Distributed Financial Risk Assessment
Group 7

Anirudh Garg (ag9563)

November 23, 2024

Abstract

This report documents the implementation of a Spark application to process, analyze, and clean stock price data stored in CSV files. The pipeline involves data profiling, statistics generation, and filling missing values, leveraging Zeppelin, Spark SQL, DataFrame transformations, and Hadoop's FileSystem API for efficient distributed processing.

1 Introduction

Stock price data, stored in CSV format, requires preprocessing and analysis for insights and downstream applications. The goal of this application is to:

- Profile data by extracting statistical information.
- Query data trends, such as the distribution of stock start dates.
- Perform data cleaning, filling missing values in the `Close` column using forward and backward filling techniques.

2 Data Acquisition and Access

- **Storage**: Data will be stored on distributed file system, HDFS, to accommodate storage needs and allow team-wide accessibility.
- **Permissions**: As these datasets are public, no additional permissions are required beyond accessing and extracting stocks related to each country from the google finance website.

- **Access Time**: Instant access; no approval needed for these publicly hosted datasets.

3 Dataset

I did the analysis on US Stocks. First, I downloaded tickers corresponding to US Stocks and then ran the `download-all-symbols.sh` file to download the data for the corresponding stocks. The stock price data was then moved to HDFS and further analysis was done using Zeppelin.

All the stocks have the following columns:

- **Date**
- **Open Price**
- **High**
- **Low**
- **Closing Price**
- **Volume**
- **Adjusted Closing Price**

4 Code Implementation

The application consists of three main components: reading and profiling stock files, querying distributions, and cleaning the data. Below is a detailed description of each section.

4.1 Reading and Listing Stock Files

The code uses the Hadoop FileSystem API to access stock files stored in HDFS. Files ending with `.csv` are filtered.

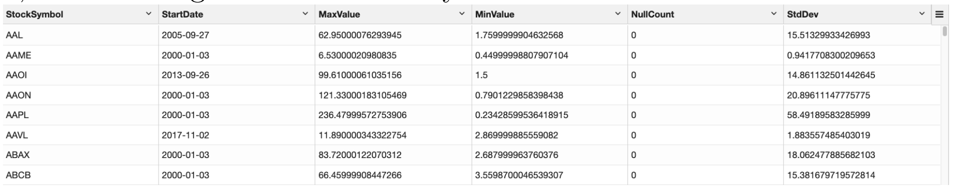
4.2 Profiling Stock Data

Each stock file is processed to extract:

- **Stock symbol:** Extracted from the file name.

- **Statistics:** Start date, minimum/maximum closing price, number of missing values, and standard deviation of closing prices.

The processing pipeline involves reading CSV files, filtering out metadata rows, and defining schema for clarity.



StockSymbol	StartDate	MaxValue	MinValue	NullCount	StdDev
AAL	2005-09-27	62.95000076293945	1.7599999904632568	0	15.51329933426993
AAME	2000-01-03	6.53000020980835	0.44999998807907104	0	0.9417708300209653
AAOI	2013-09-26	99.61000061035156	1.5	0	14.861132501442645
AAON	2000-01-03	121.33000183105469	0.7901229858398438	0	20.89611147775775
AAPL	2000-01-03	236.47999572753906	0.23428599536418915	0	58.49189583285999
AAVL	2017-11-02	11.890000343322754	2.869999885559082	0	1.883557485403019
ABAX	2000-01-03	83.72000122070312	2.687999963760376	0	18.062477885682103
ABCB	2000-01-03	66.45999908447266	3.5598700046539307	0	15.381679719572814

4.3 Querying Stock Data Distribution

The results are converted to a Spark DataFrame and SQL queries are run to analyze various metric. We analyze the stocks with the highest max values, lowest min values, and largest standard deviations. Also, we analyze the distribution of stock start dates over the years. This would help us in picking a start date to analyse stocks which have start dates before a certain cut off date.

4.4 Cleaning Stock Data

Missing values in the `Close` column are handled using forward filling (propagating the last non-null value) and backward filling (using the next non-null value).

5 Results and Discussion

- The profiling process extracted key statistics like start date, price ranges, and null counts.
- The SQL query showed trends in stock availability over years, useful for identifying market patterns and also the highest, lowest and highest std dev stock prices over the years
- The cleaning step ensured no missing values in the `Close` column, enabling downstream analysis without errors.

6 Conclusion

This application effectively processes and cleans stock price data using Apache Spark. The approach can scale to large datasets and supports extensibility for additional analyses.