In [1]:
```
# Import necessary libraries
import subprocess
import sys
import argparse
import json
import os
import gc
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import PyPDF2
import re
import nltk
import emoji
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline
import logging
import traceback
import numpy as np
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import cross_val_score
```

In [2]:
```
# Download necessary NLTK data
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('stopwords')
nltk.download('vader_lexicon')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\ELITEBOOK\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\ELITEBOOK\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ELITEBOOK\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\ELITEBOOK\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\ELITEBOOK\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

Out[2]:
```
True
```

In [3]:
```
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

# Function to extract sentences from PDFs using PyPDF2
def read_pdf_sentences(file_path):
    sentences = []
    try:
        with open(file_path, "rb") as file:
            reader = PyPDF2.PdfReader(file)
            for page in reader.pages:
                text = page.extract_text()
                if text:
                    sentences.extend(sent_tokenize(text))
    except Exception as e:
        print(f"Error reading PDF file {file_path}: {str(e)}")
    return sentences

def extract_and_merge(pdf_path, csv_path):
    try:
        print(f"Attempting to read PDF files from: {pdf_path}")
        if not os.path.exists(pdf_path):
            raise FileNotFoundError(f"PDF directory not found: {pdf_path}")

        pdf_files = [os.path.join(pdf_path, file) for file in os.listdir(pdf_path) if
        print(f"Found {len(pdf_files)} PDF files")

        pdf_sentences = []
        for file in pdf_files:
            pdf_sentences.extend(read_pdf_sentences(file))

        print(f"Extracted {len(pdf_sentences)} sentences from PDF files")
        pdf_df = pd.DataFrame({'content': pdf_sentences})

        print(f"Attempting to read CSV file: {csv_path}")
        if not os.path.exists(csv_path):
            raise FileNotFoundError(f"CSV file not found: {csv_path}")

        news_data = pd.read_csv(csv_path, encoding='latin1')
        content_column = next((col for col in news_data.columns if col.lower().strip()
        if content_column is None:
            raise KeyError(f"No 'content' column found in the CSV file: {csv_path}")
        news_data_paragraphs = []
        for content in news_data[content_column].dropna():
            paragraphs = content.split('\n\n')
            news_data_paragraphs.extend(paragraphs)

        print(f"Extracted {len(news_data_paragraphs)} paragraphs from CSV file")
        news_df = pd.DataFrame({'content': news_data_paragraphs})

        merged_data = pd.concat([pdf_df, news_df], ignore_index=True)
        print(f"Merged data shape: {merged_data.shape}")

        return merged_data

    except Exception as e:
        print(f"Error in extract_and_merge: {str(e)}")
        print(f"Current working directory: {os.getcwd()}")
        print(f"Contents of current directory: {os.listdir('.')}")
        if os.path.exists(pdf_path):
            print(f"Contents of PDF directory: {os.listdir(pdf_path)}")
        raise
```

In [4]:
```
# Assign Sentiment Analyzer Score
sid = SentimentIntensityAnalyzer()

def assign_sentiment_scores(text):
    scores = sid.polarity_scores(text)
    return scores['compound']

def assign_scores(data):
    data['sentiment'] = data['content'].apply(assign_sentiment_scores)
    return data

# Function to assign direction and new_direction based on sentiment scores
def assign_directions(data):
    data['direction'] = data['sentiment'].apply(lambda x: 'bearish' if x < 0.0 else ('
    data['new_direction'] = data['sentiment'].apply(lambda x: 2 if x < 0.0 else (1 if
    return data

# Function to preprocess individual text
def preprocess_text(text):
    lemmatizer = WordNetLemmatizer()
    stop_words = set(stopwords.words('english'))

    # Lowercase the text
    text = text.lower()

    # Remove emojis
    text = emoji.replace_emoji(text, '')

    # Remove emoticons (this is a basic implementation, might need refinement)
    text = re.sub(r'[:;=]-?[()DPp]', '', text)

    # Remove punctuation and numbers
    text = re.sub(r'[^\w\s]', '', text)
    text = re.sub(r'\d+', '', text)

    # Remove extra spaces
    text = re.sub(r'\s+', ' ', text).strip()
    # Tokenize
    tokens = word_tokenize(text)

    # Remove stop words and lemmatize
    tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in stop_words]

    try:
        tokens = [lemmatizer.lemmatize(word) for word in tokens if word not in stop_wo
    except LookupError:
        # If lemmatization fails, just use the original tokens
        tokens = [word for word in tokens if word not in stop_words]

    return ' '.join(tokens)

