sentiments correctly.
- The F1-score is 0.44, with an accuracy of 44.2 percent, which is an improvement over previous models but still not ideal.
- The decision tree performs better than previous models with Word2Vec and GloVe but slightly worse than XGBoost when using Sentence Transformer embeddings.

Points To Note:

- The Decision Tree model generalizes better than Random Forest and XGBoost models trained on static embeddings.
- The performance is comparable to the XGBoost model trained on Sentence Transformer embeddings, with slightly better interpretability but slightly lower predictive power.
- The model still struggles with sentiment separation, particularly for neutral and positive cases.

Model Performance Summary

```
In [ ]: models_train_comp_df = pd.concat(
            [
              model_wv_rfc_train_score.T,
              model_gv_rfc_train_score.T,
              model_st_rfc_train_score.T,
              model_wv_xgb_tuned_train_score.T,
              model_gv_xgb_tuned_train_score.T,
```

```
        model_st_xgb_tuned_train_score.T,
        model_st_sdt_tuned_train_score.T,
    ],axis=1
)

models_train_comp_df.columns = [
    "Model-1 (Word2Vec+RandomForest)",
    "Model-2 (GloVe+RandomForest)",
    "Model-3 (SentenceTransformer+RandomForest)",
    "Model-4 (Word2Vec+XGB[tuned])",
    "Model-5 (GloVe+XGB[tuned]))",
    "Model-6 (SentenceTransformer+XGB[tuned])",
    "Model-7 (SentenceTransformer+DecisionTree[truned])",
]

print("Training performance comparison:")
models_train_comp_df
```

```
Training performance comparison:
```

Out[ ]:

	Model-1 (Word2Vec+RandomForest)	Model-2 (GloVe+RandomForest)	Model-3 (SentenceTransformer+RandomForest)	Model-4 (Word2Vec+XGB[tuned])	(GloV
F1	1.0	1.0	0.918718	0.703309	
Recall	1.0	1.0	0.918033	0.725410	
Accuracy	1.0	1.0	0.918033	0.725410	
Precision	1.0	1.0	0.920300	0.780201	

In [ ]:

```
models_val_comp_df = pd.concat(
    [
        model_wv_rfc_val_score.T,
        model_gv_rfc_val_score.T,
        model_st_rfc_val_score.T,
        model_wv_xgb_tuned_val_score.T,
        model_gv_xgb_tuned_val_score.T,
        model_st_xgb_tuned_val_score.T,
        model_st_sdt_tuned_val_score.T,
    ],axis=1
)

models_val_comp_df.columns = [
    "Model-1 (Word2Vec+RandomForest)",
    "Model-2 (GloVe+RandomForest)",
    "Model-3 (SentenceTransformer+RandomForest)",
    "Model-4 (Word2Vec+XGB[tuned])",
    "Model-5 (GloVe+XGB[tuned]))",
    "Model-6 (SentenceTransformer+XGB[tuned])",
    "Model-7 (SentenceTransformer+DecisionTree[truned])",
]

print("Validation performance comparison:")
models_val_comp_df
```

```
Validation performance comparison:
```

Out[ ]:

	Model-1 (Word2Vec+RandomForest)	Model-2 (GloVe+RandomForest)	Model-3 (SentenceTransformer+RandomForest)	Model-4 (Word2Vec+XGB[tuned])	(GloV
F1	0.300824	0.322092	0.469761	0.358814	
Recall	0.346154	0.346154	0.480769	0.403846	
Accuracy	0.346154	0.346154	0.480769	0.403846	
Precision	0.266346	0.301158	0.467432	0.323626	

Training Performance Analysis:

- Models **1 (Word2Vec + Random Forest)**, **2 (GloVe + Random Forest)**, and **5 (GloVe + XGB)** exhibit **perfect scores (F1, Recall, Accuracy, Precision = 1.0)**, which suggests **overfitting**.
- **Sentence Transformer-based models (3, 6, and 7)** have slightly lower scores in training (~91.8%), indicating **better generalization** compared to overfitted models.
- **Model 4 (Word2Vec + XGB)** has the lowest training score (~70.3% F1-score), showing that Word2Vec is not as effective as Sentence Transformer embeddings.

Validation Performance Analysis:

- **Models 1, 2, and 5** (Word2Vec/GloVe + Random Forest/XGB) perform **poorly on validation**, with **F1-scores around 30-35%**, confirming **overfitting**.

- **Model 3 (SentenceTransformer + Random Forest)** achieves **the highest validation F1-score among Random Forest models (0.47)**, showing that **contextual embeddings improve sentiment classification**.
- **Model 6 (SentenceTransformer + XGB) and Model 7 (SentenceTransformer + DecisionTree)** perform **the best overall on validation**, with:
  - **Model 6 (XGB) F1-score: 0.41**
  - **Model 7 (DecisionTree) F1-score: 0.45**
  - **Model 7 has the best precision (0.497)**, making it more reliable for classification.

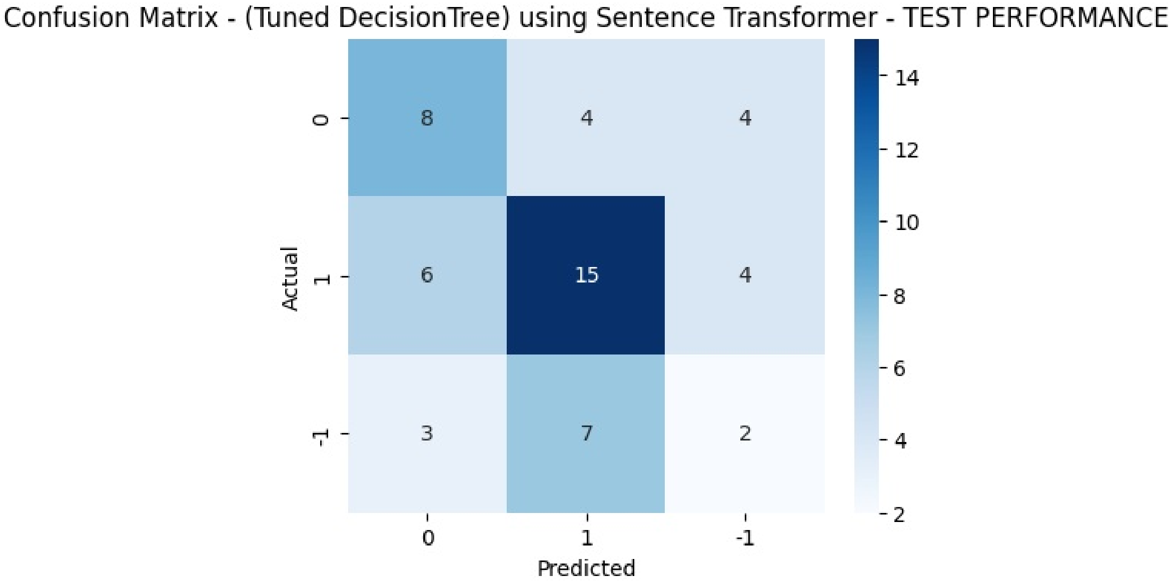
Final Model Selection

Best Model: Model 7 (SentenceTransformer + DecisionTree)

- It **generalizes better** than overfitted models.
- **F1-score (0.45) is the highest on validation**, indicating better sentiment prediction.
- **Precision (0.497) is the best**, meaning it reduces false positives.
- **DecisionTree is interpretable**, making it easier to analyze feature importance.

Final Performance Check with Test Data

In [ ]:
```
y_test_pred = model_st_sdt_tuned.predict(x_test_st)
reverse_label_map = {0: -1, 1: 0, 2: 1}
y_test_pred = pd.Series(y_test_pred).map(reverse_label_map)
display_confusion_matrix_pre_predicted(model_st_sdt_tuned,y_test_pred,y_test, '(Tuned DecisionTree) using Senter
```



In [ ]:
```
model_st_sdt_tuned_test_score = print_evaluation_scores_pre_predicted(model_st_sdt_tuned,y_test_pred,y_test, '(
model_st_sdt_tuned_test_score
```

```
(Tuned DecisionTree) using Sentence Transformer - TEST SCORE
```

Out[ ]:

	F1	Recall	Accuracy	Precision
0	0.465005	0.471698	0.471698	0.459481

Observations:

- The model generalizes well to unseen data with an F1-score of 0.465, which is consistent with its validation performance.
- The confusion matrix shows that misclassification is still present, particularly in differentiating between neutral and positive/negative sentiments.
- The precision and recall values are balanced, indicating that the model does not heavily favor one class over another.
- The accuracy of 47.2% suggests that while the model is better than random guessing, it still has room for improvement.
- Compared to previous models, this one performs better on real-world test data due to the Sentence Transformer embeddings capturing contextual sentiment effectively.

