

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Eykhmann** Jméno: **Victoria** Osobní číslo: **456115**
Fakulta/ústav: **Fakulta elektrotechnická**
Zadávající katedra/ústav: **Katedra počítačů**
Studijní program: **Otevřená informatika**
Studijní obor: **Softwarové systémy**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Predikce ve financnictví (ERP)

Název bakalářské práce anglicky:

Forecasting in Finance (ERP)

Pokyny pro vypracování:

1. Get acquainted with ERP systems and provided DB export
2. Visualize the data (x-y plots, time restriction, histograms)
3. Implement various classification and prediction methods for selected timespan according to supervisor's instructions
4. Test the solution
5. Visualize the results
6. Unit test and document the project

Seznam doporučené literatury:

- [1] Marianne Bradford - Modern ERP: Select, Implement, and Use Today's Advanced Business Systems, 2008
- [2] Bret Wagner and Ellen Monk - Concepts in Enterprise Resource Planning, 2001
- [3] Kamel Nidal, Aamir Saeed Malik - EEG/ERP Analysis: Methods and Applications, 2014

Jméno a pracoviště vedoucí(ho) bakalářské práce:

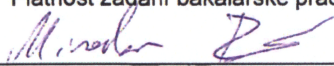
Ing. Miroslav Burša, Ph.D., oddělení biomedicínského inženýrství a asistivní technologie CIIRC


Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

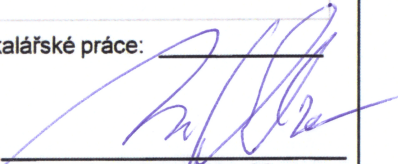
Datum zadání bakalářské práce: **31.01.2018**

Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **30.09.2019**


Ing. Miroslav Burša, Ph.D.
podpis vedoucí(ho) práce


podpis vedoucí(ho) ústavu/katedry


prof. Ing. Pavel Ripka, CSc.
podpis děkana(ky)

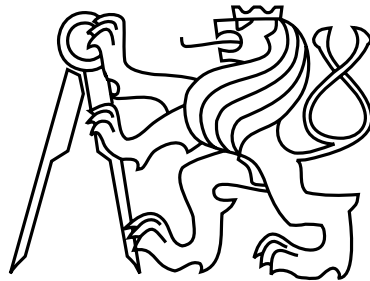
III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

Datum převzetí zadání

Podpis studentky

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering



Bachelor's Project

Forecasting in Finance

Eykhmann Victoria

Supervisor: Ing. Miroslav Burša, PhD

Study Programme: Open Informatics

Field of Study: Software Systems

May 24, 2018

Aknowledgements

I would like to thank Ing. Miroslav Burša, Ph.D. for guidance and regular consultations during writing this work. Also I would like to thank Michal Bouška for his helping.

Declaration

I hereby declare that I have completed this thesis independently and that I have listed all the literature and publications used.

I have no objection to usage of this work in compliance with the act §60 Zákon č. 121/2000Sb. (copyright law), and with the rights connected with the copyright act including the changes in the act.

In Prague on May 25, 2018

.....

Abstract

The aim of this thesis was to design and develop a web application for data specialist to import data from database, check the suitability of a dataset for classification and to generate a valid classifier that can be further used to classify incoming data. We have used real data from manufacturing companies (manufacture orders).

The application is implemented using Java EE and Spring framework and is designed to be run online. It is able to visualize data (to verify the data), train multiple classifiers on the whole dataset (or a subset). The best (selected) classifier can be exported and handed to the production team.

The application has been tested on three real datasets and was developed under a supervision of a commercial company that develops ERP systems.

Abstrakt

Cílem této práce bylo navrhnout a vyvinout webovou aplikaci pro datového specialistu, která umožňuje import dat z databáze, kontrolu validity datové sady pro klasifikaci a vytvoření (naučení) klasifikátoru, který lze dále použít pro klasifikaci nově příchozích dat. Využili jsme skutečné údaje z výrobních firem (příkazy k výrobě).

Aplikace je implementována pomocí Java EE a frameworku Spring. Aplikace je navržena tak, aby mohla být provozována online. Aplikace umožňuje vizualizovat data (verifikace datové sady), trénovat více klasifikátorů na celé datové sadě (nebo na podmnožině). Nejvhodnější (vybraný) klasifikátor lze exportovat a předat k použití v produkčním prostředí.

Aplikace byla testována na třech reálných datových sadách a byla vyvíjena pod dohledem společnosti, která vyvíjí ERP systémy (a která poskytla reálná data).

Contents

1	Introduction	1
1.1	Financial forecast	1
1.2	Problem statement	1
2	Analysis	3
2.1	Data specialist's workflow	3
2.2	Requirements	6
2.2.1	Functional requirements	6
2.2.2	Non functional requirements	7
3	Technology	9
3.1	Software development	9
3.2	Technology review	10
3.2.1	Server-side technology	10
3.2.2	Client-side technology	12
3.3	Architecture	13
4	Design & Implementation	17
4.1	GUI Design	17
4.2	Design	19
4.3	Components	22
5	Testing	31
5.1	Use cases	31
5.2	Code analysis	32
5.3	Unit tests	32
5.4	Cross browser testing	32
5.5	Conclusion	34
6	Conclusion	35
6.1	Further work	36
A	User Guide	39
A.1	Requirements:	39
A.2	Running	40

B Table of parameters	41
C Sprints	45
D List of abbreviations	47
E Content of CD	49

List of Figures

2.1	Process for data specialist	4
3.1	Waterfall method	9
3.2	Sprint	10
3.3	Top 15 development languages, ranked by IEEE Spectrum in 2017[5]	11
3.4	Comparison between JavaScript chart libraries	13
3.5	Client Server	13
3.6	Model-view-presenter and Model-View-Presenter	14
3.7	Java Python communication	15
4.1	Graph	18
4.2	Time Series	18
4.3	Train classifier	19
4.4	Use classifier	19
4.5	Class diagram	20
4.6	Use cases related to statistical graphs include importing data, selecting parameter for coloring data, displaying four types of graphs, toggling between linear/logarithmic scales and selecting dataset	21
4.7	Use cases for viewing one or several products on the time series graph and selecting timespan for graph	22
4.8	Use cases related to classifier include training one or several classifiers, displaying their accuracy, testing and export of a selected classifier	22
4.9	Upload new dataset for statistical graphs	23
4.10	Coloring by selected parameter	23
4.11	X logarithmic	23
4.12	Y logarithmic	23
4.13	Scatter plot with linear scales	24
4.14	Scatter plot with logarithmic scales	24
4.15	Histogram shows distribution of command price. Logarithmic y axis was used.	25
4.16	Boxplots	25
4.17	List for x axis	26
4.18	Choosing dataset element	26
4.19	Time series for a single product	26
4.20	Several products in one graph	27
4.21	Timespan selection in time series graph component	27

4.22	Form for training classifier	28
4.23	List of trained classifiers	28
4.24	Classifier results	29
4.25	Classifiers results	29
4.26	Use classifier page	30
4.27	Download classifier	30
5.1	Rendered main page of application in Chrome browser	32
5.2	Rendered main page of application in Edge	33
5.3	Rendered main page of application in Firefox browser	33
5.4	Rendered main page of application in Internet explorer browser	34
5.5	Rendered main page of application in Opera browser	34

List of Tables

5.1	Functional requirements implementation	31
B.1	Human readable names	41
B.2	Description in EN	42

Chapter 1

Introduction

1.1 Financial forecast

The purpose of the financial forecast is to evaluate current and future fiscal conditions to guide programmatic decisions. A financial forecast is a fiscal management tool that presents estimated information based on past, current, and projected financial conditions. This will help identify future revenue and expenditure trends that may have an immediate or long-term influence on government policies, strategic goals, or community services. An effective forecast allows for improved decision-making in maintaining fiscal discipline and delivering essential community services.[1]

1.2 Problem statement

A big problem of manufacturing companies is fixed financial means in warehouses, in materials, and in semi-products; which are lying unused for a prolonged period of time and represent a significant fixation of financial means.

The main goal of this project is to create a robust tool that can be used by data specialists, to evaluate the current situation in stock and make decisions about adding semi-products to the manufacturing chain. Having such an application would tremendously simplify the process of further decision making.

Production planner, which makes decision about what and when products will be produced, decides this based on:

- leadership priority (priority customer, new opportunities)
- current orders (waiting, in process)
- available resources (worker, machine, material)
- estimated needs (warehouse status)
- own experiences

The planner primarily focuses on production and operation, not on warehouse status. This can lead to long-term fixation of the funds in the warehouse and inefficient use of storage capacities.

The goal of the joint project Asseco and CVUT was to provide an executive employee a simplified financial view of the future status of the warehouse. It attempts to develop a Decision Support System that will notify production planners, given a high enough probability, if the semi-finished products will remain in the warehouse for a long time.

We focused on individual production orders. For the semi-finished product we already have historical information on how much money (and how long) it was in the warehouse before and after the end of the production order. Therefore, we can estimate similar parameters for new production orders. If we keep track of the expensive and long-lying semi-finished products, we can assign them a numerical value, which will rate the production order. We called this value a *penalty*, and it represents the value of a semi-product in a warehouse accumulated over time.

The next decision was to separate products into different classes according to their penalty. Products were divided into three classes: with high, low, and medium penalty. Categories are introduced so that there are always the same amount of manufacture orders in each class. A simple semaphore like UI element was created, that colorizes production order with low penalty in green color, products with high penalty in red, and others in yellow.

Chapter 2

Analysis

Following python scripts are available:

- `sql_connect.py`. Connects to database and downloads raw data.
- `prepare_data.py`. Adds statistical data.
- `generate_classifier_description.py`. Generates parameters for classifier.
- `own_classifier.py`. Trains classifier with parameters that gets from previous script.

