Master Thesis Project

A System of Automated Web Service Selection

*Author:* Oleksandr Malyutin *Supervisor:* Prof. Welf Löwe *Reader:* Dr. Mauro Caporuscio *Examiner:* Prof. Andreas Kerren *Semester:* VT 2016

*Course Code:* 5DV00E

*Subject:* Computer Science

Abstract

In the modern world, service oriented applications are becoming more and more popular from year to year. To remain competitive, these Web services must provide the high level of quality. From another perspective, the end user is interested in getting the service, which fits the user's requirements the best: for limited resources get the service with the best available quality. In this work, the model for automated service selection was presented to solve this problem. The main focus of this work was to provide high accuracy of this model during the prediction of Web service’s response time. Therefore, several machine learning algorithms were selected and used in the model as well as several experiments were conducted and their results were evaluated and analysed to select one machine learning algorithm, which coped best with the defined task. This machine learning algorithm was used in final version of the model.

As a result, the selection model was implemented, whose accuracy was around 80% while selecting only one Web service as a best from the list of available. Moreover, one strategy for measuring accuracy has also been developed, the main idea of which is the following: not one but several Web services, the difference in the response time of which does not exceed the boundary value, can be considered as optimal ones. According to this strategy, the maximum accuracy of selecting the best Web service was about 89%. In addition, a strategy for selecting the best Web service from the end- user side was developed to evaluate the performance of implemented model.

Finally, it should also be mentioned that with the help of specific tool the input data for the experiments was generated, which allowed not only generating different input datasets without huge time consumption but also using the input data with the different type (linear, periodic) for experiments.

**Keywords:** Web service, service selection, service recommendation, decision making, machine learning.

Preface

I would like to thank all people who supported me during my work on this master thesis. Without their help and effort, it would be impossible for me to complete my studies at Linnaeus University and especially this master thesis. I want to thank:

* My parents – they provided me an excellent opportunity to come to Sweden and get this valuable high-level education. They, as well as Yuliia Cherbadzhy, supported me along all my studies, persuading me, that I have enough intelligence and skills to complete my studies and this master thesis successfully.
* Jonas Lundberg, Vira Liubchenko and all people, who participate in the creation and continuous improvement of student exchange program between Linnaeus University in Sweden and Odessa National Polytechnic University in Ukraine.
* Andreas Heberle – he provided a valuable opportunity to participate in Baden-Württemberg Scholarship program and write my master thesis at the University of Applied Science in Karlsruhe, in Germany. He helped, along with Jens Kirchner and Welf Löwe, by giving advice in difficult situations, offering new ideas and evaluating the existing ones.

Contents

1. Introduction 1
   1. Background 1
   2. Previous Research 2
   3. Problem Formulation 2
   4. Motivation 3
   5. Research Question 4
   6. Scope/Limitation 4
   7. Target Group 5
   8. Outline 6
2. Literature Overview 7
   1. Structure and Goals of Literature Overview Section 7
   2. Literature Overview Strategy 8
   3. Service Selection Model Overview 9
   4. Selection Algorithms Overview 15
   5. Framework Overview 18
   6. Literature Overview Conclusion 19
3. Model Design 20
   1. Structure and Goals of Model Design Section 20
   2. Overview of Selected Machine Learning Algorithms 20
      1. Multilayer Perceptron 20
      2. Support Vector Machines 23
      3. Decision Trees 27
   3. Architecture Design 29
   4. Class Diagram 32
   5. The Usage of Weka Framework Components 39
   6. Graphical User Interface 41
   7. Model Design Chapter Conclusion 45
4. Model Evaluation 47
   1. Structure and Goals of Model Evaluation Section 47
   2. Overview of Accuracy Measuring Techniques 47
   3. Input Data Collection 50
   4. Input Data Transformation 56
   5. Conduction of Experiments 57
      1. First Experiment Conduction 57
      2. Second Experiment Conduction 58
      3. Third Experiment Conduction 63
5. Analysis 68
   1. First Experiment Analysis 68
   2. Second Experiment Analysis 68
   3. Third Experiment Analysis 69
   4. Reliability 70
6. Discussion 71
7. Conclusion 73
   1. Further Research 73 References 75
8. Appendix 1. Tables with raw data of experiments I
   1. Table with accuracy values for 100 datasets for each observed machine learning algorithm I

# Introduction

In this chapter first of all the background of the master thesis is presented as well as a short overview of the previous research. Based on this information, a thesis goal was formulated as well as goal criteria. Moreover, a “Motivation” and “Target group” sections contain an explanation, why the defined research problem is actual and should be investigated as well as a category of people, for whom the information from this master thesis can be interesting. The limitations of the work were also discussed to define the boundaries of the presented study clearly. At the end of this chapter, a brief overview of all master thesis chapters content is presented.

## Background

In the modern world, service-oriented applications become widely spread in business and academic circles and the number of such applications increasing steadily from year to year [1]. Service Oriented Architecture (SOA) was introduced to be able to solve a large problem in an efficient way. The main idea of SOA is to split the large task into a set of smaller but related subtasks [2]. The SAO is not technology but design philosophy, brings independence from any vendor, product, technology or industrial trend [3]. One of the possible SOA implementation is Web service. According to World Wide Web Consortium, the definition of Web service is the following:

*“A Web service is a software system designed to support interoperable machine-to-machine interaction over a network.”* [4]

Web services operate with WSDL (Web Service Description Language) to access its interface description and with SOAP (Simple Object Access Protocol) to provide an exchange of structured information [5]. In the service-oriented research field, the question of selecting the most efficient Web service can be defined as fundamental as well as non-trivial one. There are many parameters and techniques, upon which one particular service from the set of others can be selected as the most efficient one. For that purpose, Quality-of-Service is implemented to calculate for each Web service a ranking value, based on non-functional characteristics [6]. QoS includes a variety of methods and techniques, which allows estimating the match the requirements of service callers to the non-functional characteristics of the service provider based on available network resources. It has been predicted that the Quality-of-Service, which are non-functional performance values of Web services, would be a substantial factor in the scope of ranking services performance from the perspective of end-user [7]. Among these non- functional values, there are availability, accessibility, integrity, performance, reliability, regulatory and security. For business applications, providing the high QoS can be a decisive factor in the commercial success of application [8].

One of the main tasks is to learn the dependency of such parameters as the distance between Web service and user, invocation time and service response time to be able to select the most efficient Web service automatically. Herewith, under efficiency of service, the response time is assumed, so the service with the lowest response time from now on will be referred as the most efficient one. In addition, the invocation time will imply a time point, when the user invokes the service and response time will imply the time gap between sending a request to the service by the user and receiving the respective response from the service. Moreover, such parameter as the cost of the service invocation should also be taken into account, because the user will eventually be asked – what price he or she is ready to pay in order to get the most efficient service. Therefore, the problem is to provide the selection of Web service with such ratio of service effectiveness and cost values, which fits the best to the user’s requirements. The service, which fits the best the user requirements, from now on will be referred as the optimal one.

## Previous Research

In the scope of selecting Web services, there are several studies, based on users and service location determination: [7], [8]. However, these approaches pick collaborative strategy – the end user’s decision depends not only on this particular user’s parameters but also on the parameters of other users, who are located within a defined geographical closeness. In this work, the implementation of Web service selection should be designed in such way, that user provides only his own data. This allows making the decision for the particular user as less dependent on other users as possible excluding the abuse of information by other users. Moreover, while using the defined list of parameters during ranking and selecting the most efficient Web service, there is a possibility to find out, what the correlation is between these parameters and what the weight of each parameter is during the ranking of services. Therefore, the development of service selection method, which will pick the most efficient service according to the user’s requirements, can be defined as an actual one.

## Problem Formulation

The goal of this work is to improve the reliability of service selection with the efficiency ratio of service and cost indicators of service operation invocation that fits best the user requirements through the model design and development as well as implementation of an information system based on this model. To achieve this goal the following objectives were conducted:

* + 1. The analysis of currently existing service selection methods was conducted.
    2. On the basis of the chosen service selection methods the model was designed.
    3. Data collection procedure was carried out. Then, the collected data was analysed, and transformed into a format, with which the model could perform computations.
    4. A verification of model was conducted through a set of experiments, and the accuracy of the model was measured.

In other words, the achievement of the defined goal will take place during two main stages: during the first stage, the processes of model design and implementation will be performed. In addition, during this stage, the preparation of input data for implemented model, which will provide automatic Web service selection, will be conducted. From the preceding list, this stage matches 1-3 items. During the second stage, three experiments will be conducted in which the accuracy of each of the selected machine learning algorithms will be evaluated. The process of choosing the optimal Web service is based on the two parameters: the response time and the cost of operation invocation. Since for the actual and predicted results (comparison of which affects the reliability of choosing the optimal web service) the cost of operation invocation is a constant parameter, the response time of a Web service is the major and the only parameter that affects the validity of Web service choice.

Machine learning algorithms predict the response time of Web services and the accuracy of these algorithms’ prediction is an important parameter, which directly affects the reliability of optimal Web service choice. Therefore, it can be argued that the algorithm with the best accuracy value can provide the best reliability value of optimal service selection. At the same time, the improvement of optimal Web service selection reliability will be assessed by increasing the accuracy of the machine learning algorithms result in the course of experiments in comparison with developed strategies, which are designed to simulate the behaviour of the user during the selection of optimal service.

Making a summary of the stages which must be completed to achieve the thesis goal, it can be said that the selected during the experiments conduction machine learning algorithm will form the basis of the model, which is designed to improve the reliability of optimal Web service selection.

## Motivation

Motivation from the end user perspective. In most cases, the main user’s interest is to get a product or service with the ratio of quality and expended resources that fits best the defined user requirements. Exactly at this, the developed system is aimed: provide the selection of the optimal Web service for the end user. Despite the fact that that the system is not used in real production environment, some of the design principles and features can be

used in future systems, which would be intended for commercial purposes. These principles are listed more detailed in the next paragraph.

Motivation from the scientific perspective. A significant amount of literature resources was collected and analysed. The closest to this study was the work in article “Appropriate Machine Learning Methods for Service Recommendation Based on Measured Consumer Experiences Within a Service Market” [9]. However, in the article mentioned above, firstly completely different machine learning algorithms were picked and analysed. Secondly, in this study, a new strategy for selecting a Web service from user side has been developed, which provides a better evaluation of machine learning algorithms effectiveness when comparing it to the service selection strategy effectiveness from the end user side. Finally, thanks to the second and third experiment conduction, there was an attempt to get the rationale for each algorithm’s obtained result. Such kind of information is not provided in every scientific research. This effort is intended to motivate other researchers to not only obtain and evaluate the results of machine learning algorithms but also to explore the reason of such obtained results. Some situations may be possible when different machine learning algorithms work with different accuracy depending on the character of the input data (linear, periodic), and there is a possibility of an outcome when the combination of different algorithms can improve the existing results.

## Research Question

As it has been defined previously, the improvement of reliability can be achieved by improving the accuracy of the Web service selection model. Therefore, the research question can be formulated as following:

*In what way can the most efficient Web service be selected automatically with at least 80% accuracy?*

Based on the data obtained during the literature overview, it can be judged that with high probability the answer to this question will be positive: there is a possibility to build a model, which provides at least 80% accuracy in selecting efficient Web service.

In addition, the number of 80% indicates that finally at least in four cases out of five the implemented model should provide the user correctly chosen most efficient Web service. At this stage, it is difficult to assess how well this value fits the user expectations. However, the strategy of choice Web service from the user side will be presented in this work later. This strategy will show results about 50% of accuracy, which is much worse than the results, obtained by implemented model.

## Scope/Limitation

At this point, some limitations should be declared, because there is no physical possibility to cover all aspects related to a defined problem in this

master thesis. First of all, it is worth mentioning that in the literature review chapter have not been considered all existing selection models for Web service selection, not all machine learning algorithms, as well as not all the frameworks and programming languages which can be suitable for the Web service selection model implementation. The reason for this is time and physical constraints, which have been met during the writing of this work. Those models of Web service selection have been selected and reviewed, which are, in the opinion of the author of this degree project, suit best for achieving the defined goal. Algorithms and machine learning framework were chosen based on the results of existing studies.

There were also some concerns about the fact that the Web service selection model could function in real time. This restriction was no longer relevant when the decision to split the model of Web service selection to the foreground and background submodels had been made.

Finally, the last limitation is related to the input data and obtained results. In this study, the obtained results were not compared with the results obtained in the course of other experiments and studies. This was not done because in different experiments different sets of data had been used. Consequently, this would require focusing only on a single study and using the set of input data from this study for experiment conduction. This would significantly narrow the possibilities to modify the input data for the experiment, and such modifications have been made for the third experiment in this study. An alternative would be a possibility to use some works and provided by them input data sets for evaluation and comparison of results of machine learning algorithms work. This was impossible due to time and physical constraints. Therefore, to evaluate the obtained results a strategy of optimal Web service selection from the user side has been developed.

## Target Group

This study is mainly aimed at professionals who work in the field of research of possible methods and techniques for selecting optimal Web service. In particular, for these specialists, the results of the experiments, as well as the methodology of carrying out the experiments themselves may be of interest. In addition, the developed strategy of choosing the optimal Web service from the user side and the strategy of calculating the accuracy of the result of the machine learning algorithms work called TOPGAP may be of interest. The strategy of accuracy calculation called TOPGAP can also be useful experts from areas, which related to regression analysis, because it, in a case of necessity, can be easily adapted for experiments, which are different from the conducted ones shown in this study.

## Outline

Chapter 2 provides an overview of existing tools, which can be used for model construction to provide automatic service selection. In addition, in this chapter, non-functional requirements have been formulated and then the appropriate methods, algorithms, approaches and libraries, which in the best way meet the defined requirements, have been selected for future use. An insight of this chapter provides an understanding of what the methods and tools exist that can be used for selection of the optimal Web service and what advantages and disadvantages each of these methods has.

Chapter 3 first provides a brief overview of the technical implementation of each selected machine learning algorithm. This is necessary to ensure that the applicant will not apply to other resources, if he is not familiar with the principle of an algorithm is presented in the study. After that, the architecture of the model is presented as well as implementation details with the help of communication and class diagrams. Finally, the description of the user interface has been provided through the set of actions the user conducts during the process of Web service selection. Familiarization with this chapter provides a clear understanding of how the model of automated service selection has been implemented and how it is functioning.

Chapter 4, first of all, presents methods, with the help of which the accuracy of machine learning algorithms has been assessed. After that, the input data collection procedure is described as well as the procedure of data preparation for further use in the experiment. Finally, the experiments are described and conducted. All necessary data to make conclusions has been collected and presented in a format, which is suitable to make conclusions. Familiarization with this chapter provides an understanding of what preparations have been done to be able to conduct experiments, how the experiments have been conducted and the outcome of these experiments.

In chapter 5, the analysis of obtained data is carried out separately for each experiment. The result of each experiment has been analysed and based on this analysis several claims are formulated. These claims are supported by statistical testing. Therefore, in the next chapter, there is a possibility to draw some substantial conclusions. Familiarization with this chapter provides an interpretation of experiment results.

