|  |  |
| --- | --- |
|  | |
| **The role of robots in stock market** | |
|  | |
| MASTERS Murodov Shohjahon | |
|  | |
|  |  |
| Choose a type  20XX |  |
|  |  |
|  | |

\*\*\* Scanned submission page 1 \*\*\*

\*\*\* Scanned submission page 2 \*\*\*

\*\*\* Scanned Thesis Author’s Statement page 1 \*\*\*

\*\*\* Scanned Thesis Author’s Statement page 2 \*\*\*

ABSTRAKT

Czech abstract

Klíčová slova: klíčové slovo, klíčové slovo

ABSTRACT

Algorithmic trading defines set of specific rules and instructions using computer programs to execute trades with low latency and no human errors.

The goal of this thesis is to build a system containing set of robots that manages automated trading. As such retrieving live and historical stock data, analyze the data by applying ai models, backward and forward testing, choosing a strategy for analyzed data and execute the trades. In technical terms building the system is represented by developing a standalone application, the robots are represented by the application’s services, and these words are used interchangeably. The system is to be capable of extension that means new analysis tools or strategies can be added by implementing a new service/robot.

<<to be adjusted>>

Keywords: automated trading, algo trading

Acknowledgements, motto and a declaration of honour saying that the print version of the Bachelor's/Master's thesis and the electronic version of the thesis deposited in the IS/STAG system are identical, worded as follows:

I hereby declare that the print version of my Bachelor's/Master's thesis and the electronic version of my thesis deposited in the IS/STAG system are identical.

CONTENTS

[Introduction 9](#_Toc22828381)

1. [Theory 10](#_Toc22828382)

[1 Heading 11](#_Toc22828383)

[1.1 Subheading 11](#_Toc22828384)

[1.2 Subheading 11](#_Toc22828385)

[1.2.1 Sub-subheading 11](#_Toc22828386)

[2 Heading 12](#_Toc22828387)

[2.1 Subheading 12](#_Toc22828388)

1. [Analysis 13](#_Toc22828389)

[3 Heading 14](#_Toc22828390)

[3.1 Subheading 14](#_Toc22828391)

[3.1.1 Sub-subheading 14](#_Toc22828392)

[3.2 Subheading 14](#_Toc22828393)

[4 Heading 15](#_Toc22828394)

[4.1 Subheading 15](#_Toc22828395)

[Conclusion 16](#_Toc22828396)

[Bibliography 17](#_Toc22828397)

[List of abbreviations 18](#_Toc22828398)

[List of figures 19](#_Toc22828399)

[List of tables 20](#_Toc22828400)

[appendices 21](#_Toc22828401)

Introduction

According to analysts, about 70% of US equities in 2013 were carried out via algorithmic trading, and the use of trading algorithms has continued to grow since then. But algorithmic trading didn’t just emerge out of nowhere. It’s good we look at the history. [1]

Algorithmic trading emerged with the advent of the internet in the late 1980s and early 1990s. However, it wasn’t until 1998 when the U.S Securities and Exchange Commission (SEC) authorized electronic exchanges that computerized high-frequency trading became mainstream.

Algorithmic trading witnessed a great boom in the late 2000s. In the early 2000s, algo trading accounted for less than 10% of equity orders, but it grew rapidly that by the end of 2009, algorithmic traders had captured 70% of the US securities markets. According to the NYSE, between 2005 and 2009 alone, algo trading volume grew by 164%.

The boom in algo trading was also accompanied by a significant decrease in trade execution time. For instance, in 2001, HFT trades had an execution time of several seconds, but by 2010, this had shrunk to milliseconds, even microseconds, and subsequently, nanoseconds in 2012.