Weekly News Summarization

Installing and Importing the necessary libraries

In [ ]:
```
# Necessary Libraries
import pandas as pd
import numpy as np
```

```python
# To visualize data
import matplotlib.pyplot as plt

# To used time-related functions
import time

# To parse JSON data
import json

# To implement progress bar related functionalities
from tqdm import tqdm
tqdm.pandas()

# To ignore unnecessary warnings
import warnings
warnings.filterwarnings('ignore')
```

Loading the data

In [ ]:
```python
# Mounting google drive and initilizing path variable
from google.colab import drive
drive.mount("/content/drive")
path = '/content/drive/MyDrive/PGPAIML/Project-6/'
```

```
Mounted at /content/drive
```

In [ ]:
```python
original_data = pd.read_csv(path+'stock_news.csv') #delimiter="\t", encoding='utf-8')
```

In [ ]:
```python
data = original_data.copy()
```

Installing and loading LLM Specific libraries

In [ ]:
```python
# Installation for GPU llama-cpp-python
!CMAKE_ARGS="-DLLAMA_CUBLAS=on" FORCE_CMAKE=1 pip install llama-cpp-python==0.2.45 --force-reinstall --no-cache
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 36.7/36.7 MB 105.1 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Installing backend dependencies ... done
  Preparing metadata (pyproject.toml) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.0/62.0 kB 174.0 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45.5/45.5 kB 246.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.6/134.6 kB 289.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.4/16.4 MB 259.3 MB/s eta 0:00:00
  Building wheel for llama-cpp-python (pyproject.toml) ... done
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This
behaviour is the source of the following dependency conflicts.
pytensor 2.27.1 requires numpy<2,>=1.17.0, but you have numpy 2.2.3 which is incompatible.
numba 0.61.0 requires numpy<2.2,>=1.24, but you have numpy 2.2.3 which is incompatible.
torch 2.5.1+cu124 requires nvidia-cublas-cu12==12.4.5.8; platform_system == "Linux" and platform_machine == "x86
_64", but you have nvidia-cublas-cu12 12.5.3.2 which is incompatible.
torch 2.5.1+cu124 requires nvidia-cuda-cupti-cu12==12.4.127; platform_system == "Linux" and platform_machine ==
"x86_64", but you have nvidia-cuda-cupti-cu12 12.5.82 which is incompatible.
torch 2.5.1+cu124 requires nvidia-cuda-nvrtc-cu12==12.4.127; platform_system == "Linux" and platform_machine ==
"x86_64", but you have nvidia-cuda-nvrtc-cu12 12.5.82 which is incompatible.
torch 2.5.1+cu124 requires nvidia-cuda-runtime-cu12==12.4.127; platform_system == "Linux" and platform_machine =
= "x86_64", but you have nvidia-cuda-runtime-cu12 12.5.82 which is incompatible.
torch 2.5.1+cu124 requires nvidia-cudnn-cu12==9.1.0.70; platform_system == "Linux" and platform_machine == "x86_
64", but you have nvidia-cudnn-cu12 9.3.0.75 which is incompatible.
torch 2.5.1+cu124 requires nvidia-cufft-cu12==11.2.1.3; platform_system == "Linux" and platform_machine == "x86_
64", but you have nvidia-cufft-cu12 11.2.3.61 which is incompatible.
torch 2.5.1+cu124 requires nvidia-curand-cu12==10.3.5.147; platform_system == "Linux" and platform_machine == "x
86_64", but you have nvidia-curand-cu12 10.3.6.82 which is incompatible.
torch 2.5.1+cu124 requires nvidia-cusolver-cu12==11.6.1.9; platform_system == "Linux" and platform_machine == "x
86_64", but you have nvidia-cusolver-cu12 11.6.3.83 which is incompatible.
torch 2.5.1+cu124 requires nvidia-cusparse-cu12==12.3.1.170; platform_system == "Linux" and platform_machine ==
"x86_64", but you have nvidia-cusparse-cu12 12.5.1.3 which is incompatible.
torch 2.5.1+cu124 requires nvidia-nvjitlink-cu12==12.4.127; platform_system == "Linux" and platform_machine == "
x86_64", but you have nvidia-nvjitlink-cu12 12.5.82 which is incompatible.
thinc 8.2.5 requires numpy<2.0.0,>=1.19.0; python_version >= "3.9", but you have numpy 2.2.3 which is incompatib
le.
langchain 0.3.19 requires numpy<2,>=1.26.4; python_version < "3.12", but you have numpy 2.2.3 which is incompati
ble.
tensorflow 2.18.0 requires numpy<2.1.0,>=1.26.0, but you have numpy 2.2.3 which is incompatible.
gensim 4.3.3 requires numpy<2.0,>=1.18.5, but you have numpy 2.2.3 which is incompatible.
```

In [ ]:
```python
# Importing the Llama class from the llama_cpp module
# For downloading the models from HF Hub
!pip install huggingface_hub==0.20.3 -q
# Importing library for data manipulation
import pandas as pd
```

```python
# Function to download the model from the Hugging Face model hub
from huggingface_hub import hf_hub_download

# Importing the Llama class from the llama_cpp module
from llama_cpp import Llama

# Importing the json module
import json
```

```
                                              0.0/330.1 kB ? eta -:--:--
                                              71.7/330.1 kB 2.6 MB/s eta 0:00:01
                                              330.1/330.1 kB 5.1 MB/s eta 0:00:00
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This
behaviour is the source of the following dependency conflicts.
peft 0.14.0 requires huggingface-hub>=0.25.0, but you have huggingface-hub 0.20.3 which is incompatible.
transformers 4.48.3 requires huggingface-hub<1.0,>=0.24.0, but you have huggingface-hub 0.20.3 which is incompat
ible.
diffusers 0.32.2 requires huggingface-hub>=0.23.2, but you have huggingface-hub 0.20.3 which is incompatible.
accelerate 1.3.0 requires huggingface-hub>=0.21.0, but you have huggingface-hub 0.20.3 which is incompatible.
```

Loading the model

In [ ]:
```python
## Model configuration
model_name_or_path = "TheBloke/Llama-2-13B-chat-GGUF"
model_basename = "llama-2-13b-chat.Q5_K_M.gguf"
model_path = hf_hub_download(
    repo_id=model_name_or_path,
    filename=model_basename
    )
```

```
llama-2-13b-chat.Q5_K_M.gguf:   0%|          | 0.00/9.23G [00:00<?, ?B/s]
```

In [ ]:
```python
llm = Llama(
    model_path=model_path, # Path to the model
    n_gpu_layers=100, #Number of layers transferred to GPU
    n_ctx=4500, #Context window
)
```

```
llama_model_loader: loaded meta data with 19 key-value pairs and 363 tensors from /root/.cache/huggingface/hub/m
odels--TheBloke--Llama-2-13B-chat-GGUF/snapshots/4458acc949de0a9914c3eab623904d4fe999050a/llama-2-13b-chat.Q5_K_
M.gguf (version GGUF V2)
llama_model_loader: Dumping metadata keys/values. Note: KV overrides do not apply in this output.
llama_model_loader: - kv   0:                       general.architecture str              = llama
llama_model_loader: - kv   1:                               general.name str              = LLaMA v2
llama_model_loader: - kv   2:                       llama.context_length u32              = 4096
llama_model_loader: - kv   3:                     llama.embedding_length u32              = 5120
llama_model_loader: - kv   4:                          llama.block_count u32              = 40
llama_model_loader: - kv   5:                  llama.feed_forward_length u32              = 13824
llama_model_loader: - kv   6:                 llama.rope.dimension_count u32              = 128
llama_model_loader: - kv   7:                 llama.attention.head_count u32              = 40
llama_model_loader: - kv   8:              llama.attention.head_count_kv u32              = 40
llama_model_loader: - kv   9:     llama.attention.layer_norm_rms_epsilon f32              = 0.000010
llama_model_loader: - kv  10:                          general.file_type u32              = 17
llama_model_loader: - kv  11:                       tokenizer.ggml.model str              = llama
llama_model_loader: - kv  12:                      tokenizer.ggml.tokens arr[str,32000]   = ["<unk>", "<s>", "</
s>", "<0x00>", "<...
llama_model_loader: - kv  13:                      tokenizer.ggml.scores arr[f32,32000]   = [0.000000, 0.000000,
0.000000, 0.0000...
llama_model_loader: - kv  14:                  tokenizer.ggml.token_type arr[i32,32000]   = [2, 3, 3, 6, 6, 6, 6
, 6, 6, 6, 6, 6, ...
llama_model_loader: - kv  15:                tokenizer.ggml.bos_token_id u32              = 1
llama_model_loader: - kv  16:                tokenizer.ggml.eos_token_id u32              = 2
llama_model_loader: - kv  17:            tokenizer.ggml.unknown_token_id u32              = 0
llama_model_loader: - kv  18:               general.quantization_version u32              = 2
llama_model_loader: - type  f32:   81 tensors
llama_model_loader: - type q5_K:  241 tensors
llama_model_loader: - type q6_K:   41 tensors
llm_load_vocab: special tokens definition check successful ( 259/32000 ).
llm_load_print_meta: format           = GGUF V2
llm_load_print_meta: arch             = llama
llm_load_print_meta: vocab type       = SPM
llm_load_print_meta: n_vocab          = 32000
llm_load_print_meta: n_merges         = 0
llm_load_print_meta: n_ctx_train      = 4096
llm_load_print_meta: n_embd           = 5120
llm_load_print_meta: n_head           = 40
llm_load_print_meta: n_head_kv        = 40
llm_load_print_meta: n_layer          = 40
llm_load_print_meta: n_rot            = 128
llm_load_print_meta: n_embd_head_k    = 128
llm_load_print_meta: n_embd_head_v    = 128
llm_load_print_meta: n_gqa            = 1
llm_load_print_meta: n_embd_k_gqa     = 5120
```

```
llm_load_print_meta: n_embd_v_gqa     = 5120
llm_load_print_meta: f_norm_eps       = 0.0e+00
llm_load_print_meta: f_norm_rms_eps   = 1.0e-05
llm_load_print_meta: f_clamp_kqv      = 0.0e+00
llm_load_print_meta: f_max_alibi_bias = 0.0e+00
llm_load_print_meta: n_ff             = 13824
llm_load_print_meta: n_expert         = 0
llm_load_print_meta: n_expert_used    = 0
llm_load_print_meta: rope scaling     = linear
llm_load_print_meta: freq_base_train  = 10000.0
llm_load_print_meta: freq_scale_train = 1
llm_load_print_meta: n_yarn_orig_ctx  = 4096
llm_load_print_meta: rope_finetuned   = unknown
llm_load_print_meta: model type       = 13B
llm_load_print_meta: model ftype      = Q5_K - Medium
llm_load_print_meta: model params     = 13.02 B
llm_load_print_meta: model size       = 8.60 GiB (5.67 BPW)
llm_load_print_meta: general.name     = LLaMA v2
llm_load_print_meta: BOS token        = 1 '<s>'
llm_load_print_meta: EOS token        = 2 '</s>'
llm_load_print_meta: UNK token        = 0 '<unk>'
llm_load_print_meta: LF token         = 13 '<0x0A>'
llm_load_tensors: ggml ctx size =    0.28 MiB
llm_load_tensors: offloading 40 repeating layers to GPU
llm_load_tensors: offloading non-repeating layers to GPU
llm_load_tensors: offloaded 41/41 layers to GPU
llm_load_tensors:        CPU buffer size =   107.42 MiB
llm_load_tensors:      CUDA0 buffer size =  8694.21 MiB
...................................................................................................
llama_new_context_with_model: n_ctx      = 4500
llama_new_context_with_model: freq_base  = 10000.0
llama_new_context_with_model: freq_scale = 1
llama_kv_cache_init:      CUDA0 KV buffer size =  3515.62 MiB
llama_new_context_with_model: KV self size  = 3515.62 MiB, K (f16): 1757.81 MiB, V (f16): 1757.81 MiB
llama_new_context_with_model:  CUDA_Host input buffer size   =    19.83 MiB
llama_new_context_with_model:      CUDA0 compute buffer size =   400.37 MiB
llama_new_context_with_model:  CUDA_Host compute buffer size =    10.00 MiB
llama_new_context_with_model: graph splits (measure): 3
AVX = 1 | AVX_VNNI = 0 | AVX2 = 1 | AVX512 = 1 | AVX512_VBMI = 0 | AVX512_VNNI = 0 | FMA = 1 | NEON = 0 | ARM_FM
A = 0 | F16C = 1 | FP16_VA = 0 | WASM_SIMD = 0 | BLAS = 1 | SSE3 = 1 | SSSE3 = 1 | VSX = 0 | MATMUL_INT8 = 0 |
```