Chapter 6 provides the general discussion about experiment results and their impact on the current work. In this chapter, it is found out, whether the goal criteria of the master thesis as well as a research question has been met or not.

Chapter 7 contains the overall conclusion about work has been done, areas for further improvement of the work as well as the benefits of the implementation of these areas and the reasons why these areas have not been worked out in this study.

# Literature Overview

## Structure and Goals of Literature Overview Section

The literature review is split into six subchapters. In the first subchapter, the brief overview of each subchapter is presented.

In the second subchapter, the literature overview strategy is provided. Firstly, the search engine and search keywords are presented, with the help of which the search has been conducted. After that, the criterion for selecting an article for further evaluation is defined. Finally, the distribution of selected articles over the years is presented. The models, which has been found in selected articles, are evaluated in next subchapter.

In the third subchapter, the non-functional requirements (NFR) for prediction model are stated. After that, the existing models have been evaluated according to stated NFR and then it has been decided, whether there is a possibility not to build an entirely new model, but take some design solutions from already existing model as a fundament and modify it to fit declared NFR. Therefore, the main goals of this subchapter are:

* + 1. Identify the NFR for service selection models.
    2. Search for existing service selection models, highlight and compare their advantages and disadvantages.
    3. Make a conclusion: either pick some design solutions from already existent models or declare that completely new model will be designed and implemented.

In the fourth subchapter, the NFR for selection algorithms are defined. This makes sense, because in both cases, when either the new model will be built completely from the scratch or the design decisions from already existent implementations will be taken, the knowledge about these selection algorithms will be required. Moreover, in this subchapter, the rough design of future selection model is already known and according to that, the NFR of selection algorithms can be defined more precisely. Therefore, the main goals of this subchapter are:

1. Identify the NFR for selection algorithms.
2. Search for existing service selection algorithms and evaluate them. The evaluation of selection algorithms will be conducting according to the existing sources of selection algorithms evaluation.
3. Select no more than three particular selection algorithms, which fits the defined NFR.

In the fifth subchapter, the NFR for the selection frameworks have been defined, with the help of which there is a possibility to implement the model and make the whole process of service selection automated. Therefore, the main goals of this are:

1. Identify the NFR for selection frameworks.
2. Search for existing selection frameworks and determine their degree of matching the defined NFR.
3. Select one concrete selection framework, which fits the defined NFR the most.

Finally, in the sixth subchapter, a conclusion about goals of literature overview section is made. At this moment, a general design solution for service selection model has been presented. This design solution is elaborated in details in next chapter of this work. Moreover, the selection algorithms have been defined, which form the basis of service selection model. Finally, the selection framework has been selected, with the help of which the model is implemented and the whole process of service selection is automated.

## Literature Overview Strategy

The literature overview has been conducted according to the strategy, whose main points are presented in this subchapter. A literature review was conducted during the period from the middle of January 2015 until the beginning of April 2015. OneSearch has been chosen as the only search engine. This decision has been made due to several reasons. The first one is that this engine is working with a large number of databases, including IEEE Xplore, Google Scholar and many others [10]. Therefore, there was no need to use each database separately, but there was a possibility to obtain a consolidated result, given by OneSearch engine. The second reason was that Linnaeus University supports this search engine. Therefore, there was a possibility to get support from library staff. Moreover, OneSearch provides not only collections articles and reference databases but also a list of available books of the university library, which can be loaned. This reason was not a decisive one, but it could be very useful in case when some technical literature would be required. Finally, the author’s membership in Linnaeus University provided access to a large number of paid articles. By using other search engines, there was a need to insert credentials each time and check, whether there was access to the desired article or not. It could speed down the process of the literature search.

The first term, which has been used as a search keyword, is “*service selection*” because this is the process, on which the current master thesis is based. Moreover, such keywords as “*ranking*” and “*Quality of Service*” have been included, because they define, in which criteria the process of service selection can be conducted. However, after conducting some preliminary search using keywords “*service selection*” and “*Quality of Service*”, it was found out, that many authors used an abbreviation of this term: “*QoS*”. At least the first 20 results, which has had the best match to the searching criteria, contains the abbreviation “*QoS*” rather than “*Quality of Service*”.

After the search, the first 50 articles, provided by the search engine, were briefly evaluated. It was decided to stop on 50 articles due to time

constraints. However, the references from one work to other literature queries were also evaluated, if they contained some service selection model description. The evaluation has been conducted by NFR, which are defined for service selection model in the next subchapter. Here a little bit of fuzzy logic has been applied, because the observed model is not required to meet all defined NFR in order to be selected. This was done because the goal of literature section was to find some solutions, which can help to build a service selection model. Despite the fact, that the model can meet not all NFR, it can have some design solutions, which can be used in the implementation of the current model. Finally, the last criteria was that the articles were selected no older than 2000.

Overall, there were 13 articles selected for further and more detailed evaluation. The distribution of literature queries, grouped by the year of publication, is presented on figure 2.1.

2.5

2

1.5

1

0.5

0

2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015

Number of queries

Figure 2.1: The distribution of literature queries by the year of publishing.

As a conclusion, in next subchapter, the models, which are presented in selected articles, have been observed and evaluated regarding the defined NFR and rational analysis.

## Service Selection Model Overview

In the scope of selecting the most efficient Web service research field, there is a significant number of models, based on different approaches and techniques. For proper observation and model evaluation, NFR should be defined, based on which the decision has been made whether the selected model or some of its design solutions could be used in this work or not. The following NFR for the service selection model were defined:

* + 1. Service selection model should focus on such characteristics as the accuracy of prediction and speed of prediction as the most important.
    2. Service selection model should have a weight function – it can be decided, how important is each parameter for the user during the service selection process.
    3. The model should be able to adapt automatically to the changing conditions such as adding new service, remove the old one. Here under automatically is meant that there should be no changes in a structure of the model to adapt to the new set of services.

However, the service selection models have been evaluated not only by the defined NFR but also according to the rational analysis: if the model had some drawback, which was not directly related to the defined NFR, but this drawback showed a lack of the observed model, this model was likely to be rejected.

Currently, there are several works, which aim is to model the QoS values as stochastic distributions. In [11] a normal distribution was adopted to present the QoS values of Web services. This approach takes a service broker scenario as an example to demonstrate the impact of stochastic QoS parameters. The user provides to broker both functional information, according to which the broker will query the Web services with proper functionality; and non-functional, according to which the broker will query the Web services with acceptable QoS parameters, such as the cost of operation invocation, response time, availability and other. The main idea is to aggregate the cost of operation invocation and other QoS parameters in respect to the regarded workflow patterns. This will result in overall cost and QoS parameters values of a specific workflow. After that, a set of restrictions will be applied to optimise the model results. The authors of this article found that the probability distribution of non-functional parameters could be retrieved in two ways: either it could be explicitly retrieved from the service provider in terms of the legal agreement, or it could be predicted based on the previous historical results.

In another resource [12] the t location-scale distribution was adopted. The authors of this article have based their work on contacts, which are also called Service level agreements (SLA). These contracts define the regulations and agreements between service provider and end user. There are contracts with hard guarantees, when, for example, the response time have to be always less than some threshold value. However, in the opinion of authors, the usage of such hard rules and boundaries is not realistic and therefore more statistical approach has been applied. The authors of work have proposed soft probabilistic contracts, which means that each of such contract contains a probability distribution of considered QoS parameters.

Finally, in [13] the beta distributions were adopted. The authors of this article have developed a multi-objective stochastic program, which is used to simultaneously optimise the selected QoS parameters, such as workflow

duration, reliability, availability and cost of service invocation. The mentioned values were then modelled as decision-depended random values.

The general problem with three defined above implementations is that different QoS have various domains, and there is a high probability that there will be a different kind of fluctuations of these QoS. Moreover, not always possible to pick an appropriate probability distribution for the observed type of QoS.

Another approach is proposing to compute the set of optimal services, based on their QoS, as fast as possible, sacrificing at the same time the precision of the result [14]. The optimal service is selected from the list of semantically equivalent services. Services can be called semantically equivalent when they provide the same function. The presented approach is based on the composition of QoS values of service and Genetic Algorithms (GA). According to the article on the topic of genetic algorithms, the definition is following:

*“Genetic algorithm (GA) is a search heuristic that mimics the process of natural selection. This heuristic (also sometimes called a metaheuristic) is routinely used to generate useful solutions to optimization and search problems.”* [15]

The main drawback of this approach is that it sacrifices accuracy for speed, but in the case of this study, accuracy and calculation speed are an equally important and there is no possibility to accept such trade-off.

There are also several approaches, based on collaborative filtering, and they are presented in following resources: [16], [17], [18], [19]. These approaches are based on collaborative filtering strategy. According to the article on the topic of collaborative filtering, the broad definition of this process is following:

*“Collaborative filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc.”* [20]

Figure 2.2. shows a schematic example of user’s rating prediction based on collaborative filtering. In this example, users leave their usage assessment of different Web services. For one particular user, the system predicts the assessment of particular Web service, which has not yet been evaluated. This prediction is based on assessments accordance of Web services, made by other users and the current user. Therefore, the end user’s decision depends not only on this particular user’s parameters but also on the parameters of other users, who have the similar needs and requirements. The user of such collaborative filtering system can share with other users the opinion and judgement about Web services and then collaborative filtering system process this shared data to provide some useful recommendations about Web service selection. However, there are some problems, which are inherent the mentioned above collaborative filtering approaches.



User 3



Server 1

User 1

Server 2

User 2

Server 3

Figure 2.2: A schematic example of a prediction of user’s assessment with the help of collaborative filtering.

The major problem with this approach is that it does not consider the user physical location as a parameter, which should be used during ranking and selection process. In article “Personalized QoS-Aware Web Service Recommendation and Visualization” [21] the authors conducted an experiment on a real data set, which was about 1.5 million service invocations of 100 existing services in the real world under real conditions. After this experiment, it has been found out that some values are highly dependent on the user physical location. As a result, the response time for the group of users, who locate close to each other, fluctuates slightly, whether the response time for users, who are located distantly from each other, varies considerably. Therefore, the precision of collaborative filtering approach is significantly degraded. The second problem is related to memory-based collaborative filtering systems [17], [19]. The number of services and users are rising steadily as well as the complexity of computations, which can be a serious barrier for such systems. When the number of active users begins to run thousands, such systems can hardly cope to work in real time.

However, there is one implementation of collaborative filtering, which manages with problems, which have been mentioned in [21]. The authors of this article were not satisfied with the performance of existing QoS prediction techniques, and they decided to take into account the relation between QoS values and end users’ physical location. In this approach the correlation between user location and QoS attributes were considered, the users location were retrieved by IP address and then the users were clustered by different regions, for which the QoS prediction was calculated. The general idea is that the closer the users are located to each other; the higher probability is that they have similar service usage experience. The authors of this article have combined model-based and memory-based collaborative filtering algorithms

and in that way have improved prediction accuracy and time complexity. As a result, the overall performance of collaborative filtering system was slightly improved, comparing to other collaborative filtering systems. To understand the reasons why this approach can be considered sufficiently, the definitions of model-based and memory-based approaches should be observed.

The memory-based approach works with user rating data. Upon this rating data, the similarity between observable items is calculated and the recommendations can be made. Despite the fact that this approach is simple to implement and it is quite effective, there are some major drawbacks. The scalability of this method is very poor because the performance decreases rapidly with the growth of dataset. Moreover, the addition of a new item can be challenging and time-consuming, requiring re-inserting all elements again into the structure because representation usually relies on a specific vector space [20].

The model-based approach is more complex because it uses data mining and machine learning algorithms, with the help of which it finds patterns and potential relationships between items and values in the training dataset. The derived trained model can then be used to make predictions on the real data set. This method is less sensitive to the sparseness of the data comparing with the memory-based approach. This brings more flexibility while adding new elements and the predictions are more precise. Not the last thing is the fact that model-based approach can provide rational for the concrete recommendation, therefore, the user can understand, on what ground current recommendation was compiled. Moreover, the model-based approach can handle with weights. However, there are a couple of drawbacks. The high cost of the model construction is the main disadvantage of this approach. Moreover, the loss of data is possible in case of usage some reductions models [20].

There is one more approach, which was described in article “The QoS Prediction of Web Service with Location Information Ensemble” [22]. This model is also using collaborative filtering approach: it arranges the properties of one particular user and its neighbours together and the comprehensive combination of these values is used for service selection process. The authors of this article also care about the precision of prediction and use probabilistic matrix factor model to improve it.

Despite all reasonable advantages of collaborative filtering, during the detailed investigation, one drawback was spotted. In the case of collaborative filtering algorithms, the data is shared between several users, so the data of the one user can be used to calculate the most efficient service for another user. It can lead to some problem because some companies can act as a user and abuse data to “improve” the rating of their own service. Consequently, significantly misrepresentation of information occurs, which can lead to the decrease of model precision. However, the model-based approach of

collaborative filtering has shown the good performance and based on data mining and machine learning algorithms, so it is worth to look in detail at machine learning algorithms.

Another approach, which is proposed in article “Appropriate Machine Learning Methods for Service Recommendation Based on Measured Consumer Experiences Within a Service Market” [9] uses a model-based approach. As it has been mentioned before in this chapter, the model-based approach uses data mining and machine learning algorithms. During Web service selection procedure, the observed approach uses several machine learning algorithms, which Naïve Bayes, Hoeffding Tree and Fast Incremental Model Trees with Drift Detection (FIMT-DD). The model of this approach is provided in figure 2.3.

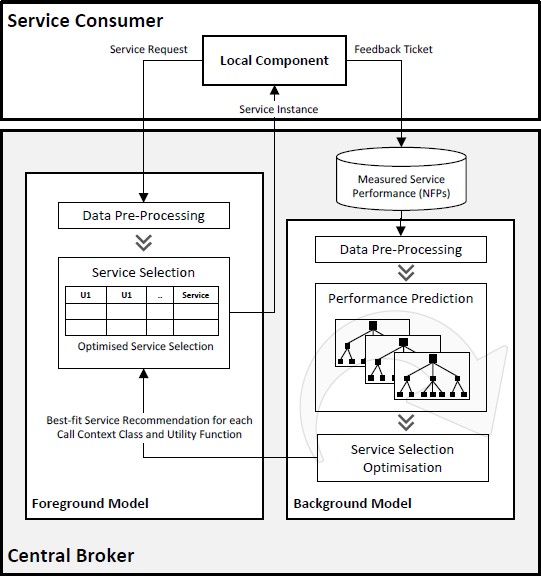


Figure 2.3: The structure of model, proposed in [9].

In observed approach, first, a measurement feedback ticket arrives at the local component. Local component is used to proceed two main tasks. The first task is to manage the dynamic bindings and requests of the optimal

Web services from a user perspective. The second task is to compute the feedback ticket after the Web service invocation. This ticket contains data about processed service invocation: non-functional parameters of a service call. This data is stored and pre-processed in the background model. After that, it is used to make predictions by learning the model. For each Web service, the utility function is calculated, based on a combination of different non-functional parameters. The Web service with the highest utility value will update the table in foreground model. In its turn, this table will be used each time, when the central broker will receive the request from a local component to determine the optimal Web service for the user. This design gives a performance advantage because learning process can be quite resource and time consuming and placing learning process in foreground model can negatively affect the overall performance of service selection model – it may happen that random selection of services can be much more effective than the usage of the model for selection the optimal service.