2.1 Data specialist's workflow

This section describes the expected data specialist's workflow, which is shown in figure 2.1. At the beginning only data from database are available. At the end, a trained and tested classifier can be exported and transferred to IT developers that will use the classifier to evaluate production orders.

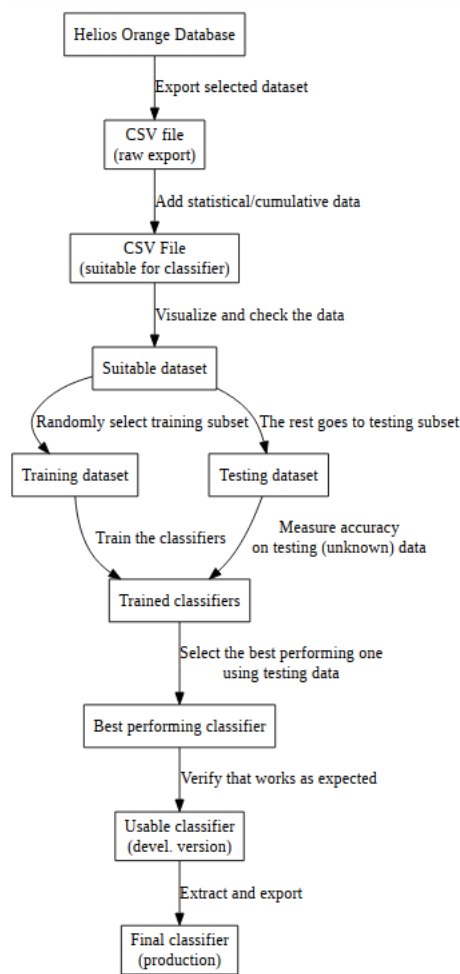


Figure 2.1: Process for data specialist

The first step is to export selected dataset from Helios Orange Database. In this step, it is necessary to download the data from the database to the disk, to CSV file type. For downloading data to the dataset use python script `sql_connect.py`. This script generates following files:

- `manufacture_command.csv`

This file contains raw data from Helios database.

- `material.csv`

This file represents the tree of dependencies of semi-finished products on materials.

- `slozitost.csv`

File contains TAC for materials. TAC is time required to perform an operation on one piece of product in minutes.

- `stock_load.csv`

This file contains information about movements of the semi-finished products in the warehouse

- `stock_price.csv`

The price of semi-product.

The next step is adding statistical/cumulative data. For this purpose `prepare_data.py` script was created. This script expects files from previous step and creates new ones:

- `days.txt`

This file contains a list of days from Helios database. For example, the day in which the new manufacture command was started.

- `manufacture_command_ex.csv`

File `manufacture_command_ex.csv` extends the previous file. Except for columns named as parameters in database, it contains additional calculated columns with statistical information. Rows mean different production orders. List of necessary column names and their description you can find in [Appendix B](#).

- `data_cont.csv`

This file has information about the amount of products in the warehouse. The first row represents dates in Unix time stamp. The first column means ids of materials. Cells shows count of material in a particular day.

The specialist will check on the graphs dependency between different parameters, and will verify distribution of numerical data. The next step is to divide the suitable dataset into two parts, the training dataset and testing dataset. The first part will be used to train classifiers, which will be tested on the second part.

If we go into details, the step of training a classifier is separated into two parts. First, you need to generate parameters for a classifier. For example, four parameters are necessary: `company`, `start_years_train(=start train years)`, `end_years_train(=end train years)`, `start_years_test(=start test years)`, `end_years_test(=end test years)`. Other parameters are optional. Script `experiments/generate_classifier_description.py` takes parameters from the specialist and sets other by default. Then script puts these parameters into specific structure in JSON format. This structure is sent to `own_classifier.py` script, that trains classifier.

The specialist will compare different classifiers and will select the one with best performance using the testing data. Then he verifies that the classifier works as expected. The final step is to extract and export the classifier.

This project is based on Michal's Bouška work. He created python scripts for working with database, and for training and using classifiers. My work was to simplify the process by providing graphical interfaces such as statistical and time series graph. This application will allow person without deep knowledge of training classifier methods to test and use them.

2.2 Requirements

This chapter defines functional and non functional requirements for the application. A functional requirement describes *what* a software application should do, while non-functional requirements place constraints on *how* the application will do so. Almost all requirements I got during the first weeks from Asseco.

2.2.1 Functional requirements

This section is separated into three parts. The first part covers requirements about statistical graphs, the second one contains requirements about time series graph, the last one includes requirements about classifiers.

Statistical graphs

1. Import data

User will be able to upload dataset with .CSV file extension and valid structure. All parameters should be read, but display only necessary ones. The parameters must be human-readable.

2. Colorize products by parameter

User is able to choose parameter for coloring the graph items (such as penalty). Application will re-colorize products taking into account selected parameter. Parameters should be colored by 1/3 principle. That means, that dataset should be sorted and divided into three parts by values. The first third will be colored in green, the second in yellow, the third in red.

3. View dependency graph (Scatter plot)

This graph shows dependency of one parameter on another. Each dot on graph represents an individual product.

- Log/Linear scales

User can switch between linear and logarithmic scales for both x and y axes.

4. View histograms

Application should display two histograms for both x and y parameters.

- Log/Linear scales

User can switch between linear and logarithmic scales.

5. View boxplots

The box plot (a.k.a. box and whisker diagram) is a standardized way of displaying the distribution of data based on the five number summary: minimum, first quartile, median, third quartile, and maximum. Application should display both boxplots for x and y parameters[2].

- Log/Linear scales

User can switch between linear and logarithmic scales.

6. Changing x-y axes
User will be able to choose parameter for x axes and for y axes on scatter graph.
7. Changing dataset
User will be able to choose dataset

Time Series graph

8. View time series graph
Graph should display quantity of material in a stock via line charts.
9. One graph includes several products
User will be able to display two or more materials in a one graph.
10. Choose timespan for time series graph

Classifiers

11. Train classifier
User will be able to train classifier by filling the form.
12. Train different classifiers
User will be able to train different types of classifiers.
13. Show accuracy of classifiers
User is able to see results of selected classifier.
14. Compare different classifiers accuracy in the same graph
Selected classifiers results should be displayed on graph in which x values represent particular classifiers, and y axes displays accuracy from zero to one.
15. Test classifier
There will be two ways to test the classifier. The first is by using already prepared test dataset, the second is by typing parameters manually via web form.
16. Export classifier
User will be able to download selected classifier for further usage.

2.2.2 Non functional requirements

1. The classifier training can be time consuming which means that the application should use an asynchronous approach.
2. Availability
All modern browsers:
 - Chrome 65
 - Edge 16
 - Firefox 59
 - Internet explorer 11
 - Opera 50

Chapter 3

Technology

3.1 Software development

For developing this application a waterfall approach was chosen with some elements of agile approach, such as Scrum.

The waterfall method is a relatively linear sequential design approach in software development. Progress flows largely in one direction through the phases of requirement analysis, system design, implementation, testing, deployment, and maintenance.

In the waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The Waterfall method can be seen in figure 3.1.

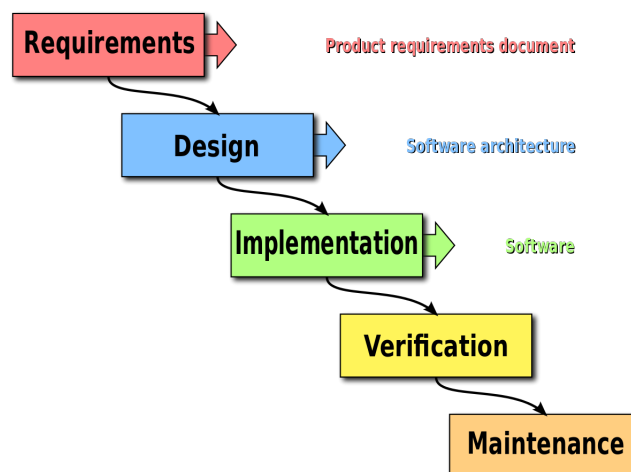


Figure 3.1: Waterfall method

For this project I chose to follow this approach because I have already had main requirements and there won't be unexpected changes during development.

From the agile software development method I get a useful tool named sprint (figure 3.2). A sprint or iteration is the basic unit of development in Scrum. The sprint is a timeboxed

effort, that is, it is restricted to a specific duration. Often sprints are limited to one calendar month. My sprints duration was one week. Each sprint starts with planning event that aims to identify the work for the sprint and make an estimated forecast for the sprint goal. Each sprint ends with a sprint review. List of all sprints see in Appendix C.

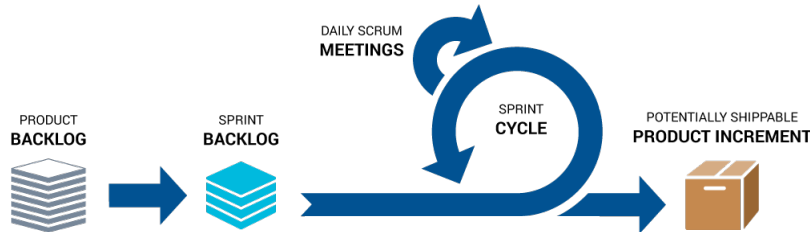


Figure 3.2: Sprint

3.2 Technology review

Description of technology is separated to two chapters. Technology on server side and technology on client side. All used technology is Open Source. This means that the programmer need not pay to use them.

3.2.1 Server-side technology

Language

There are many languages for writing web application, but there is no best language for web applications, because each language is unique and has its own advantages and disadvantages.

