# Financial Surveillance Using Big Data

Project CS,
Uppsala University

Final Presentation
2018/01/11 Uppsala

# Members of Project CS 2017

Asa

Daniel

Emanuel

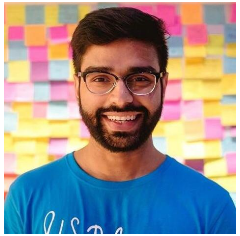
Ludvig

Martin

Michael

Philip

Rahul



Satya

# Topics

- Background
- Apache Spark
- Environment
- Parsing
- Spoofing
- Machine Learning
- Demo
- Conclusions and Future Works

# Background

# MO' DATA MO' PROBLEMS

- Scila currently offers a solution focused on surveilling data in real time
- From stream processing to batch processing
- Apache Spark is designed to make this process more efficient

# Proof of Concept with Apache Spark

Can we use Apache Spark to process and present the data produced by Scila?

Can we provide tools to run pattern detection and help visualize the data?

# Major components

- Parsing the data and implementing functionality for querying it
- A Proof of Concept spoofing detection algorithm
- Machine Learning techniques for anomaly detection

# Apache Spark

# What is Apache Spark?

The world's largest open source project in data processing.

The API allows for easy execution of streaming, machine learning or SQL workloads.

In-memory cluster computing where the key is its ability to iterate multiple transformations in memory before writing back to disk.

Spark is ***not*** a file distribution framework. Spark is designed to work together with HDFS, not replace Hadoop entirely.

Written in Scala, API also supports Java and Python

# How we use data in Spark

Spark Dataset (Available since Apache Spark 1.6)

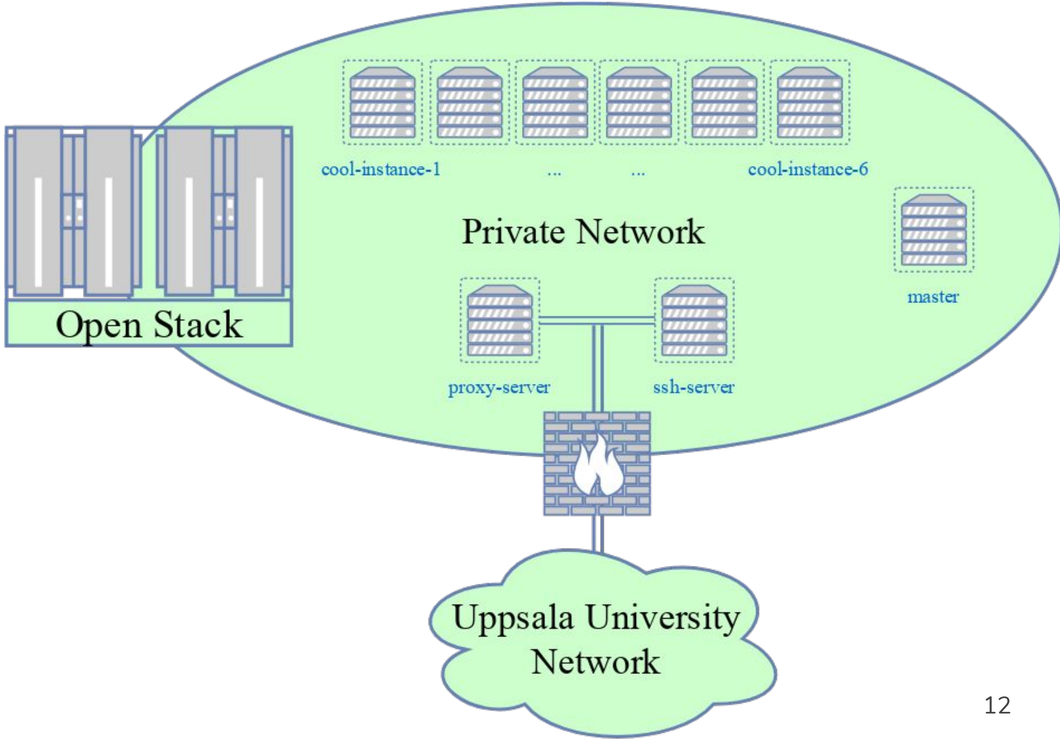
Offers high-level domain specific language operations (e.g. select, join, sum, count)

Datasets leverage Sparks fast in-memory encoding

# Environments

# Cluster

- OpenStack
    a. 6 Workers
    b. 1 Master
    c. 1 Proxy server
    d. 1 SSH server
- Private network
- Public access
    a. Proxy server
    b. SSH server



# Hadoop Distributed File System (HDFS)

What?