The financial industry is evolving. The reality is that computers today are playing a bigger role in buying and selling stocks than ever before. By some estimates today, computer trading makes up more than 90% of trades. Whether the amount of trading done by computers is 87% or 93 is irrelevant though. At this point, it's pretty clear that computers are more than half of all trades out there. We refer to this type of trading as algorithmic. It's driven not by an individual saying it's time to buy or sell, but instead by a computer operating a preset program to buy or sell a stock. Now, despite that, humans still play a role in trading. In particular, humans are important for designing algorithms. Computers will carry out the program to buy or sell a stock, but it's humans that decide whether that program makes sense and whether they will make money based on it. Algo Trading essentially comes in two flavors: market making and data mining. Market making makes trades based on capitalizing on the bid-ask spread. This is typically associated with professional financial firms, groups like Goldman Sachs or JP Morgan, or with high-frequency traders. So individual traders and hedge funds, things like that, they're not necessarily involved in market making in most cases. Where individuals and hedge funds are going to be more involved is the data mining side. In particular, data mining involves trading based on patterns in data like stock prices and outside information. The basic idea here is that you're looking for correlations between stock prices and other data points. You're looking for a relationship between a stock's price and some external set of data that'll tell us whether it's time to buy or sell. From there, we can then write a computer program that actually executes the trades.

Algorithmic trading can be done based on the relationship between customers and suppliers. For example, if we have a small under-the-radar firm that reports earnings, frequently has implications or read throughs for larger customers or suppliers that are related to that company. For example, think about Apple and some of the small vendors it uses. When a small supplier comes out and reports earnings, that has implications for how many iPhones Apple will sell. As a result, smart traders will look at what's going on with Apple's suppliers and try to understand how they should trade Apple based on that. Now, that's been done for a long time. Where algorithmic trading takes it to a new level is by automatically incorporating all this information in some sort of formulaic way and then making a trading decision based on that.

Algorithmic trading can be done based on the relationship between customers and suppliers. For example, if we have a small under-the-radar firm that reports earnings, frequently has implications or read throughs for larger customers or suppliers that are related to that company. For example, think about Apple and some of the small vendors it uses. When a small supplier comes out and reports earnings, that has implications for how many iPhones Apple will sell. As a result, smart traders will look at what's going on with Apple's suppliers and try to understand how they should trade Apple based on that. Now, that's been done for a long time. Where algorithmic trading takes it to a new level is by automatically incorporating all this information in some sort of formulaic way and then making a trading decision based on that.

Now that we understand the basics around practically putting together a trading algorithm, it's important to understand a few pragmatic rules. First of all, buying and selling with algorithms is more complicated than it seems. The reality is that the strategies and rules behind an algorithm are just the beginning. The process for buying or selling securities based on those algorithms is going to vary significantly. High frequency trading firms, for example, will often tap directly into a network for an exchange in order to trade. In other words, they want that algorithm to be able to seamlessly make all of the decisions, and then trade directly through the exchange, to do this as fast as possible. Outside of the high frequency traders though, most quant firms have separate systems for research versus trading. So you might develop an algorithm, and then test it on one system, and then put it into practice on a completely different system. Practically speaking, automating an algorithm to make trade decisions directly can actually be very dangerous. A lot of the time when you hear about significant market volatility in the stock market, these days, it often gets blamed on computer traders making decisions without human intervention. There's really a big raft of factors that you need to take into account when you're considering whether a computer should be allowed to make those trades directly, or if a human needs to be involved. Number one, you need to consider the liquidity in a stock. If this liquidity isn't there for a particular security, it can be very hard for a computer to trade that security and not move the price, and thus distort the returns around that trading strategy. In particular, this can be reflected in bid-ask spread imbalances. If the bid and the ask are too wide on a particular security, it can be tough for a computer to take that into account appropriately, and it may mean greater uncertainty in any algorithm you try to develop. Finally, it's really important that humans continue to be involved in evaluating the trade strategies that algorithms are based on. If you remove the human element, the human break, if you will, in evaluating these trading strategies, then it can potentially lead to unforeseen consequences. There was a firm a few years ago called Knight Capital, for example, that got into trouble, and reportedly part of the problem was that Knight Capital simply didn't have enough human involvement with some of their algorithmic trading. So human beings continue to play an important sanity check in considering any of the trades that computers are going to make, and checking to make sure that a particular algorithm seems like it'll hold water, seems like it'll continue to be valid going forward, rather than being a statistical artifact based on a limited subset of data. One final consideration to take into account when you're thinking about these trading decisions is what's called make-or-take liquidity. At-large asset managers, a key consideration is reducing trading costs and make-or-take liquidity plays a critical role in this. In particular, market firms like the NYSC or the NASDAQ, really value liquidity. Liquidity means providing a source of securities when everyone else wants to buy stock, and it means being willing to buy securities when everyone else is trying to sell. So ask yourself, will your firm be providing liquidity by buying when everyone else is selling? If so, you could be a liquidity provider. Providing that liquidity in the market can actually reduce your trading costs a lot. Firms like the NYSC and the NASDAQ pay rebates. They pay fees, they pay special cash bonuses, if you will, to firms that provide liquidity. Similarly, if you take liquidity from the market, if you're demanding it, if you won't step in and buy, when everyone else is selling, then you end up paying a higher price for any trades that you might want to make. Take into account carefully, whether your firm can provide liquidity in the markets, and whether this can actually help you to enhance your algorithmic trading strategies. There are a number of quantitative firms out there that are excellent at providing liquidity in the market, and essentially reducing their trading costs to very near zero. Dimensional Fund Advisors, one of the largest asset managers in the world, and a quantitative investment shop, counts the ability to provide liquidity to the market as a key advantage in lowering trading costs.