- **Java** is an object-oriented programming language. The key benefit of Java is "Write once, run anywhere" (WORA). That means, compiled Java code expected to run on any Java virtual machine (JVM) regardless of computer architecture[3].
- **Python** is an object-oriented programming language. It supports multiple programming paradigms including procedural, functional and imperative. Python has a design philosophy that emphasizes code readability, notably using significant whitespace[4].

To decide what programming language to work in, I look to the IEEE Spectrum Ranking for help; it ranks the popularity of programming languages. Rankings are created by weighing and combining 12 metrics from 10 sources[5]. It takes into consideration Google search, Google trends, Stack Overflow questions, GitHub tags, Hacker News and others. The top 15 languages are shown in figure 3.3.

To choose the best programming language suitable for me I take into account my own experience. During my studies in University, I used Java. Considering that Java was ranked as one of the best languages for web development, the application will be written in Java.

Spring

Spring is a popular application framework for the Java platform. The main reason behind the creation of Spring Framework is to make web development easier and faster. Framework

Language Rank	Types	Spectrum Ranking
1. Python	🌐 🖥️	100.0
2. Java	🌐 📱 🖥️	99.4
3. C#	🌐 📱 🖥️	88.6
4. JavaScript	🌐 📱	85.5
5. PHP	🌐	81.4
6. Go	🌐 🖥️	76.1
7. Ruby	🌐 🖥️	72.4
8. Scala	🌐 📱	68.3
9. HTML	🌐	67.0
10. Perl	🌐 🖥️	57.6
11. Lua	🌐 🖥️	52.7
12. Rust	🌐 🖥️	52.7
13. Processing	🌐 🖥️	49.8
14. D	🌐 🖥️	38.8
15. Clojure	🌐 🖥️	24.6

Figure 3.3: Top 15 development languages, ranked by IEEE Spectrum in 2017[5]

is separated to several modules that provide a range of services such as Spring Core Container, aspect-oriented programming (AOP), authentication, model-view-controller (MVC) and others[6]. The key feature of Spring Core is the Dependency Injection (DI), concrete example of Inversion of Control (IoC). Dependency Injection is a technique that eliminates dependency between components, helps reuse classes and tests these classes separately.

Maven

Maven is tool for building and managing Java-based projects. All configurations are located in the POM file that contains information about the project and configuration details used by Maven to build the project[7]. Maven dynamically downloads libraries from central repository.

Vaadin

Vaadin Framework is a Java UI framework that simplifies web app development. Code is written in Java and executed on the server's JVM, while the UI is rendered as HTML5 in the browser. The framework also automates all the communication between the browser and the server.

Lombok

Project Lombok is also licensed under the MIT license. Lombok is java library that writes all getters/setters, constructor and other features for you.

```
@Data
```

```
public class User implements Serializable {
```

```
    private String firstName;
```

```

private String lastName;
private int age;

public User(String firstName, String lastName, int age) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.age = age;
}
}

```

In this simple example annotation `@Data` will generate getters and setter for all variables: `firstName`, `lastName`, `age`.

3.2.2 Client-side technology

HTML 5

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications[8]. It describes the structure of Web pages by providing tagged elements, building blocks of HTML page.

CSS

Cascading Style Sheets is a language used to style and lay out web pages. For example, CSS used to alter the font, text size, background color and other decorative features.

JavaScript

JavaScript is a high-level, interpreted programming language. It is used to make dynamic webpages interactive, for example providing 2D/3D graphics, which is useful for my application.

JavaScript library for plotting statistical graphs

A lot of JavaScript graphics libraries aid in the development and display of graphic elements like particles, motion, animation, plotting[9]. At the beginning of the development I used **Chart.js** library. It offers basic chart types, like line charts, histograms and time series. However I was required to create more complex charts with additional functionality.

Upon brief research, I found a better library that suits my needs. **Plotly.js** offers a lot of features, any chart can be zoomed in, zoomed out, auto-scaled, it offers more charts, like box plots, that I have to implement in my application.

Adil Baaaj has done research about JavaScript Chart Libraries[10]. He reviewed the presence of 7 types of visualizations:

Basic: line, bar, column, scatter plot, bubble, donut / pie, and area charts

Scientific: contour plot, heat map, ternary plot, polar graph, carpet plot, and parallel coordinates plot

Financial: time series, candlesticks

Statistical: error bars, box plots, 2D density plots, tree map

3D: point cloud, line, ribbon, surface, and mesh plots

Map: choropleth maps, scatter plot on maps, bubble maps, line on maps

	Basic charts	Statistical charts	Scientific charts	Financial charts	3D charts	Map charts
Chart.js						
Highcharts						
C3						
NVD3						
Chartist						
Plotly.js						
Victory						

	At least half of the chart provided from this chart type
	At least one chart provided from this chart type
	No chart provided from this chart type

Figure 3.4: Comparison between JavaScript chart libraries

JavaScript Object Notation (JSON)

To serialize and deserialize graphs, I use JSON format. Reading JSON file is much faster than CSV file.

3.3 Architecture

In this section I describe the architecture of my application. Software application architecture is the process of defining a structured solution that meets all of the technical and operational requirements, while optimizing common quality attributes such as performance, security, and manageability [11].

Client-server

The client-server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and those who demand the service, called clients[12].

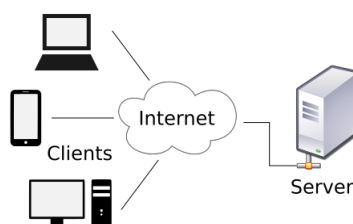


Figure 3.5: Client Server

Thick vs Thin Client

A thin client machine is going to communicate with a central processing server, meaning there is little hardware and software installed on the user's machine. Other words, thin client is a way of saying that all UI logic runs in a web server and the only communication is when the web browser requests a new full HTML page to show.

In contrast, a thick client will provide users with more features, graphics and choices making

the applications more customizable [13]. Thick client can be a JavaScript application running in a web browser, for example. The UI logic is running in the client.

In my application, I use JavaScript library Plotly.js, that enables one to play with graphs and there is thus no necessity to send any request to the server. Vaadin framework combines the strengths of each model.

Model–view–presenter

Model–view–presenter (MVP) is a derivation of the model–view–controller (MVC) architectural pattern, and is used mostly for building user interfaces. Model–view–presenter divides an application into three interconnected parts. Model is responsible for managing the data of the application. The view means presentation of the model in a particular form. The presenter responds to the user input and performs interaction on the data model objects. One big difference about MVC from MVP is that the View does not directly bind to the Model. Figure 3.6 shows differences between MVP and MVC.

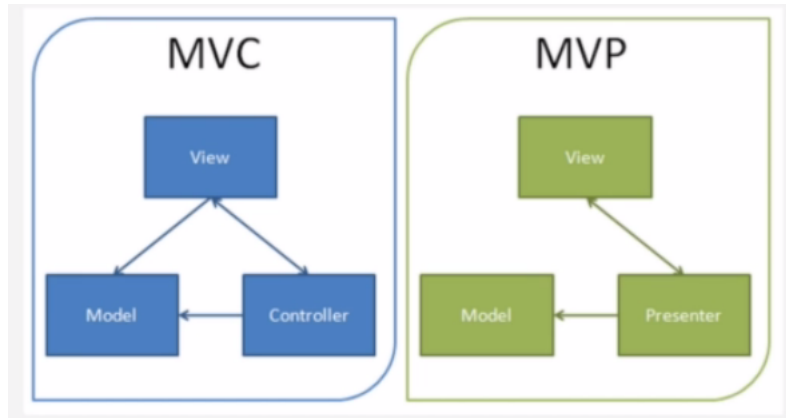


Figure 3.6: Model–view–presenter and Model-View-Presenter

Spring DI

Inversion of Control is a principle in software engineering by which the control of objects or portions of a program is transferred to a container or framework. Dependency injection (DI) is a pattern through which to implement IoC, where the control being inverted is the setting of object’s dependencies.

Spring Web

I use spring web for REST API.

Java Python communication

Follow figure 3.7 shows how Java communicates with Python scripts. Java calls Python scripts via command line, so three pipes are opened: OutputStream in which Java writes, InputStream and ErrorStream from which Java reads.

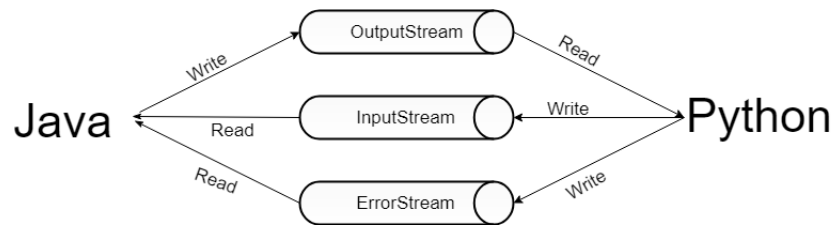


Figure 3.7: Java Python communication

Chapter 4

Design & Implementation

4.1 GUI Design

This section shows a mock-up design of the application. The application contains three pages: "Graph", "Train classifier", and "Use classifier". Description of each tab clarified below. All tabs have a navigation bar on top, that allows navigation between pages.

1. Graph page

This page contains two tabs, for statistical graphs and for time series graph. By default, the statistical graph tab is displayed. The page has a *select* element for choosing a dataset and an "Add" button for adding a new dataset. These two elements meet the requirements 1 and 7.

- Figure 4.1 displays the first page "Graph" in which the Graph tab is opened. The tab contains two lists for selecting x and y parameters for axes (see requirement 6). Two checkboxes for each axis are placed to the right from each list and serve to toggle between linear/logarithmic scales. Four graphs situated in this tab are: Scatter plot (requirement 3), two Histograms for x and y parameters (requirement 4) and a Boxplot (requirement 5). The select element for choosing the parameter for coloring the graph items is placed above two lists (requirement 2).