Model metadata: {'tokenizer.ggml.unknown_token_id': '0', 'tokenizer.ggml.eos_token_id': '2', 'general.architectu
re': 'llama', 'llama.context_length': '4096', 'general.name': 'LLaMA v2', 'llama.embedding_length': '5120', 'lla
ma.feed_forward_length': '13824', 'llama.attention.layer_norm_rms_epsilon': '0.000010', 'llama.rope.dimension_co
unt': '128', 'llama.attention.head_count': '40', 'tokenizer.ggml.bos_token_id': '1', 'llama.block_count': '40',
'llama.attention.head_count_kv': '40', 'general.quantization_version': '2', 'tokenizer.ggml.model': 'llama', 'ge
neral.file_type': '17'}

Aggregating the data weekly

In [ ]:
```python
data["Date"] = pd.to_datetime(data['Date'])  # Convert the 'Date' column to datetime format.
```

In [ ]:
```python
# Group the data by week using the 'Date' column.
weekly_grouped = data.groupby(pd.Grouper(key='Date', freq='W'))
```

In [ ]:
```python
weekly_grouped = weekly_grouped.agg(
    {
        'News': lambda x: ' || '.join(x)  # Join the news values with ' || ' separator.
    }
).reset_index()

print(weekly_grouped.shape)
```

```
(18, 2)
```

In [ ]:
```python
weekly_grouped.shape
```

Out[ ]:
```
(18, 2)
```

In [ ]:
```python
# creating a copy of the data
data_1 = weekly_grouped.copy()
```

Utility Functions

In [ ]:
```python
#Defining the response function
# IMPORTANT INFO
# The prompt at the end --> '> ```json' helped enforce the model to generate JSON Reference: https://www.reddit

def generate_response(prompt, news):
    input_text =   f"""
      [INST]
      {prompt}
      [/INST]
```

```
        News Articles: {news}
        > ```json
        """
    # print(input_text)
    model_output = llm(
      input_text,
      max_tokens=1024,
      temperature=0,
      top_p=0.95,
      repeat_penalty=1.2,
      top_k=50,
      stop=['>```', 'INST'], #<-- added '>```' as a stop sequence to stop generating anything after the JSON Re
      echo=False,
      seed=1,
    )

    final_output = model_output["choices"][0]["text"]
    return final_output
```

In [ ]:
```
# defining a function to parse the JSON output from the model
def get_json_data(json_str):
    import json
    try:
        # Find the indices of the opening and closing curly braces
        json_start = json_str.find('{')
        json_end = json_str.rfind('}')

        if json_start != -1 and json_end != -1:
            extracted_category = json_str[json_start:json_end + 1]  # Extract the JSON object
            data_dict = json.loads(extracted_category)
            return data_dict
        else:
            print(f"Warning: JSON object not found in response: {json_str}")
            return {}
    except json.JSONDecodeError as e:
        print(f"Error parsing JSON: {e}")
        return {}
```

Create a suitable prompt to use for the LLM to get the responses in the desired JSON format

In [ ]:
```
# Creating the prompt

# IMPORTANT NOTE
# After lot of experiments, this prompt worked best to return JSON data from the LLaMa model seletecd to be use
# Even with this prompt, the model generated invalid JSON at least one instance.

prompt = """

### Role:
You are an expert financial analyst specializing in **stock market impact analysis**. Your expertise lies in an

### Instructions:
1. **Read all the news articles** carefully. Each article is separated by "||".
2. **Identify events** within the news articles that are likely to influence stock prices positively or negativ
3. **Classify events** as **Positive** if they are expected to boost stock prices (e.g., strong earnings report
4. **Classify events** as **Negative** if they are likely to lower stock prices (e.g., poor earnings reports, r
5. **Rank the top three** most impactful positive and negative events based on their expected influence on the
6. **Ensure conciseness** in summarizing events while retaining critical financial implications.

### Task:
Analyze the provided **weekly news data** to determine the **top three positive** and **top three negative even

{
  positive:{"1":"first_positive_event_here", "2":"second_positive_event_here", "3":"third_positive_event_here"}
  negative:{"1":"first_negative_event_here", "2":"second_negative_event_here", "3":"third_negative_event_here"}
}
"""
```