- Java based file system that provides scalable and reliable data storage

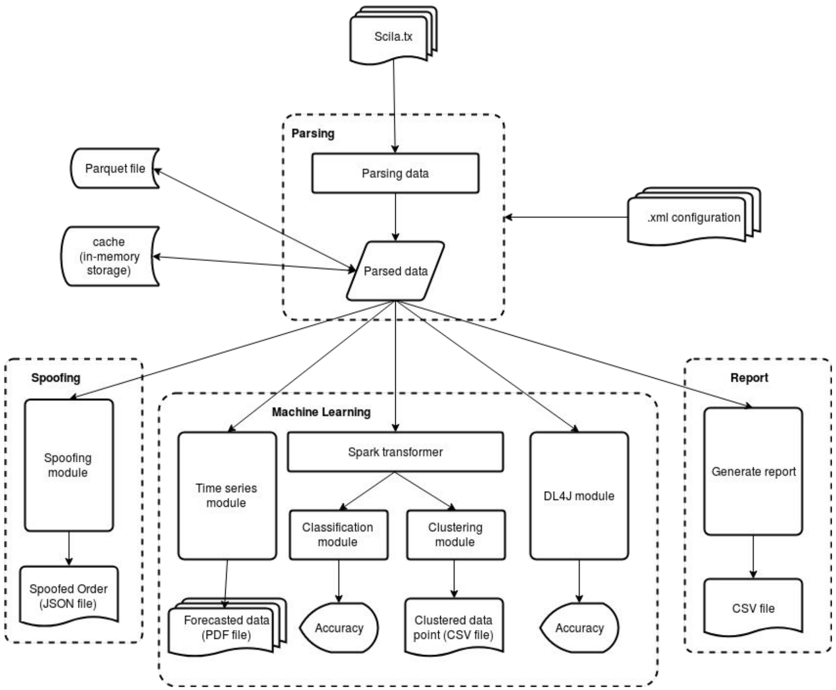
Why?

- Scalable
- Fault tolerant
- Distributed Storage system
- Good for clusters

# Other tools

- Docker
  - Worker image
  - Private network
    - Enables DNS support
  - Mount data-folder at some mounting point
- Spring
  - Loading beans through xml files.
  - No need recompile parameter change
  - Easy swap of implementations classes
  - Specify different profiles for different environments

# System overview



# Parsing

# The provided data

- Historical financial transaction data
- Scila Message Types
- Structure of the input data
  - Each line:
    - Size in bytes
    - JSON header
    - JSON message

