PART 1: SENTIMENT ANALYSIS USING TEXTBLOB

Objective (What are we trying to learn?)

The goal is to:

- Convert **unstructured text (reviews)** into **structured sentiment labels**
- Understand **how people feel overall** about a product/service
- Learn how **TextBlob** performs sentiment analysis

In real-world data science:

- Reviews, tweets, comments, news = **text**
- Models cannot directly use text → must convert to **numerical or categorical features**

Dataset Description

The given dataset contains:

- A column with **text reviews** (for example: review)

Each row = **one review**

What is TextBlob doing internally?

TextBlob uses:

- A **lexicon-based approach**
- Each word has an associated sentiment score
- Overall sentiment = average polarity of words

Polarity range:

Polarity Value	Meaning
+1	Extremely positive
0	Neutral
-1	Extremely negative

Code Explanation (Part 1)

Import libraries

```
import pandas as pd
from textblob import TextBlob
import matplotlib.pyplot as plt
```

Why?

- pandas → data handling
- TextBlob → sentiment analysis
- matplotlib → visualization

Load dataset

```
df = pd.read_csv("Reviews.csv")
```

Reads the CSV into a DataFrame where:

- Rows = reviews
- Columns = attributes (review text, etc.)

Sentiment function

```
def get_sentiment(text):
    polarity = TextBlob(str(text)).sentiment.polarity
    if polarity > 0:
        return "Positive"
    elif polarity < 0:
        return "Negative"
    else:
        return "Neutral"
```

Step-by-step logic:

1. Convert text to string (handles missing values)
2. Extract **polarity**
3. Apply rule-based classification:
    - > 0 → Positive
    - < 0 → Negative
    - = 0 → Neutral

This satisfies the **assignment requirement** of classifying into 3 categories.

Apply sentiment analysis

```
df["Sentiment"] = df["review"].apply(get_sentiment)
```

What happens here?

- Function runs on **every review**
- Output stored in a **new column**
- Dataset now contains sentiment labels

First deliverable achieved

Sentiment distribution

```
sentiment_counts = df["Sentiment"].value_counts()
sentiment_percent = df["Sentiment"].value_counts(normalize=True) * 100
```

Why this step?

- To understand **overall opinion**
- Normalization converts counts → percentages

Visualization

```
sentiment_counts.plot(kind="bar")
```

Interpretation:

- Taller bar = more reviews in that category
- Helps visually understand sentiment dominance

Brief Analysis (What you should conclude)

Typical observations:

- Majority reviews are **positive**
- Negative reviews identify dissatisfaction
- Neutral reviews are informational or emotionally balanced

Key learning:
Sentiment analysis helps summarize large volumes of text **quickly and objectively**.

PART 2: STOCK TREND PREDICTION USING SENTIMENT + TIME SERIES

Objective (Why this part exists)

Here we **combine two data science domains**:

1. **Natural Language Processing** (news sentiment)
2. **Time Series Forecasting** (stock prices)

Real-world motivation:
Markets react not only to past prices, but also to **news and emotions**

Dataset Preparation

We need **two datasets**:

Stock Prices

- Daily Close price
- Date-based time series

Source:

- Yahoo Finance

News Headlines

- One or more headlines per day
- Each headline influences market sentiment

Source:

- Kaggle news dataset

Sentiment Annotation (Part 2 reuse)

Apply same sentiment logic

```
news["Sentiment"] = news["Headline"].apply(get_sentiment)
```

Consistency is important:

- Same sentiment logic as Part 1
- Ensures comparable interpretation

Convert labels → numeric

```
sentiment_map = {"Positive": 1, "Neutral": 0, "Negative": -1}
news["Sentiment_Score"] = news["Sentiment"].map(sentiment_map)
```

Why numeric?

- Neural networks **cannot process text**
- Numbers allow mathematical learning

Align News with Stock Prices

Daily aggregation

```
daily_sentiment = news.groupby("Date")["Sentiment_Score"].mean()
```

Reason:

- Multiple headlines per day
- We compute **average sentiment** for that day

Merge with stock prices

```
data = pd.merge(prices, daily_sentiment, on="Date", how="left")
data["Sentiment_Score"].fillna(0, inplace=True)
```

Important rule from assignment:

If news is missing → treat as **neutral sentiment**

This prevents data loss and bias.

Feature Scaling

```
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data[["Close", "Sentiment_Score"]])
```

Why scaling is mandatory for LSTM?

- LSTM uses gradient descent
- Different scales cause unstable training
- Scaling ensures faster and stable convergence

Time Series Sequence Creation

```
def create_sequences(data, time_steps=60):
```

What this does conceptually:

- Uses last **60 days** of data to predict **next day**
- Preserves temporal order

This is **how LSTM learns patterns over time**

LSTM Model Explanation

Why LSTM?