Checking the model output on a sample

In [ ]:
```
news = data_1.loc[3, 'News']
print(len(news.split(' ')))
news
```

1038

Out[ ]: ' The Swiss National Bank (SNB) governor, Andrea Maechler, stated that negative interest rates and foreign currency market intervention are necessary to prevent a strong Swiss franc from causing deflation in the country. She emphasized that price stability is the bank\'s mandate, and the exchange rate significantly impacts monetary conditions and inflation. Switzerland, || The Dow, S&P 500, and Nasdaq experienced significant losses on Tuesday despite White House economic adviser Lawrence Kudlow denying reports that trade talks between the U.S. and China had been canceled. The International Monetary Fund\'s bearish outlook on global growth and weak existing home sales data also || IBM\'s stock price increased after hours due to better-than-expected earnings and revenue, with its cloud computing business contributing positively. || Huawei is expanding its presence in Europe with the launch of the new Honor View20 smartphone, which offers advanced camera features at a lower price point than rivals Samsung and Apple. The phone includes a 48 mega pixel camera that combines multiple images into one high-quality photo, as well as a second camera for || Foxconn, the world\'s largest contract manufacturer for Apple iPhones, is trying to recruit more than 50,000 employees across its China campuses in Q1 2019 amid reports of mass layoffs. Last week, the Nikkei reported that Foxconn had let go around 50,0 || Amazon is launching its long-awaited direct fulfillment and delivery network in Brazil after facing complications from the country\'s complex tax system and logistics. Starting Tuesday, Amazon will directly sell over 800 suppliers\' merchandise, including L\'Oreal and Black & Decker, totaling 320,0 || Tesla is considering Tianjin Lishen as a potential battery supplier for its new Shanghai electric car factory, but no agreement has been reached yet. The companies are still negotiating details such as the size of the order and the battery cell size required by Tesla. Lishen had previously quoted Tesla for batteries, but no agreement was signed || TomTom, a Dutch digital mapping company, agreed to sell its fleet management business to Bridgestone for €910 million. The sale comes as TomTom faces competition from Google, which has entered the market and struck deals with Renault and Volvo. Despite concerns over future uncertainty, CEO Harold Goddijn plans to keep the remaining || Japan Display shares surged on reports of potential funding from Taiwan\'s TPK Holding and China\'s Silk Road Fund, in exchange for a 30% stake. Discussions were described as advanced, with the company previously denying similar reports. The Apple supplier has faced losses due to competition from Chinese rivals and slow smart || The White House reportedly rejected a scheduled meeting with Chinese officials this week due to disagreements over intellectual property rules, causing cautious trading in Asian stocks. China\'s Shanghai Composite and Shenzhen Component saw minimal changes, while the Hang Seng Index edged higher. The U.S. denied cancelling the meeting || Trump expressed optimism about ongoing trade negotiations with China, stating that the U.S. was doing well and that China wants to make a deal. However, Trump also threatened to increase tariffs on Chinese imports unless China addresses intellectual property concerns and other trade issues. The White House economic adviser believes a deal could be reached by March 1, || Texas Inquiries reported fourth-quarter earnings exceeding analysts\' expectations, but missed revenue forecasts due to weak global smartphone sales. The company expects lower first-quarter earnings and revenue, leading to a slight increase in share price during after-hours trading. TI earned $1.26 per share on $3 || FireEye\'s stock price surged after Baird added it to their "Fresh Picks" list, citing a recent decline in shares and confident 2019 guidance. With an outperform rating and $23 price target, analysts expect a strong year for the cybersecurity company\'s major products like Man || IBM, Procter & Gamble, United Technologies, and Comcast stocks surged in premarket trade on Wednesday due to better-than-expected fourth quarter results and raised full year profit guidance or forecasts. Capital One and Kimberly Clark stocks declined as they reported earnings below analysts\' estimates. Apple stock rose amid a report || In 2018, Google and Facebook broke their previous records for annual lobbying expenditures. Google spent $21 million, up from $18 million in 2017, with nearly half spent in Q4 when CEO Sundar Pichai testified before Congress. Facebook shelled out $13 million, || Apple, under new leadership in Project Titan, has let go over 200 team members from its autonomous vehicle unit. The dismissals were anticipated internally and Doug Field, an Apple veteran and former Tesla engineering VP, is now co-leading the project with Bob Mansfield. Affected employees are reportedly moving to || The U.S. and China are far from resolving their trade disputes, but there\'s a possibility they will reach an agreement before the March 1 deadline, according to U.S. Commerce Secretary Wilbur Ross. A Chinese delegation of 30 members is set to visit Washington next week for trade talks. Ross downplay || Starbucks, which introduced Europe\'s coffee culture to Americans, is facing pressure from Wall Street to replicate success in China. Despite opening stores at a rate of nearly 600 per year, sales growth has slowed due to an economic downturn and the US-China trade war. Same store sales in China were up only || Mastercard, a U.S. payments card company listed on NYSE as MA, is still determined to apply for a bankcard clearing license in China despite voluntarily withdrawing an application in 2018. The company aims to present another application soon, as American Express became the first U.S. card network to gain direct access || Mastercard continues its pursuit of a bankcard clearing license in China, with plans to submit another application soon. The company previously applied in 2017 but withdrew voluntarily. American Express recently became the first US card network to gain direct access to China\'s payments network, bypassing UnionPay\'s monopoly. Master || The Indian Cellular and Electronics Association (ICEA) representing major smartphone makers such as Apple, Samsung, Oppo, and Foxconn, among others, submitted a 174-page document to the government calling for increased export credits on devices, tariff cuts on machinery imports, and other measures aimed at making India a'

In [ ]:
```
%%time
summary = generate_response(prompt, news)
print(summary)
```

```
llama_print_timings:        load time =    1002.10 ms
llama_print_timings:      sample time =     106.82 ms /   202 runs   (    0.53 ms per token,  1891.08 tokens per second)
llama_print_timings: prompt eval time =    3758.97 ms /  1948 tokens (    1.93 ms per token,   518.23 tokens per second)
llama_print_timings:        eval time =   11575.53 ms /   201 runs   (   57.59 ms per token,    17.36 tokens per second)
llama_print_timings:       total time =   15997.93 ms /  2149 tokens
```

```
 {
         "positive": {
           "1": "IBM's better-than-expected earnings and revenue boost the company's stock price",
           "2": "Huawei's expansion into Europe with the launch of the new Honor View20 smartphone",
           "3": "Apple's new Shanghai electric car factory is considering Tianjin Lishen as a potential battery
supplier"
         },
         "negative": {
           "1": "Foxconn's layoffs of over 50,000 employees across its China campuses",

           "2": "Tesla's potential funding from Taiwan's TPK Holding and China's Silk Road Fund for Japan Displa
y",
           "3": "The U.S. and China's ongoing trade disputes, with no agreement reached yet"
         }
       }
```
 {
         "positive": {
           "1": "IBM's better-than-expected earnings and revenue boost the company's stock price",
           "2": "Huawei's expansion into Europe with the launch of the new Honor View20 smartphone",
           "3": "Apple's new Shanghai electric car factory is considering Tianjin Lishen as a potential battery
supplier"
         },
         "negative": {
           "1": "Foxconn's layoffs of over 50,000 employees across its China campuses",
           "2": "Tesla's potential funding from Taiwan's TPK Holding and China's Silk Road Fund for Japan Displa
y",
           "3": "The U.S. and China's ongoing trade disputes, with no agreement reached yet"
         }
       }
```
CPU times: user 14.3 s, sys: 1.7 s, total: 16 s
Wall time: 16 s
```

In [ ]:
```
get_json_data(summary)
```

Out[ ]:
```
{'positive': {'1': "IBM's better-than-expected earnings and revenue boost the company's stock price",
  '2': "Huawei's expansion into Europe with the launch of the new Honor View20 smartphone",
  '3': "Apple's new Shanghai electric car factory is considering Tianjin Lishen as a potential battery supplier
"},
 'negative': {'1': "Foxconn's layoffs of over 50,000 employees across its China campuses",
  '2': "Tesla's potential funding from Taiwan's TPK Holding and China's Silk Road Fund for Japan Display",
  '3': "The U.S. and China's ongoing trade disputes, with no agreement reached yet"}}
```

In [ ]:
```
model_response_parsed = pd.json_normalize(get_json_data(summary), max_level=0)
model_response_parsed['positive'] = model_response_parsed['positive'].astype(str).str.replace("{","");
model_response_parsed['positive'] = model_response_parsed['positive'].astype(str).str.replace("}","");
model_response_parsed['positive'] = model_response_parsed['positive'].astype(str).str.replace(", ","\n");

model_response_parsed['negative'] = model_response_parsed['negative'].astype(str).str.replace("{","");
model_response_parsed['negative'] = model_response_parsed['negative'].astype(str).str.replace("}","");
model_response_parsed['negative'] = model_response_parsed['negative'].astype(str).str.replace(", ","\n");
positive = model_response_parsed.loc[0, 'positive']
negative = model_response_parsed.loc[0, 'negative']
print('POSITIVE EVENTS:\n' + positive)
print('NEGATIVE EVENTS\n' + negative)
```

```
POSITIVE EVENTS:
'1': "IBM's better-than-expected earnings and revenue boost the company's stock price"
'2': "Huawei's expansion into Europe with the launch of the new Honor View20 smartphone"
'3': "Apple's new Shanghai electric car factory is considering Tianjin Lishen as a potential battery supplier"
NEGATIVE EVENTS
'1': "Foxconn's layoffs of over 50,000 employees across its China campuses"
'2': "Tesla's potential funding from Taiwan's TPK Holding and China's Silk Road Fund for Japan Display"
'3': "The U.S. and China's ongoing trade disputes
with no agreement reached yet"
```

Observations

The model generated this particular JSON output perfectly. But this is not the case for all news feds to the model with the specified prompt. In at least one case, given the concatenated news articles, the model generated invalid JSON.

Generating the model output on the weekly data based on the engineered prompt.