Therefore, to make a conclusion of this subchapter, a model-based

approach has been picked. In the next subchapter, the literature overview has been continued to find out, which machine learning algorithms the model- based approach provides. The decision to select the model-based approach has been made because this approach meets all defined NFR: as it has been mentioned before, is quite effective, it allows adding new elements to it, it has weight function and the user data can be independent in such model. The designed in this work selection model has also been divided into foreground and background submodels, as it is described in [9] because in that case the learning speed of machine algorithms is not a crucial parameter, which can have a dramatic effect on the performance of Web service selection model.

## Selection Algorithms Overview

As defined in the previous subchapter, it was decided to pick a model-based approach. However, before starting to set up the NFR and select machine learning algorithms according to them, it should also be decided, from which category the machine learning algorithms should be selected for designed model. Machine learning algorithms are divided into several groups, according to different selection criteria. One of such criteria can be a type of output data. There are the several categories of these algorithms: classification, clustering and regression. An observation of these categories has been presented and based on that, it has been decided, which of them suits for model design better.

The first category is classification, in which the inputs are divided into several classes and then when new input is coming, it should be assigned to one or more of existing classes. Clustering is the special case of classification with the only difference: at the beginning, the classes are not clearly defined but formed during the model functioning. For Web service selection model

there is a need to have the predictable response time of each service for each time point. This is required for one major reason – there is a need to put in correspondence the response time of each service to the cost of the call operation to calculate the rankings of all services invocation and select one or more services, which fit the user requirements most. However, and it has been described in [23], the classification and clustering algorithms do not allow to conduct such outputs. They only can classify for each time point the service according to one or more parameters and this classified service can be assumed as the best one.

Therefore, the regression algorithms should suit to the defined task better. Regression analysis is highly used in prediction tasks because the output of regression algorithms is continuous. Therefore, for defined task, the regression algorithm is able to, according to the previously obtained data, predict the response time for each service for each time point. This feature gives a huge flexibility, makes it possible to put in correspondence the response time of each service to the cost of the operation call and select the best service not only according to the response time or other non-constant parameters but also according to the user preferences.

As a result, the machine learning algorithms, which are used for regression analysis tasks, have been observed and selected. Therefore, the following NFR for these machine learning algorithms were defined:

* + 1. High accuracy of prediction;
    2. High performance of prediction;
    3. High performance of learning and accommodating to changes.

These NFR are arranged in decreasing order of importance, so for conducted experiment it is more important to have the high accuracy of prediction that the ability to accommodate to the changes fast.

In the article “Appropriate Machine Learning Methods for Service Recommendation” [9], the comprehensive review of six classification techniques was presented: Decision Trees, Neural Networks, Naive Bayes, k- Nearest Neighbours algorithm (k-NN), Support Vector Machines (SVM) and Rule Learners. In this resource, each algorithm was evaluated by measuring the following parameters:

* Accuracy in general;
* Speed of learning with respect to number of attributes and the number of instances;
* Speed of classification;
* Tolerance to missing values;
* Tolerance to irrelevant attributes;
* Tolerance to redundant attributes;
* Tolerance to highly interdependent attributes (e.g. parity problems);
* Dealing with discrete/binary/continuous attributes;
* Tolerance to noise;
* Dealing with danger of overfitting;
* Attempts for incremental learning;
* Explanation ability/transparency of knowledge/classifications;
* Model parameter handling.

To meet defined NFR, which have been stated to our selection algorithms, the following parameters should be taken into account:

* Accuracy in general;
* Speed of learning with respect to number of attributes and the number of instances;
* Speed of classification;
* Model parameter handling (under a question).

Other parameters, which has been presented in this survey, can be ignored. To make a short version of the table with algorithms comparison only by required parameters, a table 2.1. is presented.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Decisi on Trees | Neural Networ ks | Naive Bayes | k-NN | SVM | Rule Learne rs |
| Accuracy in general | 2 | 3 | 1 | 2 | 4 | 2 |
| Speed of  classification | 4 | 4 | 4 | 1 | 4 | 4 |
| Speed of learning with respect to number of attributes and the number of  instances | 3 | 1 | 4 | 3 | 1 | 2 |
| Model parameter  handling | 3 | 1 | 4 | 3 | 1 | 3 |

Table 2.1: Comparison of machine learning techniques (Note: the bigger number is, the better the result).

Based on the high levels of such parameters as accuracy and speed of classification, two algorithms for the future model were selected: Neural Networks and SVM. In Neural Network field, there are two main categories – single layered perceptron and multilayer perceptron (MLP). Single layer perceptron can only handle with values, which are linearly separable. Therefore, the only acceptable choice is multilayer perceptron. Backpropagation has been selected as a training technique because it is the most common decision for multilayer perceptron [24].

Decision trees algorithm has also been selected. Despite the fact that decision trees algorithm has the same indicator of accuracy as k-NN and Rule Learners, the other indicators of this algorithm are better, so among these three algorithms the decision trees has been selected as the most suitable one.

Therefore, to make a conclusion of this subchapter, three machine learning algorithms has been selected:

1. Multilayer perceptron with back propagation learning technique;
2. Support Vectors Machine;
3. Decision Trees.

## Framework Overview

As defined in the previous subchapter, three machine learning algorithms have been picked. To select framework, which was used to implement and evaluate these three selected algorithms, the following NFP were considered:

* + 1. The framework should support such types of machine learning as multilayer perceptron with backpropagation learning technique, support vector machines and decision trees. This requirement is the crucial one, because if the framework does not support any of these machine learning types, it will be impossible to measure the performance and accuracy of algorithms properly.
    2. The framework should have a high number of libraries, extensions and plugins so there will be a good perspective to use this framework for other machine learning methods to compare obtained results in a similar runtime environment.
    3. The framework should support automation, providing either functionality within application or API, which can be used from a separate application.
    4. The framework should support online learning. Online learning allows to process data in sequential order: at each step, the model can be updated for better predictions on further steps [25]. As an opposition, there is a batch learning, according to which the model first should be trained on the entire training set at once. Because in the model the processes of training and selecting the most efficient Web service can alternate, therefore, support for online learning is one of the major NFR for the selection of the framework.

In addition, one important but not mandatory selection factor is a programming language, on which the framework is written. With all conditions and characteristics being equal, preference will be given to the framework, which is written on Java, because the applicant has already a thorough knowledge of Java language. Therefore, there is no need to learn a new programming language. This factor will reduce the time, required for implementation of the Web service selection model.

There is a poll of framework usage [26], where in the first place is the tool, called RapidMiner. It is a powerful application, which provides a variety of data mining and machine learning techniques and algorithms. It also has a large variety of existing extensions to enlarge the functionality. However, RapidMiner does not provide online learning when data is available in a sequential format and it can be helpful to map data from dataset to the corresponding labels [27]. R language is, because of its statistical background, very powerful tool, which provides a significant amount of implemented machine learning algorithms. Nevertheless, for the same reason, this language does not provide the proper level of automation. Excel and SQL are on the third and fourth place respectively in the ranking. The main problems of these tools are their low level of automation – there is a need to build an additional application to make the whole process automated. Moreover, they are more suitable as data storage tool, but not as data processing tool because of lack of existing algorithms. Python is the next candidate in presented poll, on which several frameworks are implemented: scikit-learn, Pylearn2 and some others less popular [28]. The problem with Pylearn2 is that currently it is not supported by any developer [29]. On the contrary, scikit-learn framework receives regular updates, supports online learning, provides the proper level of automation through API and have some build-in capabilities to visualize result. At this point, it was marked as a potential candidate for model implementation. The next candidate is Weka framework. It has all advantages as RapidMiner do but also supports online learning. Moreover, comparing with the potential candidate scikit-learn, Weka also has the GUI model builder called “KnowledgeFlow”. This can be useful for some preliminary tests of selected techniques or model visualization. Finally, this framework is written in Java while scikit-learn is written in Python. Therefore, as a result of this subchapter, the Weka framework was selected for implementation of service selection model.

## Literature Overview Conclusion

To make a conclusion of this chapter, three milestones, were achieved: first, the NFR for selection model were defined and a review of currently existing models and approaches was conducted. As a result, a model-based approach has been selected, which is based on machine learning algorithms. In addition, a decision to divide the model into foreground and background submodels and to place learning process into background submodel has been made. After that, the NFR for machine learning algorithms were defined and upon these requirements, three algorithms were selected: multilayer perceptron, Support Vectors Machine (SVM) and decision trees. Finally, the NFR for framework were defined and the framework Weka was selected in order to implement and evaluate the selection model and to make the whole process automated.

# Model Design

## Structure and Goals of Model Design Section

The model design chapter is split into seven subchapters. In the first subchapter, the brief overview of each subchapter is presented. In the second subchapter, there is an overview of selected machine learning algorithms. The short overview of functioning principle of each algorithm is presented. The purpose of this procedure is to provide technical details of each algorithm to understand their basic workflow. In the third subchapter, the architectural design is presented. The main design principles are listed here, as well as used patterns. In addition, the communication diagram shows the interaction between core components of the system. In the fourth subchapter, the class diagram with detailed components description is presented. This information provides a clear understanding of what every core element of the system does, which methods and fields and which responsibilities it has. In the fifth subchapter, the usage of Weka components is demonstrated. This subchapter provides an understanding of how the Weka components have been used and where they have been placed in the system. The sixth subchapter provides the description of the user interface and what actions the user conducts during the process of Web service selection. Finally, in the seventh subchapter, a short conclusion about model design has been made. As a result of this whole chapter, the model has been designed and presented, which is used during evaluation in next chapter.

## Overview of Selected Machine Learning Algorithms

### Multilayer Perceptron

In broad scope, the artificial neural network can be defined as a mathematical model, which maps data vectors from one space into another [30]. The main functioning principle is very similar to functioning principles of biological neural networks – the network of neuron cells in a vital organism. Artificial neural network refers to statistical learning algorithms and is used to approximate or estimate the input values. Artificial neural networks can be used as for clustering and classification tasks as well as for tasks of the regression analysis. The artificial neural network consists of basic units, called neurons, which computes the result values from the input data. On figure 3.1. the example of the artificial neural network is presented.



Figure 3.1: The structure of neural network.

On figure 3.1, the rounded circles represent the core units of artificial neural network – neurons and the arrows represent the direction of connection between these neurons. Each neuron in such network deals only with signals, which it periodically receives, and with signals, which it periodically sends. Despite the fact, that each neuron has a very simple structure, a large number of neurons being connected in a network can perform complex tasks.

Neural networks are not programmed in the usual sense of this word’s definition, but they are trained. The possibility of learning is one of the main advantages of neural networks over traditional algorithms. In technical aspect, training is a process of finding the coefficients of the connections between neurons. During the training, the neural network is able to identify complex relationships between input and output data.

One possible usage of the artificial neural network is data prediction. The ability of the neural network to predict proceed directly from its ability to compile and release hidden dependencies between the input and output data, which is the core idea of regression analysis. After training, the neural network is able to predict the next value of a certain sequence based on several previous values and any currently existing factors. This ability of artificial neural network have been used to predict the response times of Web services in this study.

One of the most common variation of artificial neural network is multilayer perceptron, which can learn and solve quite complex problems. Multilayer perceptron is a feedforward neural artificial network. Feedforward neural artificial network has an acyclic structure, so there are no cycles or loops in such network. Usually, the structure of multilayer perceptron consists of four or more layers. On figure 3.2. the structure of multilayer perceptron is presented and three dots notation on this figure mean that there can be more than two hidden layers in multilayer perceptron.

There is one layer with input neurons, which does not have inputs from other neurons. This is a layer of sensors or receptors, which receive data from outside of multilayer perceptron. These elements are called S-elements. Depending on the input data, each S-element can be in one of two states – rest or excitation, and only in the second case, it transmits a single signal to the next layer, to associative elements. A-elements are called associative because each such element generally corresponds to a set of (association) S- elements. A-element is activated, when the number of signals from S- elements on its input exceeds a certain value θ. The signals from the excited A-elements are transferred afterwards to the adder R, and besides the signal from the i-th associative element is passed by a factor wi. [31] This factor is called the weight of A-R relations.



...

Wi

...

Wi

Wi

Wi

...

Wi

...

Output layer R-neurons

Hidden layer A-neurons

Input layer S-neurons

Figure 3.2: The structure of multilayer perceptron.

As well as the A-elements, the R-element calculates the sum of the input signals multiplied by the weight. R-element, and with it the multilayer

perceptron, produces a value, which should be predicted by this perceptron. Mathematically, the function realized R-element might be written as

𝑛

𝑓(𝑥) = 𝑠𝑖𝑔𝑛(∑ 𝑤𝑖𝑥𝑖 − 𝜃)

𝑖=1

The training of multilayer perceptron is a process of changing the weighting coefficients wi relations A-R. Weight relationships S-A and thresholds A-elements are randomly selected at the beginning and then do not change.

For training multilayer network a backpropagation learning algorithm has been used, in which the error signal, which calculation is based on the perceptron outputs with each piece of data, is transmitted to its inputs. After that, this error signal is compared to the expected value and corresponding neurons’ weights are adjusted.

After training, the perceptron is ready to work in the recognition mode. In this mode, perceptron imposed previously unknown to him objects and must determine to which class they belong. Perceptron work is as follows: upon presentation of an object, excited A-elements transmit a signal to R- element and this signal equals to the sum of the corresponding coefficients wi. This output signal is a value, which should be predicted by perceptron.

### Support Vector Machines

Support vector machines (SVM) is a set of similar algorithms, which are used for data analysis and patterns recognition in classification and regression tasks. To understand the functioning principle of SVM, firstly a simple example should be observed. For training purpose, there is a set of entities, each of them belongs to one of two classes. SVM then maps these entities into the points in two-dimensional space. After that, the hyperplane is build, which for current example is a line. This line is distant from entities as much as possible and clearly divides the entities into two categories. The corresponding representation can be observed on figure 3.3.



L1

L2

L3

Figure 3.3: The representation of several lines used for classification of mapped entities.

On figure 3.3, ten entities were mapped in two-dimensional space. These entities belong to one of two classes: either “square” or “circle”. Several lines can be built to separate the space for classes: L1, L2 and L3.

However, the core idea of SVM is to maximise the gap between classes, which leads to such classification that is more precise. For example, on figure 5 the line L2 can be picked up because it has the maximum gap between objects of two classes: line L3 is too close to the objects of class “square” and line L1 is close to objects of both classes. After that, new entities can be mapped in this two-dimensional space and depending on which side they relatively to a line are, each entity can be associated with the corresponding class. Overall, a core feature of SVM method is a continuous decrease of the empirical classification error and increase the gap, so the SVM method is also known as the method of the classifier with the maximum gap.

