news_analysis

March 24, 2025

1 Step 1: Dataset Preparation

The Dataset of Spotify Stock News

[106]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio

from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk
from scipy import stats
from sklearn.model_selection import TimeSeriesSplit
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import warnings
warnings.filterwarnings('ignore')
```

[107]:
```
#load the csv file that contains the news from the finhub api 03/2024 to 02/2025
df_news = pd.read_csv("/Users/armandocriscuolo/c2025/data_science_project_2025/
 ↪code/Data-Science-Project/Data-Question-2-B/news_analysis/
 ↪spotify_news_2024_2025_finhub.csv")

#print the first 5 rows of the dataframe
print(df_news.head())

print("\n\n-----------------------------------\n\n")

#load the csv file that contains the stock data from the marketstack api 03/
 ↪2024 to 02/2025
df_stock = pd.read_csv("/Users/armandocriscuolo/c2025/data_science_project_2025/
 ↪code/Data-Science-Project/Data-Question-2-B/news_analysis/
 ↪spotify_stock_data_20250323_193740.csv")

#print the first 5 rows of the dataframe
```

```python
print(df_stock.head())
```

```
  category    datetime                                           headline  \
0  company  1740759981  Palantir, Nvidia Nixed But Netflix Remains On …
1  company  1740684439  Tracking Chase Coleman's Tiger Global Portfoli…
2  company  1740664816  Spotify Technology (SPOT) is Attracting Invest…
3  company  1740658999  YouTube Surpasses Competitors in Streaming and…
4  company  1740593693  Spotify CEO Wants EU To Penalize Apple For Def…

          id                                              image related  \
0  132937882  https://media.zenfs.com/en/ibd.com/fc3f416bcfc…    SPOT
1  132901707  https://static.seekingalpha.com/cdn/s3/uploads…    SPOT
2  132937883  https://media.zenfs.com/en/zacks.com/bdc2850a4…    SPOT
3  132937884  https://media.zenfs.com/en/us.finance.gurufocu…    SPOT
4  132937885  https://media.zenfs.com/en/Benzinga/20dc68a2c9…    SPOT

        source                                            summary  \
0        Yahoo  A volatile market has shaken off Nvidia and Pa…
1  SeekingAlpha  Tiger Global's 13F reveals a $26.46B portfolio…
2        Yahoo  Spotify (SPOT) has been one of the stocks most…
3        Yahoo  Nielsen ranks YouTube ahead of Netflix and Hul…
4        Yahoo  Spotify Technology’s (NYSE:SPOT) CEO, Daniel E…

                                                 url        date
0  https://finnhub.io/api/news?id=578a14cc0987e6c…  2025-02-28
1  https://finnhub.io/api/news?id=c7d354e4ade5e63…  2025-02-27
2  https://finnhub.io/api/news?id=cec976f64d52fff…  2025-02-27
3  https://finnhub.io/api/news?id=1fd8ee4da1b2b5e…  2025-02-27
4  https://finnhub.io/api/news?id=e4ed9eb39665a16…  2025-02-26


-----------------------------------



    open      high      low   close     volume  adj_high  adj_low  adj_close  \
0  584.25  609.9200  580.000  608.01  4531895.0  609.9200  580.000     608.01
1  611.00  613.0000  586.000  590.76  1191837.0  613.0000  586.000     590.76
2  595.62  608.5294  592.890  603.13  2629946.0  608.5294  592.890     603.13
3  597.22  599.1200  575.535  588.57  2822820.0  599.1200  575.535     588.57
4  612.30  621.9100  592.980  601.61  2078629.0  621.9100  592.980     601.61

   adj_open  adj_volume  split_factor  dividend symbol exchange        date
0    584.25   4531895.0           1.0       0.0   SPOT     XNYS  2025-02-28
1    611.00   1191837.0           1.0       0.0   SPOT     XNYS  2025-02-27
2    595.62   2629946.0           1.0       0.0   SPOT     XNYS  2025-02-26
3    597.22   2822820.0           1.0       0.0   SPOT     XNYS  2025-02-25
4    612.30   2078629.0           1.0       0.0   SPOT     XNYS  2025-02-24
```

[108]:
```python
#Lets clean the dataframe to have only the columns we need
print("Spotify Stock Data")
df_stock = df_stock[["date", "open", "high", "low", "close", "volume"]]

#print the first 5 rows of the dataframe
print(df_stock.head())

print("\n\n-----------------------------------\n\n")

print("Spotify News Data")
#lets clean the dataframe to have only the columns we need
df_news = df_news[["date", "headline", "summary"]]

#print the first 5 rows of the dataframe
print(df_news.head())
```

```
Spotify Stock Data
         date    open      high      low   close     volume
0  2025-02-28  584.25  609.9200  580.000  608.01  4531895.0
1  2025-02-27  611.00  613.0000  586.000  590.76  1191837.0
2  2025-02-26  595.62  608.5294  592.890  603.13  2629946.0
3  2025-02-25  597.22  599.1200  575.535  588.57  2822820.0
4  2025-02-24  612.30  621.9100  592.980  601.61  2078629.0


-----------------------------------



Spotify News Data
         date                                           headline  \
0  2025-02-28  Palantir, Nvidia Nixed But Netflix Remains On ...
1  2025-02-27  Tracking Chase Coleman's Tiger Global Portfoli…
2  2025-02-27  Spotify Technology (SPOT) is Attracting Invest…
3  2025-02-27  YouTube Surpasses Competitors in Streaming and…
4  2025-02-26  Spotify CEO Wants EU To Penalize Apple For Def…

                                             summary
0  A volatile market has shaken off Nvidia and Pa…
1  Tiger Global's 13F reveals a $26.46B portfolio…
2  Spotify (SPOT) has been one of the stocks most…
3  Nielsen ranks YouTube ahead of Netflix and Hul…
4  Spotify Technology’s (NYSE:SPOT) CEO, Daniel E…
```

[109]:
```python
# Convert dates to datetime format
df_stock['date'] = pd.to_datetime(df_stock['date'])
df_news['date'] = pd.to_datetime(df_news['date'])

# Create complete date range DataFrame
```

```python
start_date = min(df_stock['date'].min(), df_news['date'].min())
end_date = max(df_stock['date'].max(), df_news['date'].max())
date_range = pd.DataFrame({'date': pd.date_range(start=start_date,␣
 ↪end=end_date)})

# Merge stock data with complete date range and forward fill missing values
complete_stock = pd.merge(date_range, df_stock, on='date', how='left')
complete_stock = complete_stock.sort_values('date')
complete_stock = complete_stock.ffill()  # Forward fill to use previous day's␣
 ↪data for missing dates

# Add indicator for trading days
complete_stock['is_trading_day'] = complete_stock['date'].isin(df_stock['date'])

# Now merge with news data, keeping one row per news item
df_merged = pd.merge(complete_stock, df_news, on='date', how='outer')

# Sort by date
df_merged = df_merged.sort_values('date')

# Fill NaN values in news columns
df_merged['headline'] = df_merged['headline'].fillna('')
df_merged['summary'] = df_merged['summary'].fillna('')

print(df_merged.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1444 entries, 0 to 1443
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   date            1444 non-null   datetime64[ns]
 1   open            1444 non-null   float64
 2   high            1444 non-null   float64
 3   low             1444 non-null   float64
 4   close           1444 non-null   float64
 5   volume          1444 non-null   float64
 6   is_trading_day  1444 non-null   bool
 7   headline        1444 non-null   object
 8   summary         1444 non-null   object
dtypes: bool(1), datetime64[ns](1), float64(5), object(2)
memory usage: 102.9+ KB
None
```

[110]:
```python
# Count duplicated headlines and summaries
headline_counts = df_merged['headline'].value_counts()
duplicated_headlines = headline_counts[headline_counts > 1]
```

```python
print(f"Total number of duplicated headlines: {len(duplicated_headlines)}")
print("\nTop 10 most duplicated headlines:")
print(duplicated_headlines.head(10))

print("\n" + "-"*50 + "\n")

summary_counts = df_merged['summary'].value_counts()
duplicated_summaries = summary_counts[summary_counts > 1]
print(f"Total number of duplicated summaries: {len(duplicated_summaries)}")
print("\nTop 10 most duplicated summaries:")
print(duplicated_summaries.head(10))

