In [ ]:

In [1]:
```python
import pandas as pd
try:
    df = pd.read_csv('/content/dataset.csv', encoding='latin1')
    print("Dataset loaded successfully!")
    print("Shape of the dataset:", df.shape)
    display(df.head())
except FileNotFoundError:
    print("Error: Dataset file not found. Please upload the dataset file or
except Exception as e:
    print(f"An error occurred while loading the dataset: {e}")
```

```
Dataset loaded successfully!
Shape of the dataset: (4846, 2)
```

	Sentiment	News_Headline
0	neutral	According to Gran , the company has no plans t...
1	neutral	Technopolis plans to develop in stages an area...
2	negative	The international electronic industry company ...
3	positive	With the new production plant the company woul...
4	positive	According to the company 's updated strategy f...

In [2]:
```python
# Check for missing values
print("Missing values per column:")
display(df.isnull().sum())

# Check the distribution of sentiment labels
print("\nDistribution of Sentiment Labels:")
display(df['Sentiment'].value_counts())

# Analyze text length of headlines
df['Headline_Length'] = df['News_Headline'].apply(len)
print("\nDescriptive statistics for Headline Length:")
display(df['Headline_Length'].describe())

# You can add more exploration steps here, e.g., word frequency analysis
```

```
Missing values per column:
```

	0
Sentiment	0
News_Headline	0

dtype: int64

```
Distribution of Sentiment Labels:
```

	count
Sentiment	
neutral	2879
positive	1363
negative	604

dtype: int64

```
Descriptive statistics for Headline Length:
```

	Headline_Length
count	4846.000000
mean	128.132068
std	56.526180
min	9.000000
25%	84.000000
50%	119.000000
75%	163.000000
max	315.000000

dtype: float64

In [3]:

```
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import pandas as pd

# Download necessary NLTK data
try:
    nltk.data.find('corpora/stopwords')
except LookupError:
    nltk.download('stopwords')
try:
    nltk.data.find('tokenizers/punkt')
except LookupError:
    nltk.download('punkt')
try:
    nltk.data.find('corpora/wordnet')
except LookupError:
    nltk.download('wordnet')


# Initialize lemmatizer and stopwords
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))
```

```python
def preprocess_text(text):
    # Ensure the input is a string
    if not isinstance(text, str):
        return "" # Return an empty string for non-string inputs

    # Remove punctuation and special characters, but keep numbers
    text = re.sub(r'[^a-zA-Z0-9\s]', '', text)
    # Convert to lowercase
    text = text.lower()
    # Tokenize the text
    tokens = nltk.word_tokenize(text)
    # Remove stop words and lemmatize the words
    tokens = [lemmatizer.lemmatize(word) for word in tokens]
    # Join tokens back into a string
    return ' '.join(tokens)

# Apply preprocessing to the News_Headline column
# Add a check for non-string types in the column before applying
if not pd.api.types.is_string_dtype(df['News_Headline']):
    print("Warning: 'News_Headline' column contains non-string types. Attemp
    df['News_Headline'] = df['News_Headline'].astype(str)


df['Cleaned_Headline'] = df['News_Headline'].apply(preprocess_text)

print("Original Headlines:")
display(df['News_Headline'].head())
print("\nCleaned Headlines:")
display(df['Cleaned_Headline'].head())
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
Original Headlines:
```

	News_Headline
0	According to Gran , the company has no plans t...
1	Technopolis plans to develop in stages an area...
2	The international electronic industry company ...
3	With the new production plant the company woul...
4	According to the company 's updated strategy f...

dtype: object

```
Cleaned Headlines:
```

	Cleaned_Headline
0	according to gran the company ha no plan to mo...
1	technopolis plan to develop in stage an area o...
2	the international electronic industry company ...
3	with the new production plant the company woul...
4	according to the company s updated strategy fo...

dtype: object

In [4]:
```
# Check the distribution of sentiment labels again to confirm
print("Distribution of Sentiment Labels after Preprocessing:")
display(df['Sentiment'].value_counts())

# Based on the distribution, we can decide if we need to handle imbalance.
# If there's significant imbalance, we can use techniques like SMOTE.
# We'll address this in the next steps if necessary.
```

```
Distribution of Sentiment Labels after Preprocessing:
```