However, this makes the SVM possible to process only linear tasks. To solve the non-linear tasks a kernel trick was applied [32], introducing ε-SV. To understand the functioning principle of ε-SV, another example should be observed, for which the training set of entities {(𝑥1, 𝑦1), … , (𝑥𝑖, 𝑦𝑖)} ∈ 𝑋 × 𝑅 will be taken, where 𝑋 represents the space of input patterns. For example, it can be currency exchange rates, which are measured within the specified period of time with corresponding econometric indicators. The main goal of ε-SV is to find a function 𝑓(𝑥), which is as flat as possible and at the same time for all instances from training set has the most deviation ε from the actual obtained values 𝑦𝑖. This means, that the algorithm does not care about errors in case they are less than ε but at the same time it does not accept any deviation, which is larger than ε. Therefore, the linear function can be presented in the following form:

𝑓(𝑥) = 〈𝑤, 𝑥〉 + 𝑏 𝑤𝑖𝑡ℎ 𝑤 ∈ 𝑋, 𝑏 ∈ 𝑅

where 〈𝑤, 𝑥〉 represents the dot product of 𝑋. To ensure the flatness of 𝑓(𝑥) there is a need to seek a small 𝑤. One of the possible solutions here is to minimize the norm [33], i.e. ‖𝑤‖2 = 〈𝑤, 𝑤〉. Therefore, this problem can be written as a convex optimization problem:

Minimize 1 ‖𝑤‖2

2

subject to { 𝑦𝑖 − 〈𝑤, 𝑥𝑖〉 − 𝑏 ≤ ε

〈𝑤, 𝑥𝑖〉 + 𝑏 − 𝑦𝑖 ≤ ε

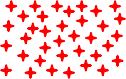
This means that such function f exists, which can approximate all pairs (𝑥𝑖, 𝑦𝑖) with ε precision. In other words, parameter ε plays a role of the threshold: all predictions have to be in ε range in order to be considered as true predicted values. On figure 3.4. the graphical representation of this formula is shown [34].



Figure 3.4: The graphical representation of SVR.

Like in classification tasks, for regression, there is also a motivation to search and then optimise the generalisation bounds. In regression tasks, the input set 𝑋 is first mapped into the *m*-dimensional feature space [35]. For that purpose, the fixed, usually nonlinear, mapping is used and, as a result, a linear model is constructed in this feature space. The example of such model is presented on figure 3.5. Despite the fact, that the model is built for non- linear mapping, the model for regression task has quite similar structure.

1



0.8

0.6

0.4

Ψ

1



0.5

0.2

0

0

-0.2

-0.4

-0.6

-0.8

-1

-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6

0.8 1

-0.5

-1

-1

-0.5

0.5 0.5 0

0

1 1

-1

-0.5

Figure 3.5: The process of mapping the input set to *m*-dimensional feature space [35].

With the help of mathematical notation, the calculation of the linear model 𝑓(𝑥, 𝑤) in feature space can be conducted according to the following formulation:

𝑚

𝑓(𝑥, 𝑤) = ∑ 𝑤𝑖𝑔𝑖(𝑥) + 𝑏

𝑗−1

where 𝑔𝑖(𝑥), 𝑗 = 1, … , 𝑚 represents a collection of transformations, which have to be nonlinear. The parameter b is named “bias” term. Often the preprocessing of data is conducted, after which the data is supposed to be zero mean. In such cases, the formulation is simplified by dropping the bias term. The loss function is used to measure the quality of estimation:

𝐿(𝑦, 𝑓(𝑥, 𝑤))

Here the mentioned before epsilon-intensive function is used:

𝐿𝑔

(𝑦, 𝑓(𝑥, 𝑤)) = { 0, 𝑖𝑓 |𝑦 − 𝑓(𝑥, 𝑤)| ≤ ε

|𝑦 − 𝑓(𝑥, 𝑤)| − ε, 𝑜𝑡ℎ𝑒𝑟𝑤𝑖𝑠𝑒

The empirical risk is calculated according to the following formula:

𝑛

𝑅 (𝑤) = 1 ∑ 𝐿

(𝑦 , 𝑓(𝑥 , 𝑤))

𝑒𝑚𝑝.

𝑛 𝑔 𝑖 𝑖

𝑖=1

Therefore, the support vector machine algorithm aims to perform linear regression in high-dimensional feature space using the epsilon-intensive loss function and at the same time to reduce the complexity of model by minimizing the ‖𝑤‖2. For that purpose, the non-negative slack values

𝖯𝑖, 𝖯∗ 𝑖 = 1, … , 𝑛 can be introduced. The purpose of these values presence is to measure the deviation of training entities, which are located outside the ε epsilon-intensive zone. Therefore, the SVR can be formulated as a task of minimisation the following functional:

𝑖

𝑛

min 1 ‖𝑤‖2 + 𝐶 ∑(𝖯

+ 𝖯∗)

2 𝑖 𝑖

𝑖=1

𝑦𝑖 − 𝑓(𝑥𝑖, 𝑤) ≤ ε + 𝖯∗

𝑖

{ 𝑓(𝑥𝑖, 𝑤) − 𝑦𝑖 ≤ ε + 𝖯𝑖

𝖯𝑖, 𝖯∗ ≥ 0 , 𝑖 = 1, … , 𝑛

𝑖

The transformation of this optimization problem can be applied, which will lead to the dual problem. The following functions provide the solution of this problem:

𝑛𝑠𝑣

𝑓(𝑥) = ∑(𝑎𝑖 − 𝑎∗) 𝐾(𝑥𝑖, 𝑥) 𝑠. 𝑡. 0 ≤ 𝑎∗ ≤ 𝐶, 0 ≤ 𝑎𝑖 ≤ 𝐶

𝑖 𝑖

𝑖=1

where 𝑛𝑠𝑣 indicates the amount of Support Vectors, which should be built. The calculation of kernel function is proceeded according the following formulation:

𝑚

𝐾(𝑥, 𝑥𝑖) = ∑ 𝑔𝑗 (𝑥)𝑔𝑗(𝑥𝑖)

𝑗=1

Finally, the good performance of SVR depends on the setting up the proper meta-parameters such as C, ε and kernel parameters. Here the constant C represents the trade-off between the complexity (flatness) of *f* function and the degree to which the deviations larger than e are tolerated. Parameter ε defined the width of ε-zone, and the training set should be positioned in the boundaries of this zone. In addition, parameter ε and the number of support vectors, which should be constructed, has the inverse relation: the lower the value of ε is, the more support vectors will be built and vice versa. Moreover, the bigger the ε value is, the more “flat” will be the estimation of the final result. As a result, both C and ε affect the complexity of the model.

In Weka, the implementation of SVM for regression is called SMOReg

[36].

### Decision Trees

Decision trees is a toll of decision support, which is used in statistics and data analysis to provide prediction models. There are two main types of decision trees: the models, where the target types take the finite values are called classification models and they are used for classification tasks. On figure 3.6. the example of the decision tree, used in classification task, is presented.

As it can be observed on figure 3.6., the decision tree representation has a flowchart structure. Each internal node contains a test condition. If this test condition is satisfied, the transition occurs on the right branch, otherwise on the left branch. Each test condition is called classification rule. Leaf nodes represent the class labels. Concluding, the main idea of the decision tree is to apply and check classification rules and by obtained results classify the object to the corresponding class label.

no

yes

destroyed

no

yes

not destroyed

no

yes

not destroyed

Building 15 years old or

more

Building with more than 1 floor

Richter scale is more than

10

Figure 3.6: Decision tree for classification task.

Richter scale is more than 5

no

yes

not destroyed

destroyed

Decision trees can be used in regression tasks, working with infinitive values and producing a numerical output. There are several algorithms, which are used to generate decision trees from entry dataset. The very first algorithm was ID3, which is used to build a decision tree for the dataset. The functioning of the algorithm is the following:

* + - 1. ID3 takes the entry dataset X as a root node.
      2. ID3 calculates the entropy value for each attribute in dataset X.
      3. ID3 splits the dataset X by using the attribute, the entropy of which is the minimum.
      4. ID3 generates the rule, containing this attribute.
      5. ID3 performs the operation on the two parts of the split X dataset again.

The criteria for stopping recursive calculations can be when either there is no attributes nor elements in the entry dataset or each element from dataset belongs to one class.

There is an ID3 algorithm improvement, called C4.5. The principle of its functioning is the following: on each tree node, the C4.5 algorithm selects the attribute, which splits the dataset in the most efficient way by information gain (an asymmetrical measure of the distance between two probability distributions). The attribute with the highest information gain is selected to make a decision rule. Then, C4.5 algorithm repeats this action on the two subsets of the initial entry dataset.

The advantage of C4.5 is that this algorithm can handle with both labelled and continues attributes. C4.5 creates a threshold value and then splits the input dataset into two subsets: in the first subset, the attribute value

is higher that the threshold value, in the second dataset the attribute value is equal or lower than the threshold value. In Weka, there is an implementation of C4.5. algorithm called J48.

Unfortunately, J48 in Weka can only be used for classification tasks, but not regression ones. For service selection model implementation the algorithm M5P has been selected. This is the implementation of Quinlan's M5 algorithm [37], which can handle with continuous numerical values and can be used for regression tasks. In this implementation, first of all, a splitting criterion was introduced to minimize the intra-subset of variation in the class values down each branch [38]. There are two criterions for stopping a splitting procedure: there are either few elements left or class values of instances, which reached the current node, variate insignificantly. In addition, the tree is pruned back from each leaf. Finally, a smoothing procedure is applied, the main purpose of which is to avoid the significant differences between subtrees. It moves from leaves to root, combining the leaf model prediction with each node and then it smoothes it with the help of value, which is predicted by the linear model of the observed node.

To make a conclusion, decision trees have several advantages: they do not need complex input data transformation. Moreover, the decision trees are easy to understand and there is a statistical measurement, with the help of which the performance of decision tree can be measured. However, they have one disadvantage – when many outcomes are expected, the calculations become more complex and resource consuming. Nevertheless, this disadvantage was not effected on developed model in a critical way, because a background submodel was introduced, which had neutralized the performance disadvantages of machine learning algorithms.

## Architecture Design

The first architectural decision was to use software architectural pattern called Model View Controller (MVC). This pattern, as its name implies, has three main components: model, view and controller. The model provides data and business logic rules, the view is responsible for the graphical user interface representation and controller provides the interaction between the model and the view. MVC architecture is shown schematically on figure 3.7.

Updates Manipulates

Controller

Model

User

Shows data

Inputs data

View

Figure 3.7: Schematic representation of MVC design pattern components.

The advantage of MVC is that it provides a strong separation between presentation (view and controller) and domain (model) [39]. This advantage can be used during the Web service selection model implementation. As a result, on the first step the separate Web service selection model can be implemented without the dependency from the user interface and after that, the user interface and supporting controllers can be implemented to enable user interaction with the model. For the developed application, the model from MVC pattern is the Web service selection model, which has been implemented in Java programming language with the help of implemented algorithms, taken from Weka framework. The view from MVC is the graphical user interface, which provides the possibility for the user to interact with Web service selection model. Finally, the controller from MVC are components, which are responsible for communication between the graphical user interface and Web service selection model. Further, in this chapter the detailed description of these components is provided.

As it was told in the previous chapter, a decision to split the whole Web service selection model into background and foreground submodels had been made. The advantage of background submodel presence is that such machine learning algorithms have been chosen before, which have high accuracy and high classification speed, but their learning speed is very low. Therefore, if the learning procedure is placed in the foreground submodel, the advantages of our system will be dramatically reduced, because the response time of the worst service can be lower than the time, required by the own system to determine the best service from the list. As a result, the learning procedure was placed in the background submodel.

On figure 3.8. the communication diagram of Web service selection model, according to the UML standard, is represented.



**Foreground**

1.3:calculateRanking()

3.1:swapModels()

**ServicePool**

1.1:triggerRequest()

1.4:sendRequest()

:MainMenuController

:SE1

1.2:classifyOneInstance()

User

:DE1

:ForegroundModel

**Background**

:DE2

:TaskToQueue

:US1

:BackgroundModel

2.2:trainFormInstances()

2.1:addInstances()

Figure 3.8: The communication diagram for Web service selection model.

On this diagram, first, the user sends the request to the Web service with the help of function *1.1:triggerRequest()*. This function is implemented in class *MeinMenuController*. After that, the object of this class invokes the function *1.2:classifyOneInstance()* of the class *ForegroundModel*. This function predicts the response times for all available Web services for defined period of time. Then *MainMenuController* invokes the function *1.3:calculateRanking ()* to determine the optimal Web service from the list of Web services, the response time of whose was predicted by *ForegroundModel*. Finally, the object of class *MainMenuController* invokes the function *1.4:sendRequest()* to the optimal Web service and returns the result of function invocation to the user. At this point, the interaction between user and Web service selection model is over. However, selection model continues to work by itself, sending the data of processed request to the *TaskToQueue* with the help of function *2.1:addInstances()*. The object of *TaskToQueue* class is used to accumulate the number of instances, on which the *BackgroundModel* should be trained. Then, the training of *BackroundModel* is processed with the help of function *2.2:trainFromInstances()*. Finally, either after some defined period of time or because of the event, triggered by the user, the object of class *ForegroundModel* should be replaced by the object of class *BackgroundModel* with the help of function *3.1:swapModels()*, which is implemented in *MainMenuController*.

## Class Diagram

On figure 3.9, the class diagram of implemented system is presented. After that, the description of core elements is provided.

**Interface *MainContrInter***

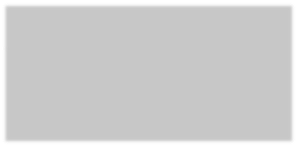
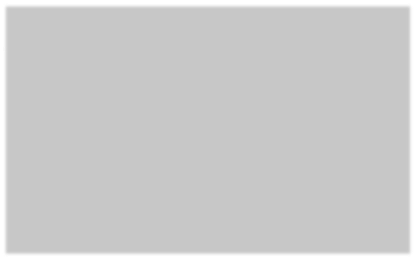
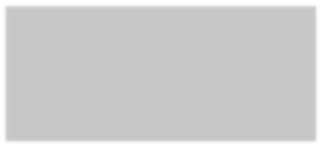
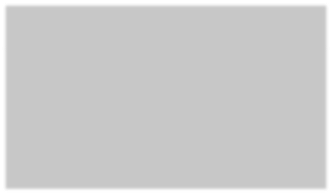
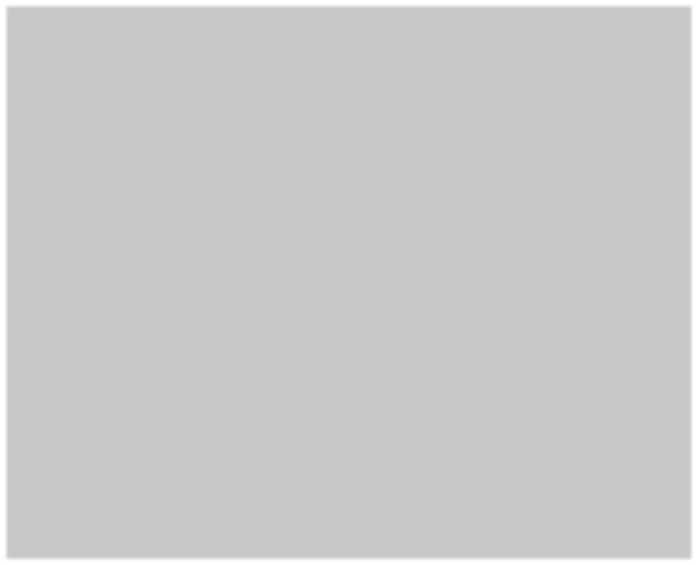
In this diagram, the objects of classes *BackgroundModel* and *ForegroundModel* use the interface *MainContrInter* to notify their current state to the components of the graphical user interface. Methods *setMessageToBMField(String):void* and *setMessageToFGMField(String)*

*:void* of mentioned interface are implemented in class *MainMenuController* and used to set in the corresponding fields of graphical user interface the current state of the objects of classes *BackgroundModel* and *ForegroundModel*. The current state is passed to this method as a string parameter. The next method is called *swapModels():void* and is implemented in *MainMenuController* class. This method is used to swap the foreground model with the background model because background model is more trained. In this method, the copying of object of class *RecognitionModel* is conducted. Next method *setMessageToEvaluationField(String):void* is also implemented in class *MainMenuController* and is used to set the value to the field, which is responsible for showing the results of model evaluation in graphical user interface. The model is evaluated according to several approaches and the accuracy of the model for each approach is presented in mentioned above field. The results of the model evaluation are passed as a string parameter.