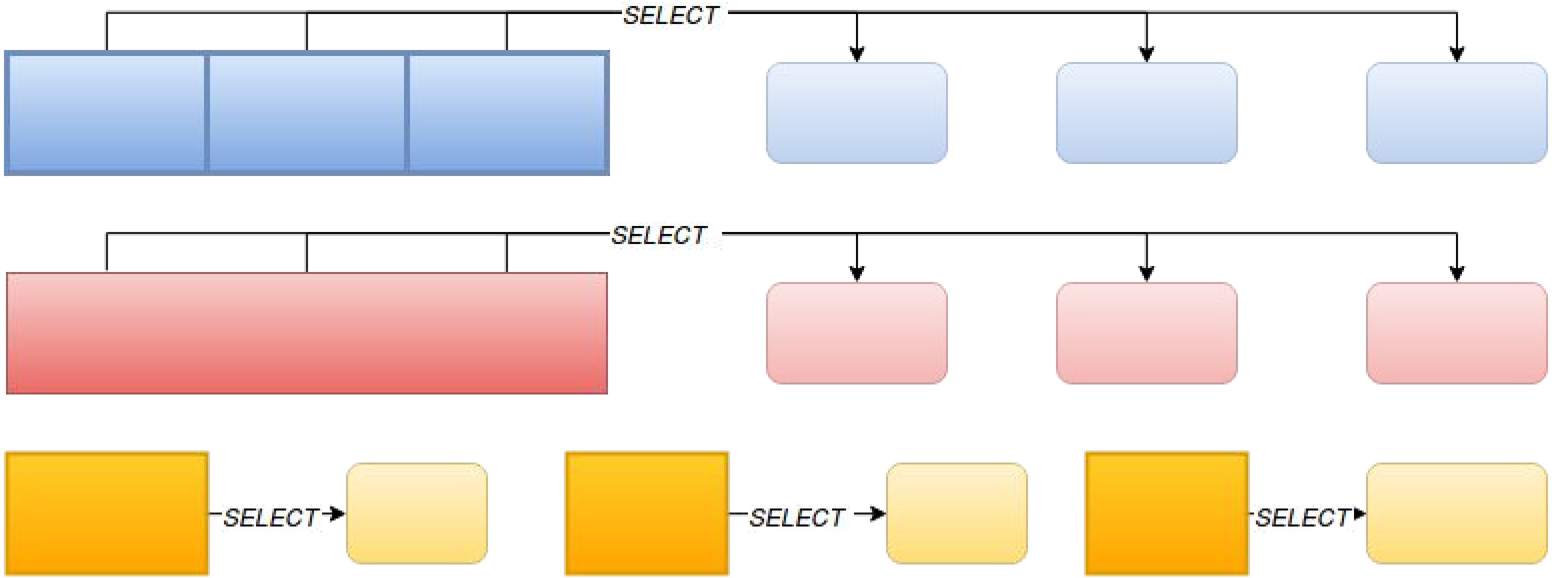
# Parsing the data

There are two main steps to go from the text file to SQL queries:

- Extract both JSON objects
- Apply Spark's JSON reader
  - Encode data
- Total memory consumption: 5.4 GB
  - Disk size packed: 2.7 GB
  - Disk size unpacked: 12.9 GB