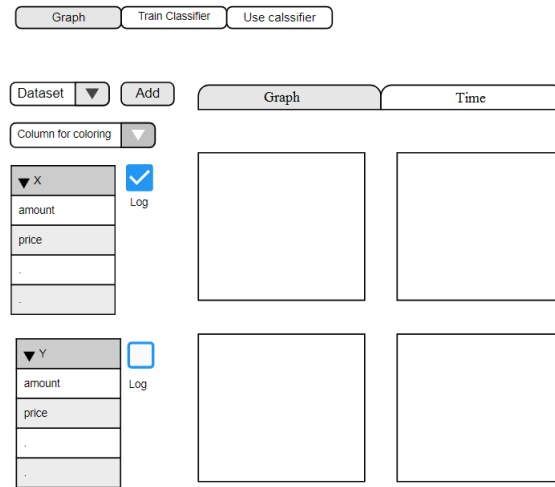


Figure 4.1: Graph

- Figure 4.2 displays the second tab "Time" for time series graph. In this tab, two elements are present: a list for selecting id of products and a time series graph (requirement 10). The list enables one to select several products and then a graph is able to display them (requirement 9).

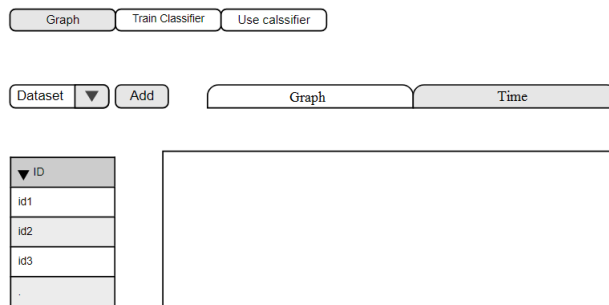


Figure 4.2: Time Series

2. Train one classifier (or all classifiers) page

This page is responsible for training classifiers (see figure 4.3). Page contains form that includes parameters for training classifiers and a button "Train" (requirement 11). After clicking that button, using the data from the submitted form, a dataset for training and testing is constructed and a classifier is trained. Button "Train all" will train all classifiers in the list (satisfies requirement 12). Trained classifiers will appear in the list right of the form. The "Show" button is responsible for displaying accuracies of selected classifiers on the graph (requirements 13, 14). By selecting a classifier or clicking on a classifier on the graph, a link appears under the graph that enables you to download the classifier (requirement 16).

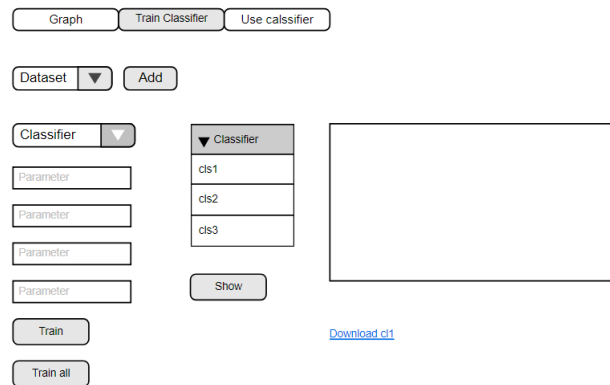


Figure 4.3: Train classifier

3. Use selected classifier page

On this page it is possible to test the classifier in two ways (requirement 15). The first way is by using an already prepared dataset. For this purpose, the page contains a grid with a test dataset. The second way is by typing parameters manually via the form. The form is placed to the right of the grid. There is a select element under the grid for choosing the type of classifier. After clicking the "Evaluate" button, your result will appear under the button and is colored in green, yellow or red according to its penalty value. The history of previous results is kept.

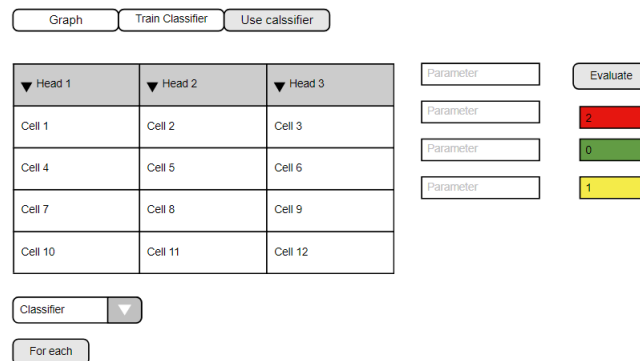


Figure 4.4: Use classifier

4.2 Design

Class diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects[14]. The application has three layers: View, Service API, Database API (figure 4.5).

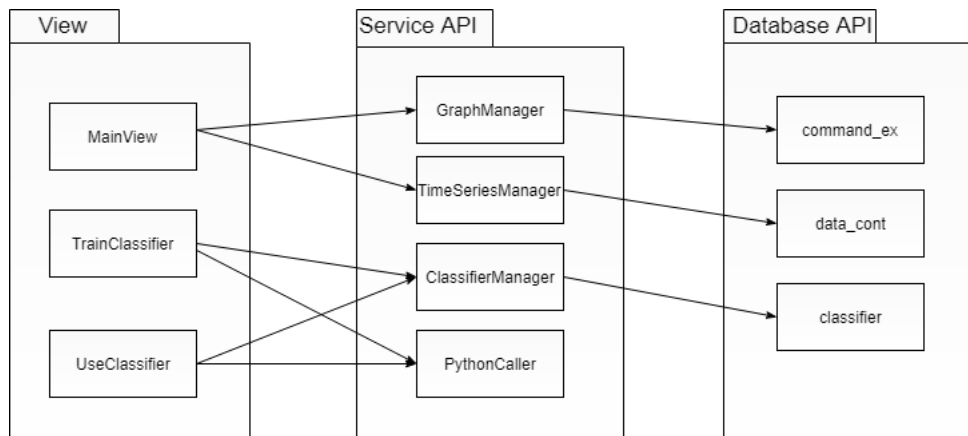


Figure 4.5: Class diagram

Package "Service API" represents the Service layer and it's responsible for retrieving and saving datasets and classifiers. The package contains four classes: GraphManager, TimeSeriesManager, ClassifierManager, PythonCaller.

- GraphManager responsibilities include: to serialize/deserialize datasets, provide list of all available datasets.
- TimeSeriesManager provides the same methods but with time series files.
- ClassifierManager is responsible for import/export of classifiers.
- The goal of PythonCaller class is to interact with python scripts, for example, train, test, and get accuracy of classifier.

Package "View" is responsible for querying the model and updating the view, reacting to user interactions updating the model. Package contains three pages: MainView, TrainClassifier and UseClassifier.

- Main page includes two tabs: for statistical graphs and for time series graph. Main page in first tab calls GraphManager to get list of datasets and to get data from particular dataset. These data it uses to display four graphs. Page calls TimeSeriesManager in second tab to get list of datasets and to get data for time series graph.
- Page TrainClassifier is responsible to train, retrieve and show accuracy of classifiers. It calls ClassifierManager to get list of all classifiers, also, it sends parameters to Python-Caller to train the classifier.
- The main goal of UseClassifier page is to use the classifier on selected command orders. ClassifierManager is called to get list of available classifiers, PythonCaller is called to evaluate command orders.

Package "Database API" provides data to Service layer. It contains three folders: "command_ex" for statistical graphs, "data_cont" for time series graph, and "classifier" for classifiers.

Use cases

The following graphs show use cases diagrams. All use cases presented satisfy the specialist's workflow steps (section 2.1).

Figure 4.6 shows use cases related to statistical graphs. This graph covers requirements 1, 2, 3, 4, 5, 6, 7. These use cases will be implemented in the first page of the application in the tab "Graph".



Figure 4.6: Use cases related to statistical graphs include importing data, selecting parameter for coloring data, displaying four types of graphs, toggling between linear/logarithmic scales and selecting dataset

Figure 4.7 shows use cases related to time series graph. This graph covers requirements 8, 9 and 10. These use cases will be implemented on the first page in the tab "Graph".

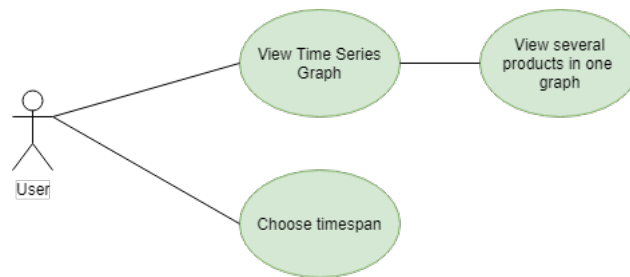


Figure 4.7: Use cases for viewing one or several products on the time series graph and selecting timespan for graph

Figure 4.8 shows use cases related to classifiers. This graph covers requirements 11, 12, 13, 14, 15 and 16. Requirement 15 will be implemented on the page "Use classifier", others will be implemented on the page "Train classifier".

