A Time Series Forecast of Stock Prices Using Twitter Sentiment and Financial Data

Ronan Downes SBA22447

September 14, 2024

Contents

1 Data Collection Processing

1.1 Data Loading

To begin the analysis, we load the datasets provided in the ZIP file. This includes tweet data and stock price data for multiple companies, including Apple (AAPL), Amazon (AMZN), Google (GOOG), Microsoft (MSFT), and Tesla (TSLA). The tweet data captures the sentiment expressed in tweets, while the stock price data provides daily stock prices for each company.

1.2 Loading Stock Price Data

The next step involved loading the stock price data for the selected companies: Apple (AAPL), Amazon (AMZN), Google (GOOG), Microsoft (MSFT), and Tesla (TSLA). This data spans the period from January 2020 to December 2020 and contains information on daily stock prices, including fields such as date, open, high, low, close, adjusted close, and trading volume.

The data for each company was stored in separate CSV files, which were imported using Python's `pandas` library.

1.3 Date Conversion

To ensure consistency in our time series analysis, we converted the date columns in both the tweet and stock price datasets to `datetime` format. This step is crucial as it allows us to properly align the data during analysis. Invalid date formats were handled using the `errors='coerce'` option, converting any problematic entries into `NaT` (Not a Time) values for easier identification and cleaning.

1.4 Sentiment Analysis

Sentiment analysis was performed on the tweet data using the `TextBlob` library. Each tweet's text was processed to calculate a sentiment polarity score, which ranges from -1 (negative) to 1 (positive). This score helps identify the overall sentiment of the tweets related to the selected stocks.

The sentiment scores were then aggregated by date to align with the daily stock price data. This aggregation allowed us to assess the average sentiment for each day, providing a temporal dimension that can be used in the forecasting analysis.

Merging Sentiment Data with Stock Prices

In this step, we merge the daily sentiment data with the stock price data for each selected company (AAPL, AMZN, GOOG, MSFT, TSLA). The sentiment data is joined on the date, allowing us to analyze how daily sentiment might correlate with stock price changes. After merging, we drop redundant columns to clean up the data.

2 Big Data Processing Environment Setup

Three virtual machines (VMs) were set up using a pre-configured Ubuntu image from OSBoxes to streamline the installation process. These VMs were cloned to serve distinct roles: `Hadoop_Spark`, `SQL`, and `NoSQL`.

2.1 Cloning Virtual Machines for Different Environments

The initial Ubuntu VM was cloned into three separate environments, ensuring consistency across all setups.

Steps to Clone:

- **Initial Setup:** A pre-configured Ubuntu VM from OSBoxes was used, which featured pre-installed Guest Additions for seamless copy-paste functionality.
- **Shutdown Before Cloning:** The VM was shut down to ensure a clean state before cloning.
- **Cloning:** In VirtualBox, full clones were created for each VM, named `SQL`, `NoSQL`, and `Hadoop_Spark`, enabling independent operation.

2.2 VM Resources Allocation

Each VM was allocated 100GB of storage, 2 processors, and 128MB of VRAM, tailored to meet their specific requirements.

2.3 Configuring the Virtual Machines

Each VM was configured according to its intended role. The detailed configuration steps, including system updates, software installations, and troubleshooting processes, are covered in the appendix.

Hadoop_Spark VM Configuration

The configuration involved setting up Java, Hadoop, and Spark, resolving environment variable issues, and recloning to ensure a clean setup. A detailed breakdown of these steps is provided in the appendix.

SQL VM Configuration

The SQL VM was set up with MySQL, including secure installation and database testing. The exact commands and configurations are documented in the appendix.

NoSQL VM Configuration

For the NoSQL VM, MongoDB was installed and configured, with service testing conducted to ensure proper functionality. Detailed instructions are in the appendix.

3 Exploratory Data Analysis and Sentiment Extraction

4 Time Series Forecasting Techniques

5 Comparative Analysis of SQL and NoSQL Databases

6 Dynamic Dashboard Design and Implementation

References

A Appendix Introduction

B Appendix Data Collection Processing

B.1 Loading Tweet Dataset

The following Python code snippet demonstrates how the tweet dataset was loaded into the environment:

```
import pandas as pd
import os
import zipfile
import warnings

warnings.filterwarnings('ignore')

# Unzip the file directly in the current directory
with zipfile.ZipFile('stock-tweet-and-price.zip', 'r') as zip_ref:
    zip_ref.extractall()

# Load the tweet dataset
tweets_df = pd.read_csv('stock-tweet-and-price/stocktweet/stocktweet.cs

# List all unique tickers in the 'ticker' column
unique_tickers = tweets_df['ticker'].unique()
print(f"Unique tickers in the dataset: {unique_tickers}")

# Display the first few rows of the tweet dataset
tweets_df.head()
```

B.2 Loading Stock Price Data

The following Python code snippet demonstrates how the stock price data for the selected companies was loaded:

```
# Choose companies
companies = ['AAPL', 'AMZN', 'GOOG', 'MSFT', 'TSLA']
stock_data = {}
```

```
# Load stock data for each company
for company in companies:
    stock_data[company] = pd.read_csv(f'stock-tweet-and-price/stockprice

# Display the first few rows of one of the company's stock data
stock_data['AAPL'].head()
```

B.3 Date Conversion

The following Python code snippet demonstrates how the date columns in the tweet and stock price datasets were converted to `datetime` format:

```
# Convert tweet dates to datetime and handle any invalid date formats
tweets_df['date'] = pd.to_datetime(tweets_df['date'], format='%d/%m/%Y',

# Convert stock prices 'Date' columns to datetime and handle invalid for
for company in companies:
    stock_data[company]['Date'] = pd.to_datetime(stock_data[company]['Da
```