Sentiment	count
neutral	2879
positive	1363
negative	604

dtype: int64

In [5]:
```
from imblearn.over_sampling import SMOTE
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split

# Separate features (X) and target (y)
X = df['Cleaned_Headline']
y = df['Sentiment']

# Split the data into training and testing sets before applying SMOTE
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar

# Vectorize the text data using TF-IDF on the training data
tfidf_vectorizer = TfidfVectorizer(max_features=5000) # You can adjust max_f
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)

# Apply SMOTE to the training vectorized data
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train_tfidf, y_t

print("Shape of training data before SMOTE:", X_train_tfidf.shape)
```

```
print("Shape of training data after SMOTE:", X_train_resampled.shape)
print("\nDistribution of Sentiment Labels in training data after SMOTE:")
display(y_train_resampled.value_counts())

# Vectorize the test data using the same TF-IDF vectorizer fitted on the tra
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

```
Shape of training data before SMOTE: (3876, 5000)
Shape of training data after SMOTE: (6909, 5000)

Distribution of Sentiment Labels in training data after SMOTE:
```

Sentiment	count
positive	2303
neutral	2303
negative	2303

dtype: int64

In [6]:

```
# Add custom feature: Number of words in cleaned headline
df['Cleaned_Word_Count'] = df['Cleaned_Headline'].apply(lambda x: len(x.spli

# Add custom feature: Average word length in cleaned headline
df['Average_Word_Length'] = df['Cleaned_Headline'].apply(lambda x: sum(len(w

print("\nDescriptive statistics for Cleaned Word Count:")
display(df['Cleaned_Word_Count'].describe())

print("\nDescriptive statistics for Average Word Length:")
display(df['Average_Word_Length'].describe())

display(df.head())
```

```
Descriptive statistics for Cleaned Word Count:
```

	Cleaned_Word_Count
count	4846.000000
mean	20.489682
std	8.792611
min	1.000000
25%	14.000000
50%	19.000000
75%	26.000000
max	52.000000

dtype: float64

```
Descriptive statistics for Average Word Length:
```

	Average_Word_Length
count	4846.000000
mean	4.902997
std	0.782964
min	2.000000
25%	4.375000
50%	4.850000
75%	5.400000
max	8.600000

dtype: float64

	Sentiment	News_Headline	Headline_Length	Cleaned_Headline	Cleaned_Wo
0	neutral	According to Gran , the company has no plans t...	127	according to gran the company ha no plan to mo...	
1	neutral	Technopolis plans to develop in stages an area...	190	technopolis plan to develop in stage an area o...	
2	negative	The international electronic industry company ...	228	the international electronic industry company ...	
3	positive	With the new production plant the company woul...	206	with the new production plant the company woul...	
4	positive	According to the company 's updated strategy f...	203	according to the company s updated strategy fo...	

In [7]:

```python
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer

# Download necessary NLTK data for SentimentIntensityAnalyzer
try:
    nltk.data.find('sentiment/vader_lexicon.zip')
except LookupError:
    nltk.download('vader_lexicon')

# Initialize SentimentIntensityAnalyzer
sid = SentimentIntensityAnalyzer()
```

```
# Add custom feature: Count of words in the cleaned headline (already done, 
# df['Cleaned_Word_Count'] = df['Cleaned_Headline'].apply(lambda x: len(x.sp

# Add custom features: Count of positive, negative, and neutral words (This 
# We can revisit this if needed, but VADER's polarity scores cover this sent

# Add custom features: Polarity scores from SentimentIntensityAnalyzer
df['VADER_Negative'] = df['Cleaned_Headline'].apply(lambda x: sid.polarity_s
df['VADER_Neutral'] = df['Cleaned_Headline'].apply(lambda x: sid.polarity_sc
df['VADER_Positive'] = df['Cleaned_Headline'].apply(lambda x: sid.polarity_s
df['VADER_Compound'] = df['Cleaned_Headline'].apply(lambda x: sid.polarity_s


print("\nDescriptive statistics for VADER Sentiment Scores:")
display(df[['VADER_Negative', 'VADER_Neutral', 'VADER_Positive', 'VADER_Comp