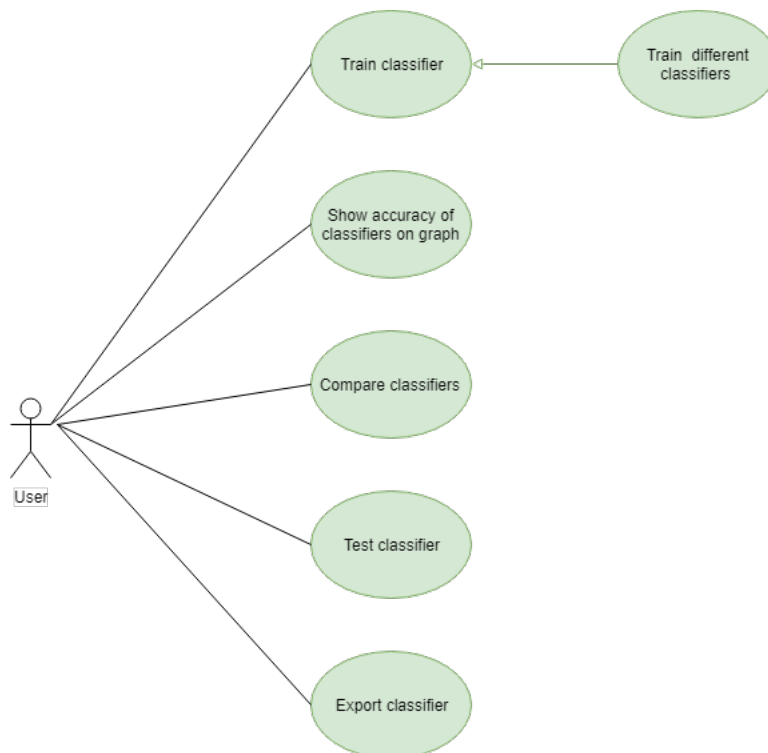


Figure 4.8: Use cases related to classifier include training one or several classifiers, displaying their accuracy, testing and export of a selected classifier

4.3 Components

This section is structured in the same way as the Analysis section. The first part covers requirements related to statistical graphs, the second includes requirements for times series graphs, and the last one contains requirements related to classifiers.

Statistical graphs

1. Import data for statistical graphs component

The application should provide an opportunity to upload a new dataset. To add a new dataset, user should click the button "Add new", then modal window then appears (figure 4.9). Example name is "hudba_manufacutre_command_ex.csv"

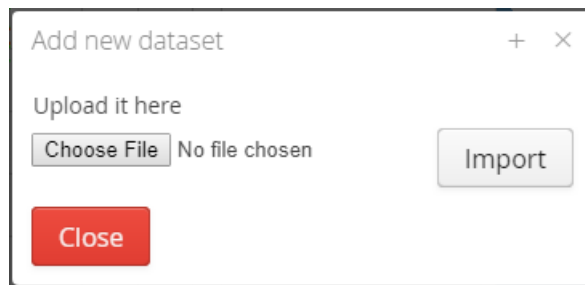


Figure 4.9: Upload new dataset for statistical graphs

2. Colorize products by parameter component

User can select parameter for coloring the x-y plot. The most relevant parameter is penalty. But the user can select any other. This adds a third parameter that can be visualized.

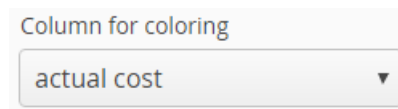


Figure 4.10: Coloring by selected parameter

3. View x-y graph (Scatter plot) component

Figure 4.13 shows scatter plot with both x and y axes linear. Logarithmic axes can be toggled for any axis.

- Linear/logarithmic scale

Changing scale type is implemented via check boxes (figure 4.11 and figure 4.12)



Figure 4.11: X logarithmic



Figure 4.12: Y logarithmic

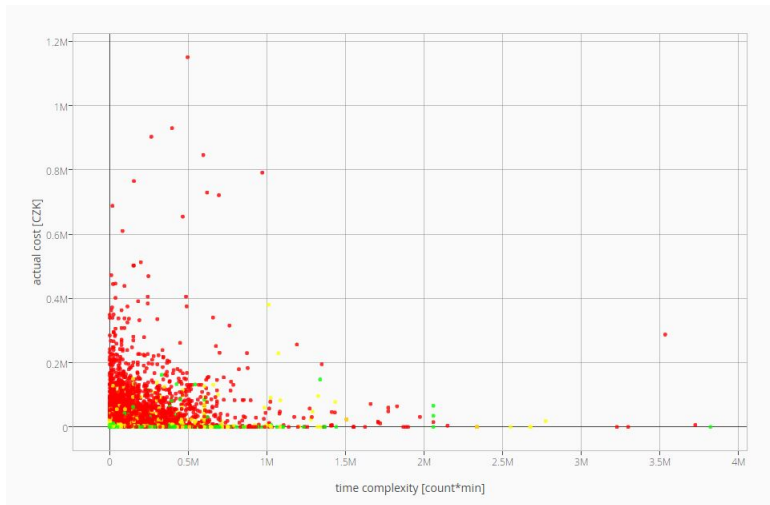


Figure 4.13: Scatter plot with linear scales

Figure 4.14 shows same parameters as figure 4.13, but using a logarithmic scale.

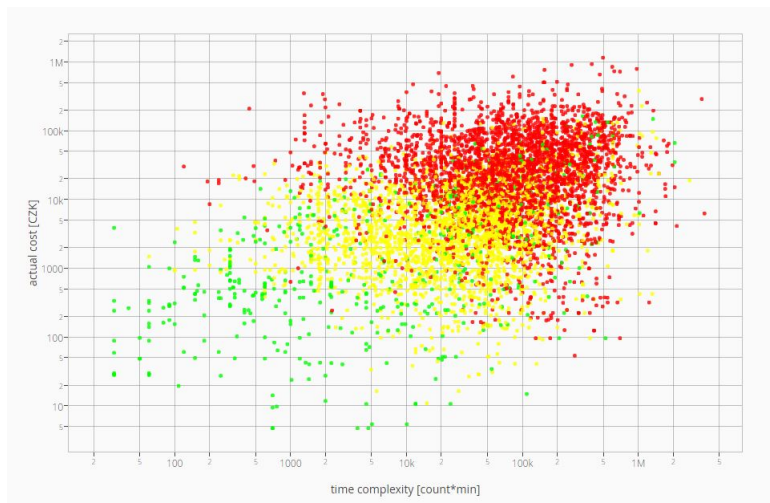


Figure 4.14: Scatter plot with logarithmic scales

4. View histograms component

For histogram graphs, the same logarithmic scale is available. On mouse-over, x and y values appear on the graph nearby the cursor (figure 4.15).

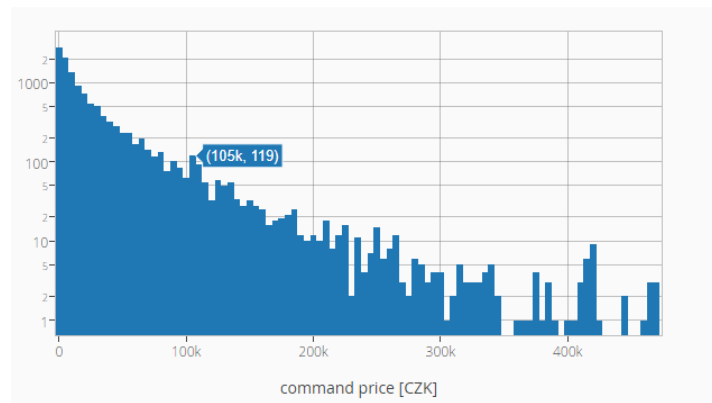


Figure 4.15: Histogram shows distribution of command price. Logarithmic y axis was used.

5. View boxplots component

Boxplots can also have logarithmic scale. JavaScript library Plotly.js provides this feature. The box plot is a standardized way of displaying the distribution of data based on the five number summary: minimum, first quartile, median, third quartile, and maximum [2]. On mouse-over these five values appear (figure 4.16).

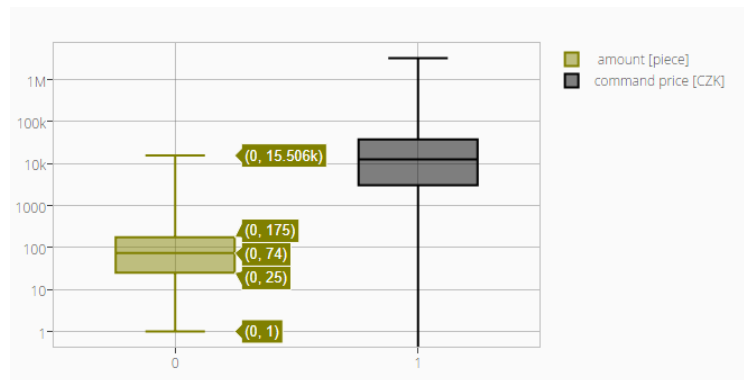


Figure 4.16: Boxplots

6. Changing x-y axes component

For changing the sources for x and y axes, two lists were created. Names of parameters were mapped to human-readable names. For example, `amount_clear` parameter in database was transformed into "amount clear". All transformation together with description of the parameters can be seen in Appendix B.

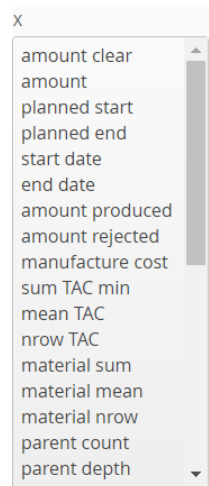


Figure 4.17: List for x axis

7. Changing dataset component

For changing dataset, the responsible element is select, from Vaadin framework.

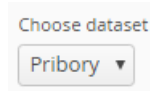


Figure 4.18: Choosing dataset element

Time Series graphs

8. View time series graph component

JavaScript library provides its own time series graph. X axis represents date. Y axis represents amount in warehouse (pieces). Figure 4.19 shows time series for a single product. Product id is displayed on the left. For anonymous reasons we cannot display product name. But, it is available in Helios database and data specialist can find it here.

