Analysis Report

‘ JSWSTEEL Stock Analysis and Trend Forecasting Using PySpark’

Chapter 1: Introduction

Stock market analysis plays a crucial role in understanding the financial health and trends of companies. With the increasing volume of trading data, it has become essential to adopt scalable tools to process and analyze this data effectively. This project aims to leverage the power of PySpark to perform large-scale analysis on historical stock data of JSW Steel Ltd. (JSWSTEEL), one of India's leading steel manufacturing companies.

The main objective of this study is to clean and preprocess raw stock data, engineer meaningful features like moving averages and daily returns, and extract valuable insights through exploratory data analysis and visualization. By detecting patterns such as moving average crossovers and measuring volatility, this project provides actionable insights for traders, investors, and financial analysts. The approach also demonstrates how big data frameworks like Apache Spark can be used to perform real-time, scalable analytics on financial datasets.

The stock market is a dynamic environment influenced by economic conditions, investor sentiment, and company performance. In this project, we conduct a comprehensive analysis of JSWSTEEL stock using PySpark. The goal is to identify patterns, trends, and technical signals that assist in informed investment decision-making.

Chapter 2: Technology Stack

In this project, a combination of modern big data tools and Python-based libraries was used to ensure efficient, scalable, and insightful stock market analysis. The technologies selected not only allowed handling of large datasets but also supported the generation of advanced visualizations and statistical computations.

2.1 PySpark

Apache Spark is a distributed computing framework ideal for processing large-scale datasets. PySpark is its Python API. We used PySpark DataFrames for:

- Reading and preprocessing CSV files
- Performing transformations and aggregations
- Applying window functions for rolling statistics
- Running SQL queries on structured data

2.2 Spark SQL

Spark SQL allows querying structured data using SQL-like syntax. This was used to:

- Perform correlation analysis
- Run aggregations for monthly and daily summaries
- Identify outliers based on return thresholds

2.3 Window Functions

Window functions enable operations across a window of rows without collapsing them. These were used to:

- Calculate previous day's close (lag)
- Compute rolling averages (SMA_50 and SMA_200)
- Measure 30-day rolling volatility

2.4 Pandas and Matplotlib

Although PySpark handles data processing, visualization was carried out using Pandas and Matplotlib by converting Spark DataFrames to Pandas. This enabled:

- Line charts for price trends and moving averages
- Histograms for return distribution
- Scatter plots for volume-price relationships

2.5 Jupyter Notebook

All code, outputs, visualizations, and documentation were integrated in Jupyter Notebook. It provided an interactive environment for:

- Writing and executing code step-by-step
- Capturing results and graphs inline
- Presenting the entire project in a clean, readable format

By combining these technologies, we achieved a seamless flow from data ingestion to insight generation, enabling comprehensive technical analysis of JSWSTEEL stock performance.

- **PySpark**: Scalable big data processing
- **Spark SQL**: Analytical queries
- **Window Functions**: Feature engineering over time-series data
- **Pandas & Matplotlib**: Visualization and plotting
- **Jupyter Notebook**: Development and documentation

Chapter 3: Dataset

The dataset used for this analysis comprises historical stock market data for **JSW Steel Ltd. (NSE: JSWSTEEL)**. The data was obtained in CSV format and includes daily records of the stock's trading performance.

3.1 Dataset Source

- **File Name**: JSWSTEEL - JSWSTEEL.csv
- **Data Format**: Comma-Separated Values (CSV)
- **Frequency**: Daily stock prices
- **Coverage**: Several years of historical trading data

3.2 Raw Columns

The original dataset included the following columns:

- **Price:** (Incorrect header, later renamed to Date)
- **Open:** Opening price of the stock
- **High:** Highest price during the trading day
- **Low:** Lowest price during the trading day
- **Close:** Closing price of the stock
- **Adj Close:** Adjusted closing price for corporate actions
- **Volume**: Total number of shares traded on that day

3.3 Initial Challenges

Upon loading the dataset into PySpark, the following issues were observed:

- **Data Type Mismatches**: All numerical columns (Open, High, Low, Close, Volume) were inferred as strings.
- **Date Format Issues**: The Price column had full datetime strings with timezone info (e.g., 2003-05-08 00:00:00+00:00), which caused parsing failures.
- **Missing Values**: Some rows had null values in essential columns like Close and Volume.
- **Extra Rows**: Header rows or misaligned data rows were present at the top (e.g., rows with all NULLs or containing metadata like JSWSTEEL.NS).