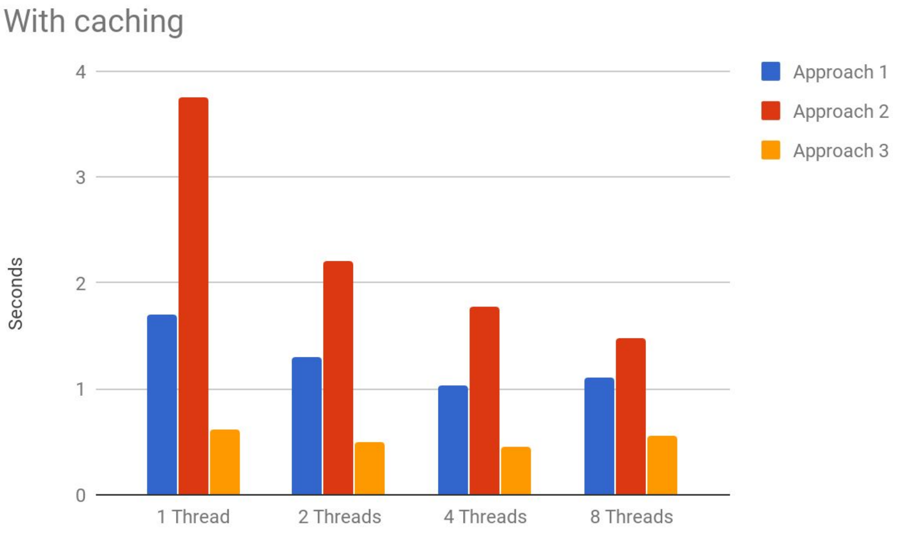
# Parsing the data

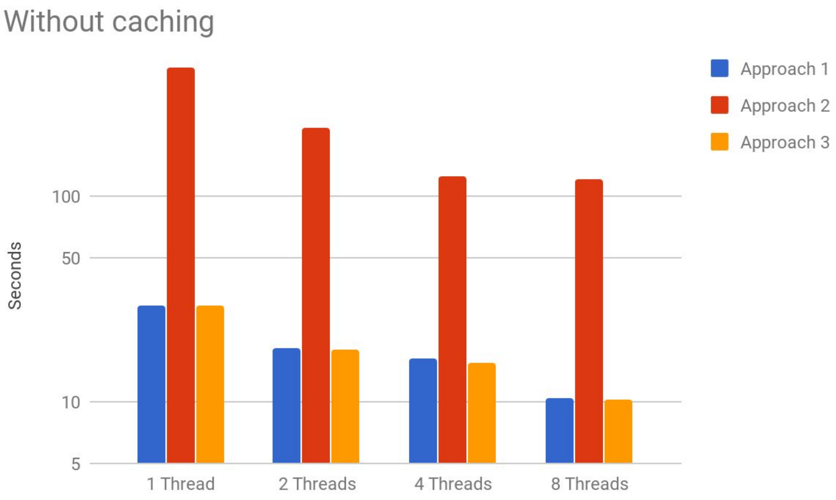
- Three approaches to construct the datasets from the input data and query them.



# Parsing benchmarks

SQL query used: Select count(*) from order_events
Data: 1.4 GB data, 4 million rows



# Spoofing

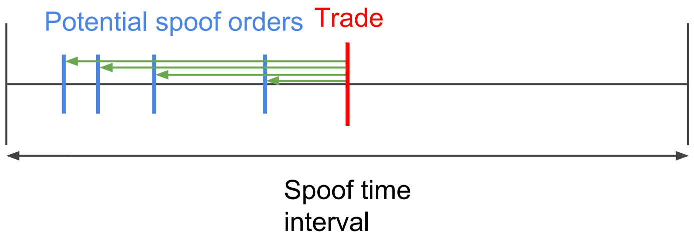
# What is spoofing

- "Spoofing" is a practice of using orders that are not intended to be executed, to manipulate prices.
- Defined differently in different markets.We have gone by the definition provided in the specifications and EU regulators for european markets.

# Spoofing detection

Multi stage filtering

1. From the data, analyze each trade.
2. Next step involves getting the suspected spoof orders.
3. Orders that are over a specified value.
4. Orders that were canceled by the same person as the trade.




# Machine Learning

# Spark MLlib Utilization

- Attempted outliers detection using unsupervised learning.
- Forecasting stock closing price using time series algorithm.
- Classifying market participant based on historical trade data using supervised learning.

# Supervised Learning

# The experiment

- Performance comparison between classifiers
- The effect of normalization
- Different size of dataset
- The effect of hyper-parameter tuning
- Different attributes
- Different type of market participant level

# The Most Interesting Result

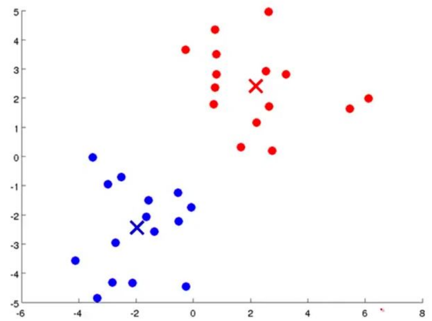
- Normalized attributes: price, volume, tradeHour, tradeMinute, tradeSecond
- Label: various level of market participant
- Data size: 1 day (11301 instances up to 22602 instances)

