

### **Financial Risk Management with Apache Spark**

Design Component

CS Capstone Project  
Saint Louis University  
December 11, 2017

Kamran Madatov

Bao Thai

John Mitton

Under Guidance from:   
Dr. Goldwasser & Dr. Chambers

Mentor:

Dr. Ahn

A written document that describes a detailed design for achieving the formal requirements. A design document should include a description of the major components, their interfaces and how they interact to form the whole. Figures should be included for clarity, such as a UML diagram of the software design or an ER-diagram for a database. This document should also contain a discussion of any third-party technologies or software packages that will be used in meeting the project goals. Teams should demonstrate that they have already evaluated and familiarized themselves with any such technologies. Finally, this document must include a proposed timeline for the remainder of the project life cycle, making sure to include specific sub-goals for the development, implementation, and testing phases of the project.

1. **Project Description**

Value at Risk (VaR) has been widely adopted in the financial industry to measure risk. It is used for regulatory compliance, understanding the risk characteristics of large portfolios, and and making informed trading decisions. Three common methods of calculating Value at Risk are variance-covariance, historical simulation, and Monte Carlo simulation. Monte Carlo simulation can be more accurate than the simple models, but it requires more computational power. Fortunately, Apache Spark provides an easy way to scale statistical problems beyond what a single server can handle. Using Spark and historical stock data, we will calculate VaR of stocks with Monte Carlo Simulation in less time.

1. **Functional Component**

**Monte Carlo**

Process

1. Historical data on the desired stock is collected from online via a web crawler.
2. The data is read from the csv file and stored in appropriate lists while the simulation runs. E.g. opening, closing, date etc.
3. An expected value and standard deviation is calculated and used to generate the normal random variables.
4. Those variables are the increase or decrease for a day. 1000 of them are generated and the mean of them is the expected increase or decrease for the day.
5. Step for is repeated for the desired number of days.

Testing

* Collected 11 years worth of data but only used the first 10 years to calculate the expected value and standard deviation. Ran the test for 1 day to 365 days and compared it to the extra year’s values. Determined the average error for 1 week 1 month and 1 year. The stocks were AAPL, JNP, CNC, TSLA.

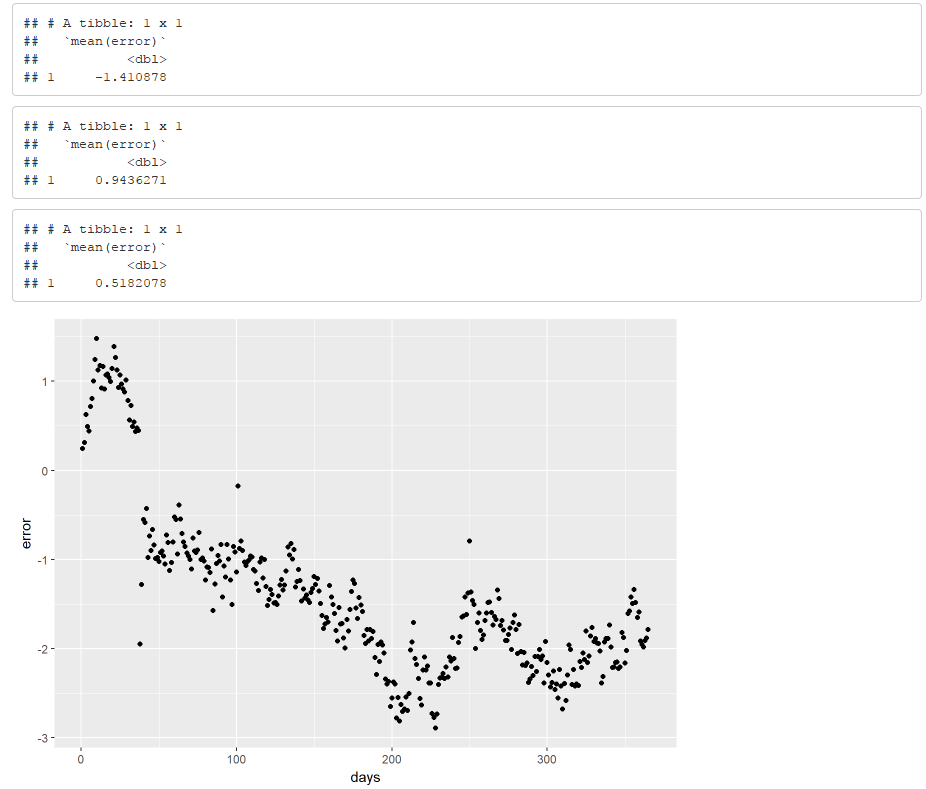
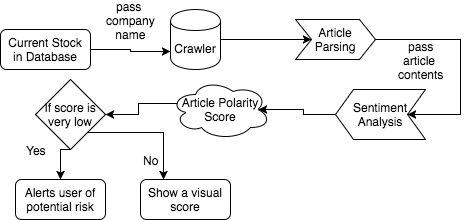


Figure X.X - Error for JNP for 1 year, 1 month, and 1 week and corresponding graph

* From the testing we concluded that the model was the most accurate up to 1 week and for less volatile stocks. So it was more accurate for CNC and JNP than AAPL and TSLA.