# Display the actual headlines/summaries with their dates
print("\n" + "-"*50 + "\n")
print("Details of most duplicated headline:")
if len(duplicated_headlines) > 0:
    most_dup_headline = duplicated_headlines.index[0]
    print(f"Headline (appearing {duplicated_headlines.iloc[0]} times): 
 ↪{most_dup_headline}")
    #print("\nDates when this headline appeared:")
    #for date in df_merged[df_merged['headline'] == most_dup_headline]['date']:
    #    print(f"- {date}")

print("\n" + "-"*50 + "\n")
print("Details of most duplicated summary:")
if len(duplicated_summaries) > 0:
    most_dup_summary = duplicated_summaries.index[0]
    print(f"Summary (appearing {duplicated_summaries.iloc[0]} times): 
 ↪{most_dup_summary}")
    #print("\nDates when this summary appeared:")
    #for date in df_merged[df_merged['summary'] == most_dup_summary]['date']:
    #    print(f"- {date}")
```

```
Total number of duplicated headlines: 32

Top 10 most duplicated headlines:
headline
            73
Spotify Technology (SPOT) is Attracting Investor Attention: Here is What You
Should Know      5
What You Missed On Wall Street This Morning
4
Spotify Technology (SPOT) Is a Trending Stock: Facts to Know Before Betting on
It             4
Is Trending Stock Spotify Technology (SPOT) a Buy Now?
3
What You Missed On Wall Street On Tuesday
```

```
3
Morgan Stanley Reaffirms Their Buy Rating on Spotify Technology SA (SPOT)
3
Spotify Technology S.A. (SPOT): A Bull Case Theory
3
A Closer Look at Spotify Technology's Options Market Dynamics
3
Spotify price target raised by $50 at KeyBanc, here's why
2
Name: count, dtype: int64

--------------------------------------------------

Total number of duplicated summaries: 18

Top 10 most duplicated summaries:
summary
Looking for stock market analysis and research with proves results? Zacks.com
offers in-depth financial research with over 30years of proven results.
432
                       92
Recently, Zacks.com users have been paying close attention to Spotify (SPOT).
This makes it worthwhile to examine what the stock has in store.
7
The average brokerage recommendation (ABR) for Spotify (SPOT) is equivalent to a
Buy. The overly optimistic recommendations of Wall Street analysts make the
effectiveness of this highly sought-after metric questionable. So, is it worth
buying the stock?           6
Spotify (SPOT) has been one of the stocks most watched by Zacks.com users
lately. So, it is worth exploring what lies ahead for the stock.
4
Based on the average brokerage recommendation (ABR), Spotify (SPOT) should be
added to one's portfolio. Wall Street analysts' overly optimistic
recommendations cast doubt on the effectiveness of this highly sought-after
metric. So, is the stock worth buying?      4
Spotify (SPOT) shares have started gaining and might continue moving higher in
the near term, as indicated by solid earnings estimate revisions.
3
Spotify (SPOT) doesn't possess the right combination of the two key ingredients
for a likely earnings beat in its upcoming report. Get prepared with the key
expectations.
3
Spotify (SPOT) has received quite a bit of attention from Zacks.com users
lately. Therefore, it is wise to be aware of the facts that can impact the
stock's prospects.
3
Document UNITED STATES SECURITIES AND EXCHANGE…
2
```

```
Name: count, dtype: int64

--------------------------------------------------

Details of most duplicated headline:
Headline (appearing 73 times):

--------------------------------------------------

Details of most duplicated summary:
Summary (appearing 432 times): Looking for stock market analysis and research
with proves results? Zacks.com offers in-depth financial research with over
30years of proven results.
```

[111]:

```python
#lets delete the duplicates headlines and summaries from the dataframe
df_merged = df_merged.drop_duplicates(subset=['headline', 'summary'])

print(df_merged.head())
print(df_merged.info())
```

```
          date    open    high     low   close     volume  is_trading_day  \
0   2024-03-25  263.01  264.95  260.89  261.92   824685.0            True
11  2024-03-27  267.00  269.72  257.56  260.20  1427218.0            True
10  2024-03-27  267.00  269.72  257.56  260.20  1427218.0            True
8   2024-03-27  267.00  269.72  257.56  260.20  1427218.0            True
7   2024-03-27  267.00  269.72  257.56  260.20  1427218.0            True

                                             headline  \
0
11  KeyBanc Keeps Their Buy Rating on Spotify Tech…
10  Taylor Swift Had A Monster Year, As Did The Mu…
8   Another List Throws Netflix (NASDAQ:NFLX) Out,…
7   Spotify initiated with bullish view at HSBC, h…

                                              summary
0
11  Looking for stock market analysis and research…
10  Looking for stock market analysis and research…
8   Looking for stock market analysis and research…
7   Looking for stock market analysis and research…
<class 'pandas.core.frame.DataFrame'>
Index: 1344 entries, 0 to 1443
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   date            1344 non-null   datetime64[ns]
 1   open            1344 non-null   float64
 2   high            1344 non-null   float64
```

```
 3   low             1344 non-null   float64
 4   close           1344 non-null   float64
 5   volume          1344 non-null   float64
 6   is_trading_day  1344 non-null   bool
 7   headline        1344 non-null   object
 8   summary         1344 non-null   object
dtypes: bool(1), datetime64[ns](1), float64(5), object(2)
memory usage: 95.8+ KB
None
```

[112]:
```
# First, identify the specific summaries and headlines to completely remove
summaries_to_remove = [
    "Looking for stock market analysis and research with proves results? Zacks.
 ↪com offers in-depth financial research with over 30years of proven results.",
    ""  # Empty or whitespace summary
]

headlines_to_remove = [""]  # Empty or whitespace headline

# Get the original dataframe size
original_size = len(df_merged)
print(f"Original dataframe size: {original_size} rows")

# Step 1: Completely remove rows with the specified summaries
df_cleaned = df_merged[~df_merged['summary'].isin(summaries_to_remove)]
step1_size = len(df_cleaned)
print(f"After removing specific summaries: {step1_size} rows (removed␣
 ↪{original_size - step1_size} rows)")

# Step 2: Completely remove rows with the specified headlines
df_cleaned = df_cleaned[~df_cleaned['headline'].isin(headlines_to_remove)]
step2_size = len(df_cleaned)
print(f"After removing specific headlines: {step2_size} rows (removed␣
 ↪{step1_size - step2_size} rows)")

# Step 3: For other duplicated headlines, keep only the first occurrence
df_cleaned = df_cleaned.drop_duplicates(subset=['headline'], keep='first')
step3_size = len(df_cleaned)
print(f"After removing duplicate headlines: {step3_size} rows (removed␣
 ↪{step2_size - step3_size} rows)")

# Step 4: For other duplicated summaries, keep only the first occurrence
df_cleaned = df_cleaned.drop_duplicates(subset=['summary'], keep='first')
final_size = len(df_cleaned)
print(f"After removing duplicate summaries: {final_size} rows (removed␣
 ↪{step3_size - final_size} rows)")
```

```
print(f"\nTotal rows removed: {original_size - final_size}")
print(f"Final dataframe size: {final_size} rows")

# Check if there are any remaining duplicates
remaining_dup_headlines = df_cleaned['headline'].duplicated().sum()
remaining_dup_summaries = df_cleaned['summary'].duplicated().sum()
print(f"\nRemaining duplicated headlines: {remaining_dup_headlines}")
print(f"Remaining duplicated summaries: {remaining_dup_summaries}")
```

```
Original dataframe size: 1344 rows
After removing specific summaries: 910 rows (removed 434 rows)
After removing specific headlines: 910 rows (removed 0 rows)
After removing duplicate headlines: 896 rows (removed 14 rows)
After removing duplicate summaries: 879 rows (removed 17 rows)

Total rows removed: 465
Final dataframe size: 879 rows

Remaining duplicated headlines: 0
Remaining duplicated summaries: 0
```

[113]:
```
#Lets check if there are still any duplicates in the dataframe