3.4 Cleaning & Transformation Summary

To make the dataset usable for analysis, the following transformations were applied:

- Renamed Price → Date
- Converted all numerical columns to DoubleType
- Removed null and invalid rows

- Extracted the date from the datetime string using substring()
- Parsed Date column using the to_date() function

3.5 Final DataFrame Schema

After preprocessing, the final DataFrame schema included:

- Date (DateType)
- Open, High, Low, Close, Adj Close, Volume (DoubleType)

The cleaned dataset became the foundation for all further analysis, including trend detection, volatility analysis, and technical indicator computation.

- **Dataset**: JSWSTEEL historical stock prices in CSV format
- **Features**:
    - Date (renamed from 'Price')
    - Open, High, Low, Close (prices)
    - Volume
    - Adjusted Close
- **Initial Issues**:
    - Data type inconsistencies (all string)
    - Missing/null values in key columns

Chapter 4: Methodology

This chapter outlines the comprehensive approach taken to analyze the JSWSTEEL stock dataset using PySpark. The methodology was structured in sequential stages, starting from raw data ingestion to generating meaningful financial indicators and uncovering patterns through analysis.

4.1 Data Cleaning and Preparation

The raw dataset initially had inconsistencies like misnamed columns, incorrect data types, and missing values. The following preprocessing steps were applied:

- **Renaming Columns**: The column labelled "Price" was actually the timestamp and was renamed to "Date".
- **Date Formatting**: Extracted the date portion from full timestamps using substring() and converted to Spark DateType using to_date().
- **Type Casting**: Columns like Open, High, Low, Close, and Volume were originally in string format and were cast to DoubleType.
- **Null Handling**: Rows with missing essential values in Close or Date were dropped to maintain integrity during calculations.
- **Chronological Ordering**: The data was sorted by date to support time-dependent operations like moving averages and lag functions.

4.2 Feature Engineering

Once the dataset was cleaned, we engineered new features to enhance the analytical capabilities:

- **Prev_Close**: Used PySpark's lag() window function to capture the previous day’s closing price.
- **Daily_Return**: Computed as the percentage change from Prev_Close to Close, using the formula:
- **SMA_50 and SMA_200**: 50-day and 200-day Simple Moving Averages, respectively. These were calculated using window functions to smooth price fluctuations and identify trends.
- **Volatility_30**: A 30-day rolling standard deviation of Daily_Return, used to measure short-term risk and variability.

4.3 Trend Detection

To classify market phases into bullish or bearish periods, we used a crossover strategy:

- If SMA_50 > SMA_200 ⇒ the market is in an **Uptrend**.
- Otherwise ⇒ the market is in a **Downtrend**.

This logic was implemented using the when() function in PySpark and added as a new column called Trend. These trends are crucial for traders who follow technical indicators.

4.4 Correlation Analysis

Correlation analysis was done using Spark SQL's corr() function to measure the relationships between numerical fields:

- **Strong Correlation**: Between Close and Open, High, Low
- **Weak Correlation**: Between Close and Volume

This analysis helped us understand how tightly different features move together, and how independent Volume is from actual price changes.

4.5 Exploratory Data Analysis (EDA)

To further understand the behavior of the stock, EDA was conducted with the following:

- **Top 10 High Volume Days**: To identify sessions of unusually high trading activity.
- **Top 10 Most Volatile Days**: Based on Daily_Return, highlighting days with sharp price movement.
- **Monthly Analysis**: Aggregated average Close and Volume for each month to assess seasonal trends.

Together, these steps created a robust analytical pipeline that transformed raw stock data into structured insights suitable for visualization and strategic interpretation.