B.4 Sentiment Analysis

The following Python code snippet demonstrates how sentiment analysis was performed on the tweets using the `TextBlob` library:

```
from textblob import TextBlob

# Apply sentiment analysis
tweets_df['sentiment'] = tweets_df['tweet'].apply(lambda tweet: TextBlob

# Aggregate sentiment by date
daily_sentiment = tweets_df.groupby('date')['sentiment'].mean().reset_in

# Display the first few rows of the aggregated sentiment data
daily_sentiment.head()
```

B.5 Merging Sentiment and Stock Price Data

The following Python code snippet demonstrates how we merged the daily sentiment data with the stock price data for each selected company:

```
# Merge sentiment data with stock price data for each selected company
for company in companies:
    df = stock_data[company]
    df = df.merge(daily_sentiment, left_on='Date', right_on='date', how=
    df.drop(columns=['date'], inplace=True)
    stock_data[company] = df

# Display the first few rows of the merged data for one company (e.g., A
stock_data['AAPL'].head()
```

C Appendix Big Data Processing Environment Setup

This appendix provides a detailed account of the setup for the Big Data Processing Environment, including the technical configurations, problem-solving steps, and the decision-making process involved in configuring the `Hadoop_Spark`, `SQL`, and `NoSQL` virtual machines.

C.1 Initial Challenges and Problem Solving

I needed copy-paste interaction between my host windos OS and my Ubuntu Linnux VMs. The isolation in a virtual machine (VM) environment is designed for security and separation of environments, which meant clipboard sharing (copy-paste) is omitted on purpose. This college project did not have security treats and I wanted to copy-paste from the Big Data Storage & Processing module of CCT MSc in Data Analytics so I needed to change this behaviour.

Initially, I attempted to enable clipboard sharing through the VirtualBox settings, setting the clipboard mode to "Bidirectional." However, this did not resolve the issue. Next, I attempted to insert the VirtualBox Guest Additions CD, a common solution that installs drivers to facilitate better interaction with the host system. Even after installing Guest Additions and restarting the VM, copy-paste functionality remained disabled, leading to further troubleshooting. Valuable time was being waisted so I needed to think outside the box!

C.2 Switching to OSBoxes.org for a Pre-Configured Image

Given these challenges, I decided to try a different approach. I discovered pre-configured Ubuntu images on OSBoxes.org, which promised a more interactive environment with functionalities like clipboard sharing already enabled. Using OSBoxes significantly streamlined the process and provided the flexibility I needed. This allowed me to seamlessly apply the content and tools from my Big Data Storage & Processing module, aligning the VM environment with my project requirements.

Key Benefits of OSBoxes.org Approach:

- The pre-built image came with Guest Additions pre-installed, fixing the clipboard sharing issue.
- The setup was faster and avoided repetitive configurations that had previously consumed time and mental effort.
- Enabled a smoother workflow for testing various configurations, directly aiding the data processing work central to my project.

C.3 Hadoop_Spark VM Configuration

I first configred the `Hadoop_Spark` VM. The setup involved the following steps:

System Update: To begin, I updated the package list and installed the latest system updates:

```
sudo apt update
sudo apt upgrade -y
```

Java Installation: Hadoop and Spark require Java, so I installed OpenJDK 11:

```
sudo apt install openjdk-11-jdk -y
java -version
```

Verification of the Java version was crucial to avoid potential errors in subsequent steps.

Setting Environment Variables: While editing the `hadoop-env.sh` file, I encountered problems with incorrectly set environment variables, particularly with `JAVA_HOME`. Several attempts to modify the file manually led to errors in executing Hadoop commands. Recognizing that too many changes had introduced errors, I decided to reclone the VM from the OSBoxes image to start fresh.

Recloning the VM: This step reinforced the importance of a clean slate when troubleshooting complex configurations. By recloning, I could isolate the root issue and proceed with confidence. I carefully re-edited the `hadoop-env.sh` file:

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
```

Applying the changes using:

```
source /usr/local/hadoop/etc/hadoop/hadoop-env.sh
```

This resolved the environment variable issue, allowing me to proceed with the Hadoop setup.

C.4 SQL VM Configuration

The configuration of the SQL VM was relatively smooth, as I had already learned from earlier mistakes. The setup included:

Install MySQL: Installation was performed using the package manager:

```
sudo apt install mysql-server -y
```

Securing the Installation: Running `mysql_secure_installation` highlighted the importance of securing the database even in a testing environment. Although some security measures were simplified for this project, the process provided hands-on experience with database hardening techniques.

C.5 NoSQL VM Configuration

For the NoSQL environment, I chose MongoDB. This decision was informed by the need to work with a flexible data model as part of my project objectives.

Install MongoDB: The installation was completed using:

```
sudo apt update
sudo apt install -y mongodb
```

Enabling MongoDB Service: Starting and enabling MongoDB to run on boot was a key step:

```
sudo systemctl start mongodb
sudo systemctl enable mongodb
```

Testing: Basic commands in the MongoDB shell were run to ensure the setup was successful.

C.6 Reflection on the Process

This appendix reflects on the problem-solving mindset required to navigate through setup challenges, from addressing virtual machine isolation to configuring complex environments. The decision to switch to a pre-configured OSBoxes VM not only streamlined the process but also exemplified the need to adapt strategies to suit project requirements. By overcoming these technical hurdles, I was able to focus more effectively on applying the concepts from my MSc course.