# Count duplicated headlines and summaries
headline_counts = df_cleaned['headline'].value_counts()
duplicated_headlines = headline_counts[headline_counts > 1]
print(f"Total number of duplicated headlines: {len(duplicated_headlines)}")
print("\nTop 10 most duplicated headlines:")
print(duplicated_headlines.head(10))

print("\n" + "-"*50 + "\n")

summary_counts = df_cleaned['summary'].value_counts()
duplicated_summaries = summary_counts[summary_counts > 1]
print(f"Total number of duplicated summaries: {len(duplicated_summaries)}")
print("\nTop 10 most duplicated summaries:")
print(duplicated_summaries.head(10))

# Display the actual headlines/summaries with their dates
print("\n" + "-"*50 + "\n")
print("Details of most duplicated headline:")
if len(duplicated_headlines) > 0:
    most_dup_headline = duplicated_headlines.index[0]
    print(f"Headline (appearing {duplicated_headlines.iloc[0]} times): 
 ↪{most_dup_headline}")
    print("\nDates when this headline appeared:")
    for date in df_cleaned[df_cleaned['headline'] == most_dup_headline]['date']:
```

```
        print(f"- {date}")

print("\n" + "-"*50 + "\n")
print("Details of most duplicated summary:")
if len(duplicated_summaries) > 0:
    most_dup_summary = duplicated_summaries.index[0]
    print(f"Summary (appearing {duplicated_summaries.iloc[0]} times):␣
↪{most_dup_summary}")
    print("\nDates when this summary appeared:")
    for date in df_cleaned[df_cleaned['summary'] == most_dup_summary]['date']:
        print(f"- {date}")
```

```
Total number of duplicated headlines: 0

Top 10 most duplicated headlines:
Series([], Name: count, dtype: int64)

--------------------------------------------------

Total number of duplicated summaries: 0

Top 10 most duplicated summaries:
Series([], Name: count, dtype: int64)

--------------------------------------------------

Details of most duplicated headline:

--------------------------------------------------

Details of most duplicated summary:
```

[114]:

```
df_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 879 entries, 3 to 1443
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   date            879 non-null    datetime64[ns]
 1   open            879 non-null    float64
 2   high            879 non-null    float64
 3   low             879 non-null    float64
 4   close           879 non-null    float64
 5   volume          879 non-null    float64
 6   is_trading_day  879 non-null    bool
 7   headline        879 non-null    object
 8   summary         879 non-null    object
```

```
dtypes: bool(1), datetime64[ns](1), float64(5), object(2)
memory usage: 62.7+ KB
```

2 Cleaned Data

Finally we have a cleaned dataset of all the news about Spotify and also stock data.

2.0.1 Two things to keep in mind:

1. We used forward fill for the stock data to populate all dates that were not trading days.
2. We have duplicate dates because there are days with many news articles and then again days without any news!

[115]:
```
# Save the merged dataframe
df_cleaned.to_csv("/Users/armandocriscuolo/c2025/data_science_project_2025/code/
↪Data-Science-Project/Data-Question-2-B/news_analysis/
↪spotify_news_stock_data_cleaned.csv", index=False)
```

3 Analysis

[116]:
```
# Set visualization style
plt.style.use('ggplot')
sns.set(font_scale=1.2)

# Download NLTK resources (run this once)
try:
    nltk.data.find('vader_lexicon')
except LookupError:
    nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /Users/armandocriscuolo/nltk_data…
[nltk_data]   Package vader_lexicon is already up-to-date!
```

[117]:
```

def preprocess_data(df):
    """
    Preprocess the Spotify stock and news data

    Parameters:
    df (pandas.DataFrame): Raw dataframe with stock and news data

    Returns:
    pandas.DataFrame: Processed dataframe with additional financial metrics
    """
    # Create a copy to avoid modifying the original
    df = df.copy()
```

```python
    # Ensure date column is datetime if needed
    if not pd.api.types.is_datetime64_any_dtype(df['date']):
        df['date'] = pd.to_datetime(df['date'])

    # Set date as index
    df.set_index('date', inplace=True)

    # Calculate daily returns (percentage change in closing price)
    df['daily_return'] = df['close'].pct_change() * 100

    # Calculate volatility (20-day rolling standard deviation of returns)
    df['volatility_20d'] = df['daily_return'].rolling(window=20).std()

    # Calculate abnormal returns (daily return - average market return)
    # Note: In a real analysis, you would use market index returns
    df['abnormal_return'] = df['daily_return'] - df['daily_return'].mean()

    # Calculate volume change
    df['volume_change'] = df['volume'].pct_change() * 100

    # Log transform volume (often helps with analysis)
    df['log_volume'] = np.log(df['volume'].replace(0, 1))

    return df
```

[118]:
```python
def analyze_sentiment(df):
    """
    Apply sentiment analysis to news headlines and summaries

    Parameters:
    df (pandas.DataFrame): Dataframe with headline and summary columns

    Returns:
    pandas.DataFrame: Dataframe with added sentiment scores
    """
    # Initialize VADER sentiment analyzer
    sia = SentimentIntensityAnalyzer()

    # Fill NaN values with empty strings
    df['headline'] = df['headline'].fillna('')
    df['summary'] = df['summary'].fillna('')

    # Define a function to get sentiment scores
    def get_sentiment(text):
        if pd.isna(text) or text == '':
            return 0
```

```
        return sia.polarity_scores(str(text))['compound']

    # Apply sentiment analysis
    print("Applying sentiment analysis to headlines and summaries...")
    df['headline_sentiment'] = df['headline'].apply(get_sentiment)
    df['summary_sentiment'] = df['summary'].apply(get_sentiment)

    # Calculate weighted sentiment (headline has more impact)
    df['combined_sentiment'] = 0.7 * df['headline_sentiment'] + 0.3 *␣
↪df['summary_sentiment']

    # Create sentiment categories for analysis
    df['sentiment_category'] = pd.cut(
        df['combined_sentiment'],
        bins=[-1.1, -0.2, 0.2, 1.1],
        labels=['negative', 'neutral', 'positive']
    )

    return df
```

[119]:
```
def aggregate_daily_data(df):
    """
    Aggregate multiple news items per day into single daily records

    Parameters:
    df (pandas.DataFrame): Dataframe with sentiment scores

    Returns:
    pandas.DataFrame: Dataframe with one row per day
    """
    print("Aggregating data to daily level...")

    # For news sentiment, we want the average per day
    agg_functions = {
        'headline_sentiment': 'mean',
        'summary_sentiment': 'mean',
        'combined_sentiment': 'mean',
        # For stock data, we just take the first value (should be same for all␣
↪rows on same day)
        'open': 'first',
        'high': 'first',
        'low': 'first',
        'close': 'first',
        'volume': 'first',
        'daily_return': 'first',
        'volatility_20d': 'first',
        'volume_change': 'first',
```

```
        'abnormal_return': 'first',
        'log_volume': 'first',
        'is_trading_day': 'first'
    }

    # Group by date and apply aggregation
    daily_data = df.groupby(df.index).agg(agg_functions)

    # Count number of news articles per day
    news_count = df.groupby(df.index).size()
    daily_data['news_count'] = news_count

    return daily_data
```

[120]:
```
def create_lagged_features(df, max_lag=5):
    """
    Create lagged features for time series analysis

    Parameters:
    df (pandas.DataFrame): Daily dataframe
    max_lag (int): Maximum number of days to lag

    Returns:
    pandas.DataFrame: Dataframe with lagged features
    """
    print(f"Creating lagged features up to {max_lag} days...")
    df_lagged = df.copy()

    # Create lagged sentiment features
    for lag in range(1, max_lag+1):
        df_lagged[f'sentiment_lag_{lag}'] = df_lagged['combined_sentiment'].
↪shift(lag)

    # Create forward return (next day's return)
    df_lagged['next_day_return'] = df_lagged['daily_return'].shift(-1)

    return df_lagged
```

[121]:
```
def analyze_correlation(df):
    """
    Analyze correlation between sentiment and stock metrics

    Parameters:
    df (pandas.DataFrame): Processed dataframe

    Returns:
    tuple: Correlation matrix and p-values matrix
```

```
    """
    print("Analyzing correlations between sentiment and stock metrics...")