- Traditional models fail with long dependencies
- LSTM remembers **long-term trends**
- Ideal for financial time series

Model structure

LSTM → LSTM → Dense

- First LSTM learns short-term patterns
- Second LSTM refines long-term dependencies
- Dense outputs predicted price

Model Training

```
model.fit(X_train, y_train)
```

Model learns:

- How **price history**
- AND **sentiment**
  together influence future price

Evaluation & Visualization

```
plt.plot(y_test)
plt.plot(predictions)
```

Interpretation:

- Closer lines = better prediction

- Sentiment-enhanced model usually:
    - Reacts faster
    - Is smoother
    - Captures turning points better

Insights & Observations

Key findings:

- Positive sentiment often **precedes price increase**
- Negative sentiment correlates with volatility
- Sentiment acts as a **leading indicator**
- Price-only models miss emotional context

OPTIONAL EXTENSION: MULTI-STOCK ANALYSIS

If repeated for multiple stocks:

- Tech stocks → highly sentiment sensitive
- Defensive stocks → less sentiment impact
- Sentiment does **not generalize uniformly**

COMPLETE LEARNINGS FROM WEEK 3 (WiDS)

---

1. Understanding the Nature of Real-World Data

One of the primary learnings from Week 3 is recognizing that real-world data is rarely clean, structured, or numerical. Data often comes from multiple sources and in different formats. In this assignment, textual data such as reviews and news headlines had to be combined with numerical stock price data.

This highlights that data science is not only about applying models but about making heterogeneous data sources compatible and meaningful. A significant portion of the work involves cleaning, preprocessing, and aligning datasets before modeling can even begin.

---

2. Conceptual Understanding of Sentiment Analysis

Sentiment analysis is not simply a task of labeling text as positive or negative. It is a method of quantifying human opinions and emotions expressed through language. Through this assignment, sentiment analysis was used to transform subjective textual data into structured information that can be analyzed statistically or used as model input.

The assignment reinforced that sentiment analysis is probabilistic in nature and may not always be perfectly accurate, but it is still highly valuable for capturing general emotional trends in large datasets.

---

3. Learning How TextBlob Performs Sentiment Analysis

TextBlob uses a rule-based, lexicon-driven approach to sentiment analysis. Each word is associated with a sentiment polarity score, and the overall sentiment of a sentence or document is computed by aggregating these scores.

This introduced the idea that effective natural language processing does not always require complex deep learning models. Simpler, interpretable approaches can be sufficient for many applications, especially when transparency and ease of implementation are important.

---

4. Interpreting Sentiment Distribution

Beyond assigning sentiment labels, the assignment emphasized analyzing the distribution of sentiments across the dataset. Calculating percentages of positive, negative, and neutral entries helped in understanding the overall public perception reflected in the data.

This step highlighted the importance of summarizing and interpreting results rather than focusing solely on individual predictions. Aggregate sentiment trends often provide more actionable insights than isolated data points.

---

5. Understanding Time Series Data and Temporal Dependency

A critical learning from the stock prediction task was understanding that time series data has inherent temporal dependencies. Stock prices are not independent observations; past prices influence future prices.

This required a shift in thinking from traditional row-wise data analysis to sequence-based modeling, where the order of observations is central to the learning process.

---

6. Importance of LSTM in Time Series Forecasting

Long Short-Term Memory (LSTM) networks were used to model stock prices because of their ability to retain information over long sequences. Unlike traditional models, LSTMs can learn long-term dependencies and patterns in sequential data.

This helped in understanding why deep learning models are particularly effective for financial time series, where trends, momentum, and historical context play a crucial role.

---

7. Integrating Sentiment Data with Numerical Features

One of the most important technical learnings was feature engineering through integration of sentiment data with stock price data. Textual sentiment was converted into numerical scores and aligned with daily stock prices.

This demonstrated how qualitative information can be transformed into quantitative signals and incorporated into predictive models. The integration of sentiment data allowed the model to capture emotional and psychological factors that influence market behavior.

---

8. Handling Missing and Imperfect Data

The assignment involved dealing with missing news data for certain dates. Instead of discarding such data, a neutral sentiment assumption was applied, as specified in the assignment instructions.

This reinforced the understanding that real-world data science requires making justified assumptions. Handling missing data thoughtfully is often preferable to removing large portions of data, which can lead to biased models.

---

9. Model Comparison and Evaluation

The task required comparing a sentiment-enhanced LSTM model with a price-only model developed earlier. This comparison emphasized the importance of evaluating models relative to baselines rather than in isolation.

The sentiment-enhanced model showed improved responsiveness to market changes, highlighting the value of incorporating additional contextual information into predictive systems.

---

10. Role of Visualization in Model Interpretation

Visualization played a key role in understanding model behavior. Plots comparing actual and predicted stock prices helped assess model performance, while sentiment versus price plots revealed relationships between emotional trends and market movements.

This reinforced that visual analysis is a critical tool for interpretation, validation, and communication of results.

---

11. Understanding Model Limitations

The assignment also highlighted that sentiment signals are inherently noisy and do not affect all stocks uniformly. Market reactions depend on sector, company fundamentals, and broader economic conditions.

Recognizing these limitations is essential for responsible data science practice and prevents overconfidence in model predictions.

---

12. Broader Skills and Professional Learning

Beyond technical concepts, Week 3 strengthened several core data science skills:

- **Structured problem-solving**
- **Feature engineering**
- **Assumption justification**
- **Model comparison**
- **Clear documentation and explanation of results**

These skills are fundamental for real-world data science projects and professional practice.