# Algorithmic Trading using the Deutsche Börse Public Dataset

Marjan Ahmed
*Dept of Computer Science*
*University of Illinois*
*at Urbana-Champaign*
Vancouver, BC, Canada
marjana2@illinois.edu

Fan Yang
*Dept of Computer Science*
*University of Illinois*
*at Urbana-Champaign*
Palo Alto, CA, United States
fanyang3@illinois.edu

Dilruba Hawk
*Dept of Computer Science*
*University of Illinois*
*at Urbana-Champaign*
New York, NY, United States
dilruba2@illinois.edu

Wang Chun Wei
*Dept of Computer Science*
*University of Illinois*
*at Urbana-Champaign*
Brisbane, QLD, Australia
wcwei2@illinois.edu

***Abstract*—This report details the findings, implementation and research method of the teams algorithmic trading model developed using machine learning frameworks implemented leveraging AWS cloud computing infrastructure. Github project website: https://github.com/weiwangchun/cs498cca**

## I. Introduction

We intend to build a cloud hosted backtesting framework for algorithmic trading. The objective is to analyze a given data set of equities containing price-volume data and develop medium to high frequency cointegration strategies for trading equities (*pairs trading*) quickly via cloud computing. A simple web interface will be constructed allowing users to control and tweak trading parameter settings, and see in-sample and out-of-sample trading performance (i.e., hypothetical profit and loss, tradings costs, turnover, portfolio risk and Sharpe ratio).

## II. Dataset

The chosen dataset is the Deutsche Borse public dataset available on Amazon Web Services (AWS). It contains trading data in 2 minute intervals for every tradeable security listed on the Eurex and Xetra trading platforms located in Frankfurt, Germany. The dataset is circa 3 GB in size with over 18,000 files and is located at: https://s3.eu-central-1.amazonaws.com/deutsche-boerse-xetra-pds/

## III. Technologies and Tools

To implement the objective of this project, we have chosen to leverage the vast computing and storage resources offered by AWS. The technologies used in this project are as follows:

- Simple Storage Service (S3) for storing the dataset and outputs.
- Elastic Cloud Compute (EC2) for initial exploratory analysis and model development.
- Elastic MapReduce (EMR) - for Deployment in Production.
- Route53 for front end web interface.

A step-by-step methodical approach was implemented starting from creating an AWS account, creating users (team members), creating roles and permissions for team members and applications using Identity Access Management (IAM), downloading the data into the created S3 bucket using python helper functions and then use PySpark, built on AWS EMR clusters for exploratory analysis of the data.

EMR cluster creation process involved creating a 3-node cluster. Selection of PySpark for model development led to inclusion of the relevant applictions, in this case, PySpark and JupyterHub. JupyterHub helps in line block execution of codes and display results subsequently helping us explore the data and keep track of every outcome. This process will help us test different predicivte models and alogrithms and fine tune the process in developing the final model before production deployment.

After the model is finalized, We intend to develop a front end Web Interface using Amazon Route 53. A domain named *datainsight.guru* is chosen where users can choose tickers and, control and tweak trading parameters to see the outcome of the predictive model.

## IV. Method Design

In terms of the overall methodology, we choose S3 to store the stock price data and choose to use PySpark built on AWS EMR clusters[1] to analyze the data. We use boto3 package in Python to upload stock price data from the Deustche public dataset to S3[2]. Lastly, we use EMR Notebook to run PySpark jobs. Announced in Nov 2018, EMR Notebook is essentially a Python Jupyter notebook pre-configured to connect to EMR and S3.

The general process of the project is as follows:

1) Stock price-volume data is uploaded on S3
2) User (via web interface) selects the time horizon and trading parameters (i.e., trading costs, trading frequencies, lookback period, size of trading portfolio, ...)
3) We iterate through the time period provided by the user, at each point $t$:
   - A training set is constructed based on a lookback period ($k$), i.e. from $t-k$ to $t-1$
   - A 'trading model' (described in the next section) will be fitted / applied to the training set

[1]We created a 3-nodes cluster following the AWS Cluster creation process.
[2]See our codes in Github

- Stocks will be brought and sold in the test set (time $t$)
- Profitability will be recorded.

4) Results will be displayed back to the user via the web interface

### A. Trading Model

Initially, we opted to examine momentum strategies (Jegadeesh and Titman, 1993) on the German market. However, given the Deutsche database has a relatively short history (circa 2 years), we decided to instead focus on higher frequency trading strategies. In particular, we will focus on implementing pairs trading (Vidyamurthy, 2004; Zhang and Zang, 2008) on relatively high frequency equities data.

