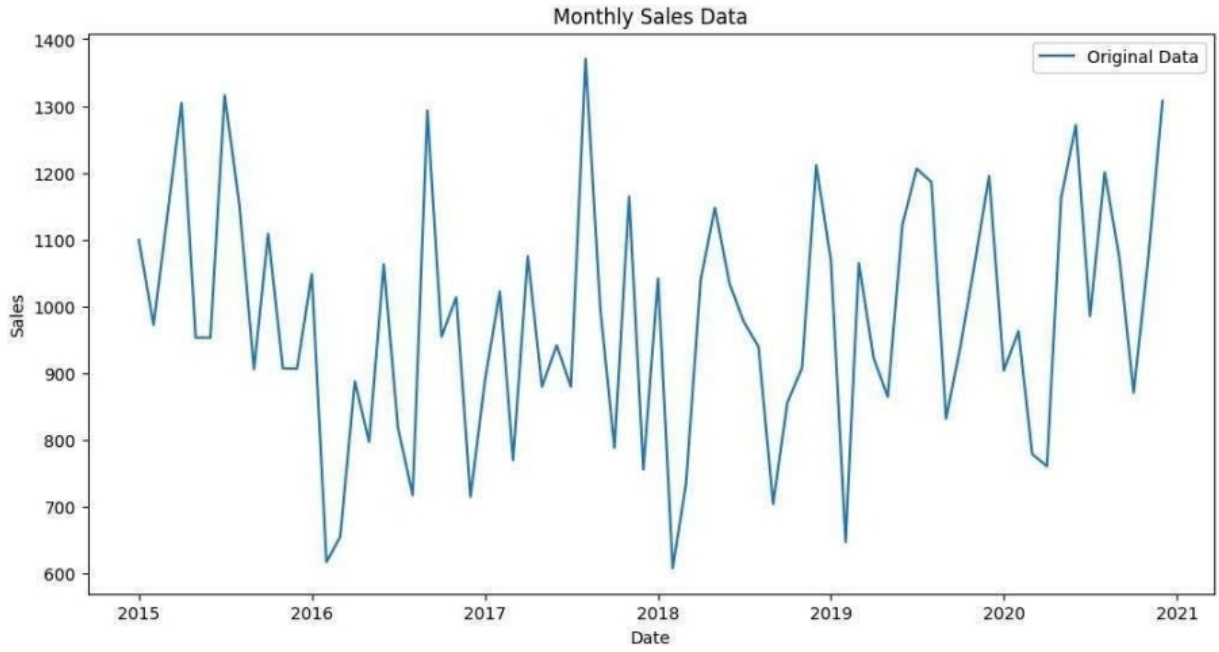
```
# If you don't have the required libraries, install them first
!pip install pandas matplotlib statsmodels
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.0.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: statsmodels in /usr/local/lib/python3.10/dist-packages (0.14.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.25.2)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.51.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (24.0)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.2)
Requirement already satisfied: scipy!=1.9.2,>=1.8 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (1.11.4)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (0.5.6)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.6->statsmodels) (1.16.0)
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA

# Create a sample time series data (monthly sales data)
np.random.seed(42)  # For reproducibility
date_rng = pd.date_range(start='2015-01-01', end='2020-12-01', freq='MS')  # Monthly start frequency
sales_data = np.random.normal(1000, 200, size=(len(date_rng)))  # Random sales data
df = pd.DataFrame(data={'Date': date_rng, 'Sales': sales_data})
df.set_index('Date', inplace=True)

# Plot the original data
plt.figure(figsize=(12, 6))
plt.plot(df['Sales'], label='Original Data')
plt.title('Monthly Sales Data')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.legend()
plt.show()

# Perform seasonal decomposition
result = seasonal_decompose(df['Sales'], model='additive', period=12)

# Plot the decomposed components
result.plot()
plt.show()

# Create an ARIMA model for forecasting (you can experiment with different orders)
model = ARIMA(df['Sales'], order=(2, 1, 2))  # ARIMA(p, d, q) parameters
arima_result = model.fit()

# Summary of the model
print(arima_result.summary())

# Forecast the next 12 months
forecast_steps = 12
forecast_result = arima_result.forecast(steps=forecast_steps)

# Plot the forecasted results
plt.figure(figsize=(12, 6))
plt.plot(df['Sales'], label='Historical Data')
plt.plot(pd.date_range(start='2021-01-01', periods=forecast_steps,  freq='MS'), forecast_result, label='Forecast', linestyle='--')
plt.title('Sales Forecast')
plt.xlabel('Date')
plt.ylabel('Sales')
plt.legend()
plt.show()
```




```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so infe
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so infe
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so infe
  self._init_dates(dates, freq)
                               SARIMAX Results
==========================================================================================
Dep. Variable:                      Sales   No. Observations:                   72
Model:                 ARIMA(2, 1, 2)   Log Likelihood                -468.809
Date:                Wed, 08 May 2024   AIC                            947.617
Time:                        11:17:15   BIC                            958.931
Sample:                    01-01-2015   HQIC                           952.116
                         - 12-01-2020
Covariance Type:                  opg
==========================================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.8217      0.170     -4.823      0.000      -1.156      -0.488
ar.L2         -0.0666      0.158     -0.420      0.674      -0.377       0.244
ma.L1          0.0781     18.954      0.004      0.997     -37.071      37.228
ma.L2         -0.9218     17.477     -0.053      0.958     -35.176      33.333
sigma2      3.004e+04   5.69e+05      0.053      0.958   -1.09e+06    1.15e+06
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 0.25
Prob(Q):                              0.95   Prob(JB):                         0.88
Heteroskedasticity (H):               0.90   Skew:                             0.14
Prob(H) (two-sided):                  0.80   Kurtosis:                         2.91
===================================================================================
```