# Function to preprocess the entire DataFrame
def preprocess_data(df):
    df_cleaned = df.copy()
    df_cleaned['content'] = df_cleaned['content'].apply(preprocess_text)
    return df_cleaned

# Count the number of bearish, bullish, and neutral sentiments
def sentiment_counts(data):
    return data['direction'].value_counts()
```

In [5]:
```
# Prepare Dataset Function
def prepare_dataset(data, sample_frac=0.1, random_state=42):
    print("Preparing dataset...")
    data = data.sample(frac=sample_frac, random_state=random_state).reset_index(drop=T

    X = data['content']
    y = data['new_direction']

    # TF-IDF Vectorization
    vectorizer = TfidfVectorizer(max_features=5000)
    X = vectorizer.fit_transform(X)

    # Split the data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st

    # Define resampling strategy
    over = SMOTE(sampling_strategy='auto', random_state=random_state)
    under = RandomUnderSampler(sampling_strategy='auto', random_state=random_state)

    # Create a pipeline with SMOTE and RandomUnderSampler
    resampling = Pipeline([('over', over), ('under', under)])

    # Apply resampling
    X_train_resampled, y_train_resampled = resampling.fit_resample(X_train, y_train)

    print(f"Dataset prepared with train size: {X_train_resampled.shape[0]} and test si
    return X_train_resampled, X_test, y_train_resampled, y_test, vectorizer
```

In [6]:
```
import sklearn
def train_and_evaluate_multiple_models(X_train, X_test, y_train, y_test):
    print("Training and evaluating multiple models...")

    models = {
        'AdaBoost': AdaBoostClassifier(estimator=DecisionTreeClassifier(max_depth=3),
        'SVM': SVC(kernel='rbf', random_state=42),
        'Random Forest': RandomForestClassifier(n_estimators=100, random_state=42),
        'Naive Bayes': MultinomialNB(),
        'Logistic Regression': LogisticRegression(random_state=42),
        'Neural Network': MLPClassifier(hidden_layer_sizes=(100,), max_iter=500, rando
    }

    results = {}

    try:
        for name, model in models.items():
            print(f"Training and evaluating {name}...")

            # Train the model
            model.fit(X_train, y_train)

            # Make predictions
            y_pred = model.predict(X_test)

            # Calculate accuracy
            accuracy = accuracy_score(y_test, y_pred)

            # Generate classification report
            report = classification_report(y_test, y_pred, target_names=['bullish', 'r
            report_df = pd.DataFrame(report).transpose()

            # Generate confusion matrix
            cm = confusion_matrix(y_test, y_pred)

            # Perform cross-validation
            cv_scores = cross_val_score(model, X_train, y_train, cv=5)

            results[name] = {
                'model': model,
                'accuracy': accuracy,
                'report': report_df,
                'confusion_matrix': cm,
                'cv_scores': cv_scores
            }

            print(f"{name} - Accuracy: {accuracy}, Cross-validation mean score: {np.me

        return results

    except Exception as e:
        print(f"An error occurred in train_and_evaluate_multiple_models: {str(e)}")
        print(f"Error details: {traceback.format_exc()}")
        return None
```

In [7]:
```
def create_comprehensive_report_multiple_models(company_name, results):
    report_data = {'Company': company_name}

    for model_name, model_results in results.items():
        report_data[f'{model_name} Accuracy'] = model_results['accuracy']
        report_data[f'{model_name} CV Mean Score'] = np.mean(model_results['cv_scores'
        report_data[f'{model_name} CV Std Score'] = np.std(model_results['cv_scores'])
        report_data[f'{model_name} Confusion Matrix'] = model_results['confusion_matri

        for class_name in ['bullish', 'neutral', 'bearish']:
            if class_name in model_results['report'].index:
                report_data[f'{model_name} Precision ({class_name})'] = model_results[
                report_data[f'{model_name} Recall ({class_name})'] = model_results['re
                report_data[f'{model_name} F1-Score ({class_name})'] = model_results['

    return pd.DataFrame([report_data])
```

In [8]:
```
def main(company_name, pdf_path, csv_path):
    try:
        logger.info(f"Processing {company_name}...")