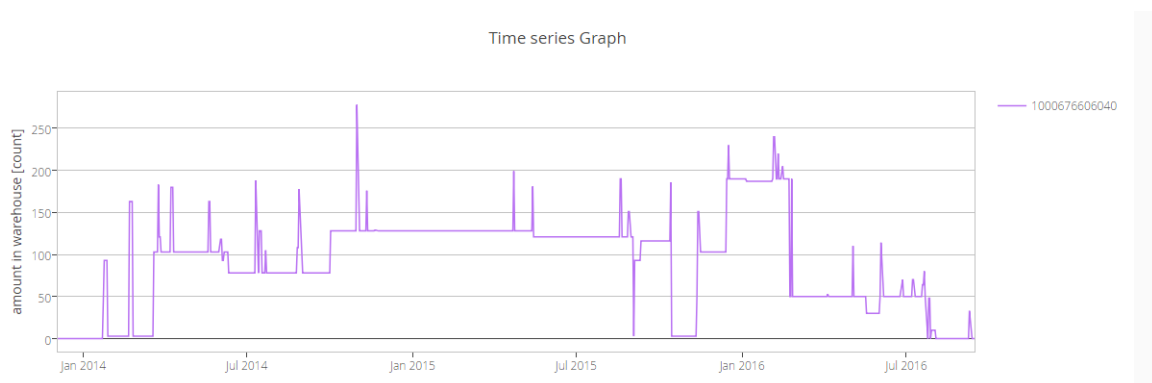


Figure 4.19: Time series for a single product

9. View time series graph (multiply products) component

Figure 4.20 shows several products in one graph. Ids of the product are displayed on the right side from the graph. As mentioned before, we do not display product name.

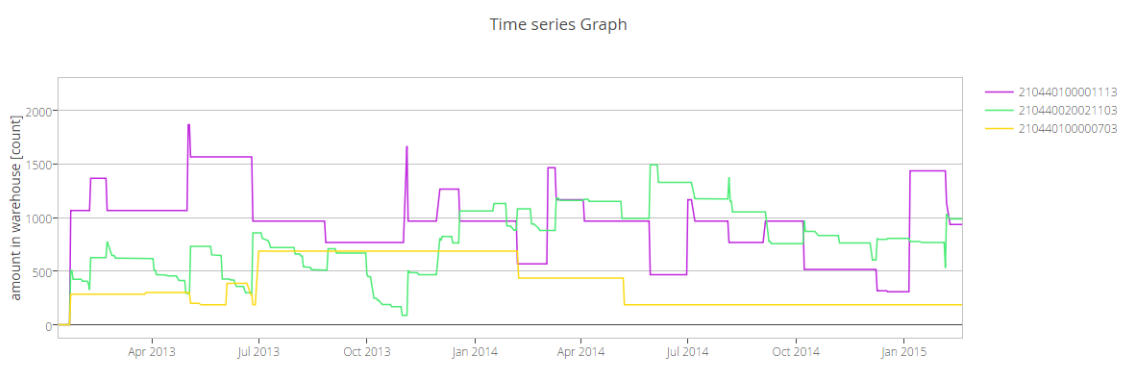


Figure 4.20: Several products in one graph

10. Time series graph: choosing timespan

This functionality is implemented using Plotly.js library (figure 4.21).

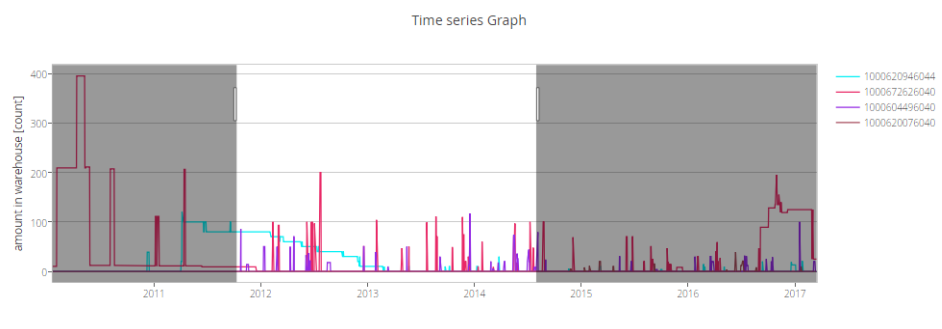
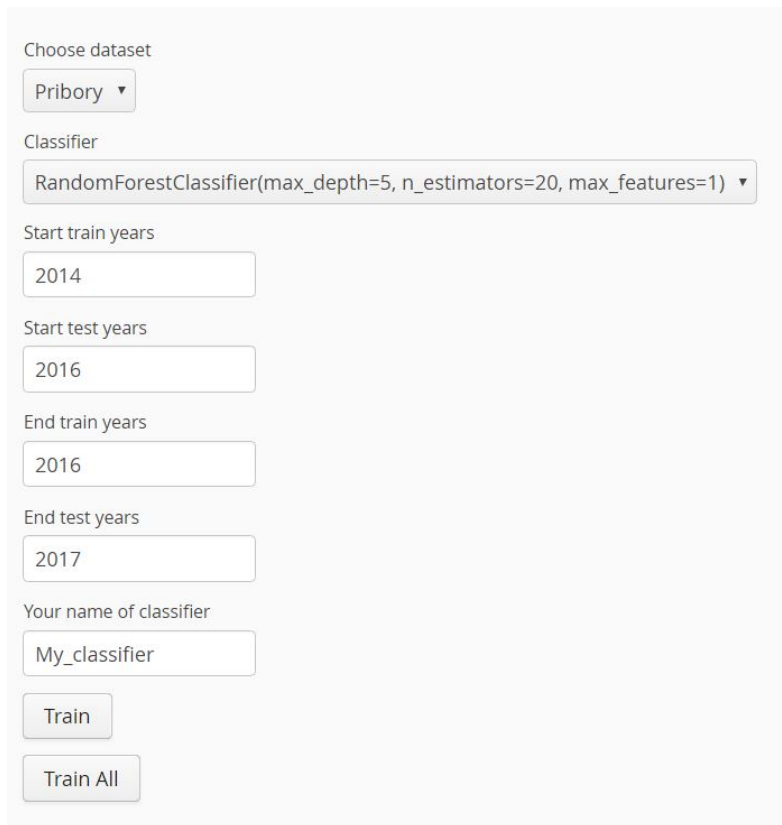


Figure 4.21: Timespan selection in time series graph component

Classifiers

11. Train classifier component

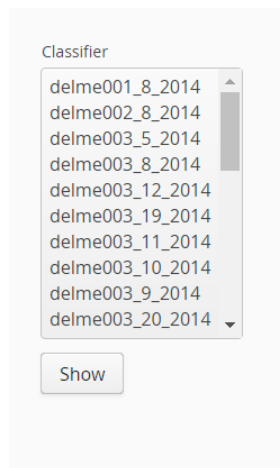
User should fill the form (see figure 4.22) and click button "Train". After clicking, using the data from the submitted form, a dataset for training and testing is constructed and a classifier is trained. Trained classifier appears in the classifiers list (figure 4.23).



The form for training a classifier includes the following fields and buttons:

- Choose dataset:** A dropdown menu with "Pribory" selected.
- Classifier:** A dropdown menu with "RandomForestClassifier(max_depth=5, n_estimators=20, max_features=1)" selected.
- Start train years:** A text input field containing "2014".
- Start test years:** A text input field containing "2016".
- End train years:** A text input field containing "2016".
- End test years:** A text input field containing "2017".
- Your name of classifier:** A text input field containing "My_classifier".
- Buttons:** "Train" and "Train All".

Figure 4.22: Form for training classifier



The list of trained classifiers is displayed in a scrollable area with the following entries:

- delme001_8_2014
- delme002_8_2014
- delme003_5_2014
- delme003_8_2014
- delme003_12_2014
- delme003_19_2014
- delme003_11_2014
- delme003_10_2014
- delme003_9_2014
- delme003_20_2014

A "Show" button is located below the list.

Figure 4.23: List of trained classifiers

12. Train different classifiers

Button "Train All" was created for this purpose. (see figure 4.22). Application creates 23 pipes that run the `own_classifier.py` script simultaneously and starts to train mul-

tively different classifiers. The same as in previous point, trained classifiers appear in the classifiers list.

13. Show accuracy of classifier component

By selecting one or several classifiers in the classifier list (figure 4.23) and clicking the "Show" button, results are displayed in the graph (see figure 4.24)

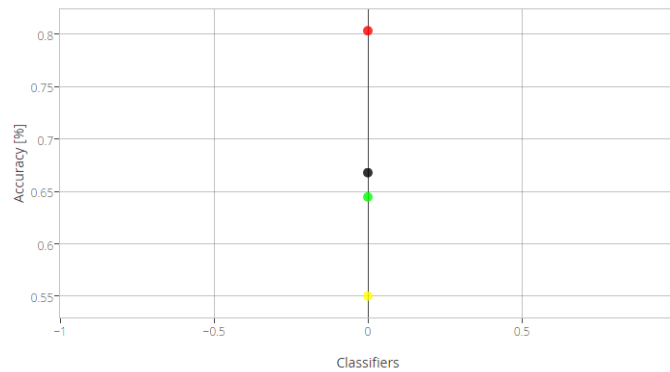


Figure 4.24: Classifier results

14. Compare different classifiers accuracy in the same graph

My solution was to create scatter graph with accuracy for various classes (red, yellow, green and overall), (see figure 4.25). X axis represents particular classifiers, Y axis represents accuracy from zero to one.