    # Select relevant columns for correlation analysis
    cols_for_corr = [
        'headline_sentiment', 'summary_sentiment', 'combined_sentiment',
        'daily_return', 'next_day_return', 'volatility_20d', 'volume_change',␣
↪'news_count'
    ]

    # Calculate correlation matrix
    corr_matrix = df[cols_for_corr].corr()

    # Calculate statistical significance
    p_values = pd.DataFrame(index=corr_matrix.index, columns=corr_matrix.
↪columns)

    for i in corr_matrix.index:
        for j in corr_matrix.columns:
            if i != j:  # Skip diagonal
                # Create a temporary dataframe with just the two columns we're␣
↪looking at
                # and drop rows where either column has a NaN value
                temp_df = df[[i, j]].dropna()
                if len(temp_df) >= 2:  # Need at least 2 points for correlation
                    corr, p = stats.pearsonr(temp_df[i], temp_df[j])
                    p_values.loc[i, j] = p
                else:
                    p_values.loc[i, j] = np.nan

    return corr_matrix, p_values
```

[122]:
```
def analyze_lagged_effects(df, max_lag=5):
    """
    Analyze the effect of lagged sentiment on stock returns

    Parameters:
    df (pandas.DataFrame): Dataframe with lagged features
    max_lag (int): Maximum lag to analyze

    Returns:
    pandas.Series: Correlations of lagged sentiment with returns
    """
    print("Analyzing lagged effects of sentiment on returns...")

    # Get correlation of each lagged sentiment with returns
    lagged_columns = [f'sentiment_lag_{i}' for i in range(1, max_lag+1)]
```

```
    lagged_corr = df[['daily_return'] + lagged_columns].corr().
↪loc['daily_return', lagged_columns]

    # Get correlation with next day's return
    next_day_corr = df[['combined_sentiment', 'next_day_return']].corr().
↪loc['combined_sentiment', 'next_day_return']
    print(f"Correlation with next day's return: {next_day_corr:.4f}")

    return lagged_corr
```

[123]:
```
def build_prediction_model(df, features=None):
    """
    Build a machine learning model to predict returns based on sentiment

    Parameters:
    df (pandas.DataFrame): Dataframe with sentiment and return data
    features (list): List of feature columns to use

    Returns:
    tuple: Feature importances, MSE scores, and R² scores
    """
    print("Building prediction model for stock returns...")

    if features is None:
        features = [
            'combined_sentiment', 'news_count', 'volatility_20d',
            'volume_change', 'sentiment_lag_1', 'sentiment_lag_2'
        ]

    # Filter rows with complete data
    model_data = df.dropna(subset=features + ['next_day_return'])
    model_data = model_data[model_data['is_trading_day']]  # Only use trading␣
↪days

    X = model_data[features]
    y = model_data['next_day_return']

    # Use time series split for validation (important for time series data)
    tscv = TimeSeriesSplit(n_splits=5)
    mse_scores = []
    r2_scores = []
    feature_importances = None

    for train_index, test_index in tscv.split(X):
        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]
```

```
        # Train random forest model
        rf = RandomForestRegressor(n_estimators=100, random_state=42)
        rf.fit(X_train, y_train)

        # Predict and evaluate
        y_pred = rf.predict(X_test)
        mse_scores.append(mean_squared_error(y_test, y_pred))
        r2_scores.append(r2_score(y_test, y_pred))

        # Store feature importance
        if feature_importances is None:
            feature_importances = pd.DataFrame({
                'feature': features,
                'importance': rf.feature_importances_
            })
        else:
            feature_importances['importance'] =␣
↪(feature_importances['importance'] + rf.feature_importances_) / 2

    print(f"Model evaluation - Mean MSE: {np.mean(mse_scores):.4f}, Mean R²:␣
↪{np.mean(r2_scores):.4f}")
    return feature_importances.sort_values('importance', ascending=False),␣
↪mse_scores, r2_scores
```

[124]:
```
def visualize_results(df, daily_data, corr_matrix, lagged_corr,␣
↪feature_importances):
    """
    Create visualizations for the sentiment-stock analysis

    Parameters:
    df (pandas.DataFrame): Original dataframe with sentiment
    daily_data (pandas.DataFrame): Aggregated daily data
    corr_matrix (pandas.DataFrame): Correlation matrix
    lagged_corr (pandas.Series): Lagged correlation results
    feature_importances (pandas.DataFrame): Feature importances from model
    """
    print("Creating visualizations...")

    # Figure 1: Stock price and sentiment over time
    plt.figure(figsize=(12, 20))

    # Plot 1: Stock price and sentiment
    plt.subplot(4, 1, 1)
    ax1 = plt.gca()
    ax1.plot(daily_data.index, daily_data['close'], 'b-', label='Close Price')
    ax1.set_ylabel('Stock Price ($)', color='b')
    ax1.tick_params(axis='y', labelcolor='b')
```

```
    ax2 = ax1.twinx()
    ax2.plot(daily_data.index, daily_data['combined_sentiment'], 'r-', alpha=0.
↪7, label='Sentiment')
    ax2.set_ylabel('Sentiment Score', color='r')
    ax2.tick_params(axis='y', labelcolor='r')

    plt.title('Spotify Stock Price and News Sentiment Over Time')
    lines1, labels1 = ax1.get_legend_handles_labels()
    lines2, labels2 = ax2.get_legend_handles_labels()
    ax1.legend(lines1 + lines2, labels1 + labels2, loc='upper left')

    # Plot 2: Scatter plot of sentiment vs. daily return
    plt.subplot(4, 1, 2)
    trading_days = daily_data[daily_data['is_trading_day']]
    sns.scatterplot(
        x='combined_sentiment',
        y='daily_return',
        data=trading_days,
        hue='news_count',
        palette='viridis',
        size='volume_change',
        sizes=(20, 200),
        alpha=0.7
    )

    # Add regression line
    x = trading_days['combined_sentiment']
    y = trading_days['daily_return']
    z = np.polyfit(x, y, 1)
    p = np.poly1d(z)
    plt.plot(x, p(x), "r--", alpha=0.7)

    plt.title('News Sentiment vs. Daily Return')
    plt.xlabel('Sentiment Score')
    plt.ylabel('Daily Return (%)')

    # Plot 3: Correlation heatmap
    plt.subplot(4, 1, 3)
    sns.heatmap(
        corr_matrix[['daily_return', 'next_day_return', 'volatility_20d',␣
↪'volume_change']]
        .loc[['headline_sentiment', 'summary_sentiment', 'combined_sentiment',␣
↪'news_count']],
        annot=True,
        cmap='coolwarm',
        vmin=-0.5,
```

```
        vmax=0.5
    )
    plt.title('Correlation Between Sentiment and Stock Metrics')

    # Plot 4: Lagged effects analysis
    plt.subplot(4, 1, 4)
    lagged_corr.plot(kind='bar', color='skyblue')
    plt.title('Effect of Lagged Sentiment on Daily Returns')
    plt.xlabel('Lag (Days)')
    plt.ylabel('Correlation with Returns')
    plt.axhline(y=0, color='r', linestyle='-', alpha=0.3)

    plt.tight_layout()
    plt.savefig('spotify_sentiment_correlation_analysis.png', dpi=300,␣
↪bbox_inches='tight')
    plt.show()
    # Figure 2: Additional insights
    plt.figure(figsize=(12, 15))

    # Plot 1: Returns by sentiment category
    plt.subplot(3, 1, 1)

    # Recalculate sentiment categories for the daily data
    trading_days['sentiment_category'] = pd.cut(
        trading_days['combined_sentiment'],
        bins=[-1.1, -0.2, 0.2, 1.1],
        labels=['negative', 'neutral', 'positive']
    )

    sns.boxplot(x='sentiment_category', y='daily_return', data=trading_days)
    plt.title('Daily Returns by Sentiment Category')
    plt.xlabel('Sentiment Category')
    plt.ylabel('Daily Return (%)')