In [ ]:
```
%%time
data_1['Key Events'] = data_1['News'].progress_apply(lambda x: generate_response(prompt,x))
```

```
  0%|          | 0/18 [00:00<?, ?it/s]Llama.generate: prefix-match hit

llama_print_timings:        load time =    1002.10 ms
llama_print_timings:      sample time =      32.12 ms /    61 runs   (    0.53 ms per token,  1898.83 tokens per
second)
llama_print_timings: prompt eval time =    9848.65 ms /  4043 tokens (    2.44 ms per token,   410.51 tokens per
```

```
second)
llama_print_timings:        eval time =    7886.82 ms /    60 runs   (  131.45 ms per token,     7.61 tokens per
second)
llama_print_timings:       total time =   17988.42 ms /  4103 tokens
 11%|█         | 2/18 [00:18<02:24,  9.01s/it]Llama.generate: prefix-match hit

llama_print_timings:        load time =    1002.10 ms
llama_print_timings:      sample time =      75.65 ms /   144 runs   (    0.53 ms per token,  1903.60 tokens per
second)
llama_print_timings: prompt eval time =    5732.99 ms /  2371 tokens (    2.42 ms per token,   413.57 tokens per
second)
llama_print_timings:        eval time =   10812.58 ms /   143 runs   (   75.61 ms per token,    13.23 tokens per
second)
llama_print_timings:       total time =   17160.14 ms /  2514 tokens
 17%|██        | 3/18 [00:35<03:06, 12.41s/it]Llama.generate: prefix-match hit

llama_print_timings:        load time =    1002.10 ms
llama_print_timings:      sample time =      60.23 ms /   115 runs   (    0.52 ms per token,  1909.22 tokens per
second)
llama_print_timings: prompt eval time =    4969.83 ms /  2223 tokens (    2.24 ms per token,   447.30 tokens per
second)
llama_print_timings:        eval time =    7964.95 ms /   114 runs   (   69.87 ms per token,    14.31 tokens per
second)
llama_print_timings:       total time =   13339.97 ms /  2337 tokens
 22%|██        | 4/18 [00:48<02:58, 12.76s/it]Llama.generate: prefix-match hit

llama_print_timings:        load time =    1002.10 ms
llama_print_timings:      sample time =     106.27 ms /   202 runs   (    0.53 ms per token,  1900.77 tokens per
second)
llama_print_timings: prompt eval time =    3519.65 ms /  1551 tokens (    2.27 ms per token,   440.67 tokens per
second)
llama_print_timings:        eval time =   13169.15 ms /   201 runs   (   65.52 ms per token,    15.26 tokens per
second)
llama_print_timings:       total time =   17394.34 ms /  1752 tokens
 28%|███       | 5/18 [01:05<03:06, 14.38s/it]Llama.generate: prefix-match hit

llama_print_timings:        load time =    1002.10 ms
llama_print_timings:      sample time =      73.15 ms /   140 runs   (    0.52 ms per token,  1913.80 tokens per
second)
llama_print_timings: prompt eval time =    6063.07 ms /  2725 tokens (    2.22 ms per token,   449.44 tokens per
second)
llama_print_timings:        eval time =   10078.85 ms /   139 runs   (   72.51 ms per token,    13.79 tokens per
second)
llama_print_timings:       total time =   16649.48 ms /  2864 tokens
 33%|███       | 6/18 [01:22<03:01, 15.14s/it]Llama.generate: prefix-match hit

llama_print_timings:        load time =    1002.10 ms
llama_print_timings:      sample time =      62.76 ms /   120 runs   (    0.52 ms per token,  1912.08 tokens per
second)
llama_print_timings: prompt eval time =    2617.92 ms /  1178 tokens (    2.22 ms per token,   449.98 tokens per
second)
llama_print_timings:        eval time =    7969.20 ms /   119 runs   (   66.97 ms per token,    14.93 tokens per
second)
llama_print_timings:       total time =   10983.92 ms /  1297 tokens
 39%|████      | 7/18 [01:33<02:31, 13.80s/it]Llama.generate: prefix-match hit

llama_print_timings:        load time =    1002.10 ms
llama_print_timings:      sample time =      90.10 ms /   171 runs   (    0.53 ms per token,  1897.89 tokens per
second)
llama_print_timings: prompt eval time =    2906.85 ms /  1337 tokens (    2.17 ms per token,   459.95 tokens per
second)
llama_print_timings:        eval time =   11717.26 ms /   170 runs   (   68.93 ms per token,    14.51 tokens per
second)
llama_print_timings:       total time =   15229.88 ms /  1507 tokens
 44%|████      | 8/18 [01:48<02:22, 14.26s/it]Llama.generate: prefix-match hit

llama_print_timings:        load time =    1002.10 ms
llama_print_timings:      sample time =      96.80 ms /   180 runs   (    0.54 ms per token,  1859.58 tokens per
second)
llama_print_timings: prompt eval time =    1465.46 ms /   598 tokens (    2.45 ms per token,   408.06 tokens per
second)
llama_print_timings:        eval time =   11626.37 ms /   179 runs   (   64.95 ms per token,    15.40 tokens per
second)
llama_print_timings:       total time =   13763.40 ms /   777 tokens
 50%|█████     | 9/18 [02:02<02:06, 14.11s/it]Llama.generate: prefix-match hit

llama_print_timings:        load time =    1002.10 ms
llama_print_timings:      sample time =      88.31 ms /   168 runs   (    0.53 ms per token,  1902.41 tokens per
second)
llama_print_timings: prompt eval time =    1603.27 ms /   679 tokens (    2.36 ms per token,   423.51 tokens per
second)
llama_print_timings:        eval time =   10690.94 ms /   167 runs   (   64.02 ms per token,    15.62 tokens per
second)
```

```
llama_print_timings:       total time =   12936.17 ms /   846 tokens
 56%|█████▌    | 10/18 [02:15<01:50, 13.75s/it]Llama.generate: prefix-match hit

llama_print_timings:        load time =    1002.10 ms
llama_print_timings:      sample time =     167.22 ms /   318 runs   (    0.53 ms per token,  1901.73 tokens per second)
llama_print_timings: prompt eval time =    1789.78 ms /   771 tokens (    2.32 ms per token,   430.78 tokens per second)
llama_print_timings:        eval time =   20189.89 ms /   317 runs   (   63.69 ms per token,    15.70 tokens per second)
llama_print_timings:       total time =   23193.60 ms /  1088 tokens
 61%|██████    | 11/18 [02:38<01:56, 16.63s/it]Llama.generate: prefix-match hit

llama_print_timings:        load time =    1002.10 ms
llama_print_timings:      sample time =      68.25 ms /   130 runs   (    0.53 ms per token,  1904.68 tokens per second)
llama_print_timings: prompt eval time =    2672.32 ms /  1231 tokens (    2.17 ms per token,   460.65 tokens per second)
llama_print_timings:        eval time =    8516.19 ms /   129 runs   (   66.02 ms per token,    15.15 tokens per second)
llama_print_timings:       total time =   11671.89 ms /  1360 tokens
 67%|██████▋   | 12/18 [02:50<01:30, 15.13s/it]Llama.generate: prefix-match hit

llama_print_timings:        load time =    1002.10 ms
llama_print_timings:      sample time =      73.99 ms /   141 runs   (    0.52 ms per token,  1905.71 tokens per second)
llama_print_timings: prompt eval time =    2895.20 ms /  1342 tokens (    2.16 ms per token,   463.53 tokens per second)
llama_print_timings:        eval time =    9366.93 ms /   140 runs   (   66.91 ms per token,    14.95 tokens per second)
llama_print_timings:       total time =   12781.68 ms /  1482 tokens
 72%|███████▏  | 13/18 [03:03<01:12, 14.42s/it]Llama.generate: prefix-match hit

llama_print_timings:        load time =    1002.10 ms
llama_print_timings:      sample time =      77.75 ms /   148 runs   (    0.53 ms per token,  1903.61 tokens per second)
llama_print_timings: prompt eval time =    3846.18 ms /  1763 tokens (    2.18 ms per token,   458.38 tokens per second)
llama_print_timings:        eval time =   10114.45 ms /   147 runs   (   68.81 ms per token,    14.53 tokens per second)
llama_print_timings:       total time =   14525.45 ms /  1910 tokens
 78%|███████▊  | 14/18 [03:17<00:57, 14.46s/it]Llama.generate: prefix-match hit

llama_print_timings:        load time =    1002.10 ms
llama_print_timings:      sample time =      71.24 ms /   136 runs   (    0.52 ms per token,  1908.93 tokens per second)
llama_print_timings: prompt eval time =    1947.91 ms /   918 tokens (    2.12 ms per token,   471.27 tokens per second)
llama_print_timings:        eval time =    8848.37 ms /   135 runs   (   65.54 ms per token,    15.26 tokens per second)
llama_print_timings:       total time =   11282.30 ms /  1053 tokens
 83%|████████▎ | 15/18 [03:29<00:40, 13.50s/it]Llama.generate: prefix-match hit

llama_print_timings:        load time =    1002.10 ms
llama_print_timings:      sample time =     113.70 ms /   215 runs   (    0.53 ms per token,  1890.97 tokens per second)
llama_print_timings: prompt eval time =    1668.47 ms /   538 tokens (    3.10 ms per token,   322.45 tokens per second)
llama_print_timings:        eval time =   13544.90 ms /   214 runs   (   63.29 ms per token,    15.80 tokens per second)
llama_print_timings:       total time =   16014.48 ms /   752 tokens
 89%|████████▉ | 16/18 [03:45<00:28, 14.26s/it]Llama.generate: prefix-match hit

llama_print_timings:        load time =    1002.10 ms
llama_print_timings:      sample time =      72.32 ms /   136 runs   (    0.53 ms per token,  1880.61 tokens per second)
llama_print_timings: prompt eval time =    3204.28 ms /  1535 tokens (    2.09 ms per token,   479.05 tokens per second)
llama_print_timings:        eval time =    9193.25 ms /   135 runs   (   68.10 ms per token,    14.68 tokens per second)
llama_print_timings:       total time =   12911.63 ms /  1670 tokens
 94%|█████████▍| 17/18 [03:58<00:13, 13.86s/it]Llama.generate: prefix-match hit

llama_print_timings:        load time =    1002.10 ms
llama_print_timings:      sample time =      70.53 ms /   134 runs   (    0.53 ms per token,  1899.85 tokens per second)
llama_print_timings: prompt eval time =    1970.63 ms /   933 tokens (    2.11 ms per token,   473.45 tokens per second)
llama_print_timings:        eval time =    8798.51 ms /   133 runs   (   66.15 ms per token,    15.12 tokens per second)
llama_print_timings:       total time =   11209.39 ms /  1066 tokens
100%|██████████| 18/18 [04:09<00:00, 13.07s/it]Llama.generate: prefix-match hit
```

```
llama_print_timings:        load time =    1002.10 ms
llama_print_timings:      sample time =     119.92 ms /   230 runs   (    0.52 ms per token,  1917.96 tokens per second)
llama_print_timings: prompt eval time =    1973.71 ms /   942 tokens (    2.10 ms per token,   477.27 tokens per second)
llama_print_timings:        eval time =   15149.75 ms /   229 runs   (   66.16 ms per token,    15.12 tokens per second)
llama_print_timings:       total time =   17887.08 ms /  1171 tokens
100%|██████████| 18/18 [04:27<00:00, 14.84s/it]
CPU times: user 3min 54s, sys: 33.6 s, total: 4min 28s
Wall time: 4min 27s
```