display(df.head())
```

```
Descriptive statistics for VADER Sentiment Scores:
```

	VADER_Negative	VADER_Neutral	VADER_Positive	VADER_Compound
count	4846.000000	4846.000000	4846.000000	4846.000000
mean	0.018802	0.879398	0.101803	0.221318
std	0.055491	0.120968	0.114028	0.311683
min	0.000000	0.000000	0.000000	-0.865800
25%	0.000000	0.802000	0.000000	0.000000
50%	0.000000	0.893000	0.083000	0.202300
75%	0.000000	1.000000	0.168750	0.440400
max	0.444000	1.000000	1.000000	0.946000

	Sentiment	News_Headline	Headline_Length	Cleaned_Headline	Cleaned_Wo
0	neutral	According to Gran , the company has no plans t...	127	according to gran the company ha no plan to mo...	
1	neutral	Technopolis plans to develop in stages an area...	190	technopolis plan to develop in stage an area o...	
2	negative	The international electronic industry company ...	228	the international electronic industry company ...	
3	positive	With the new production plant the company woul...	206	with the new production plant the company woul...	
4	positive	According to the company 's updated strategy f...	203	according to the company s updated strategy fo...	

In [8]:

```
import nltk
from nltk.corpus import opinion_lexicon

# Download necessary NLTK data for Opinion Lexicon
try:
    nltk.data.find('corpora/opinion_lexicon')
except LookupError:
    nltk.download('opinion_lexicon')

# Get positive and negative word lists from the opinion lexicon
positive_words = set(opinion_lexicon.positive())
negative_words = set(opinion_lexicon.negative())

# Add custom feature: Count of positive words in cleaned headline
df['Positive_Word_Count'] = df['Cleaned_Headline'].apply(lambda x: sum(1 for

# Add custom feature: Count of negative words in cleaned headline
df['Negative_Word_Count'] = df['Cleaned_Headline'].apply(lambda x: sum(1 for

print("\nDescriptive statistics for Positive Word Count:")
display(df['Positive_Word_Count'].describe())

print("\nDescriptive statistics for Negative Word Count:")
display(df['Negative_Word_Count'].describe())

display(df.head())
```

```
Descriptive statistics for Positive Word Count:
```

	Positive_Word_Count
count	4846.000000
mean	0.350392
std	0.634579
min	0.000000
25%	0.000000
50%	0.000000
75%	1.000000
max	6.000000

dtype: float64

```
Descriptive statistics for Negative Word Count:
```

	Negative_Word_Count
count	4846.000000
mean	0.155592
std	0.412069
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	4.000000

dtype: float64

	Sentiment	News_Headline	Headline_Length	Cleaned_Headline	Cleaned_Wo
0	neutral	According to Gran , the company has no plans t...	127	according to gran the company ha no plan to mo...	
1	neutral	Technopolis plans to develop in stages an area...	190	technopolis plan to develop in stage an area o...	
2	negative	The international electronic industry company ...	228	the international electronic industry company ...	
3	positive	With the new production plant the company woul...	206	with the new production plant the company woul...	
4	positive	According to the company 's updated strategy f...	203	according to the company s updated strategy fo...	

In [9]:

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from imblearn.over_sampling import SMOTE
import numpy as np
import pandas as pd # Import pandas

# Separate features (X) and target (y) including all custom features
X = df[['Cleaned_Headline', 'Cleaned_Word_Count', 'Average_Word_Length', 'VA

from sklearn.preprocessing import LabelEncoder

# Initialize encoder
label_encoder = LabelEncoder()

# Fit and transform target labels
y = label_encoder.fit_transform(df['Sentiment'])


# Split the data into training and testing sets before applying SMOTE
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

# Vectorize the text data using TF-IDF on the 'Cleaned_Headline' column of t
tfidf_vectorizer = TfidfVectorizer(max_features=5000) # You can adjust max_f
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train['Cleaned_Headline'])

# Get the custom features for the training set
X_train_custom = X_train[['Cleaned_Word_Count', 'Average_Word_Length', 'VADE

# Combine TF-IDF features and custom features for the training set
X_train_combined = np.hstack((X_train_tfidf.toarray(), X_train_custom))
```

```
# Apply SMOTE to the combined training data
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train_combined,

print("Shape of training data before SMOTE:", X_train_combined.shape)
print("Shape of training data after SMOTE:", X_train_resampled.shape)
print("\nDistribution of Sentiment Labels in training data after SMOTE:")
display(pd.Series(y_train_resampled).value_counts()) # Convert to pandas Ser

# Vectorize the test data using the same TF-IDF vectorizer fitted on the tra
X_test_tfidf = tfidf_vectorizer.transform(X_test['Cleaned_Headline'])

# Get the custom features for the test set
X_test_custom = X_test[['Cleaned_Word_Count', 'Average_Word_Length', 'VADER_

# Combine TF-IDF features and custom features for the test set
X_test_combined = np.hstack((X_test_tfidf.toarray(), X_test_custom))

print("\nShape of test data:", X_test_combined.shape)
```

```
Shape of training data before SMOTE: (3876, 5008)
Shape of training data after SMOTE: (6909, 5008)

Distribution of Sentiment Labels in training data after SMOTE:
```