Chapter 5: Visualization and Analysis

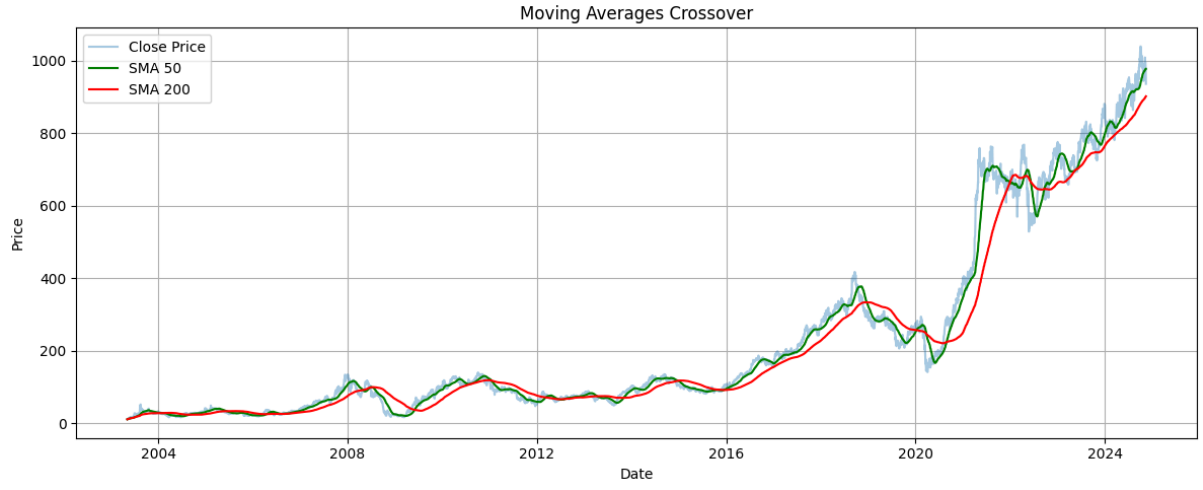
Data visualization is an essential part of stock analysis as it allows for intuitive understanding of trends, patterns, and anomalies. In this project, we converted PySpark DataFrames into Pandas DataFrames and used Matplotlib to plot various graphs that helped illustrate our findings.

5.1 Closing Price Over Time

A line plot of the Close price across the timeline provided a high-level view of how the stock has moved historically. This helped identify major bullish and bearish phases visually.

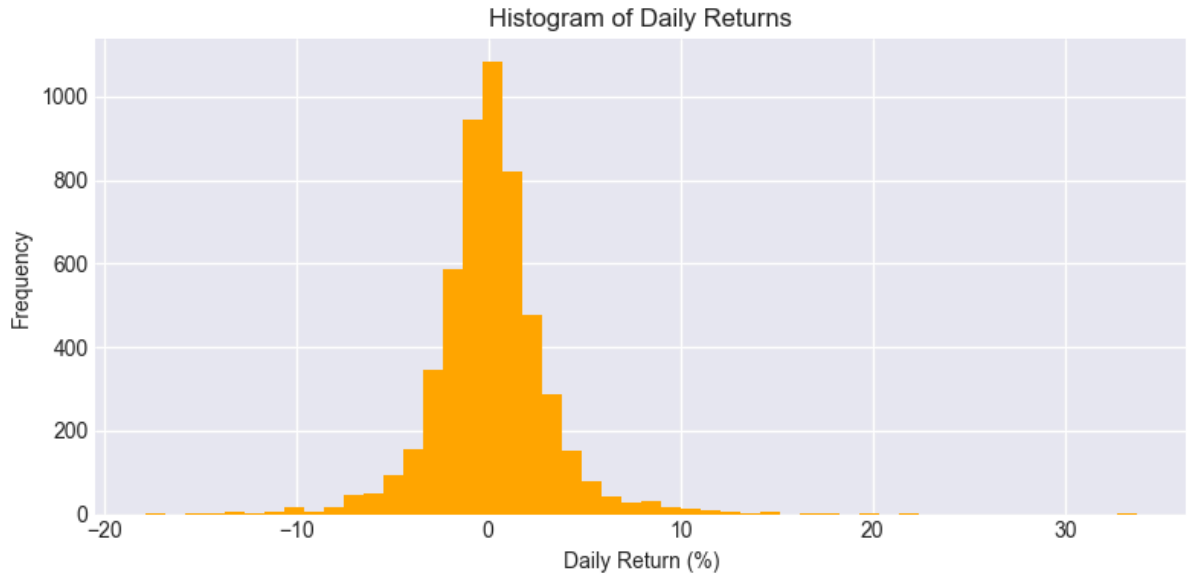
5.2 Moving Averages (SMA_50 & SMA_200)