**Interface *MainContrInter***

In this diagram, the objects of classes *BackgroundModel* and *ForegroundModel* use the interface *MainContrInter* to notify their current state to the components of the graphical user interface. Methods *setMessageToBMField(String):void* and *setMessageToFGMField(String)*

*:void* of mentioned interface are implemented in class *MainMenuController* and used to set in the corresponding fields of graphical user interface the current state of the objects of classes *BackgroundModel* and *ForegroundModel*. The current state is passed as a string parameter. The next method is called *swapModels():void* and is implemented in *MainMenuController* class. This method is used to swap the foreground model with the background because background model is more trained. In this method, the copying of object of class *RecognitionModel* is conducted. Next method *setMessageToEvaluationField(String):void* is also



BackgroundModel

RecognitionModel

-modelToRecognize:RecognitionModel

-taskToTrain:TaskToQueue

-interfaceMainContr:MainContrInter

-trainingThread:BMTrainingThread

+isTraining:boolean

+trainFromInstances(Instances):void

-trainedInstances:int

-mapOfclassifiers:Map<String, ClassifierImpl>

-workers:ExecutorService

+trainFromInstances(Instances):void

+classifyOneInstance(double):List<ServiceObject>

TaskToQueue

BackgroundModel:BMTrainingThread

+run():void

-instancesToTrain:Queue<Instances>

+isQueueEmpty():boolean

+addInstances(Instances):void

+retrieveFirstElement():Instances

+getSizeOfQueue():int

ForegroundModel

-modelToRecognise: RecognitionModel

-interfaceMainContr:MainContrInter

+classifyOneInstance(double):List<ServiceObject>

+copyRecognitionModel(RecognitionModel):void

EvaluationController

-interfaceMainContr:MainContrInter

+evaluateData(File,int,int,int):void

ServiceObject

-serviceName:String

-responceTims:int

-operationCost:int

-normalizedRespTime:float

-normalizedOpCost:float

-overAllRanking:float

-diffInPerc:float

+normalizeRespTimeInArray(List<ServiceObject>):List<ServiceObject>

+normalizeCostInArray(List<ServiceObject>):List<ServiceObject>

+computeOverallRanking(List<ServiceObject>, float, float):List<ServiceObject>

+computePercDiff(List<ServiceObject>):List<ServiceObject>

<<Interface>>

MainContrInter

MainMenuController

+setMessageToBMField(String):void

+setMessageToFGMField(String):void

+swapModels():void

+setMessageToEvaluationField(String):void

-modelBackground:BackgroundModel

+modelForeground:ForegroundModel

+operationCost:HashMap<String, Integer>

+initialize(URL,ResourceBundle):void

+triggerRequest(Request):Response

+calculateRanking(List<ServiceObject>):ServiceObject

+sendRequest(Request):Response

-setActionsAndListeners():void

EvaluationProcedure

-date:double

-actualList:List<ServiceObject>

-predictedList:List<ServiceObject>

+addReadedItem(String, double):void

+predictAndCalculate(int,int):void

+proceedRandomEvaluation():int

+proceedSimpleEvaluation():int

+processTopRangeEvaluation(int):int

+predictValues(int,int):void

+getTheLeastRankingPredictedServiceName():String

+getTheListOfServicesInRange(int):List<String>

+calculateValues(int,int):void

+getTheLeastRankingActualServiceName():String

ClassifierImpl

-instancesToTrain:Instances

-machineAlgorithm:MultilayerPerceptron

-serviceName:String

-trainedInstances:int

-Attribute1:Attribute

-Attribute2:Attribute

+addInstance(Instance):void

+trainInstances():void

+classifyOneInstance(double):ServiceObject

Figure 3.9: Class diagram of implemented system.

implemented in class *MainMenuController* and used to set the value to the field, which is responsible for showing the results of model evaluation in graphical user interface. The model is evaluated according to several approaches and the accuracy of the model for each approach is presented in mentioned above field. The results of the model evaluation are passed as a string parameter.

**Class *MainMenuController***

Class *MainMenuController* is used to ensure the functioning the elements of graphical user interface of the application. This class also stores the objects of classes *BackgroundModel* and *ForegroundModel* and plays the role of mediator between the objects of these classes and graphical user interface. This class also implements the methods of interface *MainContrInter* and has a field *modelBackground:BackgroundModel*, which is an object of background model and is used to be trained on the input training dataset. There is also a field *modelForeground: ForegroundModel*, which is an object of foreground model and is used to determine the most efficient Web service for defined time point. There is a special collection *operationCost:HashMap<String, Integer>*, which is used to store the operation cost for each Web service. To initialize all elements of graphical user interface, the method *initialize(URL,ResourceBundle):void* is invoked, which also with the help of method *setActionsAndListeners():void* defines the behaviour of graphical interface elements during the user interaction. The next function, called *triggerRequest(Request):Response*, is used to trigger the request from the user. This function passes data from a user action to the controller and gets the predicted response times of Web services for defined time point with the help of function *classifyOneInstance( double):List<ServiceObject>* of class *ForegroundModel*. The function *triggerRequest(Request):Response* returns the object of class *Response*, which contains all data about response: headers, status codes and return value. As a parameter, this function takes *Request* object, which contains request headers and body value. The description of Request and Response classes are not presented, because they are first of all out of the scope of service selection model and secondly their structure depends on the frameworks or libraries, which will be used for network communication. The function *calculateRanking(List<ServiceObject>):ServiceObject* uses the data about Web services predicted response times for the defined time point and based on this data determine the optimal Web service, which should be invoked. The function *sendRequest(Request):Response* is used to send the user’s request to the particular Web service, which has been determined as an optimal one previously.

**Class *RecognitionModel***

Class *RecognitionModel* is used to store the classifiers for all available Web services. These classifiers conduct the regression analysis. This class provides access to the available functions of these classifiers. For that purpose, a special collection *mapOfclassifiers:Map<String, ClassifierImpl>* was defined, which stores for each Web service the classifier – object of class *ClassifierImpl*. There is also an object *workers:ExecutorService*, which is used to manage a pool of several independent threads and therefore the objects of class *ClassifierImpl* from collection *mapOfclassifiers* can use these threads to conduct learning procedure in parallel. Moreover, this object counts the number of instances, on which the training of objects of class *ClassifierImpl* from collection *mapOfclassifiers* was performed and stores this value in variable *trainedInstances:int*. In order to launch the parallel training of objects of class *ClassifierImpl* from collection *mapOfclassifiers,* there is a function *trainFromInstances(Instances):void*, which takes the input training data set as a parameter. For prediction the response times of Web services for defined time point, the function *classifyOneInstance(double)*

*:List<ServiceObject>* can be invoked, to which the defined time point is passed as an input parameter.

**Class *BackgroundModel***

Class *BackgroundModel* implements the functionality of background model and used to initiate the training process of an object of class *RecognitionModel* on the input training dataset. In order to provide access to the classifiers of Web services the object *modelToRecognize*

*:RecognitionModel* is presented. There is also an object *taskToTrain*

*:TaskToQueue*, which plays the role of the queue, saving the data, with the help of which the training of background model should be conducted. Interface *interfaceMainContr:MainContrInter* provides a possibility to gain access to the graphical user interface components. The thread *trainingThread:BMTrainingThread* is used to conduct the process of training of overall background model and the variable *isTraining:boolean* plays the role of the monitor for thread *BMTrainingThread*, indicating, whether the training of background model is currently running or not. The function *trainFromInstances(Instances):void* of this class is used to conduct the training of background model on input data set. This function is invoked in thread *BMTrainingThread*.

**Class *BMTrainingThread***

Class *BMTrainingThread* is used to conduct the training of the model itself without any influence to other components. The defined method *run():void* has been implemented exactly to provide this functionality.

**Class *ForegroundModel***

Class *ForegroundModel* implements the functionality of foreground model and used to initialize the classification process of an object of class *RecognitionModel* for input dataset. In order to provide the access to the classifiers objects of each Web service there is an object *modelToRecognise:RecognitionModel* and to provide access to the graphical user interface components there is an interface *interfaceMainContr*

*:MainContrInter*. The next function *classifyOneInstance(double)*

*:List<ServiceObject>* is used to predict the data of Web services for time point, which is defined as an input parameter for this function. All data about Web services is stored in objects of class *ServiceObject*. Another function of an observed class is called *copyRecognitionModel(RecognitionModel):void* and it conducts the copying of object of class *RecognitionModel* from background to foreground model. The copying is carried out only in the case when the background model is not training (the variable *isTraining* of class *BackgroundModel* is set to false).

**Class *TaskToQueue***

Class *TaskToQueue* is used to store the input data, which later is used for background model training. This data is stored in a collection of type queue *instancesToTrain:Queue<Instances>*. There are also basic functions provided and implemented, which enables the basic operations on the elements from collection: function *isQueueEmpty():boolean* to check, whether there are any instances available for training or not; function *addInstances(Instances):void* to add new instances to the queue; functions *retrieveFirstElement():Instances* and *getSizeOfQueue():int* to return the first element and size of queue correspondingly.

**Class *ClassifierImpl***

Class *ClassifierImpl* presents the implementation of the classifier for certain Web service and provides all functions to be able to work with this classifier. First of all, it saves the set of data *instancesToTrain:Instances*, on which the machine learning algorithm *machineAlgorithm* should be trained. There is also an object *machineAlgorithm:MultilayerPerceptron* of framework Weka, which provides the implementation of certain machine learning algorithm. For this case, the multilayer perceptron is selected. The variable *serviceName:String* saves the name of Web service, to which the defined classifier refers. The variable *trainedInstances:int* stores the number of instances, on which the training of classifier has been conducted. Variables *Attribute1:Attribute* and *Attribute2:Attribute* are the first and the second attribute und these attributes defines correspondingly the Web service invocation timestamp and Web service response time for that time point. The function *addInstance(Instance):void* adds to the queue *instancesToTrain* the new instance, on which the classifier *machineAlgorithm* should be trained.

After adding some amount of instances to the queue, first of all the function *trainInstances():void* proceeds the training of classifier *machineAlgorithm* on the training set *instancesToTrain* and then, after successful training, the function *classifyOneInstance(double):ServiceObject* can conduct the regression analysis with the help of object *machineAlgorithm* for defined as a parameter timestamp.

**Class *EvaluationProcedure***

Class *EvaluationProcedure* contains dataset, which has been predicted with the help of classifiers and dataset of real data for defined time point. The variable *date:double* stores the time point value, for which the data in arrays *actualList* and *predictedList* are stored. Correspondingly, the arrays *actualList:List<ServiceObject>* and *predictedList:List<ServiceObject>* stores the real data (used for model evaluation) and predicted with the help of classifiers data for defined time point date. The function *addReadedItem(String,double):void* adds new Web service and correspondingly the response time of this Web service. The function *predictAndCalculate(int,int):void* first of all predicts the service response time with the help of classifiers and then calculates the real data set. This operation is conducted for the time point, defined in variable date. The *function proceedRandomEvaluation():int* conducts the accuracy measurement according to the RANDOM approach: as an optimal a random service is selected and if this service is the same as the service, obtained as a return value of the function *getTheLeastRankingActualServiceName()*, this function returns 1, which indicates the correctness of the choice. Otherwise, the function returns 0, which indicates the falsity of choice. The next two functions conduct the accuracy measurement. The function *proceedSimpleEvaluation():int* conducts the accuracy measurement according to the approach TOP1: the function *getTheLeastRankingPredicted ServiceName()* returns the optimal Web service and if this Web service is the same as the result of function *getTheLeastRankingActualServiceName()*, the function *proceedSimpleEvaluation():int* returns 1, which indicates the correctness of the choice. Otherwise, the function returns 0, which indicates the falsity of choice. The function *processTopRangeEvaluation(int):int* conducts the accuracy measurement according to the approach TOPGAP: the function *getTheListOfServicesInRange()* returns the list of optimal Web services and if at least one of these Web services is the same as the result of function *getTheLeastRankingActualServiceName()*, the function *processTopRangeEvaluation(int):int* returns 1, which indicates the correctness of the choice. Otherwise, the function returns 0, which indicates the falsity of choice. The int parameter of function *processTopRange Evaluation(int):int* defines the gap in percentages. More information about evaluation procedure and approaches TOP1 and TOPGAP are provided in

next chapter. The function *predictValues(int,int):void* is used to calculate the data for the array *predictedList*. As parameters, a percentage ratio of Web service response time and service operation invocation cost are passed. The function *getTheLeastRankingPredictedServiceName():String* returns the Web service with the lowest overall score and the function *getTheListOfServices InRange(int):List<String>* returns the list of Web services, the response time of which has been predicted. The overall score of these Web services differs from the lowest overall score not more that the value, which has been passed to this function as a parameter. The function *calculateValues(int,int):void* is used to calculate the values for array *actualList*. As parameters, a percentage ratio of Web service response time and service operation invocation cost is passed. The function *getTheLeastRankingActualServiceName():String* returns the Web service with the lowest overall score among the services. This overall score is based on the real (control data).

**Class *EvaluationController***

Class *EvaluationController* is used to trigger the process of accuracy measurement for the model of Web service selection. After that, this class shows the corresponding result in an appropriate component in graphical user interface. There is an interface *interfaceMainContr:MainContrInter* in this class, which provides the access to the components of graphical user interface. The function *evaluateData(File,int,int,int):void* proceeds the accuracy measurement according to the three selected approaches: RANDOM, TOP1 and TOPGAP. These approaches are observed in details in the next chapter of this work. As parameters, a file is passed, which contains the input data, the percentage ratio of the response time and operation invocation cost and finally the value of gap, required by the approach TOPGAP.