	count
2	2303
1	2303
0	2303

dtype: int64

```
Shape of test data: (970, 5008)
```

In [ ]:

In [10]:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score

# Train a Logistic Regression model
print("\nTraining Logistic Regression model...")
lr_model = LogisticRegression(max_iter=1000) # Increased max_iter for conver
lr_model.fit(X_train_resampled, y_train_resampled)

# Evaluate the Logistic Regression model
y_pred_lr = lr_model.predict(X_test_combined)
print("\nLogistic Regression Model Performance:")
print("Accuracy:", accuracy_score(y_test, y_pred_lr))
print(classification_report(y_test, y_pred_lr))
```

```
Training Logistic Regression model...

Logistic Regression Model Performance:
Accuracy: 0.7546391752577319
              precision    recall  f1-score   support

           0       0.63      0.77      0.69       121
           1       0.84      0.80      0.82       576
           2       0.66      0.66      0.66       273

    accuracy                           0.75       970
   macro avg       0.71      0.74      0.72       970
weighted avg       0.76      0.75      0.76       970
```

In [13]:

```python
import numpy as np
import pandas as pd # Import pandas

# Example real headlines (you can replace this list with any headlines you w
real_headlines = [
    "India's HDFC Bank reports 12.2% profit growth in Q1 due to higher inter
    "Wells Fargo exit ban revives fears about doing business in China",
    "American Express allays competition concerns after profit beat",
    "Hedge funds lure record inflows in first half of 2025",
    "Trump signs stablecoin law as crypto industry aims for mainstream adopt
    "U.S. equities edge lower after tariff headlines and mixed earnings",
    "Lofty U.S. stock market valuations bank on earnings strength",
    "Dow Jones futures: Tesla, Google Step Up With Earnings Due; AI Stock Br
    "Take Five: global markets to test industrial sector gains as earnings r
    "Why stock market fell today: Sensex settles 501 pts lower, Nifty below 
]

print("Testing multiple real headlines:")

for real_headline in real_headlines:
    print(f"\nOriginal Headline: {real_headline}")

    # Preprocess the headline using the same function used for the training
    cleaned_real_headline = preprocess_text(real_headline)
    print(f"Cleaned Headline: {cleaned_real_headline}")

    # Calculate custom features for the real headline directly
    tokens = cleaned_real_headline.split()
    cleaned_word_count = len(tokens)
    average_word_length = sum(len(word) for word in tokens) / cleaned_word_c

    # Calculate VADER sentiment scores
    vader_scores = sid.polarity_scores(cleaned_real_headline)
    vader_negative = vader_scores['neg']
    vader_neutral = vader_scores['neu']
    vader_positive = vader_scores['pos']
    vader_compound = vader_scores['compound']

    # Calculate Opinion Lexicon word counts
    positive_word_count = sum(1 for word in tokens if word in positive_words
    negative_word_count = sum(1 for word in tokens if word in negative_words
```

```
    # Arrange custom features in a NumPy array
    real_headline_custom = np.array([[cleaned_word_count, average_word_lengt

    # Vectorize the cleaned headline using the fitted TF-IDF vectorizer
    real_headline_tfidf = tfidf_vectorizer.transform([cleaned_real_headline]

    # Combine TF-IDF features and custom features for the real headline
    real_headline_combined = np.hstack((real_headline_tfidf.toarray(), real_

    # Predict the sentiment using the trained Logistic Regression model
    predicted_sentiment_encoded = lr_model.predict(real_headline_combined)