Extracting the model output to parsable JSON format

In [ ]:
```python
data_1['model_response_parsed'] = data_1['Key Events'].apply(get_json_data)
```

```
Warning: JSON object not found in response:  * Dow Jones Industrial average, the Dow Jones, Inc. average, Inc. average, Inc. Dow Jones, Inc. Dow Jones, Inc. Dow Jones, Inc. average, Inc. Dow Jones, Inc. Dow Jones, Inc. Dow Jones, Inc. Dow Jones, Inc. Dow
```

Formatting the model output

In [ ]:
```python
# Splitting the generated JSON into POSITIVE & NEGETIVE collumns
model_response_parsed = pd.json_normalize(data_1['model_response_parsed'], max_level=0)

# Formatting the summarized news events by removing some characters and adding new-line at the end of each ranke
model_response_parsed['positive'] = model_response_parsed['positive'].astype(str).str.replace("{","");
model_response_parsed['positive'] = model_response_parsed['positive'].astype(str).str.replace("}","");
model_response_parsed['positive'] = model_response_parsed['positive'].astype(str).str.replace(", ","\n");

model_response_parsed['negative'] = model_response_parsed['negative'].astype(str).str.replace("{","");
model_response_parsed['negative'] = model_response_parsed['negative'].astype(str).str.replace("}","");
model_response_parsed['negative'] = model_response_parsed['negative'].astype(str).str.replace(", ","\n");
model_response_parsed
```

Out[ ]:

	positive	negative
0	nan	nan
1	'1': "AMS's light and infrared proximity senso...	'1': "Geely's flat sales forecast for 2019"\n'...
2	'1': "Apple's partnership with Verizon"\n'2': ...	'1': 'Falling iPhone sales'\n'2': "China's wea...
3	'1': "IBM's better-than-expected earnings and ...	'1': "Foxconn's layoffs of over 50,000 employe...
4	'1': "Apple's strong earnings report"\n'2': "A...	'1': 'Weak iPhone sales in China'\n'2': "Corni...
5	'1': 'Apple acquisition target'\n'2': 'Strong ...	'1': 'Fears of a Chinese economic slowdown'\n'...
6	'1': "Apple's cybersecurity and media content ...	'1': 'Belgium may have to recover around €790 ...
7	'1': "Garmin's strong earnings and revenue dri...	'1': 'WhatsApp security bug allows iPhone user...
8	'1': "Huawei's new folding phone\nthe Mate X\n...	'1': "AAC Technologies Holdings' significant d...
9	'1': 'Fitbit introduced its most affordable sm...	'1': "Mozilla\nthe Firefox browser maker\nis c...
10	'1': "Oracle's revenue drop warning"\n'2': "Ap...	'1': "Boeing's 737 Max crashes"\n'2': "Faceboo...
11	'1': "Foxconn's new factory in Wisconsin"\n'2'...	'1': "Facebook's regulatory risks and poor ear...
12	'1': "Apple's new TV service launches with ori...	'1': 'Yield curve inversion raises recession f...
13	'1': "Apple's price cuts in China"\n'2': "Swat...	'1': "Apple's price cuts in China"\n'2': "Sams...
14	'1': "Apple's partnership with Oprah Winfrey a...	'1': 'Mobile phone shipments to China dropped ...
15	'1': "Terry Gou's plan to step down from Foxco...	'1': "Foxconn's reliance on Apple"\n'2': "Qual...
16	'1': "Snap's better-than-expected earnings"\n'...	'1': "Taiwan's declining export orders"\n'2': ...
17	'1': "Samsung Electronics' weakest profit in o...	'1': 'Disappointing earnings from Google press...

In [ ]:
```python
# Creating the final output dataframe, with weekly news articles, summarized top three positive events and top 

final_output = pd.concat([data_1.reset_index(drop=True),model_response_parsed],axis=1)
final_output.drop(['Key Events','model_response_parsed'], axis=1, inplace=True)
final_output.columns = ['Week End Date', 'News', 'Top-3 Weekly positive news events with ranks', 'Top-3 Weekly 

final_output
```

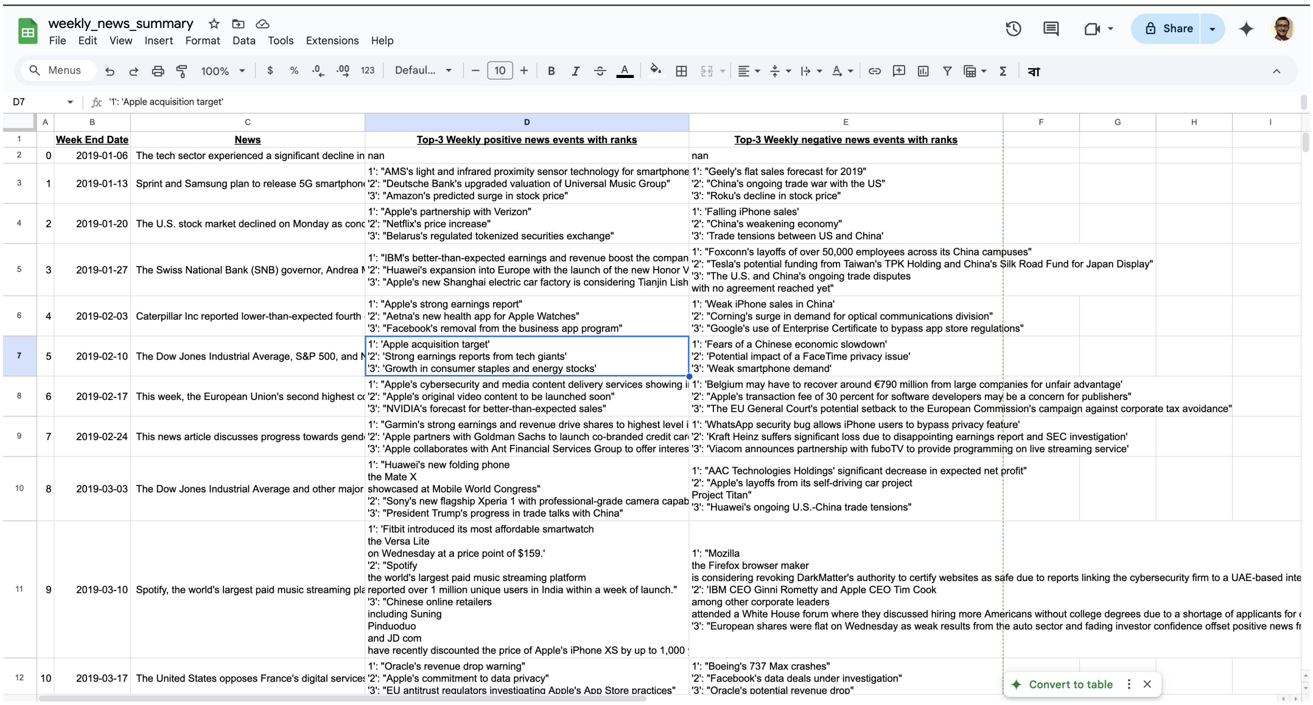
Out[ ]:

	Week End Date	News	Top-3 Weekly positive news events with ranks	Top-3 Weekly negative news events with ranks
0	2019-01-06	The tech sector experienced a significant dec...	nan	nan
1	2019-01-13	Sprint and Samsung plan to release 5G smartph...	'1': "AMS's light and infrared proximity senso...	'1': "Geely's flat sales forecast for 2019"\n'...
2	2019-01-20	The U.S. stock market declined on Monday as c...	'1': "Apple's partnership with Verizon"\n'2': ...	'1': 'Falling iPhone sales'\n'2': "China's wea...
3	2019-01-27	The Swiss National Bank (SNB) governor, Andre...	'1': "IBM's better-than-expected earnings and ...	'1': "Foxconn's layoffs of over 50,000 employe...
4	2019-02-03	Caterpillar Inc reported lower-than-expected ...	'1': "Apple's strong earnings report"\n'2': "A...	'1': 'Weak iPhone sales in China'\n'2': "Corni...
5	2019-02-10	The Dow Jones Industrial Average, S&P 500, an...	'1': 'Apple acquisition target'\n'2': 'Strong ...	'1': 'Fears of a Chinese economic slowdown'\n'...
6	2019-02-17	This week, the European Union's second highes...	'1': "Apple's cybersecurity and media content ...	'1': 'Belgium may have to recover around €790 ...
7	2019-02-24	This news article discusses progress towards ...	'1': "Garmin's strong earnings and revenue dri...	'1': 'WhatsApp security bug allows iPhone user...
8	2019-03-03	The Dow Jones Industrial Average and other ma...	'1': "Huawei's new folding phone\nthe Mate X\n...	'1': "AAC Technologies Holdings' significant d...
9	2019-03-10	Spotify, the world's largest paid music strea...	'1': 'Fitbit introduced its most affordable sm...	'1': "Mozilla\nthe Firefox browser maker\nis c...
10	2019-03-17	The United States opposes France's digital se...	'1': "Oracle's revenue drop warning"\n'2': "Ap...	'1': "Boeing's 737 Max crashes"\n'2': "Faceboo...
11	2019-03-24	Facebook's stock price dropped more than 3% o...	'1': "Foxconn's new factory in Wisconsin"\n'2'...	'1': "Facebook's regulatory risks and poor ear...
12	2019-03-31	This news article reports that the S&P 500 In...	'1': "Apple's new TV service launches with ori...	'1': 'Yield curve inversion raises recession f...
13	2019-04-07	Apple and other consumer brands, including LV...	'1': "Apple's price cuts in China"\n'2': "Swat...	'1': "Apple's price cuts in China"\n'2': "Sams...
14	2019-04-14	In March, mobile phone shipments to China dro...	'1': "Apple's partnership with Oprah Winfrey a...	'1': 'Mobile phone shipments to China dropped ...
15	2019-04-21	The chairman of Taiwan's Foxconn, Terry Gou, ...	'1': "Terry Gou's plan to step down from Foxco...	'1': "Foxconn's reliance on Apple"\n'2': "Qual...
16	2019-04-28	Taiwan's export orders continued to decline f...	'1': "Snap's better-than-expected earnings"\n'...	'1': "Taiwan's declining export orders"\n'2': ...
17	2019-05-05	Spotify reported better-than-expected Q1 reve...	'1': "Samsung Electronics' weakest profit in o...	'1': 'Disappointing earnings from Google press...