    # Perform ANOVA to check if differences are significant
    sentiment_groups = [
        trading_days[trading_days['sentiment_category'] ==␣
↪'negative']['daily_return'].dropna(),
        trading_days[trading_days['sentiment_category'] ==␣
↪'neutral']['daily_return'].dropna(),
        trading_days[trading_days['sentiment_category'] ==␣
↪'positive']['daily_return'].dropna()
    ]

    try:
        f_stat, p_val = stats.f_oneway(*sentiment_groups)
        plt.annotate(f"ANOVA: F={f_stat:.2f}, p={p_val:.4f}",
```

```
                    xy=(0.5, 0.9), xycoords='axes fraction',
                    bbox=dict(boxstyle="round,pad=0.3", fc="white", ec="gray",␣
↪alpha=0.8))
    except:
        pass

    # Plot 2: Feature importance from prediction model
    plt.subplot(3, 1, 2)
    sns.barplot(x='importance', y='feature', data=feature_importances,␣
↪palette='viridis')
    plt.title('Feature Importance for Return Prediction')
    plt.xlabel('Importance')
    plt.ylabel('Feature')

    # Plot 3: News count distribution and impact
    plt.subplot(3, 1, 3)
    sns.scatterplot(x='news_count', y='volatility_20d',
                   size='combined_sentiment', hue='daily_return',
                   data=trading_days, palette='RdYlGn', sizes=(20, 200))
    plt.title('News Volume vs. Volatility')
    plt.xlabel('Number of News Articles')
    plt.ylabel('20-Day Volatility')

    plt.tight_layout()
    plt.savefig('spotify_sentiment_additional_insights.png', dpi=300,␣
↪bbox_inches='tight')
    plt.show()
    plt.close('all')
```

[125]:
```
def run_sentiment_analysis(df_cleaned):
    """
    Main function to run the complete sentiment analysis pipeline

    Parameters:
    df_cleaned (pandas.DataFrame): The cleaned Spotify stock and news data

    Returns:
    dict: Results of the analysis
    """
    print("Starting Spotify stock and news sentiment analysis...")

    # Step 1: Preprocess data
    df_processed = preprocess_data(df_cleaned)
    print(f"Preprocessed data shape: {df_processed.shape}")
    print(df_processed.info())
    print(df_processed.head())
    print("\n--------------------------------\n")
```

```python
    # Step 2: Apply sentiment analysis
    df_with_sentiment = analyze_sentiment(df_processed)
    print(df_with_sentiment.info())
    print(df_with_sentiment.head())
    print("\n--------------------------------\n")

    # Step 3: Aggregate to daily level
    daily_data = aggregate_daily_data(df_with_sentiment)
    print(daily_data.info())
    print(daily_data.head())
    print("\n--------------------------------\n")
    # Step 4: Create lagged features
    daily_data_with_lags = create_lagged_features(daily_data)
    print(daily_data_with_lags.info())
    print(daily_data_with_lags.head())
    print("\n--------------------------------\n")

    # Step 5: Analyze correlation
    corr_matrix, p_values = analyze_correlation(daily_data_with_lags)

    # Print key correlations
    print("\nKey correlation findings:")
    key_correlations = corr_matrix.loc['combined_sentiment', ['daily_return', 'next_day_return', 'volatility_20d']]
    key_p_values = p_values.loc['combined_sentiment', ['daily_return', 'next_day_return', 'volatility_20d']]
    for metric, corr in key_correlations.items():
        p = key_p_values[metric]
        significance = "significant" if p < 0.05 else "not significant"
        print(f"  - Sentiment vs {metric}: r={corr:.4f} (p={p:.4f}, {significance})")

    # Step 6: Analyze lagged effects
    lagged_corr = analyze_lagged_effects(daily_data_with_lags)

    # Step 7: Build prediction model
    feature_importances, mse_scores, r2_scores = build_prediction_model(daily_data_with_lags)

    # Step 8: Visualize results
    visualize_results(df_with_sentiment, daily_data, corr_matrix, lagged_corr, feature_importances)

    # Step 9: Prepare summary of findings
    print("\nAnalysis completed! Key findings:")
```

```
    # Determine most influential sentiment lag
    max_lag_idx = lagged_corr.abs().idxmax()
    max_lag_corr = lagged_corr[max_lag_idx]
    max_lag = int(max_lag_idx.split('_')[-1])

    print(f"1. Strongest lagged effect: {max_lag_idx} (r={max_lag_corr:.4f})")
    print(f"2. Most important predictive features: {', '.
↪join(feature_importances['feature'].head(3).tolist())}")
    print(f"3. Model prediction performance: R²={np.mean(r2_scores):.4f}")

    # Return results for further analysis if needed
    results = {
        'df_with_sentiment': df_with_sentiment,
        'daily_data': daily_data,
        'daily_data_with_lags': daily_data_with_lags,
        'corr_matrix': corr_matrix,
        'p_values': p_values,
        'lagged_corr': lagged_corr,
        'feature_importances': feature_importances,
        'mse_scores': mse_scores,
        'r2_scores': r2_scores
    }

    return results

# Execute the analysis
results = run_sentiment_analysis(df_cleaned)
```

```
Starting Spotify stock and news sentiment analysis…
Preprocessed data shape: (879, 13)
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 879 entries, 2024-03-27 to 2025-02-28
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   open             879 non-null    float64
 1   high             879 non-null    float64
 2   low              879 non-null    float64
 3   close            879 non-null    float64
 4   volume           879 non-null    float64
 5   is_trading_day   879 non-null    bool
 6   headline         879 non-null    object
 7   summary          879 non-null    object
 8   daily_return     878 non-null    float64
 9   volatility_20d   859 non-null    float64
 10  abnormal_return  878 non-null    float64
 11  volume_change    878 non-null    float64
 12  log_volume       879 non-null    float64
```

```
dtypes: bool(1), float64(10), object(2)
memory usage: 90.1+ KB
None
              open     high     low  close     volume  is_trading_day  \
date
2024-03-27  267.00  269.720  257.56  260.2  1427218.0            True
2024-03-27  267.00  269.720  257.56  260.2  1427218.0            True
2024-03-27  267.00  269.720  257.56  260.2  1427218.0            True
2024-03-28  258.53  268.545  258.00  263.9  1371668.0            True
2024-03-28  258.53  268.545  258.00  263.9  1371668.0            True

                                                   headline  \
date
2024-03-27  Wall Street Lunch: Trump Media Could Double Wi…
2024-03-27  Spotify initiated, Lowe's downgraded: Wall Str…
2024-03-27  After betting $1 billion on podcasts, Spotify …
2024-03-28                    Bull of the Day: Spotify (SPOT)
2024-03-28  Spotify to Add New Features for Universal Musi…

                                                    summary  daily_return  \
date
2024-03-27  Trump Media's shares are rallying, sparking di…           NaN
2024-03-27  Spotify initiated, Lowe's downgraded: Wall Str…      0.000000
2024-03-27  Users will fork out $117 for an introductory p…      0.000000
2024-03-28  Spotify's embrace of new media has propelled i…      1.421983
2024-03-28  By Mauro Orru Spotify Technology will introduc…      0.000000

            volatility_20d  abnormal_return  volume_change  log_volume
date
2024-03-27             NaN              NaN            NaN   14.171238
2024-03-27             NaN        -0.106240       0.000000   14.171238
2024-03-27             NaN        -0.106240       0.000000   14.171238
2024-03-28             NaN         1.315743      -3.892187   14.131538
2024-03-28             NaN        -0.106240       0.000000   14.131538