    # Inverse transform the predicted sentiment to get the original label
    predicted_sentiment = label_encoder.inverse_transform(predicted_sentimen

    print(f"Predicted Sentiment: {predicted_sentiment[0]}")
    # You can manually compare the predicted sentiment with what you expect
```

```
Testing multiple real headlines:

Original Headline: India's HDFC Bank reports 12.2% profit growth in Q1 due t
o higher interest income
Cleaned Headline: india hdfc bank report 122 profit growth in q1 due to high
er interest income
Predicted Sentiment: positive

Original Headline: Wells Fargo exit ban revives fears about doing business i
n China
Cleaned Headline: well fargo exit ban revives fear about doing business in c
hina
Predicted Sentiment: negative

Original Headline: American Express allays competition concerns after profit
beat
Cleaned Headline: american express allays competition concern after profit b
eat
Predicted Sentiment: neutral

Original Headline: Hedge funds lure record inflows in first half of 2025
Cleaned Headline: hedge fund lure record inflow in first half of 2025
Predicted Sentiment: negative

Original Headline: Trump signs stablecoin law as crypto industry aims for ma
instream adoption
Cleaned Headline: trump sign stablecoin law a crypto industry aim for mainst
ream adoption
Predicted Sentiment: neutral

Original Headline: U.S. equities edge lower after tariff headlines and mixed
earnings
Cleaned Headline: u equity edge lower after tariff headline and mixed earnin
gs
Predicted Sentiment: negative

Original Headline: Lofty U.S. stock market valuations bank on earnings stren
gth
Cleaned Headline: lofty u stock market valuation bank on earnings strength
Predicted Sentiment: positive

Original Headline: Dow Jones futures: Tesla, Google Step Up With Earnings Du
e; AI Stock Breaks Out
Cleaned Headline: dow jones future tesla google step up with earnings due ai
stock break out
Predicted Sentiment: neutral

Original Headline: Take Five: global markets to test industrial sector gains
as earnings ramp up
Cleaned Headline: take five global market to test industrial sector gain a e
arnings ramp up
Predicted Sentiment: positive

Original Headline: Why stock market fell today: Sensex settles 501 pts lowe
r, Nifty below 25,000; 5 reasons
Cleaned Headline: why stock market fell today sensex settle 501 pt lower nif
```

```
ty below 25000 5 reason
Predicted Sentiment: negative
```

In [12]:
```
# Train a Support Vector Machine model
print("\nTraining Support Vector Machine model...")
svm_model = SVC()
svm_model.fit(X_train_resampled, y_train_resampled)

# Evaluate the Support Vector Machine model
y_pred_svm = svm_model.predict(X_test_combined)
print("\nSupport Vector Machine Model Performance:")
print("Accuracy:", accuracy_score(y_test, y_pred_svm))
print(classification_report(y_test, y_pred_svm))
```

```
Training Support Vector Machine model...

Support Vector Machine Model Performance:
Accuracy: 0.5639175257731959
              precision    recall  f1-score   support

           0       0.37      0.48      0.42       121
           1       0.74      0.58      0.65       576
           2       0.43      0.56      0.49       273

    accuracy                           0.56       970
   macro avg       0.51      0.54      0.52       970
weighted avg       0.61      0.56      0.58       970
```

In [ ]:
```

```

In [11]:
```
!pip install gensim
```