We overlaid 50-day and 200-day simple moving averages over the price line to visualize bullish and bearish crossovers. These crossovers serve as potential buy/sell signals.



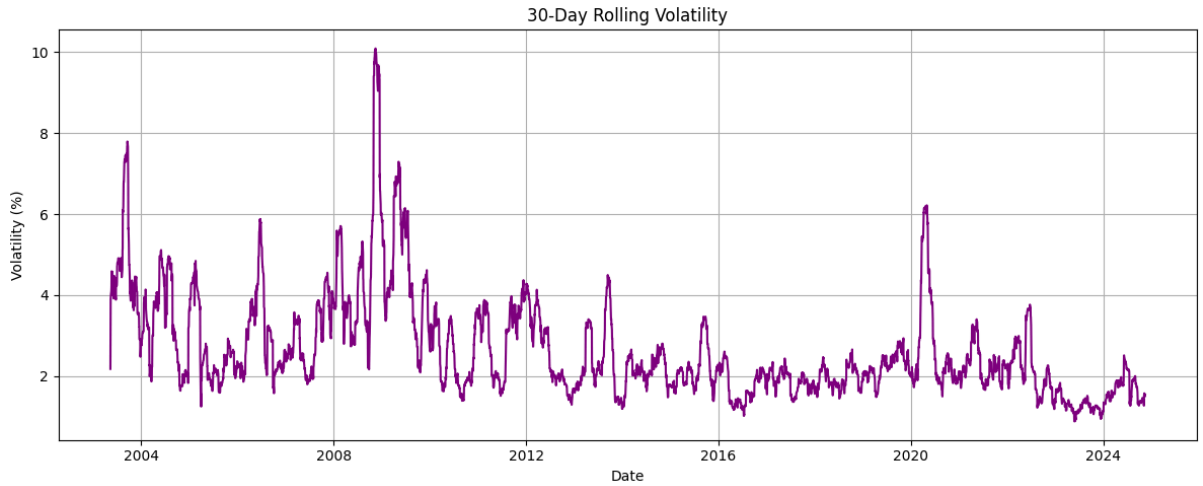
5.3 Daily Return Distribution

The histogram of Daily_Return values revealed the frequency of gains/losses. It also showed the distribution skewness and outliers.



5.4 Volatility Over Time

We tracked 30-day rolling volatility to observe risk changes. Higher volatility often indicated unstable or event-driven market phases.



Chapter 6: Key Insights

This chapter summarizes the most valuable findings from the analysis of JSWSTEEL stock data. Through extensive feature engineering, trend identification, correlation analysis, and visualizations, several patterns emerged that are highly relevant to investors, traders, and analysts.

6.1 Trend-Based Insights

- The comparison of short-term (SMA_50) and long-term (SMA_200) moving averages revealed clear bullish and bearish market phases.
- When SMA_50 consistently stayed above SMA_200, the stock tended to be in an upward trend (bullish).
- Conversely, SMA_50 crossing below SMA_200 often marked the beginning of a downward trend (bearish).

6.2 Volume Analysis

- Volume spikes were frequently observed on days with extreme daily returns.
- This suggests that large trades or news-driven events likely caused volatility and liquidity surges.
- However, on regular days, volume showed only a weak correlation with price, suggesting that other technical or fundamental indicators may be driving daily movement.

6.3 Volatility Insights

- Rolling 30-day volatility helped detect market phases that were highly uncertain or reactive.
- These volatility clusters typically occurred around inflection points in the price trend.
- Traders may consider reducing exposure during these periods if risk tolerance is low.