        # Load and preprocess data
        raw_data = extract_and_merge(pdf_path, csv_path)
        data_with_sentiment = assign_scores(raw_data)
        data_with_directions = assign_directions(data_with_sentiment)
        cleaned_data = preprocess_data(data_with_directions)

        # Display sentiment counts
        counts = sentiment_counts(cleaned_data)
        logger.info(f"{company_name} Sentiment Counts:")
        logger.info(counts)

        # Prepare dataset
        X_train, X_test, y_train, y_test, vectorizer = prepare_dataset(cleaned_data)

        # Train and evaluate multiple models
        results = train_and_evaluate_multiple_models(X_train, X_test, y_train, y_test)

        if results is None:
            logger.error(f"Training and evaluation failed for {company_name}")
            return None

        # Create comprehensive report
        comprehensive_report = create_comprehensive_report_multiple_models(company_nam

        return comprehensive_report

    except Exception as e:
        logger.error(f"Error processing {company_name}: {str(e)}")
        logger.error(traceback.format_exc())
        return None
```

In [9]:
```
if __name__ == "__main__":
    # Define paths for each company
    companies = {
        'Lloyds': {
            'pdf_path': 'data/lloyds',
            'csv_path': 'data/lloyds/lloyds_news.csv'
        },
        'IAG': {
            'pdf_path': 'data/iag',
            'csv_path': 'data/iag/iag_news.csv'
        },
        'Vodafone': {
            'pdf_path': 'data/vodafone',
            'csv_path': 'data/vodafone/vodafone_news.csv'
        }
    }

    all_reports = []
    for company_name, paths in companies.items():
        try:
            logger.info(f"Starting processing for {company_name}")
            company_report = main(company_name, paths['pdf_path'], paths['csv_path'])

            if company_report is not None:
                all_reports.append(company_report)

        except Exception as e:
            logger.error(f"Failed to process {company_name}: {str(e)}")

    # Combine all reports into a single DataFrame
    if all_reports:
        combined_report = pd.concat(all_reports, ignore_index=True)
        combined_report.to_csv('comprehensive_classification_report_adaboost.csv', inc
        logger.info("Comprehensive classification report for all companies saved to CS
    else:
        logger.warning("No reports were generated.")
```

```
INFO:__main__:Starting processing for Lloyds
INFO:__main__:Processing Lloyds...
Attempting to read PDF files from: data/lloyds
Found 20 PDF files
Extracted 66875 sentences from PDF files
Attempting to read CSV file: data/lloyds/lloyds_news.csv
Extracted 1834 paragraphs from CSV file
Merged data shape: (68709, 1)
INFO:__main__:Lloyds Sentiment Counts:
INFO:__main__:direction
neutral    27490
bullish    26909
bearish    14310
Name: count, dtype: int64
Preparing dataset...
Dataset prepared with train size: 6618 and test size: 1375
Training and evaluating multiple models...
Training and evaluating AdaBoost...
```

```
C:\Users\ELITEBOOK\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:5
27: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be remo
ved in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\ELITEBOOK\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:5
27: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be remo
ved in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\ELITEBOOK\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:5
27: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be remo
ved in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\ELITEBOOK\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:5
27: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be remo
ved in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\ELITEBOOK\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:5
27: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be remo
ved in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\ELITEBOOK\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:5
27: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be remo
ved in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
```

```
AdaBoost - Accuracy: 0.7258181818181818, Cross-validation mean score: 0.7582316578863
839
Training and evaluating SVM...
SVM - Accuracy: 0.7934545454545454, Cross-validation mean score: 0.8396774016756753
Training and evaluating Random Forest...
Random Forest - Accuracy: 0.7978181818181819, Cross-validation mean score: 0.84511912
18347024
Training and evaluating Naive Bayes...
Naive Bayes - Accuracy: 0.6567272727272727, Cross-validation mean score: 0.7064107482
536486
Training and evaluating Logistic Regression...
Logistic Regression - Accuracy: 0.7745454545454545, Cross-validation mean score: 0.81
17239040631358
Training and evaluating Neural Network...
```