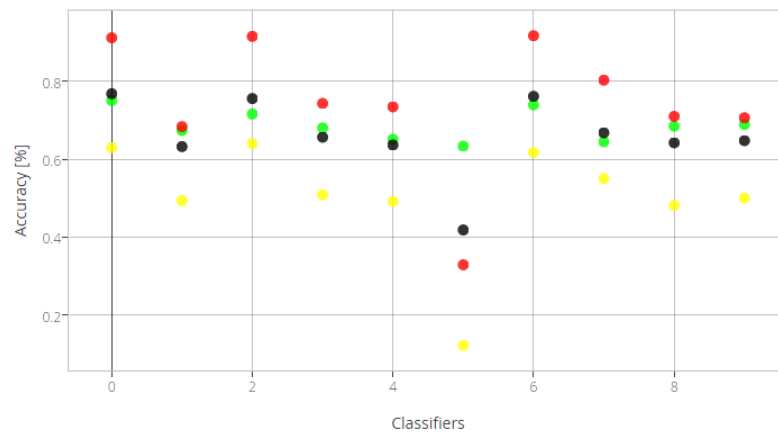


Figure 4.25: Classifiers results

15. Test classifier

There are two ways of testing the classifier. The first way is by using already prepared dataset. For this purpose, a grid view is implemented. Rows represent particular command orders. Columns represent order parameters needed for the classifier. The

second way is by typing parameters manually via form. The form is placed to the right of the grid. More information you can find in Chapter 4.1. It is also possible to test all generated classifiers on a particular product by clicking "For each" button (figure 4.26).

The screenshot shows a web interface for testing classifiers. On the left is a table with columns: command_price, start_month, mrow_TAC, sum_TAC, material_sum, material_mrow, parent_places.o., parent_places.o., parent_places.o., parent_places.o., parent_depth, parent_count, and end_class_price. The table contains multiple rows of data, with one row highlighted in blue. Below the table is a 'For Each' button and a dropdown menu for selecting a classifier, currently set to 'classifier_0_2014'. On the right is a form with input fields for various parameters: command_price (99388), start_month (6), mrow_TAC (19), sum_TAC (56040), material_sum (5000), material_mrow (5.0), parent_places.o. (0.0), parent_places.o. (0.0), parent_places.o. (0.0), parent_places.o. (0.0), parent_depth (2.0), parent_count (0.0), end_class_price (134195.2499), and an 'Evaluate' button. Below the form, a list of classifiers is shown with their respective values and status indicators (e.g., 'classifier_0_2014' is highlighted in red).

Figure 4.26: Use classifier page

16. Export classifier

After selecting a classifier, a link appears under the graph. By clicking this link, user is able to download the classifier (figure 4.27) in pickle format.

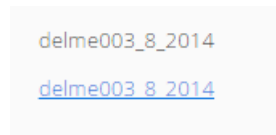


Figure 4.27: Download classifier

Chapter 5

Testing

This chapter describes the steps that were taken to evaluate the quality of the application.

5.1 Use cases

The first step was to go through all functional requirements, validate each use case, and observe expected results. Table 5.1 shows that all functional requirements related to statistical graphs, time series graph and classifiers were successfully implemented.

Table 5.1: Functional requirements implementation

Functional requirement	Implemented	Tested
Import data	✓	✓
Colorize products by parameter	✓	✓
View dependency graph (Scatter plot)	✓	✓
View histogram	✓	✓
View boxplots	✓	✓
Changing x-y axes	✓	✓
Changing dataset	✓	✓
View time series graph	✓	✓
Time series graph includes several products	✓	✓
Choose timespan for time series graph	✓	✓
Train classifier	✓	✓
Train different classifiers	✓	✓
Show accuracy of classifiers	✓	✓
Compare different classifiers accuracy in the same graph	✓	✓
Test classifiers	✓	✓
Export classifiers	✓	✓

5.2 Code analysis

Static code analysis was made with FindBugs plugin for IntelliJ IDEA. It finds bug patterns, such as null pointer dereferences, infinite recursive loops and so on. After analysis of the project files, analyzer did not find any critical bugs. Minor bugs were reviewed and fixed.

5.3 Unit tests

Unit testing is a level of software testing where individual units/components of a software are tested. It is performed by using the White Box Testing method in which the internal structure/design/implementation of the item being tested is known to tester. To test software JUnit was used. JUnit is a unit testing framework for the Java programming language [15]. The following tests were done:

- Add new dataset/ time series file
Add new dataset, and make sure, that new dataset was added, that GraphManager(or TimeSeriesManager) refreshes list of existing datasets.
- Serialize datasets/time series graph
Test converting dataset from CSV format to JSON format. This test trying to convert files in appropriate format to another format.

All unit test are passing with no errors.

5.4 Cross browser testing

The purpose of CrossBrowserTesting is to perform cross-browser testing on web applications and mobile applications. The following screenshots of the main page were made from different browsers. Functionality was preserved, small differences were in rendering without any impact on functionality nor usability of the application.

- Page rendered in Chrome browser version 65 (figure 5.1). It provides all functionality.

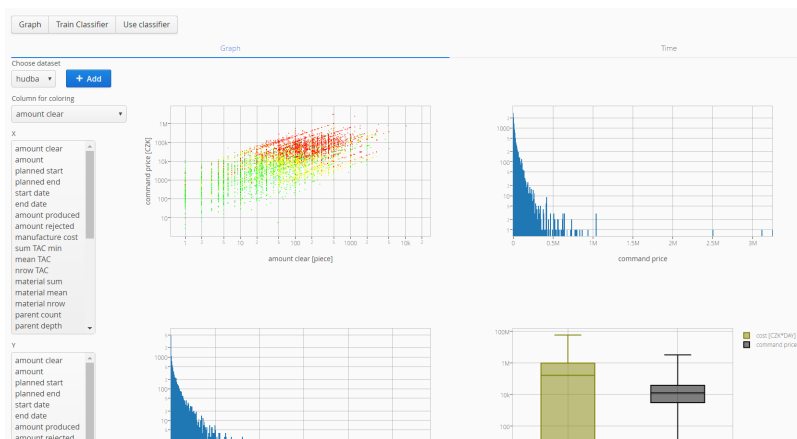


Figure 5.1: Rendered main page of application in Chrome browser

- Page rendered in Edge browser version 16 (figure 5.2). It provides all functionality.

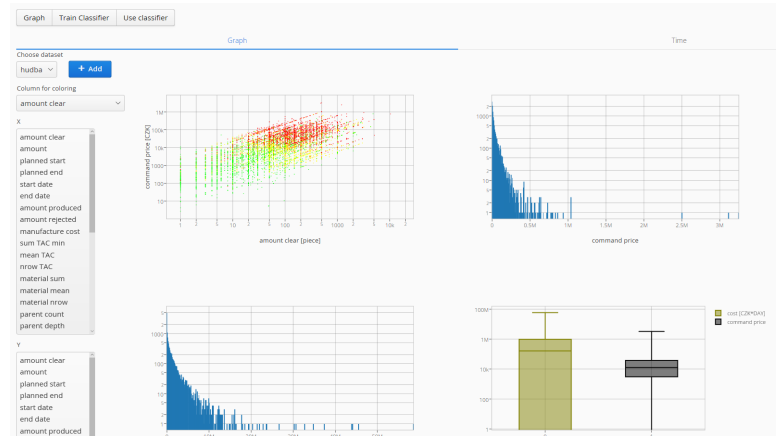


Figure 5.2: Rendered main page of application in Edge

- Page rendered in Firefox browser version 59 (figure 5.3). It provides all functionality.

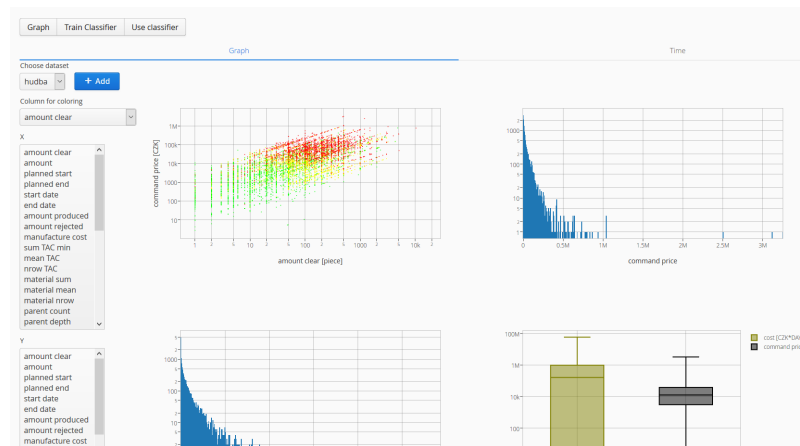


Figure 5.3: Rendered main page of application in Firefox browser

- Page rendered in Internet explorer browser version 11 (figure 5.4). It provides all functionality.

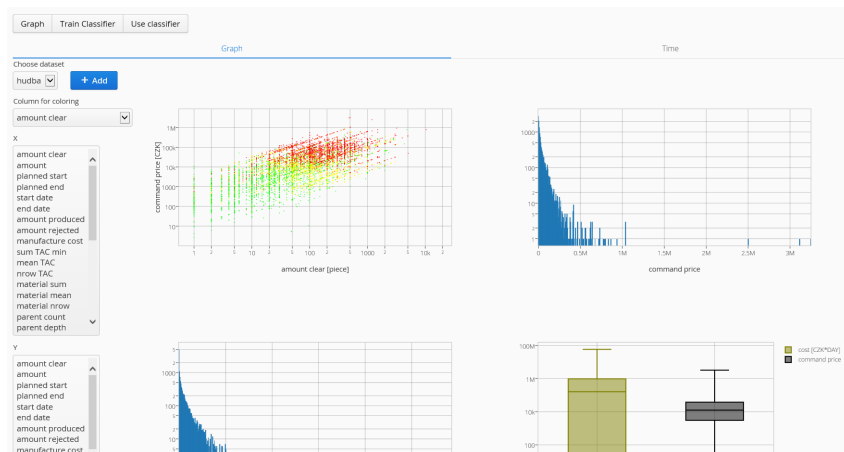


Figure 5.4: Rendered main page of application in Internet explorer browser

- Page rendered in Opera browser version 50 (figure 5.5). It provides all functionality.