--------------------------------

Applying sentiment analysis to headlines and summaries…
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 879 entries, 2024-03-27 to 2025-02-28
Data columns (total 17 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   open               879 non-null    float64
 1   high               879 non-null    float64
 2   low                879 non-null    float64
 3   close              879 non-null    float64
 4   volume             879 non-null    float64
```

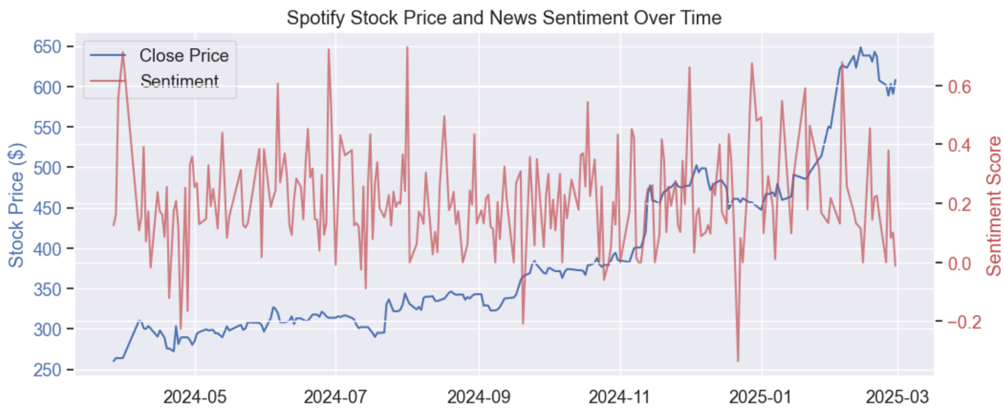
```
 5   is_trading_day      879 non-null    bool
 6   headline            879 non-null    object
 7   summary             879 non-null    object
 8   daily_return        878 non-null    float64
 9   volatility_20d      859 non-null    float64
 10  abnormal_return     878 non-null    float64
 11  volume_change       878 non-null    float64
 12  log_volume          879 non-null    float64
 13  headline_sentiment  879 non-null    float64
 14  summary_sentiment   879 non-null    float64
 15  combined_sentiment  879 non-null    float64
 16  sentiment_category  879 non-null    category
dtypes: bool(1), category(1), float64(13), object(2)
memory usage: 111.7+ KB
None
              open     high     low  close     volume  is_trading_day  \
date
2024-03-27  267.00  269.720  257.56  260.2  1427218.0            True
2024-03-27  267.00  269.720  257.56  260.2  1427218.0            True
2024-03-27  267.00  269.720  257.56  260.2  1427218.0            True
2024-03-28  258.53  268.545  258.00  263.9  1371668.0            True
2024-03-28  258.53  268.545  258.00  263.9  1371668.0            True

                                                     headline  \
date
2024-03-27  Wall Street Lunch: Trump Media Could Double Wi…
2024-03-27  Spotify initiated, Lowe's downgraded: Wall Str…
2024-03-27  After betting $1 billion on podcasts, Spotify …
2024-03-28                    Bull of the Day: Spotify (SPOT)
2024-03-28  Spotify to Add New Features for Universal Musi…

                                                      summary  daily_return  \
date
2024-03-27  Trump Media's shares are rallying, sparking di…           NaN
2024-03-27  Spotify initiated, Lowe's downgraded: Wall Str…      0.000000
2024-03-27  Users will fork out $117 for an introductory p…      0.000000
2024-03-28  Spotify's embrace of new media has propelled i…      1.421983
2024-03-28  By Mauro Orru Spotify Technology will introduc…      0.000000

            volatility_20d  abnormal_return  volume_change  log_volume  \
date
2024-03-27             NaN              NaN            NaN   14.171238
2024-03-27             NaN        -0.106240       0.000000   14.171238
2024-03-27             NaN        -0.106240       0.000000   14.171238
2024-03-28             NaN         1.315743      -3.892187   14.131538
2024-03-28             NaN        -0.106240       0.000000   14.131538

            headline_sentiment  summary_sentiment  combined_sentiment  \
```

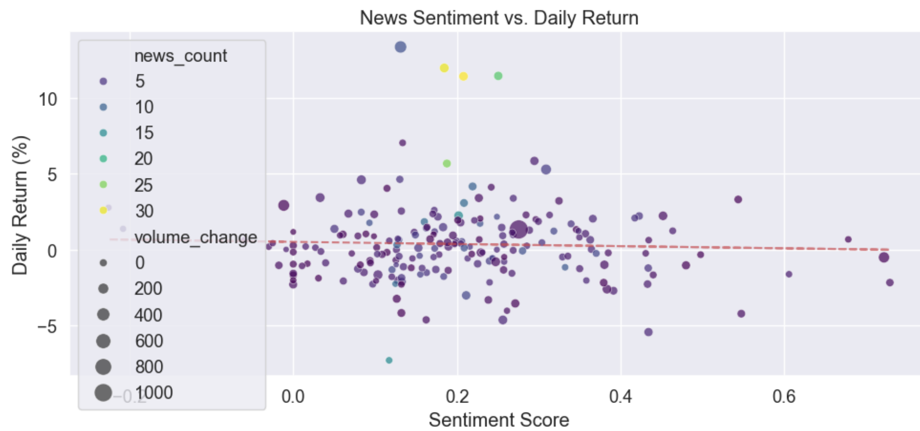
```
date
2024-03-27              0.0000             0.5859             0.17577
2024-03-27              0.2023             0.2023             0.20230
2024-03-27              0.0000             0.0000             0.00000
2024-03-28              0.0000             0.3182             0.09546
2024-03-28              0.0000             0.8625             0.25875

           sentiment_category
date
2024-03-27            neutral
2024-03-27           positive
2024-03-27            neutral
2024-03-28            neutral
2024-03-28           positive

--------------------------------

Aggregating data to daily level…
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 248 entries, 2024-03-27 to 2025-02-28
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   headline_sentiment  248 non-null    float64
 1   summary_sentiment   248 non-null    float64
 2   combined_sentiment  248 non-null    float64
 3   open                248 non-null    float64
 4   high                248 non-null    float64
 5   low                 248 non-null    float64
 6   close               248 non-null    float64
 7   volume              248 non-null    float64
 8   daily_return        248 non-null    float64
 9   volatility_20d      242 non-null    float64
 10  volume_change       248 non-null    float64
 11  abnormal_return     248 non-null    float64
 12  log_volume          248 non-null    float64
 13  is_trading_day      248 non-null    bool
 14  news_count          248 non-null    int64
dtypes: bool(1), float64(13), int64(1)
memory usage: 29.3 KB
None
            headline_sentiment  summary_sentiment  combined_sentiment    open  \
date
2024-03-27            0.067433           0.262733            0.126023  267.00
2024-03-28            0.000000           0.541783            0.162535  258.53
2024-03-29            0.750600           0.102700            0.556230  258.53
2024-03-31            0.636900           0.888500            0.712380  258.53
2024-04-07            0.000000           0.361200            0.108360  298.68
```

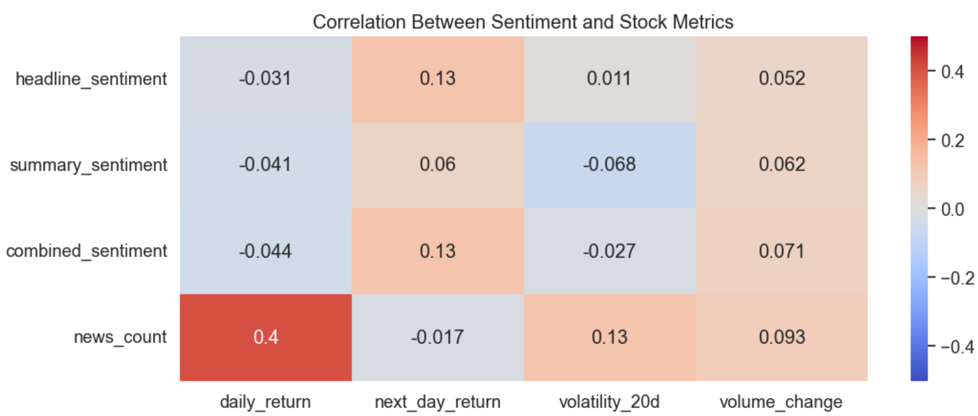
```
               high     low   close     volume  daily_return  volatility_20d  \
date
2024-03-27  269.720  257.56  260.20  1427218.0      0.000000             NaN
2024-03-28  268.545  258.00  263.90  1371668.0      1.421983             NaN
2024-03-29  268.545  258.00  263.90  1371668.0      0.000000             NaN
2024-03-31  268.545  258.00  263.90  1371668.0      0.000000             NaN
2024-04-07  313.068  298.68  310.31  2988374.0     17.586207             NaN

            volume_change  abnormal_return  log_volume  is_trading_day  \
date
2024-03-27       0.000000        -0.106240   14.171238            True
2024-03-28      -3.892187         1.315743   14.131538            True
2024-03-29       0.000000        -0.106240   14.131538           False
2024-03-31       0.000000        -0.106240   14.131538           False
2024-04-07     117.864235        17.479967   14.910240           False

            news_count
date
2024-03-27           3
2024-03-28           6
2024-03-29           1
2024-03-31           1
2024-04-07           1