```
Collecting gensim
  Downloading gensim-4.3.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x
86_64.whl.metadata (8.1 kB)
Collecting numpy<2.0,>=1.18.5 (from gensim)
  Downloading numpy-1.26.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x
86_64.whl.metadata (61 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.0/61.0 kB 2.5 MB/s eta 0:0
0:00
Collecting scipy<1.14.0,>=1.7.0 (from gensim)
  Downloading scipy-1.13.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x
86_64.whl.metadata (60 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60.6/60.6 kB 4.9 MB/s eta 0:0
0:00
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.1
1/dist-packages (from gensim) (7.3.0.post1)
Requirement already satisfied: wrapt in /usr/local/lib/python3.11/dist-packa
ges (from smart-open>=1.8.1->gensim) (1.17.2)
Downloading gensim-4.3.3-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86
_64.whl (26.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 26.7/26.7 MB 52.9 MB/s eta 0:00:
00
Downloading numpy-1.26.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86
_64.whl (18.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.3/18.3 MB 67.2 MB/s eta 0:00:
00
Downloading scipy-1.13.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86
_64.whl (38.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 38.6/38.6 MB 12.6 MB/s eta 0:00:
00
Installing collected packages: numpy, scipy, gensim
  Attempting uninstall: numpy
    Found existing installation: numpy 2.0.2
    Uninstalling numpy-2.0.2:
      Successfully uninstalled numpy-2.0.2
  Attempting uninstall: scipy
    Found existing installation: scipy 1.15.3
    Uninstalling scipy-1.15.3:
      Successfully uninstalled scipy-1.15.3
ERROR: pip's dependency resolver does not currently take into account all th
e packages that are installed. This behaviour is the source of the following
dependency conflicts.
opencv-python-headless 4.12.0.88 requires numpy<2.3.0,>=2; python_version >=
"3.9", but you have numpy 1.26.4 which is incompatible.
thinc 8.3.6 requires numpy<3.0.0,>=2.0.0, but you have numpy 1.26.4 which is
incompatible.
tsfresh 0.21.0 requires scipy>=1.14.0; python_version >= "3.10", but you hav
e scipy 1.13.1 which is incompatible.
Successfully installed gensim-4.3.3 numpy-1.26.4 scipy-1.13.1
```

In [9]:

```
# Word2vec

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
```

```python
import gensim.downloader as api

# Step 1: Encode target labels
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(df['Sentiment'])

# Step 2: Prepare features
X = df[['Cleaned_Headline', 'Cleaned_Word_Count', 'Average_Word_Length',
        'VADER_Negative', 'VADER_Neutral', 'VADER_Positive',
        'VADER_Compound', 'Positive_Word_Count', 'Negative_Word_Count']]

# Step 3: Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

# Step 4: Load pre-trained Word2Vec Google News
wv = api.load('word2vec-google-news-300')  # Pre-trained 300d Google News mo

# Step 5: Tokenize headlines
X_train_tokens = X_train['Cleaned_Headline'].apply(str.split)
X_test_tokens  = X_test['Cleaned_Headline'].apply(str.split)

# Step 6: Convert each headline to average Word2Vec vector
def document_vector(tokens):
    valid = [t for t in tokens if t in wv]
    if not valid:
        return np.zeros(wv.vector_size)
    return np.mean(wv[valid], axis=0)

X_train_w2v = np.vstack(X_train_tokens.apply(document_vector))
X_test_w2v  = np.vstack(X_test_tokens.apply(document_vector))

# Step 7: Extract custom features
custom_cols = ['Cleaned_Word_Count', 'Average_Word_Length',
               'VADER_Negative', 'VADER_Neutral', 'VADER_Positive',
               'VADER_Compound', 'Positive_Word_Count', 'Negative_Word_Count

X_train_custom = X_train[custom_cols].values
X_test_custom  = X_test[custom_cols].values

# Step 8: Combine embeddings + custom features
X_train_combined = np.hstack((X_train_w2v, X_train_custom))
X_test_combined  = np.hstack((X_test_w2v,  X_test_custom))

# Step 9: Apply SMOTE
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train_combined,

# Step 10: Print shapes and label distribution
print("Before SMOTE:", X_train_combined.shape)
print("After  SMOTE:", X_train_resampled.shape)
print(pd.Series(y_train_resampled).value_counts())
print("Test data shape:", X_test_combined.shape)
```

```
Before SMOTE: (3876, 308)
After  SMOTE: (6909, 308)
2    2303
1    2303
0    2303
Name: count, dtype: int64
Test data shape: (970, 308)
```

Task

Generate a markdown summary of the financial news headline sentiment analysis project, including data preprocessing, feature engineering, handling of class imbalance, model training and evaluation (focusing on Logistic Regression), and a real headline prediction example.

Summarize the project goal

Subtask:

Briefly state the objective of the project (Financial News Headline Sentiment Analysis).

Reasoning: State the objective of the project in a markdown block.

In [14]:
````
```markdown
The objective of this project is to perform sentiment analysis on financial
````

```
  File "/tmp/ipython-input-14-2680515063.py", line 1
    ```markdown
    ^
SyntaxError: invalid syntax
```

Reasoning: State the objective of the project in a markdown block.

In [15]:
```
# The objective of this project is to perform sentiment analysis on financia
# This helps in understanding the market sentiment conveyed by news.
```