# Result

| Class | Unique labels | Logistic Regression | SVM | Random Forest | MLP |
|---|---|---|---|---|---|
| askEndUserRef | 38 | 3.88% | 3.34% | 9.57% | 6.02% |
| askUser | 19 | 9.99% | 5.99% | 12.88% | 10.58% |
| askMember | 7 | 30.98% | 24.03% | 31.45% | 30.98% |
| bidEndUserRef | 38 | 3.46% | 3.31% | 7.07% | 6.56% |
| bidUser | 19 | 7.57% | 6.23% | 11.15% | 10.17% |
| bidMember | 7 | 31.96% | 27.25% | 32.62% | 32.29% |
| allEndUserRef | 38 | 3.19% | 3.19% | 7.10% | 3.85% |
| allUser | 19 | 9.53% | 5.85% | 11.04% | 9.31% |
| allMember | 7 | 31.12% | 24.06% | 31.11% | 31.15% |

# Unsupervised Learning

# Unsupervised

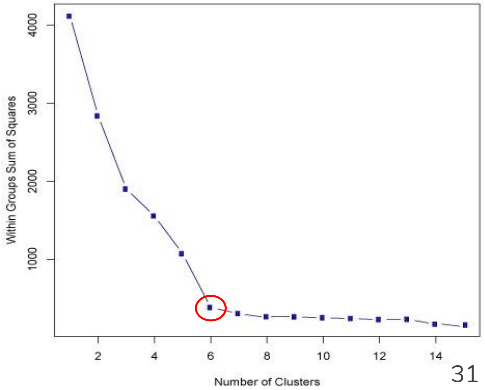



Unsupervised Learning Algorithm:

- K-Means
- Gaussian Mixture Models

How to determine data as outliers? → Distance threshold / Weight probability

How to determine number of clusters ? → Elbow method

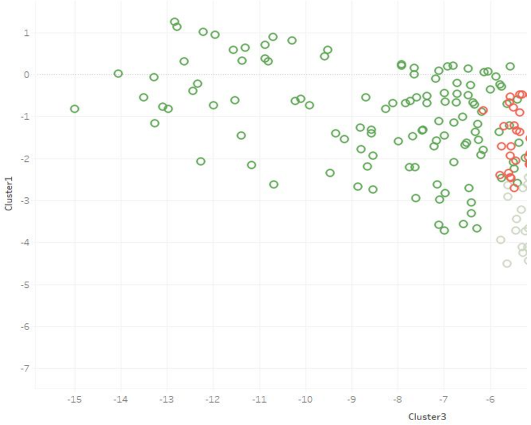


# Unsupervised Experiment - K Means

Threshold = 0.9

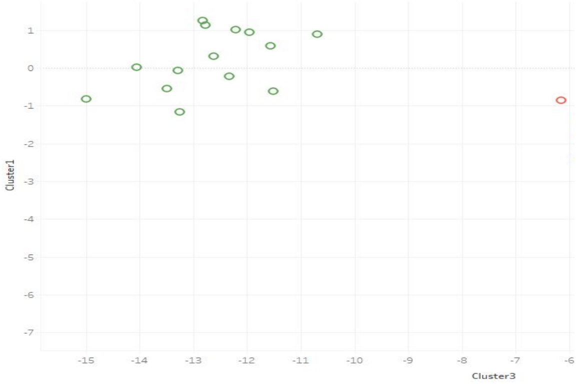
Average distance of each cluster

| Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 | Cluster 5 |
|---|---|---|---|---|
| 2.8637 | 7.0018 | 295.0512 | 5.7225 | 3.912 |




Distance of potential outliers for each clusters

| Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 | Cluster 5 |
|---|---|---|---|---|
| 88.1736 | 12.2218 | 428.9979 | 79.812 | 92.296 |
| 75.1058 | 11.8306 | 356.2108 | 79.1606 | 90.6174 |
| 75.0593 | 11.3948 | 347.68 | 77.8769 | 90.3598 |
| 69.3013 | 11.125 | 347.6308 | 74.7163 | 89.3841 |
| 69.2294 | 10.7322 | 346.2992 | 74.2232 | 87.7603 |