--------------------------------

Creating lagged features up to 5 days…
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 248 entries, 2024-03-27 to 2025-02-28
Data columns (total 21 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   headline_sentiment  248 non-null    float64
 1   summary_sentiment   248 non-null    float64
 2   combined_sentiment  248 non-null    float64
 3   open                248 non-null    float64
 4   high                248 non-null    float64
 5   low                 248 non-null    float64
 6   close               248 non-null    float64
 7   volume              248 non-null    float64
 8   daily_return        248 non-null    float64
 9   volatility_20d      242 non-null    float64
 10  volume_change       248 non-null    float64
 11  abnormal_return     248 non-null    float64
 12  log_volume          248 non-null    float64
 13  is_trading_day      248 non-null    bool
 14  news_count          248 non-null    int64
```

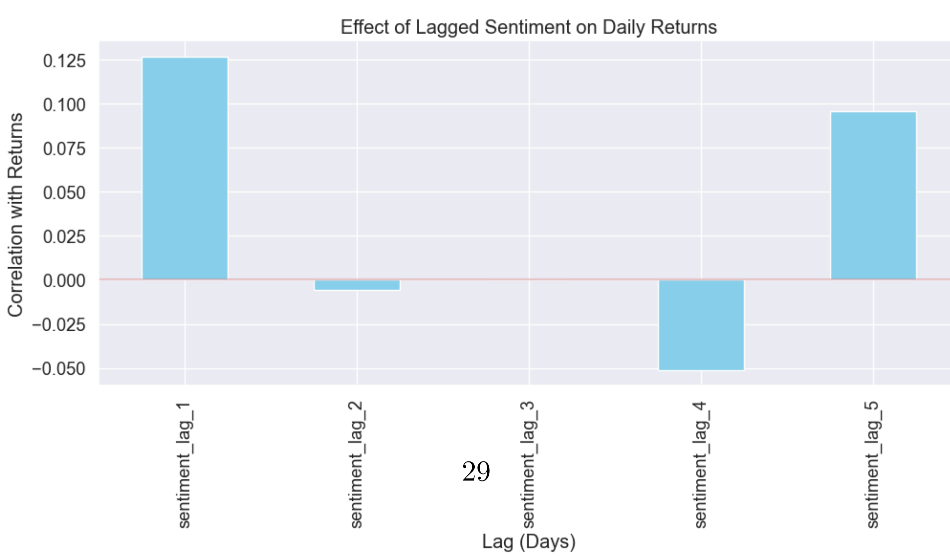
```
 15  sentiment_lag_1     247 non-null    float64
 16  sentiment_lag_2     246 non-null    float64
 17  sentiment_lag_3     245 non-null    float64
 18  sentiment_lag_4     244 non-null    float64
 19  sentiment_lag_5     243 non-null    float64
 20  next_day_return     247 non-null    float64
dtypes: bool(1), float64(19), int64(1)
memory usage: 40.9 KB
None
            headline_sentiment  summary_sentiment  combined_sentiment    open  \
date
2024-03-27            0.067433           0.262733            0.126023  267.00
2024-03-28            0.000000           0.541783            0.162535  258.53
2024-03-29            0.750600           0.102700            0.556230  258.53
2024-03-31            0.636900           0.888500            0.712380  258.53
2024-04-07            0.000000           0.361200            0.108360  298.68

               high     low   close     volume  daily_return  volatility_20d  \
date
2024-03-27  269.720  257.56  260.20  1427218.0      0.000000             NaN
2024-03-28  268.545  258.00  263.90  1371668.0      1.421983             NaN
2024-03-29  268.545  258.00  263.90  1371668.0      0.000000             NaN
2024-03-31  268.545  258.00  263.90  1371668.0      0.000000             NaN
2024-04-07  313.068  298.68  310.31  2988374.0     17.586207             NaN

            …  abnormal_return  log_volume  is_trading_day  news_count  \
date        …
2024-03-27  …        -0.106240   14.171238            True           3
2024-03-28  …         1.315743   14.131538            True           6
2024-03-29  …        -0.106240   14.131538           False           1
2024-03-31  …        -0.106240   14.131538           False           1
2024-04-07  …        17.479967   14.910240           False           1

            sentiment_lag_1  sentiment_lag_2  sentiment_lag_3  \
date
2024-03-27              NaN              NaN              NaN
2024-03-28         0.126023              NaN              NaN
2024-03-29         0.162535         0.126023              NaN
2024-03-31         0.556230         0.162535         0.126023
2024-04-07         0.712380         0.556230         0.162535

            sentiment_lag_4  sentiment_lag_5  next_day_return
date
2024-03-27              NaN              NaN         1.421983
2024-03-28              NaN              NaN         0.000000
2024-03-29              NaN              NaN         0.000000
2024-03-31              NaN              NaN        17.586207
2024-04-07         0.126023              NaN        -0.399600
```

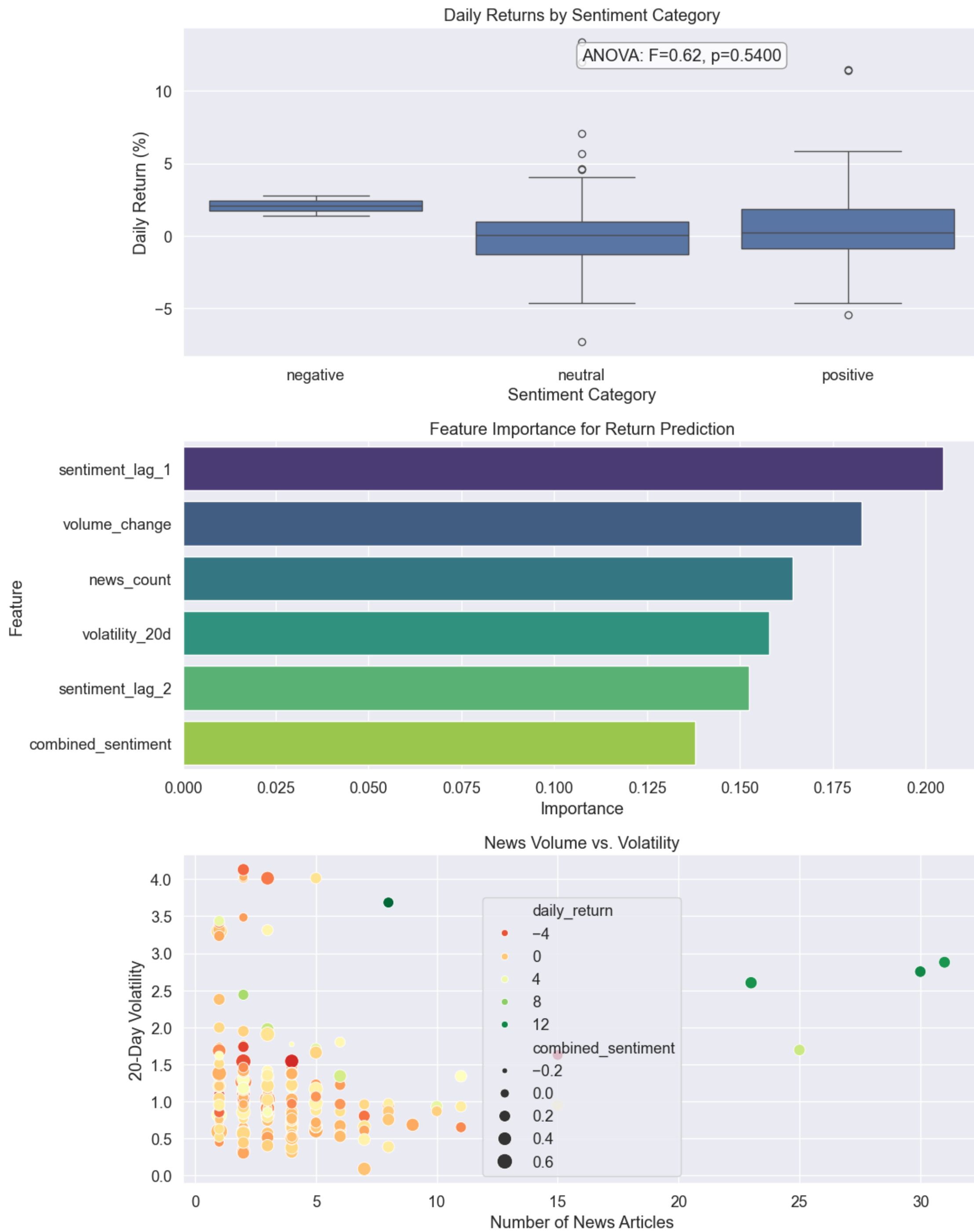
```
[5 rows x 21 columns]