Figure 5.5: Rendered main page of application in Opera browser

5.5 Conclusion

All test were passed and application is ready to use. We have successfully implemented and verified the functionality of all the functional requirements 1 to 16 (see section 2.2) and all non-functional requirements (1, 2) (see section 2.2).

Chapter 6

Conclusion

In my thesis I've implemented a web app for data specialist that can import a dataset, check the validity (suitability) of the dataset using various visualizations. If the data is suitable, he can train multiple classifiers for prediction, compare their performance and finally export selected classifier for using in production.

The following goals were achieved:

- Get acquainted with ERP systems and provided DB export.
I have understood the aim of enterprise resource planning. I was able to import the provided CSV files and use the data in my application.
- Visualize the data (x-y plots, time restriction, histograms).
I have implemented data visualization using Plotly.js library. All required types of graphs are available and working.
- Implement various classification and prediction methods for selected timespan.
Classifiers from the previous work were integrated into the application and GUI was implemented to ease work. Timespan selection was also implemented using a form, where user can choose the required timespan.
- Test the solution.
The application was tested. It is working (online), all required functionality has been tested to work. Information about testing can be found in chapter 5.
- Visualize the classifiers results.
Visualization of classifier results was made with the same JavaScript library. The graph component provides opportunity to compare different classifiers and select one for export and further usage.
- Unit test and project documentation.
Unit tests and documentation were written. All tests were passed. User guide with installation instructions was also provided (available in readme.md file). Installation to Docker image is possible using provided Dockerfile.

The application was presented to Asseco Solutions, Inc. company with positive feedback.

6.1 Further work

The application is ready to use, in the future, some improvements can be done.

- User Interface could be more user friendly and more interactive. This would need more testing with users (UX tests).
- Better support for mobile platforms could be provided. In time of developing, only x86 platform was used.
- Classifier training performance can be improved, especially with python scripts. Classifiers could be trained in parallel on grid nodes. It should be faster than training classifiers on local PC.
- Verification by data specialist is to be performed. Data specialist was kept aside to have no inner information. We will describe the problem to him and observe his understanding of the "unknown" application.

Bibliography

- [1] Government Finance Officers Association. *Financial Forecasting*. URL: <http://gfoa.org/financial-forecasting-budget-preparation-process>.
- [2] Roald Hoffmann. *Box Plot: Display of Distribution*. URL: <http://www.physics.csbsju.edu/stats/box2.html>.
- [3] Herbert Schildt. *Java The Complete Reference*. 2015.
- [4] David Ascher and Mark Lutz. *Learning Python*. 1999.
- [5] Nick Diakopoulos and Stephen Cass. *IEEE Spectrum Ranking*. 2017. URL: <https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2017>.
- [6] Ashish Sarin. *Getting started with Spring Framework*. 2016.
- [7] The Apache Software Foundation. *Introduction to the POM*. 2018. URL: <http://maven.apache.org/guides/introduction/introduction-to-the-pom.html>.
- [8] Jon Duckett. *HTML and CSS: Design and Build Websites*. 2011.
- [9] Jon Duckett. *JavaScript & JQuery*. 2014.
- [10] Adil Baa]. *Compare the Best Javascript Chart Libraries*. URL: <https://blog.sicara.com/compare-best-javascript-chart-libraries-2017-89fbe8cb112d>.
- [11] MSDN. *What is Software Architecture?* URL: <https://msdn.microsoft.com/en-us/library/ee658098.aspx>.
- [12] Wikipedia contributors. *Client-server model* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 7-May-2018]. 2018. URL: https://en.wikipedia.org/w/index.php?title=Client%E2%80%93server_model&oldid=837529203.
- [13] Vangie Beal. *The Differences Between Thick, Thin and Smart Clients*. URL: https://www.webopedia.com/DidYouKnow/Hardware_Software/thin_client_applications.asp.
- [14] Allen I. Holub. *Allen Holub's UML Quick Reference*. 2017.
- [15] Wikipedia contributors. *JUnit* — *Wikipedia, The Free Encyclopedia*. [Online; accessed 7-May-2018]. 2018. URL: <https://en.wikipedia.org/w/index.php?title=JUnit&oldid=832731550>.

Appendix A

User Guide

You should go through these steps to use the application. All steps are required except those that are marked as optional.

A.1 Requirements:

1. Download or clone project from <https://gitlab.com/eykhmvic/eClubAsseco>. Let's call installed folder MAIN.
2. Be sure that you install following libraries in python
 - scipy
 - graphviz
 - jupyter
 - matplotlib
 - numpy
 - pandas
 - pymongo
 - sacred
 - scikit-learn
3. Change PYTHON_PATH in MAIN/src/main/java/my/asseco/Tools/Constants to your python path.
4. Change PYTHONPATH to MAIN/pythonModule/pythonDir
5. (Optional) Put data in MAIN/pythonModule/pythonDir/manufacture_command_older. For example, "hudba_manufacture_command_ex.csv".
6. (Optional) Put data in MAIN/pythonModule/pythonDir/data_cont folder. For example, "pribory_data_cont.csv"

A.2 Running

Running the project from command line

Run follow command in the project root directory:

```
mvn clean install spring-boot:run
```

Running the project from IDE

Navigate to the Main class and run it as a Java application.

After the server has started, point your browser to <http://localhost:8080> to see the resulting application

Appendix B

Table of parameters

Table B.1: Human readable names

Name	Human readable name
amount_clear	amount clear
amount	amount
planed_start	planned start
planed_end	planned end
start_date	start date
end_date	end date
amount_produced	amount produced
amount_rejected	amount rejected
manufacture_cost	manufacture cost
sum_TAC	sum TAC min
mean_TAC	mean TAC
nrow_TAC	nrow TAC
material_sum	material sum
material_mean	material mean
material_nrow	material nrow
parent_count	parent count
parent_depth	parent depth
parent_pieces_order_count	parent pieces order count
ongoing_same_pieces	ongoing same pieces
ongoing_same_command	ongoing same command
ongoing_all_command	ongoing all command
ongoing_all_pieces	ongoing all pieces
time_complexity	time complexity
command_price	command price
actual_pieces	actual pieces
actual_cost	actual cost
cost_sin_2_6_60	cost [CZK*DAY]
penalty	penalty

end_stock_price	end stock price
start_month	start month
cost_penalty_60	cost penalty

Table B.2: Description in EN

Name	Units	Description
amount_clear	piece	pieces clean (after removal of defective pieces)
amount	piece	the quantity required on the manufacturing order
planned_start		Planned date of start of manufacture command
planned_end		Planned date of end of manufacture command
start_date		Actual date of start of manufacture command
end_date		Actual date of end of manufacture command
amount_produced	piece	Amount produced
amount_rejected	piece	Amount rejected
manufacture_cost	CZK	Unit price
sum_TAC	min	Amount of TAC needed to create the product
mean_TAC	min	Average TAC needed to create the product
nrow_TAC	count	Number of TAC entries needed to create the product (number of steps)
material_sum	CZK	Sum of the materials needed to create the product
material_mean	piece	Average amount of materials needed to create the product
material_nrow	count	Number of materials needed to create the product
parent_count	count	Number of unique parents in the semi-product tree
parent_depth	count	Maximum depth of parents in the semi-product tree
parent_pieces_order_count	count	Number of parent orders at start_date
ongoing_same_pieces	piece	Sum of products on manufacturing order from ongoing_same_command
ongoing_same_command	count	Number of running manufacturing order on the day of the production order with the same product
ongoing_all_command	count	Number of running manufacturing order on the day of the production order entry
ongoing_all_pieces	piece	Sum of pieces from ongoing_all_command
time_complexity	count*min	Time complexity (amount*sum_TAC)
command_price	CZK	Order price (amount * manufacture_cost)
actual_pieces	piece	Actual number of semi-products (of the same type) in stock at the day of the production order entry
actual_cost	CZK	Actual price
penalty	count(0-60)	Penalty from 0 to 60
end_stock_price	CZK	Reflects the target money, that are blocked in the storage after

		finishing the manufacture order
start_month	month	Month from date of assignment
cost_penalty_60	count	Penalty

Appendix C

Sprints

1. Meet Asseco. Get data from Michal. Import the CSV ('extended')
2. Prepare first 'Hello World' with simple (fake) graph
3. Put online first online demo (histograms for 'pibory' dataset).
4. Introduce 'scatter plots' (X-Y) with semaphore coloring (by penalty)
5. Visualize time series (from '_cont' CSV file)
6. Include other companies dataset(s)
7. GUI improvements, usability; Choose the correct way to implement classifier
8. Implement classifier (Random Forest, WEKA, use Michal's scripts (Python, BASH))
9. Visualize experiment results
10. Selection and export of the 3 (BSF) classifiers
11. Unit testing, Documentation
12. Final thesis document,UML models

Appendix D

List of abbreviations

- AOP** Aspect-oriented programming
- API** Application Programming Interface
- CSS** Cascading Style Sheets
- CSV** A comma separated values file
- DI** Dependency Injection
- ERP** Enterprise Resource Planning
- GUI** Graphical user interface
- HTML** Hypertext Markup Language
- IDE** Integrated Development Environment
- IEEE** Institute of Electrical and Electronics Engineers
- IoC** Inversion of Control
- JS** JavaScript
- JSON** JavaScript Object Notation
- JVM** Java Virtual Machine
- MIT** Massachusetts Institute of Technology
- MVC** Model-View-Controller
- MVP** Model-View-Presenter
- POM** Project Object Model
- UI** User Interface
- UML** The Unified Modeling Language
- WORA** Write once, run anywhere

Appendix E

Content of CD

- code (Contains source of the code)
 - README.md
 - LICENSE
 - assecoApp
 - JavaDoc
- thesis
 - source (Contains source of the thesis)
 - BP_Eykhmann_Victoria.pdf