Over user selected trading frequencies (highest frequency = 2 minutes, lowest frequency = 1 day), we systematically search for cointegrated pairs of stocks on the Xetra and Eurex stock exchanges using both the Engle-Granger and Johansen methods. Our trading strategy involves simultaneously buying and selling top cointegrated pairs whose spread has diverged, i.e., betting on convergence.

## V. Preliminary Evaluation Results

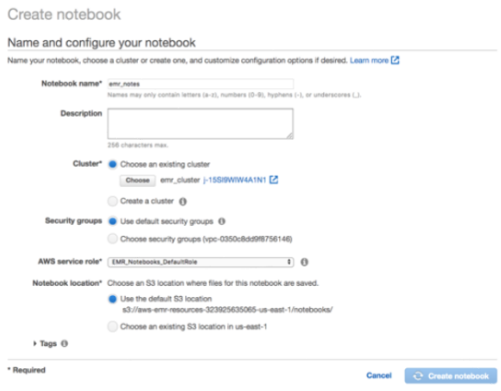
Since the main objective of CS 498 Cloud Computing Application is to understand cloud computing applications and their successful deployment, we emphasized the preliminary evaluation results on demostrating our ability to successfully create and deploy the cloud computing technologies listed in the Method Design section of this report.

The various stages of deployment that we have completed so far:

1) Using IAM to create users and apply roles and permissions for accessing the applications

| User name | Groups | Access key age | Password age | Last activity | MFA |
|---|---|---|---|---|---|
| dilruba | CCA | 3 days | 3 days | None | Not enabled |
| fan | CCA | 3 days | 3 days | None | Not enabled |
| marzan | CCA | 3 days | 3 days | 3 days | Not enabled |
| wang | CCA | 3 days | 3 days | Today | Not enabled |

2) Creating Roles for enabling application and user access to applictions
3) Creating S3 bucket for data storage and access
4) IAM Role for S3 Bucket access by other applications used in this project
5) Create EMR cluster (see Figure 1)
6) Configuration for Python Jupyter Notebook to connect to EMR and S3




7) Run PySpark job on Jupyter Notebook (see Figure 2)

## VI. Discussion

Much of the team discussions focused on selecting the appropriate technologies that can help us achieve our objective of developing a successful application thats ready for real world usage. Cost and the computing power of the technologies was also an item of discussion. AWS can be quite expensive if we dont select the proper technologies that can efficient execute the tasks needed. For example, PySpark framework built on EMR was chosen to achieve efficiency and speed that will not only help us reduce cost but deliver fast results to the users.

## VII. Future Work

The projects initial set up and testing of the applications seamless integration with each other has been completed. Now the framework or the infratructre is in place, next steps include creating or selecting the appripriate predictive algorithms to help us build a successful model. After the model is successully created, the project will move into production phase, interconnected applications checked for seamless integration and development of the front end user interface by using Amazons Route 53 application.

## VIII. Division of Work

Fan Yang and Wang Chun Wei, contributed to developing the Python helper codes for connecting to and downloading the data into S3. Along with their prior Macine Learning knowledge, and in collaboartion with Marjan Ahmed and Dilruba Hawk, the dataset was chosen and initial model development strategy such as selecting the appropriate Machine Learning model (for example, regression) were chosen. Marjan Ahmed and Dilruba, along with inputs from Fan Yang and Wang Chun Wei, created the Amazon account, created users, roles and permissions using IAM and created EC2 and S3. Fan Yang also tested the jupyter notebook and ran a computing job using the EMR cluster. Dilruba will help with developing the front end web application suing Route 53 for this project.

## References


[1] Jegadeesh, N., and S. Titman (1993) Returns to buying winners and selling losers: Implications for stock market efficiency, *Journal of Finance*, 48, 65 - 91

[2] Vidyamurthy, G. (2004) *Pairs Trading: Quantitative Methods and Analysis*, New Jersey: John Wiley & Sons.