<<<to be completed>>>

**Technical introduction**

The system application is to be implemented in Java using spring ecosystem framework.

Spring Boot has become the almost default way to build Spring-based Java applications. The rapid development and built-in operational onboarding allow you to focus on your business and customers, and not on building new applications and wiring enterprise equipment. Spring Boot, a project introduced in a tweet, has become the most efficient, productive, and powerful way to build applications using Java for enterprises, as well as in cloud native situations. While there are many other frameworks on the market, many of which have their own value statements, nothing today tops Spring in popularity and effectiveness.

Maven which is a project management tool is used to manage application dependencies, plugins and executions…

Today's Java developer has several options when building his or her applications. Apache Maven not only solves the build needs of most developers but extends itself to include such areas as dependency management, unit testing, documentation and reporting. Now, one can achieve these same tasks with many other tools, but Apache Maven provides a coherent, clean and extensible framework to build applications, either as run-of-the-mill components or as unique, complex, and customized builds all with the same tooling.

PostgreSQL

is a robust relational database management system or RDBMS. According to the folks at db-engines.com, it ranks as the fourth most popular database platform in the world behind industry powerhouses, Oracle, MySQL and Microsoft's SQL Server. But unlike those three platforms Postgres has seen a steady increase in popularity over the past decade. This trend in popularity can be attributed to a number of factors. First, PostgreSQL is an open source project which makes it free for any organization of any size to pick up and apply to their needs. Compare that with some of the other popular platforms that can cost thousands or hundreds of thousands of dollars to deploy across an organization and you'll see why Postgres is a good choice from a simple financial perspective. Second, Postgres has been in active development for over 35 years and has gained a reputation for being a reliable, robust, and secure data platform. This is in large part due to its nature as an open source project where people from all over the world collaborate to examine source code, help identify and fix bugs, and develop new features. Finally, as more large businesses push Postgres to its limits, it gives smaller users more confidence in the platform's capabilities, which makes transitioning from a paid commercial database platform more justifiable. After all, if organizations the size of Reddit, Apple and Yahoo trust Postgres to meet their needs then it's definitely up to the task of handling anything that I can throw at it. Now, I mentioned that Postgres was a relational database management system so let's unpack exactly what that means. An RDBMS creates a protective wrapper around your data. It gives your data a place to live, controls how it gets entered, and under what circumstances it can be viewed, modified, or deleted. The RDBMS also secures your data and ensures that only people in applications that have expressed permission to view and change the data have the ability to do so and keeps everyone else locked out. Finally, an RDBMS manages the performance of the system which helps you enter data and retrieve information in the most efficient ways possible. To do all of this, PostgreSQL uses a server/client model. The Postgres server handles the storage, security and maintenance of your data by storing it in a container called a database. Each Postgres server can hold many different databases. A client application is then connected to the server so that end users can input data, modify that data and then retrieve that data back out again. The beauty of this arrangement is that the server can exist on a centralized computer that provides a single point of contact for all of the different users in an organization. Then each user uses their own desktop computer, mobile device or web app, running a client application to access that shared resource pool. In this way, everyone across the organization is getting the same information, updating the same data tables and viewing the same detailed reports. So in order to learn about the PostgreSQL platform, we really need to dive into two components; how the database server operates and how to use an interface client. In this course we'll be using a client called pgAdmin to log into the Postgres server and tell it what we want to do. In order to perform all of the tasks that a PostgreSQL server needs to do to manage your data, you'll need to become familiar with the SQL programming language. SQL stands for structured query language and it's a common language used by data science professionals in practically every database application in use today. This makes the ability to read and write SQL Commands a very transferable skill that's highly sought after in the workplace. Unlike many traditional programming languages SQL uses human readable syntax that's straightforward and flexible. So don't let the idea of learning a programming language scare you away. I'll introduce you to the basics in this course and you'll be writing your own SQL commands in no time. But if writing code is also not your thing, don't worry. The pgAdmin Graphical Interface client that we're going to use will give us lots of graphical tools that'll make executing commands a lot easier. So that's the basics of what PostgreSQL does and how it can be used to help protect and store your valuable data.