# Unsupervised Experiment - Gaussian Mixture Models

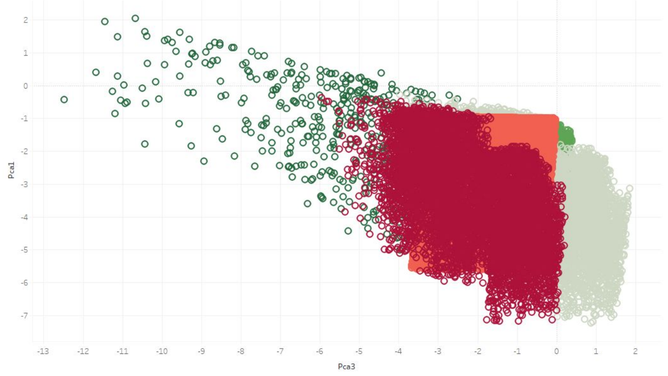
Threshold = 0.5

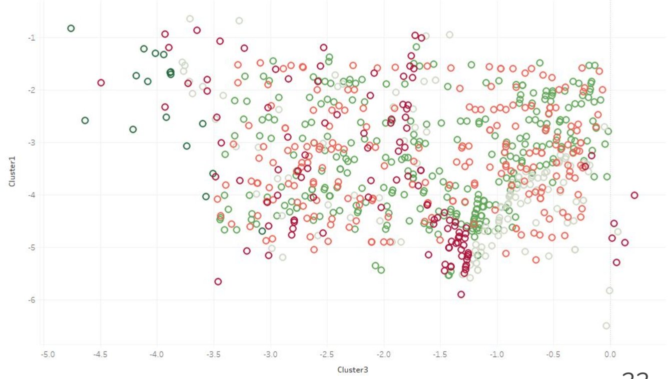
Weight probability for potential member clusters

| Data Point | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 | Cluster 5 |
|---|---|---|---|---|---|
| 1 | 0.0011 | 0.9968 | 0 | 0.0021 | 0 |
| 2 | 0.0012 | 0.9964 | 0 | 0.0024 | 0 |
| 3 | 0.0009 | 0.997 | 0 | 0.0021 | 0 |
| 4 | 0.001 | 0.9966 | 0 | 0.0024 | 0 |
| 5 | 0.0009 | 0.9971 | 0 | 0.0021 | 0 |

Weight probability for ambiguous cluster members

| Data Point | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 | Cluster 5 |
|---|---|---|---|---|---|
| 1 | 0.4138 | 0 | 0.1283 | 0.4495 | 0.0084 |
| 2 | 0.4166 | 0 | 0.1293 | 0.4457 | 0.0084 |
| 3 | 0.0012 | 0.4061 | 0.2033 | 0.3887 | 0.0008 |
| 4 | 0.1744 | 0.4061 | 0.0006 | 0.4176 | 0.0012 |
| 5 | 0.0012 | 0.4061 | 0.2033 | 0.3887 | 0.0008 |



# Time Series Model

# Sample Dataset used for prediction

| Date | Stock | Closing Price |
|---|---|---|
| 2017-07-27 | GOOGE590 | 146200000 |
| 2017-07-09 | GOOGQ590 | 123500000 |
| 2017-07-07 | GOOGQ600 | 129900000 |
| 2017-06-23 | SCILASEK | 16700000 |
| 2017-07-12 | ERICSEK | 66700000 |
| 2017-06-16 | AAPLUSD | 96910000 |

# ARIMA model

- ARIMA model stands for **A**uto**R**egressive **I**ntegrated **M**oving **A**verage model
- Three stages of ARIMA :
  - **Integrated (I)**
  - **Auto-Regressive (AR)**
  - **Moving Average (MA)**
- It is usually notated with ARIMA(p, d, q) where p, d and q are the orders of each of AR, I and MA part
- ***Final Goal*** : Each of the three stages is an effort to make the final residuals display no pattern at all i.e the model should have extracted most information from the data

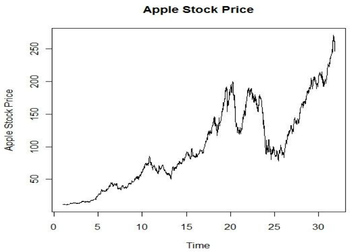
http://ucanalytics.com/blogs/arima-models-manufacturing-case-study-example-part-3/