**Articles - Sentiment Analysis**

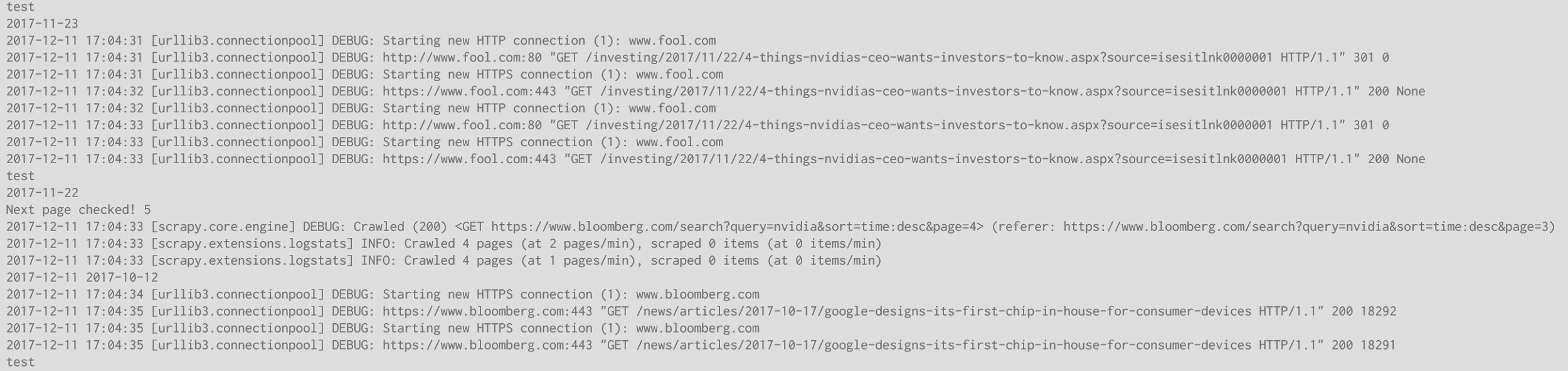
****

Input - the name of the company

* E.g. - “Apple”, “Microsoft”