[3] Zhang, H. and Zang, Q. (2008) Trading a meanreverting asset: Buy low and sell high. *Automatica* Vol. 44, 1511-1518

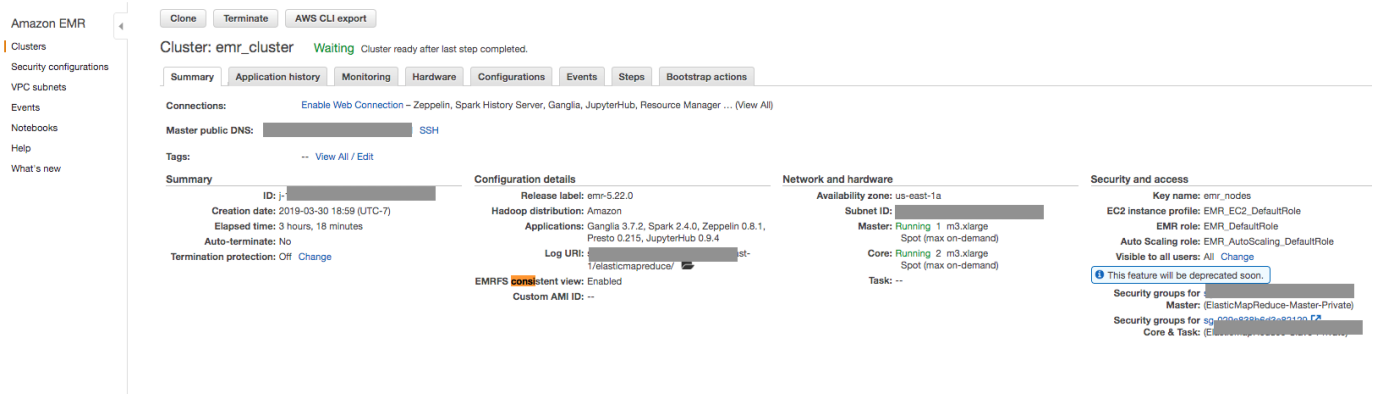



Fig. 1. EMR Cluster

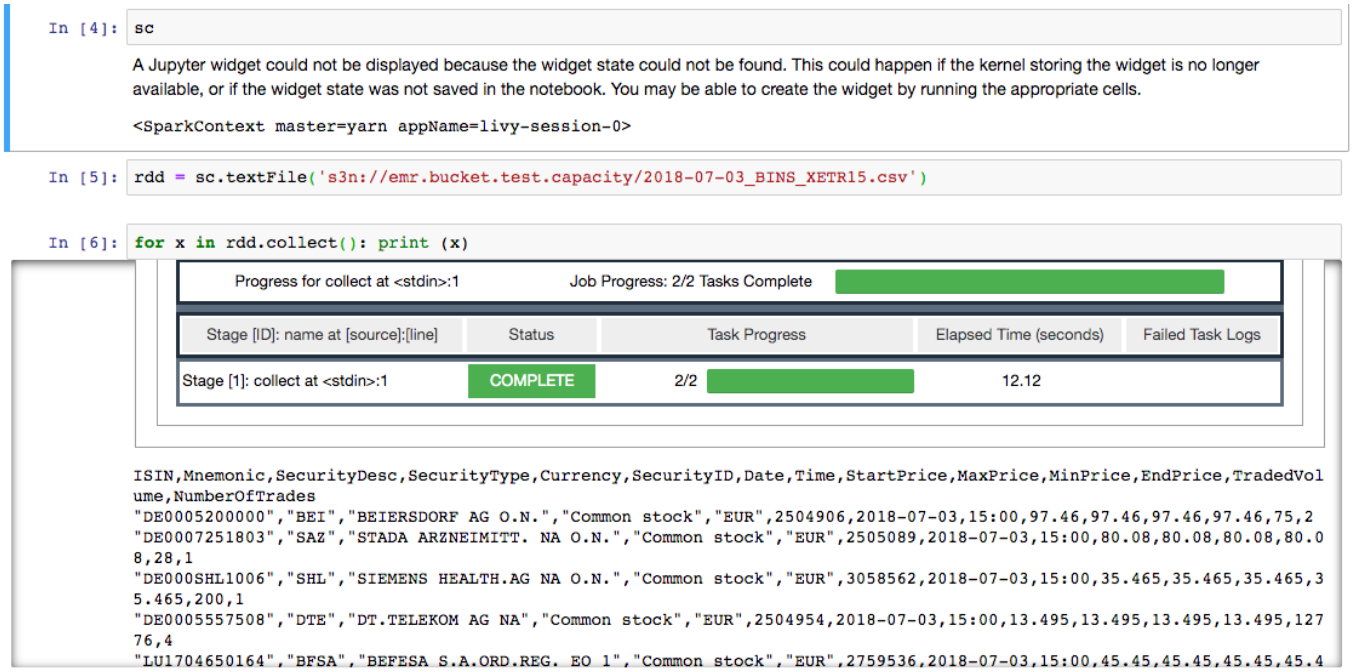



Fig. 2. Running PySpark on Jupyter Notebooks