Flyway

Flyway is a good way to migrate databases with code updates in a managed way, and it has other features, like the ability to detect inconsistent and failed migrations. And that's really important, because as we've discussed earlier, one of the things we have to watch out for is having a partially implemented migration or set of changes. Now, there are a number of use cases for Flyway, specifically schema changes. So again, things like adding tables, dropping columns, adding other constraints, dropping indexes, those kinds of things. And also reference data changes, so if we have, for example, lookup codes and things like that and we want to change those across all of our database instances, Flyway's a good way to do that. And then also, it supports bulk data changes, as well.

DL4J Spark libs for ML

OpenAPI (swagger) api documentation

Today's most successful digital platforms are architected with APIs that expose core business capabilities for internal and external developers that build new applications at scale. Well-crafted APIs are more widely adopted by the developer community, giving their creators a competitive advantage in terms of innovation and new products. The key to developing successful APIs is using tools like Swagger to create better designs and documentation so that it is effortless for developers to use the API. Hi, I'm Kevin Bowersox. I've been a full stack developer for over 10 years, and I have seen how Swagger can build higher-quality APIs that gain adoption. Swagger helps you design, build, and document your APIs using a design-first approach so that web, mobile, and third-party applications can quickly integrate and consume their core capabilities and data. In this course, I'll show you how to build API definitions with the OpenAPI Specification that accelerate delivery of APIs through generated documentation, client SDKs, server stubs, and API server virtualization. We'll use Swagger's open source tools to master key sections of the OpenAPI Specification. Then we will advance to SwaggerHub and learn how its team collaboration tools streamline the API development lifecycle. To cap off the course, we'll complete a project that uses an API definition file to rapidly establish an API-based solution that spans multiple languages, frameworks, and cloud-based services. So if you are ready to build better APIs with higher adoption, then let's get started.

GUI with react.js (beyond the scope of the project ??)

|  |  |
| --- | --- |
|  | Short term trading … |

stock Pattern recognation

text

CNN

text

Most Common Repeated Patterns

text

Head-to-Shoulder Cups …..

text

Heading

text

Subheading

text

|  |  |
| --- | --- |
|  | Analysis |

Heading

text

Subheading

text

Sub-subheading

text

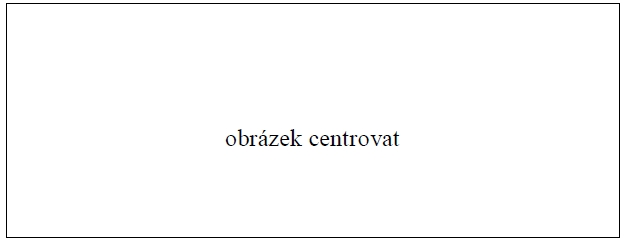


Figure 1 Caption of figure

text

Subheading

text

Table 1 Caption of table

|  |  |  |  |
| --- | --- | --- | --- |
| **Záhlaví tabulky 1** | **Záhlaví tabulky 2** | **Záhlaví tabulky 3** | **Záhlaví tabulky 4** |
| První řádek | 0,98 | 123,97 | 1258,58 |
| Druhý řádek | 1,5875 | 11,0334 | 251,005 |

Heading

text

Subheading

text

Conclusion

text

Bibliography

according to the citation style

List of abbreviations

ABC First abbreviation – meaning

B Second abbreviation – meaning

C Third abbreviation – meaning

List of figures

[Figure 1 Caption of figure 2](#_Toc22828145)

List of tables

[Table 1 Caption of table 2](#_Toc22828164)

appendices

Appendix P I: Appendix title

Appendix P II: Appendix title

Appendix P i: Appendix title

[1] Rejaul Karim. The History of Algorithmic Trading.