Written Report

Data Analysis of Apple Stock Prices using PySpark and yfinance Library

Authors:

Meron Habtemichael
MD Islam

Introduction:

In this project, we analyze the stock price data of Apple Inc. using PySpark, a distributed computing framework for big data processing. The goal of this project is to demonstrate how PySpark can be used to handle large-scale financial data processing tasks and perform analysis on them. We use the yfinance library to fetch historical stock price data for Apple Inc., and preprocess the data to calculate rolling mean and standard deviation of the closing price over a rolling window of 200 days. We then analyze the resulting data to identify trends and volatility in the stock price.

Data Set and Preprocessing:

The data set used in this project is historical stock price data for Apple Inc., which is fetched from the yfinance library. The data set consists of the following columns: Date, Open, High, Low, Close, Adj Close, and Volume. But then again we reduced the columns to contain only Date, Open, and Close. We convert the pandas DataFrame to a PySpark DataFrame using the createDataFrame method of the SparkSession. We then calculate the rolling mean and standard deviation of the closing price over a rolling window of 200 days using the withColumn method and the Window function from the PySpark API. We drop null values from the resulting DataFrame using the na.drop method.

Working Mechanism/Principle:

The working mechanism of our project is as follows: We fetch historical stock price data from yfinance and convert it to a PySpark DataFrame. We preprocess the data by calculating rolling mean and standard deviation of the closing price over a rolling window of 200 days. We then analyze the resulting data to identify trends and volatility in the stock price. PySpark is used to perform these tasks, which allows us to handle large-scale data processing tasks in a distributed and parallel manner.

Strengths and Weaknesses/Limitations:

The strengths of our project include the ability to handle large-scale data using PySpark, which can significantly reduce processing time compared to traditional data processing methods. PySpark is designed to handle distributed and parallel processing, which allows it to process large volumes of data more efficiently. The weaknesses or limitations of our project include the need for specialized hardware and infrastructure to run PySpark, which may not be available to all users. Additionally, PySpark may require more expertise and resources to set up and configure compared to other data processing tools.

Results:

We visualize the results of our analysis by plotting the rolling mean and standard deviation of the closing price for Apple Inc. over time. The resulting plot shows trends and volatility in the stock price, which can be useful for making investment decisions. The plot shows that the stock price for Apple Inc. has generally been increasing over time, with occasional dips and spikes in volatility.

For visualization purposes, we converted the PySpark dataframe to Pandas dataframe. Note that converting a PySpark DataFrame to a Pandas DataFrame can be memory-intensive, and may not be suitable for very large data sets. It is generally recommended to use PySpark for data processing and analysis whenever possible, and to only convert to Pandas for visualization or other specific tasks that require a Pandas DataFrame.

Performance Evaluation:

Overall, our project demonstrates the ability of PySpark to handle large-scale data processing tasks, and shows how it can be used for analyzing stock price data. We have learned how to use PySpark for data processing and analysis, and how to work with financial data using the yfinance library. The performance of our analysis

depends on several factors, including the size and complexity of the data set, the hardware and infrastructure used to run PySpark, and the specific algorithms and techniques used for analysis.

References:

1. https://pypi.org/project/yfinance/