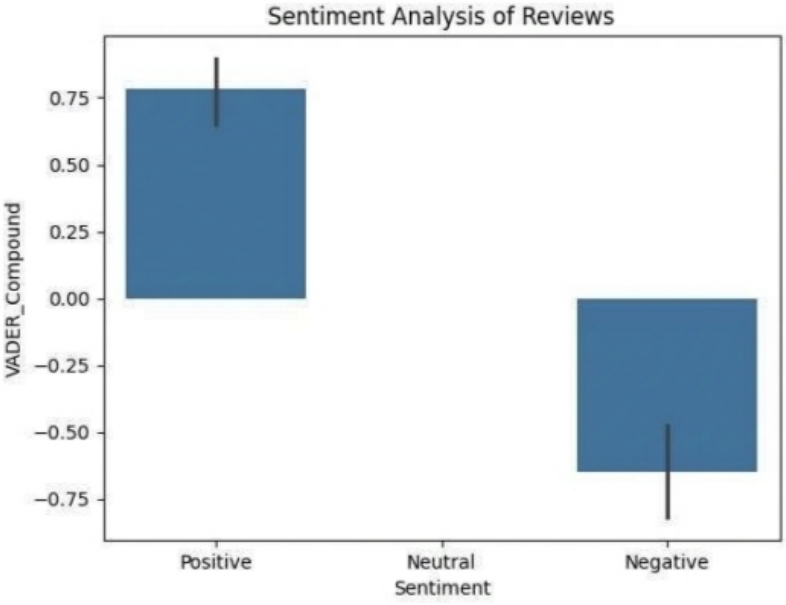
```
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```


```
# If you don't have the required libraries, install them first
!pip install pandas nltk textblob vaderSentiment
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.0.3)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: textblob in /usr/local/lib/python3.10/dist-packages (0.17.1)
Collecting vaderSentiment
  Downloading vaderSentiment-3.3.2-py2.py3-none-any.whl (125 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 126.0/126.0 kB 2.0 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.25.2)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.12.25)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.4)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from vaderSentiment) (2.31.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->vaderSentiment) (2024.2.2)
Installing collected packages: vaderSentiment
Successfully installed vaderSentiment-3.3.2
```