# Working of ARIMA model

| Date | Stock | closingPrice |
|---|---|---|
| 2017-07-21 | AAPLUSD | 97030000 |
| 2017-05-14 | AAPLUSD | 96930000 |
| 2017-06-12 | Cocoa3m | 5900000000 |
| 2017-06-02 | Cocoa3m | 5780000000 |
| 2017-06-09 | Cocoa3m | 5710000000 |
| 2017-07-12 | ERICSEK | 66700000 |
| 2017-05-04 | Eurodollar3m | 105080000 |
| 2017-08-31 | GOOGQ590 | 119900000 |
| 2017-06-28 | GOOGQ595 | 126800000 |
| 2017-07-15 | Gold1m | 1652000000 |
| 2017-06-24 | HIQ3SEK | 111200000 |
| 2017-04-30 | LMECopper3m | 1700000 |
| 2017-06-01 | LMECopper3m | 1725000 |
| 2017-06-18 | NatGas17 | 3579000 |
| 2017-05-09 | PhelixDayBase | 35000000 |
| 2017-07-27 | SCILA2SEK | 16700000 |

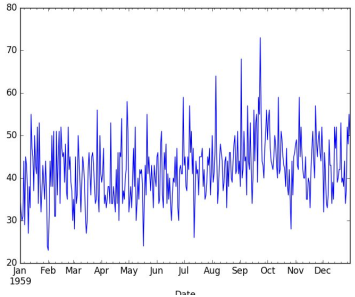
Stock time series dataset

Visualize the data




**Identify the order for Integration stage (Differencing)**

Stationary series



# Working (contd.)

Identifying parameters for AR/MA i.e **p** and **q** orders

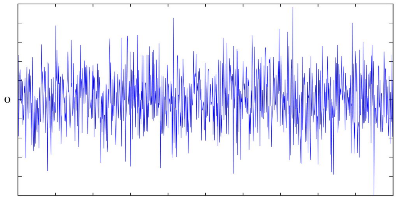
*Model :*

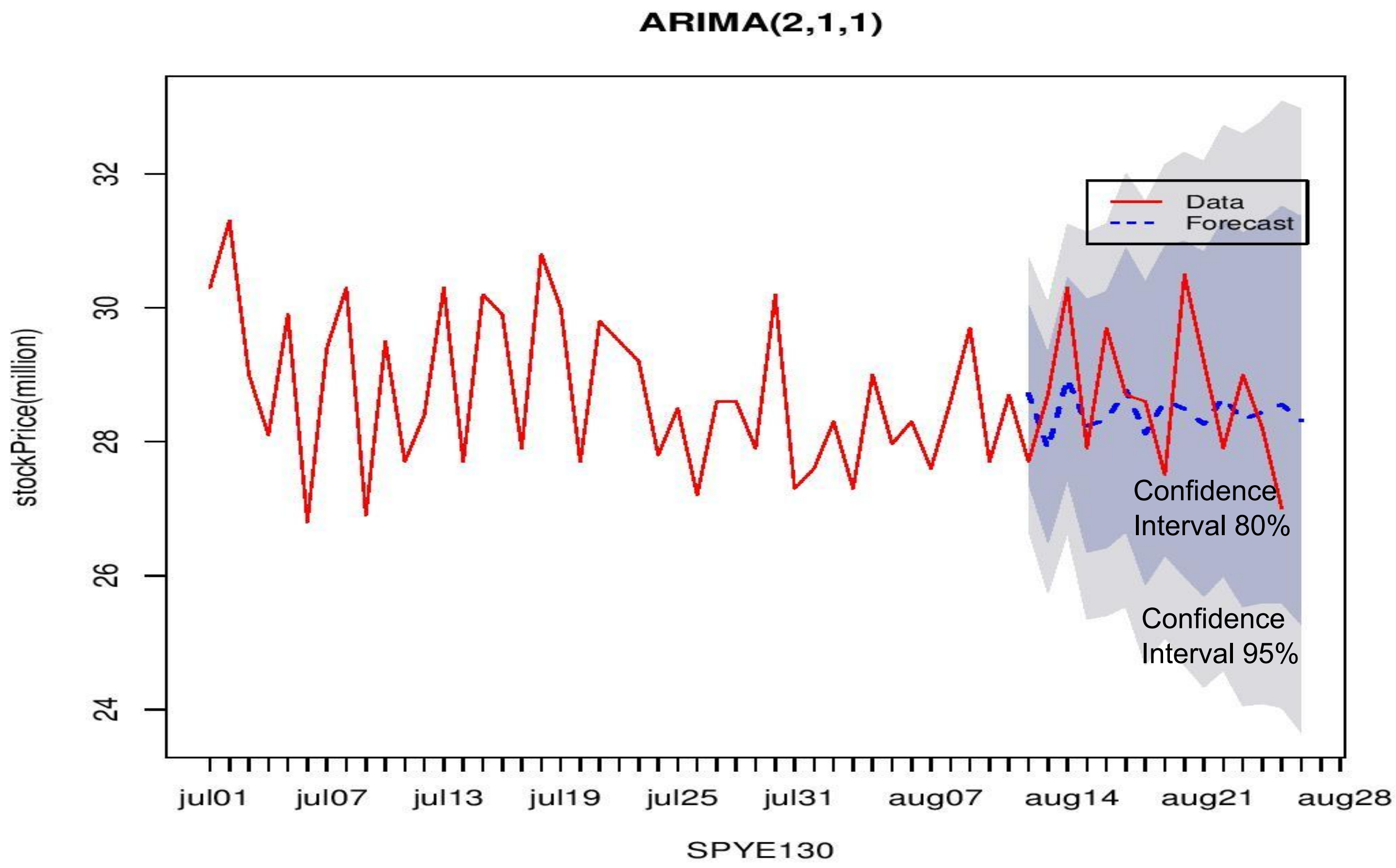
$$Y_t = c + \underbrace{\phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p}}_{\text{Auto-Regression}} + \underbrace{\theta_1 e_{t-1} + \cdots + \theta_q e_{t-q}}_{\text{Moving Average}} + e_t$$

Example : AR(1) can be represented as

$$Y_t = c + \phi_1 Y_{t-1} + e_t$$

Noise (No pattern at all)




ARIMA(2,1,1)
32
30
28
26
24
stockPrice(million)
Data
Forecast
Confidence Interval 80%
Confidence Interval 95%
jul01
jul07
jul13