6.4 Return and Outlier Detection

- By computing Daily_Return, we observed that most price movements fall within a small percentage range.
- However, a few dates had outliers with return values exceeding ±3 standard deviations, which may signal earnings announcements, macroeconomic changes, or external shocks.

6.5 Correlation Summary

- Strong correlations were found between Close and Open, High, and Low, indicating internal consistency in the price data.
- Volume, however, had a very low correlation with these variables, reinforcing that volume alone is not a reliable predictor of price movement.

These insights formed the foundation of the strategic recommendations that follow and provide a data-driven understanding of the stock's historical behavior.

Insight Area	Observation
Trend	SMA crossovers help indicate buy/sell signals
Volume	Volume spikes often align with price movements
Volatility	Periodic risk clusters detected by rolling stddev
Outliers	Significant spikes found using 3-sigma rule
Correlation	Strong between Open/High/Low/Close; weak with Volume

Chapter 7: Strategic Recommendations

Based on the analysis conducted in the previous chapters, we can derive several practical recommendations tailored to different types of market participants. These recommendations are grounded in the observed trends, volatility patterns, and behavior of the JSWSTEEL stock over time.

7.1 For Long-Term Investors

- Focus on identifying **golden cross** patterns where the 50-day SMA crosses above the 200-day SMA. This is often a signal of a strong, sustained uptrend.
- Exit or reconsider positions when a **death cross** appears (SMA_50 drops below SMA_200), which may indicate a longer-term decline.
- Use moving averages to avoid short-term noise and focus on consistent trends.

7.2 For Short-Term Traders

- Watch for **spikes in volume and daily return**, as these often correspond to high-volatility events which can provide quick entry/exit opportunities.
- Track the **30-day rolling volatility**: trade more actively during periods of high volatility, but be cautious of risk.
- Use scatter plots and return histograms to assess typical price behaviors and detect when market movements deviate significantly from the norm.

7.3 For Risk-Averse Participants

- Avoid trading during periods with elevated rolling volatility, as identified through the 30-day standard deviation of daily returns.
- Stick to periods where moving averages show alignment and price movements are more stable.
- Use SMA trends to identify steady growth phases and avoid speculative or reactive trades.

7.4 For Financial Analysts and Data Scientists

- Incorporate the engineered features like SMA, volatility, and daily return into predictive or classification models.
- Use trend and volume insights to segment the data for more granular behavior analysis.
- This analysis can serve as the basis for building machine learning pipelines for trend forecasting or anomaly detection.

These strategic insights and techniques form the foundation of robust, data-driven decision-making frameworks. Whether for investment, trading, or advanced financial modeling, they enhance clarity and confidence in market participation.

- **Long-Term Investors**: Focus on golden/death crossovers (SMA trends)
- **Short-Term Traders**: Trade during volume and volatility spikes
- **Risk-Averse Users**: Avoid periods with high rolling volatility
- **Market Analysts**: Use combined indicators for pattern recognition

Chapter 8: Conclusion

The analysis of JSWSTEEL stock using PySpark demonstrates how big data tools can be effectively applied to financial datasets. Starting from raw historical stock data, we successfully implemented a full data pipeline involving cleaning, transformation, feature engineering, visualization, and insight generation.

Through the use of technical indicators such as Simple Moving Averages (SMA_50 and SMA_200), Daily Return, and Rolling Volatility, we were able to classify market trends, detect outliers, and observe volume-driven price behavior. By applying both statistical techniques and visual analysis, we uncovered patterns that have practical implications for investors, traders, and analysts.

Some of the key takeaways from this project include:

- **SMA crossovers** serve as effective indicators of long-term trend reversals.
- **High volatility periods** can be identified using rolling standard deviation of returns, helping in risk management.
- **Volume spikes** often align with market-moving events and can signal trading opportunities.
- **Correlations** between price fields help validate the consistency of financial data.

In conclusion, this project not only provided technical insights into JSWSTEEL's trading behavior but also laid a strong foundation for more advanced projects involving forecasting, machine learning, or real-time stock analysis using tools like Kafka and Spark Streaming.

The methodologies applied here can be generalized to other stocks or financial instruments, making it a valuable template for scalable, data-driven financial analysis.