Saving the Final output for the users

In [ ]:

```python
# Saving the final output
final_output.to_csv(path+'weekly_news_summary.csv', index=True)
```



	A	B	C	D	E
1		Week End Date	News	Top-3 Weekly positive news events with ranks	Top-3 Weekly negative news events with ranks
2	0	2019-01-06	The tech sector experienced a significant decline in	nan	nan
3	1	2019-01-13	Sprint and Samsung plan to release 5G smartphone	1': "AMS's light and infrared proximity sensor technology for smartphone '2': "Deutsche Bank's upgraded valuation of Universal Music Group" '3': "Amazon's predicted surge in stock price"	1': "Geely's flat sales forecast for 2019" '2': "China's ongoing trade war with the US" '3': "Roku's decline in stock price"
4	2	2019-01-20	The U.S. stock market declined on Monday as conc	1': "Apple's partnership with Verizon" '2': "Netflix's price increase" '3': "Belarus's regulated tokenized securities exchange"	1': 'Falling iPhone sales' '2': "China's weakening economy" '3': 'Trade tensions between US and China'
5	3	2019-01-27	The Swiss National Bank (SNB) governor, Andrea M	1': "IBM's better-than-expected earnings and revenue boost the compan '2': "Huawei's expansion into Europe with the launch of the new Honor V '3': "Apple's new Shanghai electric car factory is considering Tianjin Lish	1': "Foxconn's layoffs of over 50,000 employees across its China campuses" '2': "Tesla's potential funding from Taiwan's TPK Holding and China's Silk Road Fund for Japan Display" '3': "The U.S. and China's ongoing trade disputes with no agreement reached yet"
6	4	2019-02-03	Caterpillar Inc reported lower-than-expected fourth	1': "Apple's strong earnings report" '2': "Aetna's new health app for Apple Watches" '3': "Facebook's removal from the business app program"	1': 'Weak iPhone sales in China' '2': "Corning's surge in demand for optical communications division" '3': "Google's use of Enterprise Certificate to bypass app store regulations"
7	5	2019-02-10	The Dow Jones Industrial Average, S&P 500, and N	1': 'Apple acquisition target' '2': 'Strong earnings reports from tech giants' '3': 'Growth in consumer staples and energy stocks'	1': 'Fears of a Chinese economic slowdown' '2': 'Potential impact of a FaceTime privacy issue' '3': 'Weak smartphone demand'
8	6	2019-02-17	This week, the European Union's second highest cc	1': "Apple's cybersecurity and media content delivery services showing ii '2': "Apple's original video content to be launched soon" '3': "NVIDIA's forecast for better-than-expected sales"	1': 'Belgium may have to recover around €790 million from large companies for unfair advantage' '2': "Apple's transaction fee of 30 percent for software developers may be a concern for publishers" '3': "The EU General Court's potential setback to the European Commission's campaign against corporate tax avoidance"
9	7	2019-02-24	This news article discusses progress towards gend	1': "Garmin's strong earnings and revenue drive shares to highest level i '2': 'Apple partners with Goldman Sachs to launch co-branded credit car '3': 'Apple collaborates with Ant Financial Services Group to offer interes	1': 'WhatsApp security bug allows iPhone users to bypass privacy feature' '2': 'Kraft Heinz suffers significant loss due to disappointing earnings report and SEC investigation' '3': 'Viacom announces partnership with fuboTV to provide programming on live streaming service'
10	8	2019-03-03	The Dow Jones Industrial Average and other major	1': "Huawei's new folding phone the Mate X showcased at Mobile World Congress" '2': "Sony's new flagship Xperia 1 with professional-grade camera capab '3': "President Trump's progress in trade talks with China"	1': "AAC Technologies Holdings' significant decrease in expected net profit" '2': "Apple's layoffs from its self-driving car project Project Titan" '3': "Huawei's ongoing U.S.-China trade tensions"
11	9	2019-03-10	Spotify, the world's largest paid music streaming pla	1': 'Fitbit introduced its most affordable smartwatch the Versa Lite on Wednesday at a price point of $159.' '2': "Spotify the world's largest paid music streaming platform reported over 1 million unique users in India within a week of launch." '3': "Chinese online retailers including Suning Pinduoduo and JD com have recently discounted the price of Apple's iPhone XS by up to 1,000	1': "Mozilla the Firefox browser maker is considering revoking DarkMatter's authority to certify websites as safe due to reports linking the cybersecurity firm to a UAE-based inte '2': 'IBM CEO Ginni Rometty and Apple CEO Tim Cook among other corporate leaders attended a White House forum where they discussed hiring more Americans without college degrees due to a shortage of applicants for '3': "European shares were flat on Wednesday as weak results from the auto sector and fading investor confidence offset positive news fr
12	10	2019-03-17	The United States opposes France's digital services	1': "Oracle's revenue drop warning" '2': "Apple's commitment to data privacy" '3': "EU antitrust regulators investigating Apple's App Store practices"	1': "Boeing's 737 Max crashes" '2': "Facebook's data deals under investigation" '3': "Oracle's potential revenue drop"

Conclusions and Recommendations

Conclusions:

- The use of Sentence Transformer embeddings combined with a Decision Tree classifier provided the most balanced performance in sentiment classification.
- Models using static embeddings like Word2Vec and GloVe exhibited severe overfitting.
- The classification models struggled most with differentiating between neutral and positive/negative sentiment.
- Class imbalance in the dataset may have contributed to misclassification.

Business Recommendations:

- The Sentence Transformer + Decision Tree model for automated sentiment analysis of financial news may need continuous monitoring.
- There is a need to enhance model performance by incorporating additional financial indicators or with financial-specific data & models.
- Applying data balancing techniques and collecting more data across all sentiment classes will be beneficial.
- Periodically train the model with updated financial news data and market conditions for future improvement.
- Use positive and negative news event summaries to train models with reinforcement learning using human-provided feedback on sentiment analysis.
- The business can use the positive and negative news event summaries to assess market sentiment and adjust their trading decisions accordingly.

This section contains an experimental model for sentiment analysis

- This is NOT part of the original assignment, Only for experimentation.

Model-8 Experiment with FinBERT

FinBERT Model Overview in the Context of Sentiment Analysis for Financial News

- **FinBERT is a pre-trained NLP model based on BERT** that has been specifically fine-tuned for **financial sentiment analysis** using datasets from financial reports, analyst statements, and market news.
- **Designed for sentiment classification in finance**, it categorizes text into **positive, negative, or neutral sentiments**, making it highly suitable for analyzing financial news and its impact on stock prices.
- Unlike generic sentiment analysis models, **FinBERT understands financial jargon and context**, making it more reliable for this project.