Process

Figure X.X - Web crawler in progress

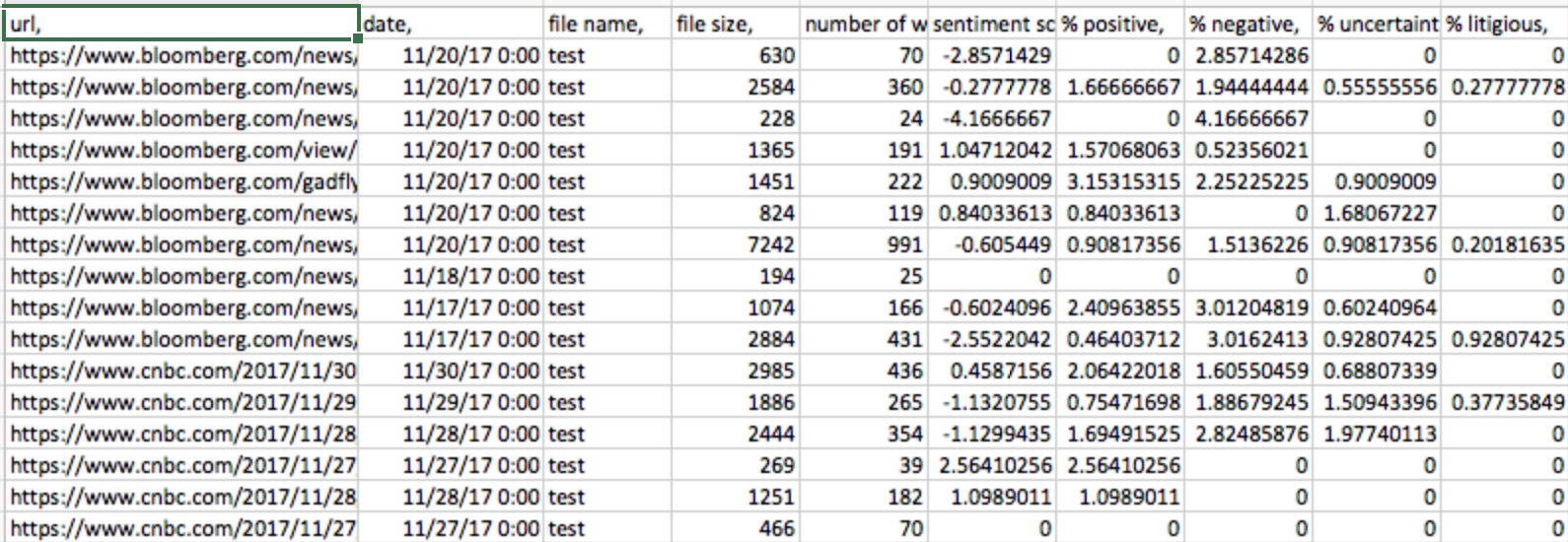


1. The input is first placed into **Scrapy (Crawler)**, with a designated parameters to crawl and a certain date restriction for a period of testing
2. Scrapy will run an automated web crawler and parse the url of the articles and pass to a **Newspaper3K(Article Parsing)** package that downloads and parse sections of the article
3. The body of the article is passed to **Loughran McDonald’s Dictionary(Sentiment Analysis)** for sentiment analysis for positive and negative scores
4. This score will measure with a standard deviation to determine risk
   1. Score average near 0 - low risk stock with steady rate
   2. Score average +-1 high risk stocks with either high returns or low returns

Output - Sentiment Score (will be processed later on for visualization on UI)

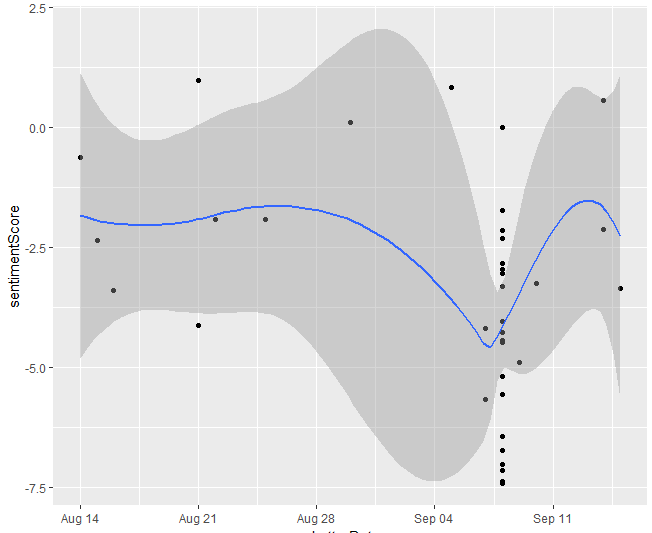
**Testing**

* An example output document from the webcrawler into article parsing, and after the scoring from sentiment analysis



* The data was plot by using the best fit trending line to determine the risks of the stocks
  + E.g. : Equifax

Figure X.X - sentiment scoring plot



* From this given risk trend line, we compare it to the actual pricing and performance

