Distributed Financial Risk Assessment

Nikhil Kommineni (Group 7)
nk3853@nyu.edu

November 23, 2024

1 Introduction

This report documents a data cleaning and profiling pipeline implemented using Apache Spark. The primary objective is to preprocess and analyze stock market data, remove inconsistencies, handle missing values, and identify outliers to prepare the data for downstream applications. The pipeline processes multiple files in a distributed environment, making it suitable for large-scale datasets.

2 Data Processing Workflow

2.1 Data Loading

The data is loaded from CSV files, with the first three rows skipped (metadata or headers). The schema is inferred automatically using Spark's `inferSchema` option. A custom schema is applied to improve column readability.

2.2 Data Cleaning

- **Null Value Handling:** Missing values in the `AdjClose` column are imputed using forward-fill and backward-fill operations via Spark window functions.
- **Outlier Detection:** Outliers are identified based on values exceeding three standard deviations from the mean and are filtered out.
- **Type Casting:** Columns such as `Date` and `AdjClose` are cast to appropriate types for analysis.

2.3 Multiple File Processing

The pipeline iterates over all files in a directory, applying the same cleaning logic to each. Processed files are saved to a designated output directory.

3 Code Implementation

Below is the implementation of the pipeline. For readability, the code is modularized and explained in corresponding sections.

3.1 Data Loading

```
val filePath = "/user/nk3853_nyu_edu/stocks/20MICRONS.NS.csv"
val rawDf = spark.read.option("header", false).option("
    inferSchema", "true").csv(filePath)
val filteredRDD = rawDf.rdd.zipWithIndex().filter { case (_, idx)
     => idx >= 3 }.map(x => x._1)
val filteredDf = spark.createDataFrame(filteredRDD, rawDf.schema)
val columnNames = Seq("Date", "AdjClose", "Close", "Open", "High"
    , "Low", "Volume")
val finalDf = filteredDf.toDF(columnNames: _*)
```

3.2 Null Value Handling

```
import org.apache.spark.sql.expressions.Window
val adjustedDf = formattedDf.withColumn("AdjClose", when($"
    AdjClose" === 0, lit(null)).otherwise($"AdjClose"))
val windowSpec = Window.orderBy("Date")
val filledDf = adjustedDf.withColumn(
    "AdjClose",
    coalesce(
        last($"AdjClose", ignoreNulls = true).over(windowSpec),
        first($"AdjClose", ignoreNulls = true).over(windowSpec)
    )
)
```

3.3 Outlier Detection

```
val stats = filledDf.agg(
  avg($"AdjClose").alias("mean"),
  stddev(col("AdjClose")).alias("std_dev")
).first()
val mean = stats.getAs[Double]("mean")
val std_dev = stats.getAs[Double]("std_dev")
val cleanedDf = filledDf.filter($"AdjClose" > mean + 3 * std_dev
    || $"AdjClose" < mean - 3 * std_dev)
```

3.4 Saving Cleaned Data

```
val outputPath = "/user/nk3853_nyu_edu/stocks/cleaned/20MICRONS.
    NS"
cleanedDf.coalesce(1).write.mode("overwrite").option("header", "
    true").csv(outputPath)
```

4 Results and Profiling

4.1 Descriptive Statistics

The cleaned dataset was analyzed to produce key statistics:

- **Null Count:** The number of null values before and after cleaning.
- **Minimum and Maximum Values:** The range of adjusted closing prices.
- **Standard Deviation:** A measure of the variability in stock prices.

4.2 Yearly Stock Distribution

Using SQL queries on the profiling results, we determined the yearly distribution of stock data from 2000 to 2024. The results can be visualized for further insights.

```
SELECT
    YEAR(CAST(StartDate AS DATE)) AS Year,
    COUNT(DISTINCT StockSymbol) AS StockCount
FROM value_distribution
WHERE YEAR(CAST(StartDate AS DATE)) BETWEEN 2000 AND 2024
GROUP BY Year
ORDER BY Year
```