Installing and Initializing the FinBERT Model

In [ ]:
```
# Installing torch dataset
!pip install transformers torch datasets
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-packages (4.49.0)
Requirement already satisfied: torch in /usr/local/lib/python3.11/dist-packages (2.5.1+cu124)
Collecting datasets
  Downloading datasets-3.3.2-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from transformers) (3.17.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.26.0 in /usr/local/lib/python3.11/dist-packages (from tra
nsformers) (0.28.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-packages (from transformers) (1.26.
4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from transformers) (2
4.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-packages (from transformers) (6.0.2
)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.11/dist-packages (from transformers)
(2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from transformers) (2.32.3)
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.11/dist-packages (from transform
ers) (0.21.0)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.11/dist-packages (from transformers)
(0.5.2)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist-packages (from transformers) (4.67.1
)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.11/dist-packages (from torch)
(4.12.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch) (3.1.5)
```

```
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from torch) (2024.10.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch) (12.4.127)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch) (12.4.127)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch) (12.4.127)
Requirement already satisfied: nvidia-cudnn-cu12==9.1.0.70 in /usr/local/lib/python3.11/dist-packages (from torch) (9.1.0.70)
Requirement already satisfied: nvidia-cublas-cu12==12.4.5.8 in /usr/local/lib/python3.11/dist-packages (from torch) (12.4.5.8)
Requirement already satisfied: nvidia-cufft-cu12==11.2.1.3 in /usr/local/lib/python3.11/dist-packages (from torch) (11.2.1.3)
Requirement already satisfied: nvidia-curand-cu12==10.3.5.147 in /usr/local/lib/python3.11/dist-packages (from torch) (10.3.5.147)
Requirement already satisfied: nvidia-cusolver-cu12==11.6.1.9 in /usr/local/lib/python3.11/dist-packages (from torch) (11.6.1.9)
Requirement already satisfied: nvidia-cusparse-cu12==12.3.1.170 in /usr/local/lib/python3.11/dist-packages (from torch) (12.3.1.170)
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in /usr/local/lib/python3.11/dist-packages (from torch) (2.21.5)
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch) (12.4.127)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.4.127 in /usr/local/lib/python3.11/dist-packages (from torch) (12.4.127)
Requirement already satisfied: triton==3.1.0 in /usr/local/lib/python3.11/dist-packages (from torch) (3.1.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-packages (from torch) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy==1.13.1->torch) (1.3.0)
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.11/dist-packages (from datasets) (17.0.0)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (from datasets) (2.2.2)
Collecting xxhash (from datasets)
  Downloading xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
Collecting multiprocess<0.70.17 (from datasets)
  Downloading multiprocess-0.70.16-py311-none-any.whl.metadata (7.2 kB)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-packages (from datasets) (3.11.12)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (2.4.6)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (25.1.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (6.1.0)
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (0.2.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.18.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->transformers) (2025.1.31)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->torch) (3.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2025.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas->datasets) (1.17.0)
Downloading datasets-3.3.2-py3-none-any.whl (485 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 485.4/485.4 kB 9.9 MB/s eta 0:00:00
Downloading dill-0.3.8-py3-none-any.whl (116 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 13.9 MB/s eta 0:00:00
Downloading multiprocess-0.70.16-py311-none-any.whl (143 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 143.5/143.5 kB 17.2 MB/s eta 0:00:00
Downloading xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.8/194.8 kB 22.1 MB/s eta 0:00:00
Installing collected packages: xxhash, dill, multiprocess, datasets
Successfully installed datasets-3.3.2 dill-0.3.8 multiprocess-0.70.16 xxhash-3.5.0
```

In [ ]:

```python
# Importing necessary libs
```

```
import torch
from transformers import AutoTokenizer, AutoModelForSequenceClassification, Trainer, TrainingArguments
from transformers import pipeline
from datasets import Dataset
import pandas as pd
from sklearn.model_selection import train_test_split
```

In [ ]:
```
# Loading FinBERT tokenizer and model
finbert_model = "ProsusAI/finbert"

tokenizer = AutoTokenizer.from_pretrained(finbert_model)
model = AutoModelForSequenceClassification.from_pretrained(finbert_model, num_labels=3)  # 3 sentiment classes
```

```
tokenizer_config.json:   0%|          | 0.00/252 [00:00<?, ?B/s]
config.json:   0%|          | 0.00/758 [00:00<?, ?B/s]
vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]
special_tokens_map.json:   0%|          | 0.00/112 [00:00<?, ?B/s]
pytorch_model.bin:   0%|          | 0.00/438M [00:00<?, ?B/s]
```

Loading, splitting and tokenizing the dataset

In [ ]:
```
df = pd.read_csv(path+'stock_news.csv') #delimiter="\t", encoding='utf-8')

# Map sentiment labels (-1, 0, 1) to FinBERT's format
label_map = {-1: 0, 0: 1, 1: 2}  # Required for compatibility
df['Label'] = df['Label'].map(label_map)

# Splitting into train and validation sets
train_texts, val_texts, train_labels, val_labels = train_test_split(df['News'], df['Label'], test_size=0.2, rand

# Tokenizing dataset
train_encodings = tokenizer(list(train_texts), truncation=True, padding=True, max_length=512)
val_encodings = tokenizer(list(val_texts), truncation=True, padding=True, max_length=512)

# Converting to Hugging Face dataset format
train_dataset = Dataset.from_dict({"input_ids": train_encodings["input_ids"], "attention_mask": train_encodings
val_dataset = Dataset.from_dict({"input_ids": val_encodings["input_ids"], "attention_mask": val_encodings["atte
```

Creating the Trainer object with train & validation dataset. And Training the model

In [ ]:
```
training_args = TrainingArguments(
    output_dir="./finbert_results",
    num_train_epochs=4,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    warmup_steps=500,
    weight_decay=0.01,
    logging_dir="./logs",
    evaluation_strategy="epoch"
)

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset
)

trainer.train()
```

```
wandb: WARNING The `run_name` is currently set to the same value as `TrainingArguments.output_dir`. If this was
not intended, please specify a different run name by setting the `TrainingArguments.run_name` parameter.
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter:
 ··········
wandb: WARNING If you're specifying your api key in code, ensure this code is not shared publicly.
wandb: WARNING Consider setting the WANDB_API_KEY environment variable, or running `wandb login` from the comman
d line.
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
wandb: Currently logged in as: amitava-basu (amitava-basu-the-university-of-texas-at-austin) to https://api.wand
b.ai. Use `wandb login --relogin` to force relogin
wandb: Using wandb-core as the SDK backend.  Please refer to https://wandb.me/wandb-core for more information.
```

Tracking run with wandb version 0.19.6

Run data is saved locally in `/content/wandb/run-20250222_202901-mpf3i5jm`

Syncing run **./finbert_results** to Weights & Biases (docs)

View project at https://wandb.ai/amitava-basu-the-university-of-texas-at-austin/huggingface

View run at https://wandb.ai/amitava-basu-the-university-of-texas-at-austin/huggingface/runs/mpf3i5jm

[140/140 00:23, Epoch 4/4]

Epoch	Training Loss	Validation Loss
1	No log	1.901664
2	No log	1.173135
3	No log	1.038923
4	No log	1.288727

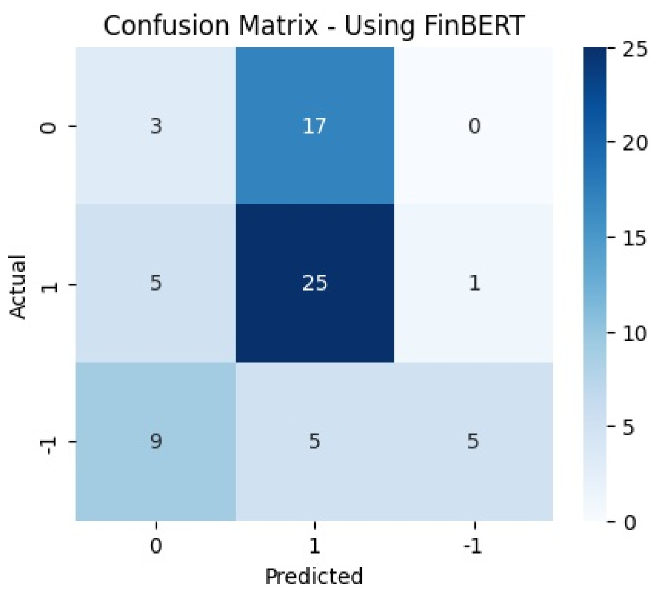
Out[ ]:
```
TrainOutput(global_step=140, training_loss=1.300225067138672, metrics={'train_runtime': 195.7809, 'train_samples_per_second': 5.7, 'train_steps_per_second': 0.715, 'total_flos': 43012877077800.0, 'train_loss': 1.300225067138672, 'epoch': 4.0})
```

Generating predictions for validation dataset, and getting scores against predicted Vs actual validation data.

In [ ]:
```python
predictions = trainer.predict(val_dataset)
preds = torch.argmax(torch.tensor(predictions.predictions), axis=1)

# Convert predictions back to original labels
reverse_label_map = {0: -1, 1: 0, 2: 1}
preds = pd.Series(preds.numpy()).map(reverse_label_map)
actual = val_labels.map(reverse_label_map)

# Compute accuracy
from sklearn.metrics import classification_report
print(classification_report(val_labels.map(reverse_label_map), preds))
```

```
              precision    recall  f1-score   support

          -1       0.18      0.15      0.16        20
           0       0.53      0.81      0.64        31
           1       0.83      0.26      0.40        19

    accuracy                           0.47        70
   macro avg       0.51      0.41      0.40        70
weighted avg       0.51      0.47      0.44        70
```

In [ ]:
```python
display_confusion_matrix_pre_predicted(model,preds, actual, 'Using FinBERT')
```



In [ ]:
```python
print_evaluation_scores_pre_predicted(model,preds, actual, 'Using FinBERT')
```

Using FinBERT

Out[ ]:

	F1	Recall	Accuracy	Precision
0	0.438786	0.471429	0.471429	0.512173

Final Observations

- The overall accuracy of the FinBERT model is approximately **47.14%**, indicating that the model is only slightly better than random guessing.

- The confusion matrix shows that the model struggles with the neutral class (0), misclassifying most neutral samples as positive (1).
- The recall (47.14%) suggests that the model is not consistently identifying all actual instances of each sentiment class, which may lead to missed sentiment signals.
- The precision (51.22%) indicates that while some of the positive predictions are correct, the model still has a high misclassification rate.
- The F1 score (43.88%) suggests that the model has a poor balance between precision and recall, indicating that further fine-tuning or additional labeled financial data needed for better performance.
- The model performs best on the positive class (1), correctly predicting 25 out of 31 samples, while performing poorly on the negative class (-1) with many misclassifications.
- To further increase performance, fine-tuning this model on a more targeted dataset may be helpful. only 349 news articles for fine-tuning is likely not sufficient for improving FinBERT's performance significantly.