```
INFO:__main__:Starting processing for IAG
INFO:__main__:Processing IAG...
```

```
Neural Network - Accuracy: 0.7629090909090909, Cross-validation mean score: 0.8188256
571510779
Attempting to read PDF files from: data/iag
Found 11 PDF files
Extracted 34291 sentences from PDF files
Attempting to read CSV file: data/iag/iag_news.csv
Extracted 2037 paragraphs from CSV file
Merged data shape: (36328, 1)
```

```
INFO:__main__:IAG Sentiment Counts:
INFO:__main__:direction
neutral    17607
bullish    12229
bearish     6492
Name: count, dtype: int64
C:\Users\ELITEBOOK\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:5
27: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be remo
ved in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
Preparing dataset...
Dataset prepared with train size: 4182 and test size: 727
Training and evaluating multiple models...
Training and evaluating AdaBoost...
```

```
C:\Users\ELITEBOOK\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:5
27: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be remo
ved in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\ELITEBOOK\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:5
27: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be remo
ved in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\ELITEBOOK\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:5
27: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be remo
ved in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\ELITEBOOK\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:5
27: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be remo
ved in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\ELITEBOOK\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:5
27: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be remo
ved in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
```

```
AdaBoost - Accuracy: 0.7042640990371389, Cross-validation mean score: 0.7400842036665
466
Training and evaluating SVM...
SVM - Accuracy: 0.7372764786795049, Cross-validation mean score: 0.8804462279844284
Training and evaluating Random Forest...
Random Forest - Accuracy: 0.781292984869326, Cross-validation mean score: 0.863952198
8418423
Training and evaluating Naive Bayes...
Naive Bayes - Accuracy: 0.657496561210454, Cross-validation mean score: 0.76136520839
40708
Training and evaluating Logistic Regression...
Logistic Regression - Accuracy: 0.7359009628610729, Cross-validation mean score: 0.83
88385839149846
Training and evaluating Neural Network...
```

```
INFO:__main__:Starting processing for Vodafone
INFO:__main__:Processing Vodafone...
```

```
Neural Network - Accuracy: 0.71939477303989, Cross-validation mean score: 0.857009540
7956189
Attempting to read PDF files from: data/vodafone
Found 14 PDF files
Extracted 51164 sentences from PDF files
Attempting to read CSV file: data/vodafone/vodafone_news.csv
Extracted 0 paragraphs from CSV file
Merged data shape: (51164, 1)
```

```
INFO:__main__:Vodafone Sentiment Counts:
INFO:__main__:direction
neutral    24998
bullish    18868
bearish     7298
Name: count, dtype: int64
```

```
Preparing dataset...
Dataset prepared with train size: 5721 and test size: 1024
Training and evaluating multiple models...
Training and evaluating AdaBoost...
```

```
C:\Users\ELITEBOOK\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:5
27: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be remo
ved in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\ELITEBOOK\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:5
27: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be remo
ved in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\ELITEBOOK\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:5
27: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be remo
ved in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\ELITEBOOK\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:5
27: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be remo
ved in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\ELITEBOOK\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:5
27: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be remo
ved in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
C:\Users\ELITEBOOK\anaconda3\Lib\site-packages\sklearn\ensemble\_weight_boosting.py:5
27: FutureWarning: The SAMME.R algorithm (the default) is deprecated and will be remo
ved in 1.6. Use the SAMME algorithm to circumvent this warning.
  warnings.warn(
AdaBoost - Accuracy: 0.7734375, Cross-validation mean score: 0.7722460072678413
Training and evaluating SVM...
SVM - Accuracy: 0.802734375, Cross-validation mean score: 0.8739760894127707
Training and evaluating Random Forest...
Random Forest - Accuracy: 0.8359375, Cross-validation mean score: 0.8783462607261734
Training and evaluating Naive Bayes...
Naive Bayes - Accuracy: 0.6748046875, Cross-validation mean score: 0.7545897334106941
Training and evaluating Logistic Regression...
Logistic Regression - Accuracy: 0.7861328125, Cross-validation mean score: 0.83621843
22227991
Training and evaluating Neural Network...
INFO:__main__:Comprehensive classification report for all companies saved to CSV.
Neural Network - Accuracy: 0.7822265625, Cross-validation mean score: 0.8680312700400
036
```