**Class *ServiceObject***

Class *ServiceObject* is used to store the data, related to Web service for the defined timestamp: the name of Web service (*serviceName:String*), the value of response time (*responceTims:int*), the cost of operation invocation (*operationCost:int*), normalized value of response time (*normalizedRespTime:float*), normalized value of the cost of operation invocation(*normalizedOpCost:float*) and finally the overall score of Web service among other Web services (*overAllRanking:float*). In addition, the difference in percentages between an overall score of Web service with the name *serviceName* and the lowest overall score among other Web services is calculated. This calculated difference is stored in variable *diffInPerc:float*. The next functions *normalizeRespTimeInArray(List<ServiceObject>):List*

*<ServiceObject>* and *normalizeCostInArray(List<ServiceObject>):List*

*<ServiceObject>* are used for normalization correspondingly the parameters

*responceTims* and *operationCost* for Web services, which are passed to these

functions as parameters. The result for each Web service is written correspondingly in variables *normalizedRespTime* and *normalizedOpCost*; the functions return the lists of Web services with defined normalized variables. There is also a function *computeOverallRanking (List<ServiceObject>, float, float):List<ServiceObject>* for calculation the overall score for Web service array and function *computePercDiff(List<ServiceObject> ):List<ServiceObject>* for computing the difference in percentage ratio between the overall score of Web service with the name *serviceName* and the lowest overall score among all available Web services.

## The Usage of Weka Framework Components

After the observation of the whole architecture, it is also worth to mention, how the Weka components have been used and in which parts they are located. In order to use Weka components there are three main steps, which should be conducted: firstly there is a need to initialize the object, which implements the corresponding machine learning algorithm, then there is a need to train this object on a training data set and finally the predictions can be made with the help of trained object. In this subchapter, all steps are observed and corresponding Java code examples are presented. For all examples, the multilayer perceptron is selected as a default machine-learning algorithm.

The initialization of multilayer perceptron object is conducted in class

*ClassifierImpl* and the corresponding code fragment is presented below:

MultilayerPerceptron machineAlgorithm = new MultilayerPerceptron();

Attribute Attribute1 =

new Attribute("timestamp"); Attribute Attribute2 =

new Attribute("responsetime");

FastVector fvWekaAttributes = new FastVector(2); fvWekaAttributes.addElement(Attribute1); fvWekaAttributes.addElement(Attribute2); Instances instancesToTrain =

new Instances("Rel", fvWekaAttributes, 10);

First, in the first line the initialization of multilayer perceptron object *machineAlgorithm* is conducted. After that, in the next two lines, the initialization of attributes *Attribute1* and *Attribute2* is done. These attributes present the timestamp of service invocation and service response time correspondingly. After that, in the next three lines, the initialization of attribute vector *fvWekaAttributes* is conducted as well as the addition of

defined before attributes *Attribute1* and *Attribute2* to this vector. Finally, with the help of attribute vector *fvWekaAttributes* the enumeration of entities *instancesToTrain* is formed, to which the new instances will be added, and these instances will be used for training the object *machineAlgorithm* of a multilayer perceptron.

The code fragment, which is used for training the multilayer perceptron is located in function *trainFromInstances(Instances):void* of the class *BackgroundModel*. The fragment of this code is presented below:

instancesToTrain.setClassIndex( instancesToTrain.numAttributes() - 1); machineAlgorithm.buildClassifier(instancesToTrain); instancesToTrain.delete();

First in the first line is defined, that the last but one attribute should be marked as a class index. After that, with the help of function *buildClassifier(),* the learning of multilayer perceptron object *machineAlgorithm* on the training dataset *instancesToTrain* is conducted. Finally, from the training set with the help of function *delete()* the instances are deleted, with the help of which the training has already been conducted.

The code fragment, which is used for predicting the service response time for the define timestamp with the help of trained multilayer perceptron object *machineAlgorithm* is located in function *classifyOneInstance(double): List<ServiceObject>* of object *RecognitionModel* and is presented below:

double date = 1431685122000d; Instance inst = new Instance(2); inst.setValue(Attribute1, date); inst.setValue(Attribute2, '?'); double predictedRespTime =

machineAlgorithm.classifyInstance(inst);

First, with the help of variable *date* the timestamp is defined, for which the response time of the service should be predicted. After that, the initialization of new instance *inst* is conducted and for this instance, the value of the variable date is set as a first attribute. The second attribute is currently unknown, because it should be predicted by trained multilayer perceptron object *machineAlgorithm*, which is done in this fragment of code by the function *classifyInstance()*. This function predicts the Web service response time *predictedRespTime* for defined timestamp *date*.

Except the multilayer perceptron also support vector regression and decision trees machine learning algorithms are used. The usage of these algorithms is the same, comparing with the described above usage of a multilayer perceptron. The only different part is the process of initialization.

The process of object initialization, which implements the support vector regression machine learning algorithm is presented below:

SMOreg machineAlgorithm = new SMOreg();

The process of object initialization, which implements the decision trees machine learning algorithm is presented below:

M5P machineAlgorithm = new M5P();

## Graphical User Interface

In this subchapter, the detailed description of the graphical user interface is presented to fulfil the overall observation of application design and structure. The user interface description is provided through the set of steps, which user should conduct to obtain the required result from the application.

For graphical user interface implementation, it was decided to use JavaFX technology. JavaFX is a set of the graphical and multimedia package that allows developers to design, build, test, debug, and deploy rich client applications that work on different platforms. This technology is quite flexible and allows writing basic application logic in Java while the graphical user interface is declared using the language JavaFX Script. Moreover, since JavaFX library written as a Java API, JavaFX application code can invoke the API from any other library written in Java. For example, the application can use JavaFX API Java library to access the specific features of the system or connect to the database server intermediate applications.

Firstly, the work with application starts from model training procedure. For that, tab *Training* should be selected. As a result, the user interface will have representation, which is shown on figure 3.10.

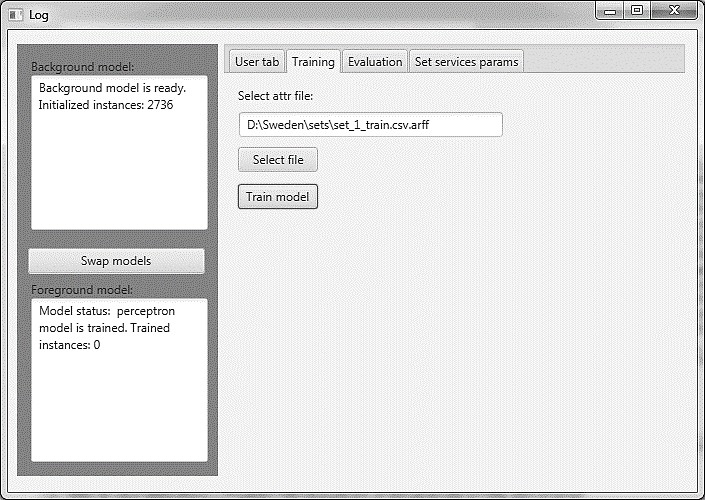


Figure 3.10: The user interface of tab *Training.*

The user can press *Select file* button and the standard dialog of operation system will appear. With the help of this dialog, the user can select the file with *.arff* extension, in which the training instances are stored. The format of the file is explained in details in next chapter. After the successful selection of a file, the absolute path to this file is shown in the field with the label *Select .arff file*. When the user pressed the *Train model* button, the process of background model training is triggered on loaded instances. After the successful training, the corresponding message appears in the field with label *Background model*. In addition, in this field is shown, on how much instances the background model has been trained on.

After conducting the training process, the user can invalidate the foreground model by pressing the button *Swap models*. The message about successful model swap will appear in the field with label *Foreground model*. After that, there is a possibility to predict the response time of Web services for defined time point. For that, the user should navigate to the tab *User tab*. As a result, the graphical user interface will have representation, shown on figure 3.11.

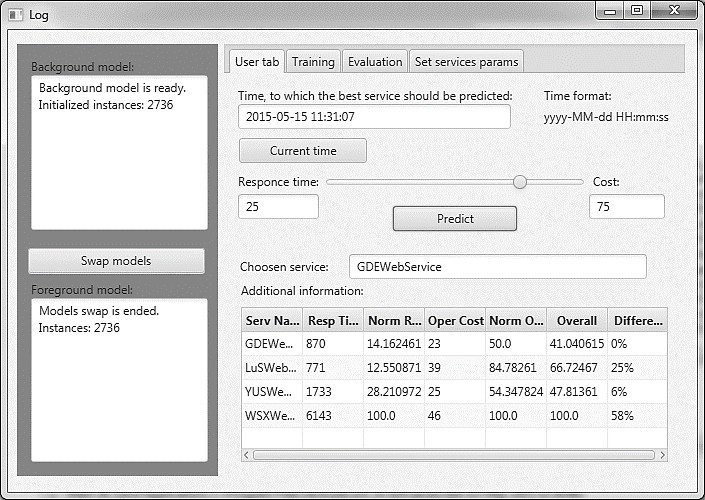


Figure 3.11: The user interface of tab *User tab.*

The time point, for which the efficient Web service should be predicted, can be defined in the field with label *Time, to which the best service should be predicted*. The format of time point is defined in the field with the label *Time format*. If the user presses the button *Current time*, the actual time will be set up as a defined time point in the field with label *Time, to which the best service should be predicted*. With the help of a slider, the user can set the ratio of response time and cost of service invocation. The corresponding values will be presented in fields with labels *Response* time and *Cost.* When the user presses the button *Predict*, the most efficient Web service will be defined and its name will be shown in the field with label *Chosen service*. Moreover, in the table will be presented intermediate data, which has been obtained during the process of most effective service determination. In this case, the cost of service invocation is generated randomly. If the user wants to define other values of invocation cost, the tab *Set services params* can be opened and its user interface is presented on figure 3.12.

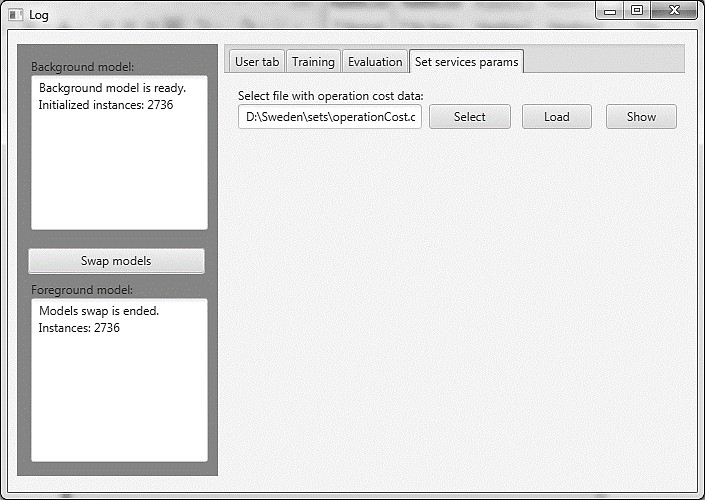


Figure 3.12: The user interface of tab *Set services params.*

After pressing the button Select file, the standard dialog of operation system appears which enables to choose the file. For this particular case, the file with extension .csv should be chosen. After the successful file selection, the absolute path to this file will appear in the field with the label *Select file with operation cost data*. When the user presses the *Load* button, the values of service invocation cost will be loaded from the file and the dialog as on figure 3.13 will be shown.

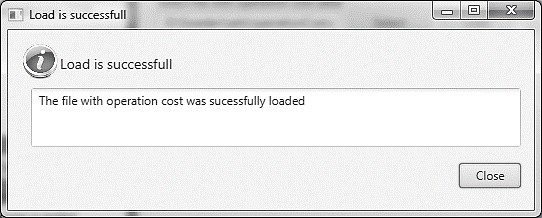


Figure 3.13: The user interface of dialog after successful loading of service invocation cost values from file.

After that, if the user will press Show button, the dialog as on figure

3.14. will appear, which will show the cost of service invocations.



Figure 3.14: The user interface of dialog, which shows the service invocation cost values.

In order to conduct the model evaluation procedure, the user should press on the tab *Evaluation*. The corresponding user interface is presented on figure 3.15.

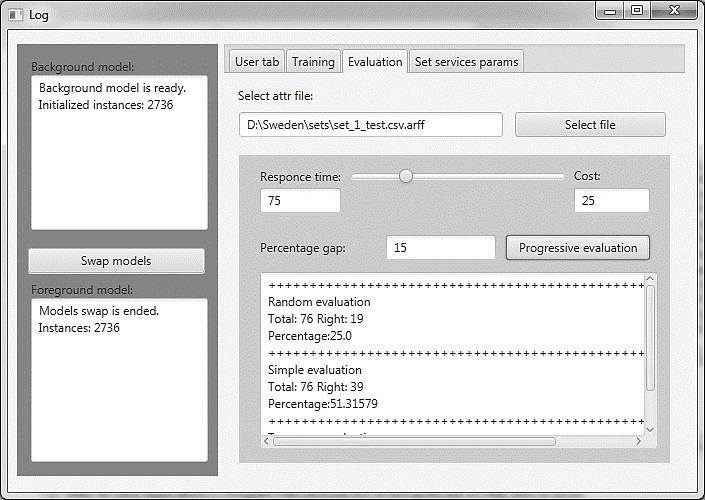


Figure 3.15: The user interface of tab *Evaluation.*

## Model Design Chapter Conclusion

Therefore, as a result of this chapter, the Web service selection system has been implemented, which is used in next chapter for Web service selection model evaluation. At the beginning of the chapter, because the machine learning algorithms are one of the core components of the developed system, the detailed description of each selected algorithms has been provided to understand algorithms functioning principle. After that, the architecture of the developed system has been described in this chapter to provide a clear understanding in what way the whole system has been built up, how it is functioning, what are the core components and their relations between each

other. This was particularly done with the help of communication diagram in subchapter 3.3 and class diagram in subchapter 3.4. Moreover, the examples of Weka components usage were presented to give an insight of how the framework with implemented machine learning algorithms had been integrated into Web service selection system. Finally, the description of the graphical user interface has been provided to understand, which steps the user conducts during the process of Web service selection.

# Model Evaluation

## Structure and Goals of Model Evaluation Section

The model evaluation chapter is split into five subchapters. In the first subchapter, the brief overview of each subchapter is presented. In the second subchapter, the accuracy measurement techniques are described. The short overview of existing techniques are presented as well as their advantages and disadvantages. Moreover, new techniques are introduced, called BEST\_FOR\_TERM and TOPGAP. In addition, the set of parameters are mentioned, which should be measured during conducting experiment. These parameters were used in efficiency estimation of machine learning algorithms. The purpose of this subchapter is to select a method or methods, which has been used to determine the effectiveness of selected machine learning algorithms. In the third subchapter, the process of data collection is presented. Here it is be discussed, how the datasets have been obtained for experiment conduction and what are the characteristics of each dataset. Moreover, in this subchapter, it is also described, how the location can effect on the dataset and based on this knowledge, the data for the third experiment have been generated. In the fourth subchapter, the process of data transformation is described. This chapter shows, how the input dataset should be transformed, so the overall model in general and Weka components, in particular, could work with it properly. Finally, in the fifth subchapter, the experiments are described because all required components are ready and presented before. The section of each experiment has a description of how the input dataset has been split into training and test set and what are the results of each experiment. As a result of this whole chapter, the results of three experiments has been obtained and based on that, in the next two chapters, these results can be interpreted and analysed and conclusions can be made.