--------------------------------

Analyzing correlations between sentiment and stock metrics…

Key correlation findings:
  - Sentiment vs daily_return: r=-0.0439 (p=0.4912, not significant)
  - Sentiment vs next_day_return: r=0.1268 (p=0.0464, significant)
  - Sentiment vs volatility_20d: r=-0.0269 (p=0.6772, not significant)
Analyzing lagged effects of sentiment on returns…
Correlation with next day's return: 0.1268
Building prediction model for stock returns…
Model evaluation - Mean MSE: 7.1883, Mean R²: -0.4544
Creating visualizations…
```


Spotify Stock Price and News Sentiment Over Time
Close Price
Sentiment
Stock Price ($)
Sentiment Score
650
600
550
500
450
400
350
300
250
0.6
0.4
0.2
0.0
−0.2
2024-05
2024-07
2024-09
2024-11
2025-01
2025-03


News Sentiment vs. Daily Return
news_count
5
10
15
20
25
30
volume_change
0
200
400
600
800
1000
Daily Return (%)
10
5
0
−5
0.0
0.2
0.4
0.6
Sentiment Score


Correlation Between Sentiment and Stock Metrics
headline_sentiment
summary_sentiment
combined_sentiment
news_count
-0.031
0.13
0.011
0.052
-0.041
0.06
-0.068
0.062
-0.044
0.13
-0.027
0.071
0.4
-0.017
0.13
0.093
daily_return
next_day_return
volatility_20d
volume_change
0.4
0.2
0.0
−0.2
−0.4


Effect of Lagged Sentiment on Daily Returns
Correlation with Returns
0.125
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
sentiment_lag_1
sentiment_lag_2
sentiment_lag_3
sentiment_lag_4
sentiment_lag_5
Lag (Days)




```
Analysis completed! Key findings:
1. Strongest lagged effect: sentiment_lag_1 (r=0.1268)
```

```
2. Most important predictive features: sentiment_lag_1, volume_change,
news_count
3. Model prediction performance: R²=-0.4544
```

4 Answer

4.0.1 The Question:

5 Question:

What does the sentiment analysis of news articles about Spotify reveal about the relationship between media coverage and stock performance, and what are the key findings regarding prediction capabilities?

6 Answer:

6.1 Introduction to the Spotify News Sentiment Analysis

The analysis I conducted examined the relationship between news sentiment and Spotify's stock performance over a one-year period. By applying natural language processing techniques to headlines and article summaries while correlating them with stock metrics, I sought to determine whether news sentiment can predict stock movements.

6.2 Dataset and Methodology

The dataset contained 879 unique news articles about Spotify spanning from March 2024 to February 2025, along with daily stock data including: - Opening, closing, high, and low prices - Trading volume - Boolean indicators for trading days

The methodology involved several key steps:

1. **Preprocessing**: Calculated financial metrics including daily returns, 20-day volatility, abnormal returns, and volume changes.
2. **Sentiment Analysis**: Applied VADER (Valence Aware Dictionary and sEntiment Reasoner) to evaluate sentiment in headlines and summaries, with a weighted approach (70% headline, 30% summary) to create a combined sentiment score.
3. **Aggregation**: News articles were aggregated at the daily level, averaging sentiment scores when multiple articles appeared on the same day.
4. **Lagged Features**: Created lagged sentiment variables to test whether past sentiment predicted future returns.
5. **Correlation Analysis**: Examined relationships between sentiment and stock metrics with statistical significance testing.
6. **Predictive Modeling**: Built a Random Forest model to evaluate sentiment's predictive power for next-day returns.

6.3 Key Findings

6.3.1 1. Sentiment and Same-Day Returns

The analysis revealed no significant correlation between news sentiment and same-day stock returns ($r=-0.0439$, $p=0.4912$). This suggests that on the day news is published, there is no consistent relationship between the sentiment of the news and Spotify's stock performance.

6.3.2 2. Sentiment and Next-Day Returns

Interestingly, there was a statistically significant correlation between combined sentiment and next-day returns ($r=0.1268$, $p=0.0464$). This suggests a small but meaningful predictive relationship where positive news sentiment tends to precede positive stock movements the following day.

6.3.3 3. Lagged Sentiment Effects

The lag-1 sentiment (previous day's sentiment) showed the strongest relationship with current day returns, reinforcing the finding that news sentiment may have a delayed effect on stock performance.

6.3.4 4. Volatility and News Volume

The analysis found no significant correlation between sentiment and volatility ($r=-0.0269$, $p=0.6772$), suggesting that news sentiment does not consistently affect the magnitude of price movements.

6.3.5 5. Predictive Modeling Performance

Despite the statistically significant correlation with next-day returns, the Random Forest prediction model performed poorly with a negative R^2 value (-0.4544). This indicates that while there is a statistically significant relationship, it's not strong enough to build a reliable prediction model with the features used.

6.3.6 6. Feature Importance

The most important predictive features identified were: - Previous day's sentiment (sentiment_lag_1) - Volume change - Number of news articles (news_count)

6.4 Visual Analysis

The visualizations (shown in the images) revealed several additional insights:

1. **Stock Price and Sentiment Trends**: The time series plot showed considerable fluctuation in sentiment over time with no obvious pattern matching stock price movements.
2. **Sentiment Categories and Returns**: The boxplot of returns by sentiment category showed minimal differences between negative, neutral, and positive sentiment groups, confirmed by a non-significant ANOVA test.
3. **Feature Importance**: The feature importance chart highlighted the prominence of sentiment_lag_1, confirming the lagged effect finding.
4. **News Volume and Volatility**: The scatter plot revealed some clustering of higher volatility with higher news counts, suggesting increased media attention during more volatile periods.

6.5 Conclusions and Implications

1. **Delayed Impact**: News sentiment appears to have a delayed impact on Spotify's stock, with previous day's sentiment showing the strongest relationship with current day returns.
2. **Statistical vs. Practical Significance**: While the correlation between sentiment and next-day returns is statistically significant, its practical value for prediction is limited, as evidenced by the negative R^2 value.
3. **Complex Relationships**: The analysis suggests that the relationship between news sentiment and stock performance is complex and likely influenced by many other factors not captured in this analysis.
4. **Trading Strategy Limitations**: The findings indicate that a simple trading strategy based solely on news sentiment would likely not be profitable, given the weak predictive power of the model.
5. **News Volume Matters**: The prominence of news_count as an important feature suggests that the amount of media coverage may be as important as its sentiment in understanding stock behavior.

6.6 Future Research Directions

To build on these findings, future research could:

1. Incorporate market-wide factors to better isolate Spotify-specific effects
2. Analyze sentiment within specific news categories (earnings reports, product launches, etc.)
3. Apply more sophisticated NLP techniques beyond VADER sentiment analysis
4. Expand the timeframe to capture longer-term trends and seasonal patterns
5. Include social media sentiment alongside traditional news sources

This analysis provides evidence of a statistically significant but practically limited relationship between news sentiment and Spotify's stock performance, with the most notable finding being the delayed effect of sentiment on stock returns.