jul19
jul25
jul31
aug07
aug14
aug21
aug28
SPYE130

# Accuracy of ARIMA model

Below is the accuracy for stock “SPYE130”

| | ME | RMSE | MAE | MPE | MAPE | MASE |
|---|---|---|---|---|---|---|
| Train | -0.086 | 0.983 | 0.820 | -0.385 | 2.881 | 0.554 |
| Test | 0.181 | 1.047 | 0.907 | 0.514 | 3.151 | 0.613 |

ME - Mean Error
RMSE - Root Mean Squared Error
MAE - Mean absolute error
MPE - Mean percentage error
MAPE - Mean absolute percentage error
MASE - Mean absolute scaled error

# Demo

# Conclusions & Future Works

# Conclusions

- Spark is good for frequent task/queries because it is using in-memory when it runs.
- Whole data parsing for frequent task and day-by-day parsing for less frequent task
- Spoofing algorithm meets all the requirement and could detect suspicious spoofed orders, but there is no best value for each parameter found.
- Spark MLlib has sufficient API for supervised learning, but not enough for unsupervised learning and even none for time series.

# Future Works

- Explore latest technology such as Apache Flink for streaming and or batch processing.
- Implement spoofing algorithm in streaming approach using Spark API.
- Implement DBSCAN in Spark for Java to use as anomalies detection model.
- Implement Support Vector Machine that could handle multiclass problem and use different kernel.
- Integrate Spark MLlib and DL4J to be able to explore more diverse type of neural network.

# Questions?

Project CS,
Uppsala University