## Overview of Accuracy Measuring Techniques

Since in experiments different models has been used, the question of model evaluation is crucial, because there is an interest in such parameter as accuracy. In classification tasks, machine learning algorithms compute accuracy relatively trivial: it is very easy to detect, whether the evaluated item was correctly assigned to a particular class or not, because output values are nominal and there is a Boolean logic here. If the object is from class1 and it has been assigned to class1, the result of detection is positive; but if the object is from class1 and has been marked as an object, which does not belong to class1 – the result of the detection is negative [40]. Moreover, these results of detection can be presented in a form of a contingency table, an example of which is shown in table 4.1.

|  |  |  |
| --- | --- | --- |
|  | Predicted: class1 | Predicted: not class1 |
| Real: class 1 | True Positive | False Negative |
| Real: not class1 | False Positive | True Negative |

Table 4.1: The example of the contingency table.

This table shows, that if the object belongs to class1 and the system has predicted, that this object also belongs to class1, the result of such detection is called True Positive (TP). On the other hand, if the object does not belong to class1 and the system has predicted exactly the same condition, the result of such prediction is called True Negative (TN). These both cases characterizes the positive result of the prediction, made by the system. In the case when system states, that the object does not belong to class1 but it according to real data it does, the result of prediction is negative and is called False Negative (FN). On the other hand, if the object does not belong to class 1, but the system detects, that the object belongs to class1, the result of detection is called False Positive (FP). Overall, the accuracy for this case can be calculated according to the following formula:

𝐴𝑐𝑐𝑢𝑟𝑎𝑐𝑦 = ∑ 𝑇𝑟𝑢𝑒 𝑃𝑜𝑠𝑖𝑡𝑖𝑣𝑒 + ∑ 𝑇𝑟𝑢𝑒 𝑁𝑒𝑔𝑎𝑡𝑖𝑣𝑒

∑ 𝑇𝑜𝑡𝑎𝑙 𝑛𝑢𝑚𝑏𝑒𝑟 𝑜𝑓 𝑖𝑛𝑠𝑡𝑎𝑛𝑐𝑒𝑠

With the regression methods, the situation of measuring accuracy is not so trivial, because these methods are working with continuous values. So, it was decided to measure accuracy by the following technique. During the process of response time prediction, the real response time of Web service was also taken into account. If the Web service with lowest predicted response time is the same as Web service with the lowest real response time, then it can be assumed that the result of the detection is positive. Otherwise, the result of detection is negative. This approach is called TOP1, which has been described in work [9]. This approach is quite trivial, however, it has a couple of disadvantages. One of the major disadvantages is that when the response time of Web services is predicted, a specific situation may occur, when two Web services can have low response time values comparing to other services and the difference between these minimum values can be 1-2

%. Factually, these two Web services can be considered as a correct choice, while a difference of 1-2% of response time is insignificant for the end user. However, it can occur that according to the real data the first service will be selected as the optimal but according to the predicted data, the second service will be selected as an optimal one. Therefore, the result of this evaluation iteration according to the approach TOP1 will be counted as a negative, while such result can be regarded as a positive because the difference 1-2% in response time is not so sufficient for the user. To cope with this problem

authors work [9] also proposed to treat the second best service as the correct choice, calling this approach TOP2.

However, in this study, the slightly different idea was brought out, which was called TOPGAP. The idea of the TOPGAP approach is more complicated that the TOP1 or TOP2 approaches. To understand it, an example should be observed. For it, the following overall score of services is presented:

*SE1 221*

*DE1 223*

*DE2 225*

*US1 320*

From this example is clear enough, that the gap between SE1, DE1 and DE2 is not big – under 1% between SE1 and DE1 and under 2% between SE1 and DE2. For the user, this gap is not significantly important, so the selection of one service from SE1, DE1 or DE2 can be accepted as a correct choice. However, the US1 service cannot be assumed as a correct selection, because the gap between SE1 and US1 is about 36.5 %. Therefore, to use TOPGAP approach, there is a need to define this percentage gap between the actual best service and the service, which can be accepted as a correct choice and can be treated as the best service. For model implementation, the default values of this gap will be equal to 7%.

Moreover, the obtained accuracy results should be compared with some reference result. The purpose of that is not only to see how good the algorithms cope with problem relatively to each other but also with the respect of real situation or other existing solutions. One the possible decision can be the comparison of experiment results with existing research results, like presented in [9]. However, despite the fact, that the model, which has been implemented in this master thesis, is based on the model, presented in resource [9] and the some of the measuring accuracy techniques are the same, the adequate comparison between results cannot be made. The main reason is that for proper comparison the same dataset or the same set of rules for dataset generation should be used. Therefore, the results of the research presented in [9] cannot be used as a reference result to evaluate the accuracy of the selected machine learning algorithms. Based on this, there is a need to develop an approach or approaches that would serve to estimate the efficiency of machine learning algorithms.

The first approach, which can be used, is called RANDOM and it has

also been presented in [9]. The essence of this approach is that at each step the most efficient Web service is chosen randomly. This approach is very simple to implement, but it has one major disadvantage – it is not based on any logical or algorithmic substantiation of a choice, which usually provides a real user. Another disadvantage is that the accuracy of this approach is reduced in proportion to the growth of Web services entities. In experiments,

the value of accuracy according to this approach was measured. However, based on the identified disadvantages, the effectiveness of machine learning algorithms were not compared to the accuracy results, obtained by the help of RANDOM approach.

As an alternative strategy, the approach called BEST\_FOR\_TERM can be used, which was developed in this study. The essence of this approach is that the user sends a request to all available Web services once and then compares the obtained response times. After that user selects the Web service with the lowest response time. Then, within a specified period of time, the user continuously selects a constant Web service with the lowest response time. After the expiration of the period, the user makes again one request to each service and selects the one with minimal response time. Again, the user chooses this service as the optimal one for the defined period of time. So, the accuracy, which has been obtained with the help of BEST\_FOR\_TERM approach, is used to estimate the accuracy of machine learning algorithms with the help of approaches TOP1 and TOPGAP.

## Input Data Collection

The data for conducting experiments has been generated. The main reason for such decision is that the process of retrieving the real data from real Web services is quite a time and resource consuming part, which is extremely difficult to be handled in the scope of this master thesis. First of all, it is very time-consuming, because for model a dataset is required, which is measured during at least several months. The other problem is that the system, which will be used to measure data, should be extremely reliable because in a case of failure there is a need either to make the measurements again or to have missing data in the dataset. Finally, unexpected or emergency situations can occur, like internet loss or hardware failure so a significant portion of data can be lost and results of experimental may be distorted.

On the contrary, generating input data with the specific tool has its own advantages: it is not so time-consuming, as a real data extraction (it takes about 5-10 minutes to generate the whole dataset). Moreover, there is no missing or misleading data in input dataset (however, missing or misleading data can be inserted in the dataset if there is a need to do so) and finally, various datasets in relation to various conditions can be generated to test different functioning scenarios.

However, there is one drawback, while generating the data with a specific tool. There is a need to define the type of each response time dataset distribution (linear, periodic, etc.). The one possible way to do that is to perform some short preliminary experiment with the usage of real Web services to obtain real data. After that, the obtained data can be analysed and based on this analysis the dataset for experiments can be generated.

Unfortunately, in the scope of this master thesis, there was no possibility to conduct such procedure due to time and resource constraint.

Another possible way is to find literature source, in which the described above experiment has already been conducted and use the evaluation results of this experiment to define the type of generated dataset. For that purpose, the article [41] has been chosen. In this article, three Web services showed different tendencies over the time: the response time of the first service almost doubled over the year; the response time of the second service stayed nearly on the same level while the response time of the third service reduced dramatically over the year. Moreover, it has been spotted, that the response time can raise during the weekends. Overall, according to the obtained information, the data for the first and second experiment has been generated, the graph of which is described on figure 4.1.

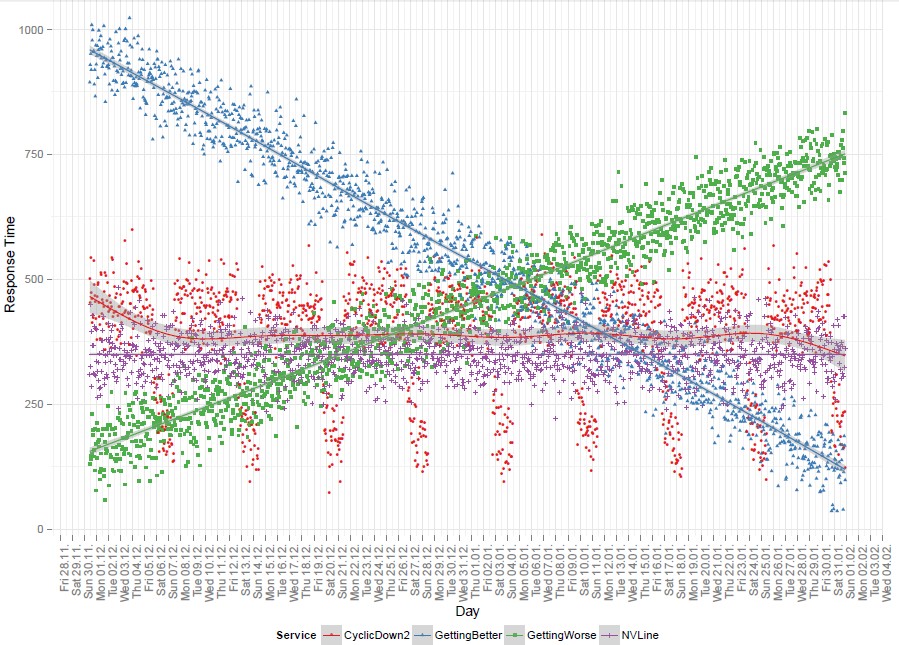


Figure 4.1: The generated response time within two months for four strategies.

In overall, four services have been presented with different response time strategy:

* CyclicDown2 – the response time of this service stays on the same

level during the time. However, there are peaks, when the response time drops to the certain level. The peak occurs once a week and lasts one day.

* GettingBetter – the response time of this service decreases linearly during the time.
* GettingWorse – the response time of this service increases linearly during the time.
* NVLine – the response time of this service stays on the same level during the time.

The structure of each dataset is the following:

* Response time is given for each service for each timestamp;
* Response time is generated for the period of 2 month;
* Response time is generated for every 1 hour;
* Overall number of instances: 4 services × 24 times pro day × 60 days = 5760.

For the third experiment, there was an aim to generate such dataset, which the response times of which were influenced by the user’s and Web service’s locations. Firstly, there was an idea to calculate the distance between user and Web service and use it in weight function with an appropriate coefficient. To implement that the URL addresses of services from their WDSL files can be retrieved. With the help of URL, the location of service can be figured out – to which country it belongs to and to which time zone. Then a map of data can be stored with the help of which the distance between user and service locations can be figured out. The one issue with this approach is that, if the USA is taken as an example, the distance between the western and southern part of this country is around 4000 kilometres. Because the time zones from URL of Web service can also be retrieved, there is a possibility to split large countries like the USA into two or more zones. There was an idea to store the data in a format “DE – USA1 = 4000” and use 4000 as a distance in kilometres if the user is located in Germany and the service is on the north side of the USA. Then there was a need to understand how to use the value of distance in calculations. The possible way was to introduce the distance value in weight function with defined coefficient. The coefficient should be chosen in such way, that the distance was not decisive, but rather a correction factor.

However, at this stage, one problem has been spotted and for its identification, one example should be considered. Suppose there is an arbitrary Web service with response time 256 ms for a certain timestamp. The distance between this Web service and the user is taken as zero: Web service and the user is located in one place and is assumed that both of them are in Sweden. Next, the technically identical copies of this Web service are placed in three remote locations: in Germany, USA and China. Therefore, for a user who is located in Sweden, the response times of respective services will be as presented in table 4.2 and on figure 4.2.

|  |  |
| --- | --- |
| Web Service location | Response time for user |
| Sweden | 256 |
| Germany | 256 + x |
| USA | 256 + y |
| China | 256 + z |

Table 4.2: The response time of services for the user, who is located in Sweden.



+ z

Sweden

China

256



+ 0

+ y

Sweden

256

+ x

USA

256

Germany 256

Figure 4.2: The schema of services and their response time for user, who is located in Sweden.

The values x, y and z will be called the delay time and the value 256 will be called the “basic response time”. The corresponding dataset can also be generated, which will describe the following situation. Therefore, the data set that was generated for the experiment does not contain the delay time. The task here is to compute the delay time for each service, according to the user location. It will be assumed that fibre optic cables provide the transmission of the signal. The speed of light in such cables is about 200 000 km/h and there is 5 ms delay for each 1000 km of transportation [42].

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Sweden | Germany | USA | Argentina |
| Sweden | 0 | 804 | 6638 | 12554 |
| Germany | 804 | 0 | 6710 | 11900 |
| USA | 6638 | 6710 | 0 | 8392 |
| Argentina | 12554 | 11900 | 8392 | 0 |

Table 4.3: The distance between the capitals of corresponding countries in km.

According to the distances between capitals, presented in table 4.3, and light speed in fibre cables, the delay time for each distance can be calculated and the results can be stored in table 4.4.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Sweden | Germany | USA | Argentina |
| Sweden | 0 | 4 | 33 | 63 |
| Germany | 4 | 0 | 34 | 60 |
| USA | 33 | 34 | 0 | 42 |
| Argentina | 63 | 60 | 42 | 0 |

Table 4.4: The rounded values of delay time for transmitting the signal between the capitals of corresponding countries.

When calculating the delay time values, such factors have not been taken into account:

* + 1. The signal is often transmitted over the networks, which are designed not in a straight line not directly and often in a broken line. Therefore, the distance between start and end destinations is higher and correspondingly higher the value of delay time.
    2. Between starting and ending point there can be some intermediate devices for routing and these devices can also increase the value of delay time.

Because our dataset is not real but generated and no reports were found, which could provide the possible ways to simulate the delay time due to non- linear signal transmission or presence the intermediate devices, in experiment these values will be neglected and the calculated delay time will be used from table 4.2. This data will be used in the first and second experiments.

Therefore, location parameter was applied in the experiment indirectly. In the article [43], the periodicity of Web service response time was demonstrated. Despite the fact that in this article 4 different strategies of Web service selection were used: Random, Latency, BW and Probe, the overall tendency of response time for all 4 strategies was quite similar: the highest response times were measured during the working days from 9 am until 5 pm with the peak of 3 pm. The chart of such peaks is presented on figure 4.3.

Median time (sec)



Figure 4.3: The chart of the response time of one web service during the day with peaks.

Because different services are located in different time zones, the peaks of their response times can occur non-simultaneously. Therefore, the third experiment focuses on the question of how well the machine learning algorithms recognize the periodicity in the dataset and what are the possible ways to optimise the dataset to handle the periodical data better.