```
import pandas as pd
from textblob import TextBlob
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import seaborn as sns
import matplotlib.pyplot as plt

# Example unstructured text data (e.g., customer reviews)
reviews = [
    "I love this product! It works so well and exceeded my expectations.",
    "The service was terrible. I will never shop here again.",
    "It's okay, but I expected it to be better.",
    "Fantastic experience! The staff was friendly, and the food was delicious.",
    "The quality is decent, but the price is too high for what you get.",
    "Worst experience ever. I'm very disappointed with the customer service.",
]

# Create a DataFrame to store the text data
df = pd.DataFrame(data={'Review': reviews})

# Initialize the VADER sentiment analyzer
analyzer = SentimentIntensityAnalyzer()

# Function to calculate sentiment scores using TextBlob and VADER
def analyze_sentiment(text):
    # Using TextBlob for polarity (scale from -1 to +1)
    blob = TextBlob(text)
    textblob_polarity = blob.sentiment.polarity

    # Using VADER for compound sentiment (scale from -1 to +1)
    vader_scores = analyzer.polarity_scores(text)
    vader_compound = vader_scores['compound']

    return textblob_polarity, vader_compound

# Apply the sentiment analysis to each review
df['TextBlob_Polarity'], df['VADER_Compound'] = zip(*df['Review'].apply(analyze_sentiment))

# Determine the overall sentiment based on the compound score from VADER
def classify_sentiment(compound_score):
    if compound_score > 0.05:
        return 'Positive'
    elif compound_score < -0.05:
        return 'Negative'
    else:
        return 'Neutral'

df['Sentiment'] = df['VADER_Compound'].apply(classify_sentiment)

# Display the DataFrame with sentiment analysis results
print(df)

# Visualize the sentiment analysis results
sns.barplot(x='Sentiment', y='VADER_Compound', data=df, order=['Positive', 'Neutral', 'Negative'])
plt.title("Sentiment Analysis of Reviews")
plt.show()
```

```
                                              Review  TextBlob_Polarity  \
0  I love this product! It works so well and exce...           0.625000
1  The service was terrible. I will never shop he...          -1.000000
2         It's okay, but I expected it to be better.           0.300000
3  Fantastic experience! The staff was friendly, ...           0.625000
4  The quality is decent, but the price is too hi...           0.163333
5  Worst experience ever. I'm very disappointed w...          -0.987500

   VADER_Compound Sentiment
0          0.8038  Positive
1         -0.4767  Negative
2          0.6486  Positive
3          0.8955  Positive
4          0.0000   Neutral
5         -0.8173  Negative
```




```
# If you don't have the required libraries, install them first
!pip install pandas matplotlib sklearn
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.0.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Collecting sklearn
  Downloading sklearn-0.0.post12.tar.gz (2.6 kB)
  error: subprocess-exited-with-error

  × python setup.py egg_info did not run successfully.
  │ exit code: 1
  ╰─> See above for output.

  note: This error originates from a subprocess, and is likely not a problem with pip.
  Preparing metadata (setup.py) ... error
error: metadata-generation-failed

× Encountered error while generating package metadata.
╰─> See above for output.

note: This is an issue with the package mentioned above, not pip.
hint: See above for details.
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
from sklearn.preprocessing import StandardScaler

# Generate synthetic data for clustering (3 clusters with some noise)
X, y = make_blobs(n_samples=300, centers=3, cluster_std=1.0, random_state=42)

# Standardize the data (important for many clustering algorithms)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Apply K-Means clustering
kmeans = KMeans(n_clusters=3, random_state=42)
y_kmeans = kmeans.fit_predict(X_scaled)
```


+ Code
+ Text
RAM
Disk
Colab AI
K-Means Clustering
Cluster Centers
5
0
5
0
5
0
5