Figure X.X - actual pricing of score

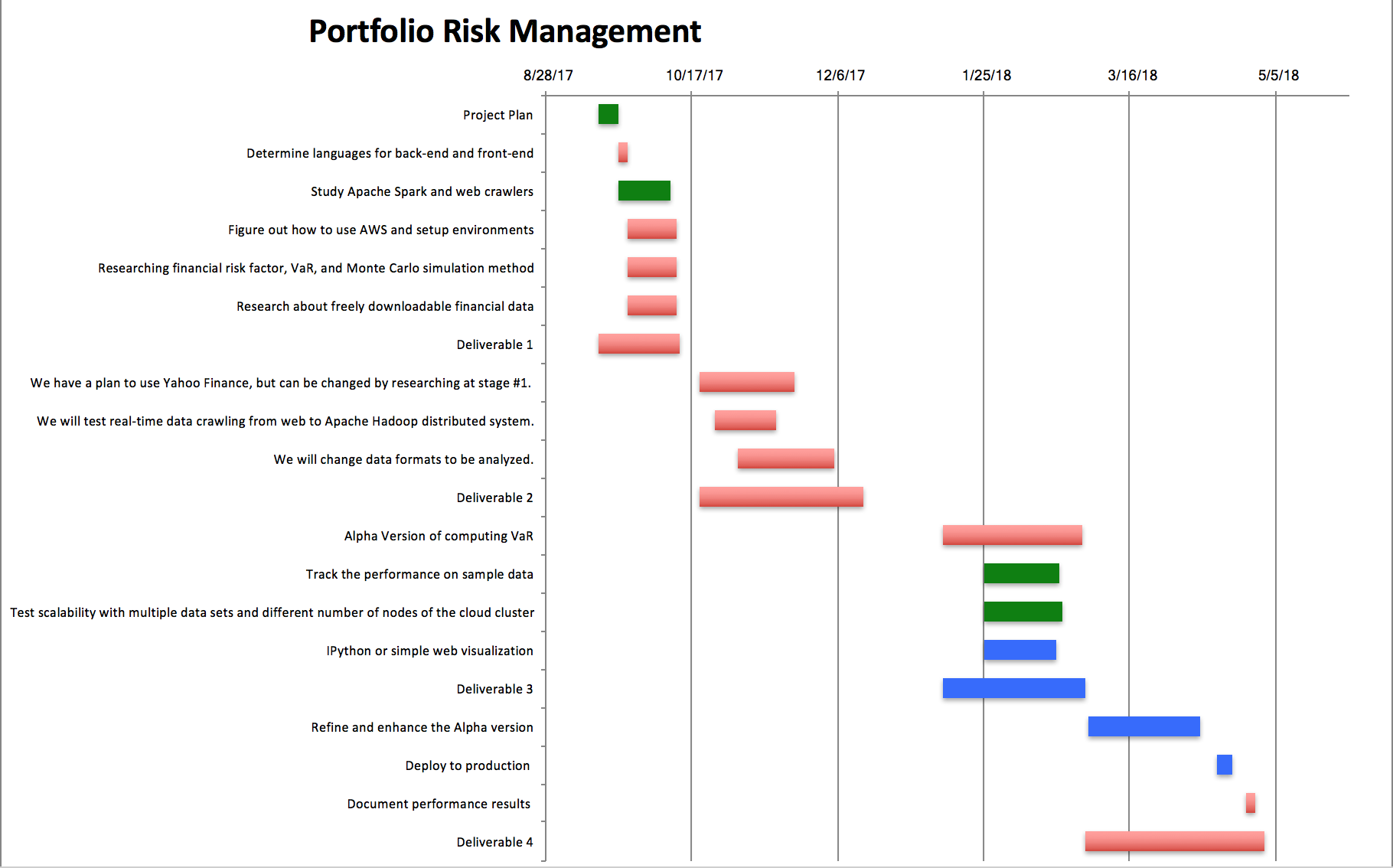


* From looking at the sentiment score, we can see equifax score is below 0 for most of August, which we interpreted as somewhat risky stock from how the articles were interpreted.
  + **Small risk negative movement** - We can see that pre-September on the actual pricing with Equifax stocks having negative movement on the price, when comparing to just below 0 sentiment average
  + **Major risk, large negative movement** - As we can see when September 7th came, there is large number of negatively scored articles, showing there is a major risk of a negative change. And we noticed that with the major sell volume of Equifax causing prices to drop

**Future Plans and Improvement**

* Future Planning
  + Apache Spark Computing
    - These web crawler and monte carlo requires a machine with high processing power and computing, so we will look into Apache Spark to set up our cloud computation system
    - Apache Sparke will hold our python functions and will run continuously with Hadoop’s cloud server to generate the monte carlo data as well as sentiment scoring.
    - We will set constraints to run and collect certain data only to prevent collection of the same data
  + Hadoop and Data Management
    - We are generating a lot of data and we want to figure out how to better manage it and store it for later purposes, such as the risk analysis and sentiment scoring
    - We are planning to use Hadoop as a cloud storage system. And this will be the middleware for storing articles that needs to be parse, and then passed to Apache Spark (sentiment scoring parsing) to obtain a scored result. And then it will be passed back again to Hadoop to store the result
* Rooms for Improvement
  + Filtering Articles
    - Some articles are irrelevant to the company, but the crawler picked it up due to having one related keyword which could cause deviation to our sentiment scoring
  + More Article Sources
    - Only two working article sources with good and correct parsing (Bloomberg and Motley Fool), currently trying to correct other sources like Wall Street Journal, CNBC, and CNN Money
  + Newer Dictionary
    - Loughran Dictionary is from 2014, could be new changes that is more correct or better score articles
  + Monte Carlo
    - Study other models to find room for improvement
    - Look into the possibility of a neural network or other deep learning

**Remainder Timeline and Planning**



**Overview**

The backend (monte carlo, webcrawler, and sentiment score) currently stands strong with minor needs of improvement. Moving forward, front end application will be designed to integrate the backend for user interface and interaction.

* Functionality of Monte Carlo: Long Term Analysis
* Functionality of WebCrawler/Sentiment Scoring: Immediate Analysis
* Goal: Combine Sentiment Scoring & Monte Carlo to improve accuracy of long term analysis
* Finally: scale the complete functionality with Amazon Web Services and Apache Spark