9

8

7

6

5

4

3

2

1

0

1:30 AM 4:30 AM 7:30 AM 10:30 AM 1:30 PM 4:30 PM 7:30 PM 10:30 PM

Time of the day

Random Latency BW Probe

The structure of dataset for the third experiment was changed completely, comparing to the dataset for the first and second experiment: the response time data for only one Web service was generated. This decision was made because otherwise the response time for the four Web services has to be generated linearly with periodic peaks at different times for each Web service. This can lead to the situation, when, along the whole timeline of generated data, according to the TOP1 strategy the best Web service will be chosen not from four but from two Web services: either Web service with the lowest linear response time or Web service with lowest peak response time. Generation of data for one Web service and subsequent usage of it for the experiment will not allow calculating such parameter as accuracy according to the user strategies or strategies TOP1 and TOPGAP. However, due to the built graphs and calculated deviation there is a chance to see, how the selected machine learning algorithms handle with periodic data. Moreover, another decision has been made to generate the response time data for Web service with a periodicity of 60 and 15 minutes to analyse, whether the increase in the number of occurrences of a positive influence on the result of algorithms. Therefore, there were two datasets: the response time of the first

one was measured every 60 minutes, the response time of the second – every 15 minutes. The measurement was conducted for the period of one week. The overall sizes of datasets are the following:

The first dataset, measurement every 60 minutes: 1 measurement per hour × 24 hours × 1 service × 7 days = 168 instances.

The second dataset, measurement every 15 minutes: 4 measurements per hour × 24 hours × 1 service × 7 days = 672 instances.

## Input Data Transformation

In order that the machine learning implementations from Weka framework can function on the input data set, the last one should be transformed into a specific format, which is discussed in this subchapter. The output data of the generator tool is saved in CSV file. Each line contains the timestamp in format “dd/mm/YYYY HH:MM”, then a name of service and a response time value of this service for defined timestamp. To make the functioning of the model proper, this file should be converted into .arff file, with which the Weka framework API is working. Moreover, despite the fact that all date values are converted into numerical values in Weka framework, one selected machine learning algorithm (M5P) is not working with values of type “date”. Therefore, the date value should be converted into numerical value. Moreover, the .arff file contains some fields with meta-data, so the CSV file has been converted into .arff file with the following result:

*@relation ServiceSelector*

*@attribute serviceInstance {WSXWebService, LuSWebService, YUSWebService, GDEWebService}*

*@attribute timestamp numeric @attribute responseTime numeric*

*@data "LuSWebService",1430661600,656 "YUSWebService",1430661600,1625 "GDEWebService",1430661600,627 "WSXWebService",1430661600,4526 "LuSWebService",1430665200,701*

Description of .afrr file core elements: after the keyword @relation there is a name of the relation, represented in this file. After keywords @attribute there are attributes and their types, which are used in this relation. For observable file, the attributes are the following:

* + 1. serviceInstance – the attribute, which defines the service name. It is a text, label value. In curly braces after the name of attributes, there is a list of possible values of this attribute.
    2. timestamp – the attribute, which defines the point of time, when the response time for the *serviceInstance* has been measured.
    3. responseTime – the attribute, which defines the measured response time for *serviceInstance* in defined *timestamp*.

After the tag @data on each line, there is a data of single instance with defined set of attributes in defined order.

To make a conclusion, the transformation part of input dataset is completed and the obtained .arff file is ready to be loaded into implemented model and used for either training or evaluation of the selection model.

## Conduction of Experiments

### First Experiment Conduction

The first experiment was conducted with the aim to simulate real user behaviour. The main idea is that user not always sends a response to the service, but after a certain period of time. For that, the generated dataset was split into training and test set with the ratio 9:1. Therefore, the model first trains on nine instances and then check the result on the 10-th instance. In the experiment, 100 datasets have been used, each contains 5760 instances. The accuracy and its deviation are presented in table 4.5 and 4.6 correspondingly.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | RANDOM | BEST\_FOR\_TERM | TOP1 | TOPGAP |
| MLP | 25.02 | 24.95 | 66.04 | 74.74 |
| SMOReg | 25.02 | 24.95 | 69.84 | 75.65 |
| M5P | 25.02 | 24.95 | 81.12 | 89.54 |

Table 4.5: The accuracy results for the first experiment.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | RANDOM | BEST\_FOR\_TERM | TOP1 | TOPGAP |
| MLP | 0 | 0 | 2.64 | 2.85 |
| SMOReg | 0 | 0 | 2.38 | 1.77 |
| M5P | 0 | 0 | 2.29 | 1.70 |

Table 4.6: The standard deviation of accuracy results for the first experiment.

Despite the fact that measuring and estimating execution time is not the goal of this project, it is useful to get an approximate idea, how much time does each algorithm require for task execution. The corresponding results are presented in table 4.7.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Real | Real in seconds | Normalized |
| MLP | 19 min 56 sec | 1196 | 0 |
| SMOReg | 33 min 9 sec | 1989 | 0.52 |
| M5P | 45 min 21 sec | 2721 | 1 |

Table 4.7: The execution time of algorithm work for the first experiment.

### Second Experiment Conduction

In the second experiment, the number of datasets was reduced to ten; however, the structure of each dataset remained the same. The evaluation procedure was also changed. In the first experiment, the model was trained on nine instances and checked on the 10-th instance. In this experiment, a model first evaluated on the i-th instance and then trained on i-th instance. This evaluation procedure gives a huge advantage: response time is predicted for each time point, so there is a possibility to build graphs, see the deviation between real and predicted response time on each step.

The accuracy and its deviation are presented in table 4.8 and 4.9 correspondingly.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | RANDOM | BEST\_FOR\_TERM | TOP1 | TOPGAP |
| MLP | 24.78 | 51.30 | 64.15 | 76.30 |
| SMOReg | 25.75 | 51.30 | 69.42 | 75.72 |
| M5P | 24.69 | 51.30 | 81.13 | 89.65 |

Table 4.8: The accuracy results for the second experiment.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | RANDOM | BEST\_FOR\_TERM | TOP1 | TOPGAP |
| MLP | 0 | 0 | 0.95 | 0.61 |
| SMOReg | 0 | 0 | 1.12 | 0.44 |
| M5P | 0 | 0 | 1.06 | 0.63 |

Table 4.9: The standard deviation of accuracy results for the second experiment.

In contrast to the first experiment, in the second it is also possible to calculate the standard deviation of the response time for each algorithm. The standard deviation for algorithm MLP is presented in table 4.10.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **CyclicDown2** | **GettingWorse** | **GettingBetter** | **NVLine** |
| Set 1 | 74.14 | 31.47 | 31.56 | 28.52 |
| Set 2 | 76.77 | 31.04 | 32.57 | 21.18 |
| Set 3 | 74.47 | 30.41 | 31.32 | 27.82 |
| Set 4 | 74.12 | 31.33 | 31.46 | 27.21 |
| Set 5 | 76.29 | 30.18 | 32.27 | 27.3 |
| Set 6 | 76.38 | 29.34 | 31.96 | 28.78 |
| Set 7 | 75.52 | 30.00 | 33.12 | 29.04 |
| Set 8 | 75.35 | 30.63 | 31.42 | 28.40 |
| Set 9 | 76.54 | 30.89 | 32.34 | 29.00 |
| Set 10 | 76.16 | 30.98 | 32.15 | 28.08 |
| **Average** | **75.574** | **30.627** | **32.017** | **27.533** |

Table 4.10: Standard deviation for MLP algorithm.

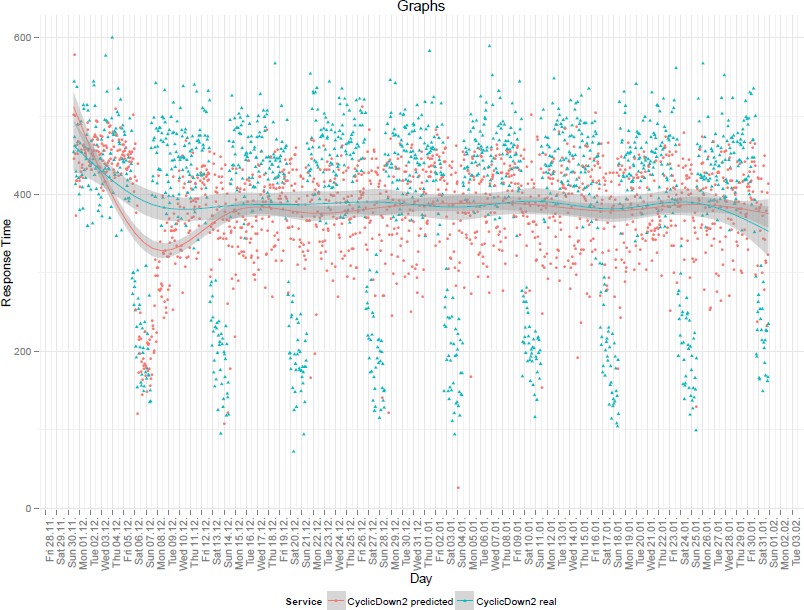
On figure 4.4. and 4.5., the real and predicted response times of MLP algorithm correspondingly for CyclicDown2 and GettingBetter generation strategies are presented.

Figure 4.4: Real and predicted response times of MLP algorithm for CyclicDown2Periodic generation strategy.

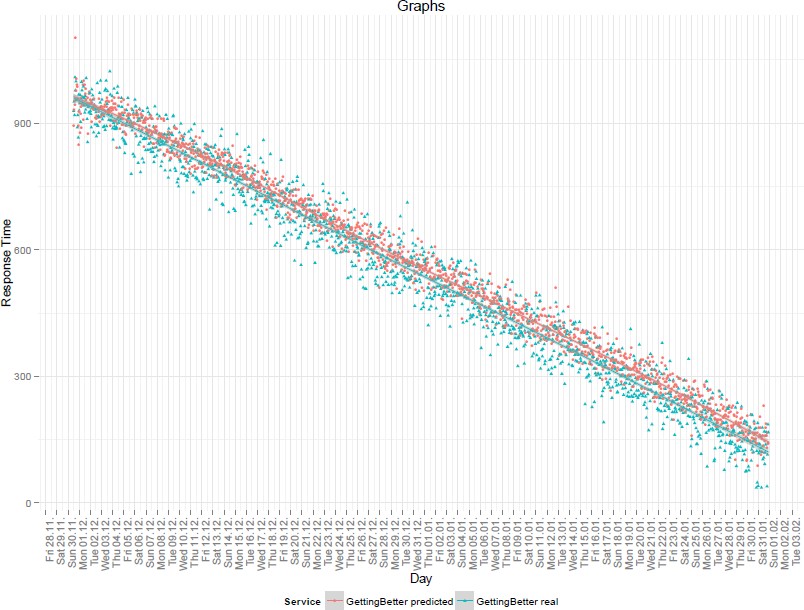


Figure 4.5: Real and predicted response times of MLP algorithm for GettingBetter generation strategy.

The standard deviation for algorithm SMOReg is presented in table

4.11.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **CyclicDown2** | **GettingWorse** | **GettingBetter** | **NVLine** |
| Set 1 | 84.68 | 25.47 | 25.81 | 25.76 |
| Set 2 | 84.88 | 25.26 | 26.67 | 24.55 |
| Set 3 | 85.15 | 25.12 | 25.62 | 24.96 |
| Set 4 | 82.98 | 25.38 | 25.16 | 25.21 |
| Set 5 | 86.96 | 25.20 | 25.53 | 24.83 |
| Set 6 | 83.94 | 25.34 | 25.72 | 26.02 |
| Set 7 | 86.89 | 25.40 | 26.24 | 26.39 |
| Set 8 | 85.25 | 24.42 | 25.17 | 25.68 |
| Set 9 | 84.96 | 25.83 | 26.05 | 26.18 |
| Set 10 | 85.48 | 25.89 | 25.51 | 25.40 |
| **Average** | **85.117** | **25.331** | **25.748** | **25.498** |

Table 4.11: Standard deviation for SMOReg algorithm.

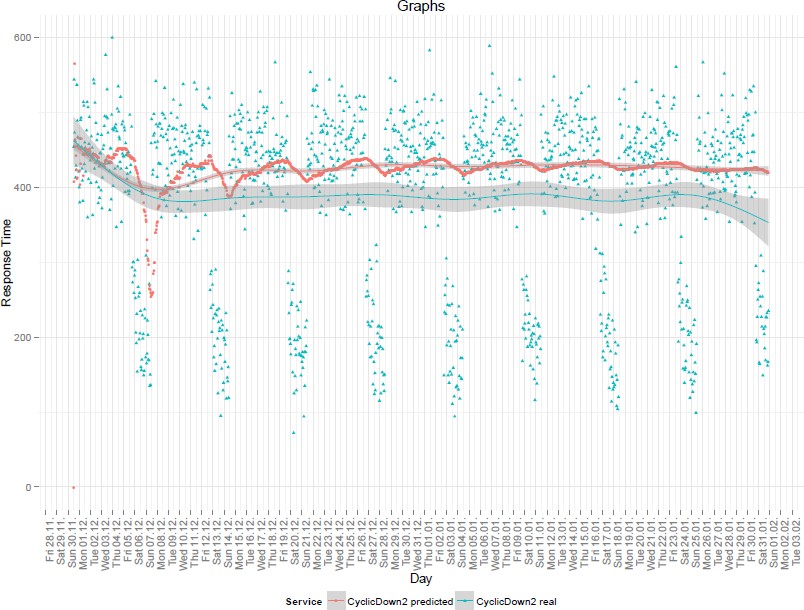
On figure 4.6 and 4.7, the real and predicted response times of SMOReg algorithm correspondingly for CyclicDown2 and GettingBetter generation strategies are presented.

Figure 4.6: Real and predicted response times of SMOReg algorithm for CyclicDown2Periodic generation strategy.

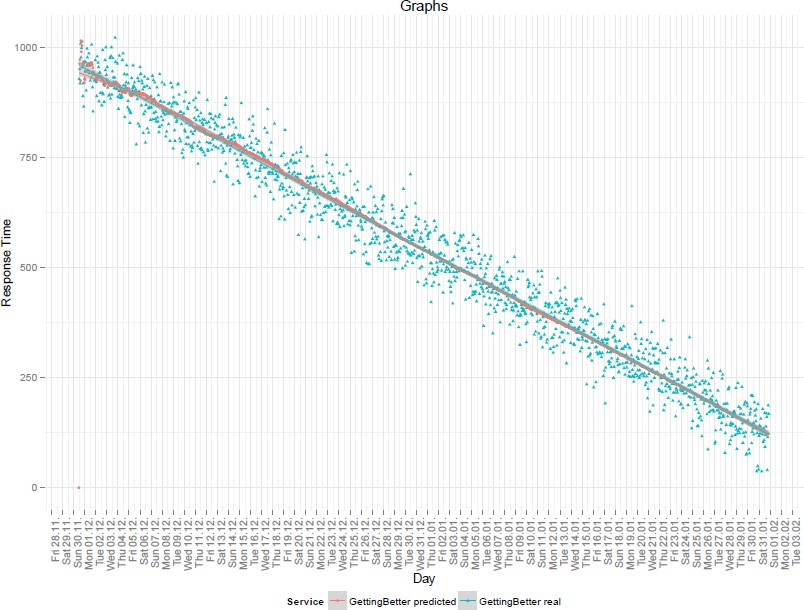


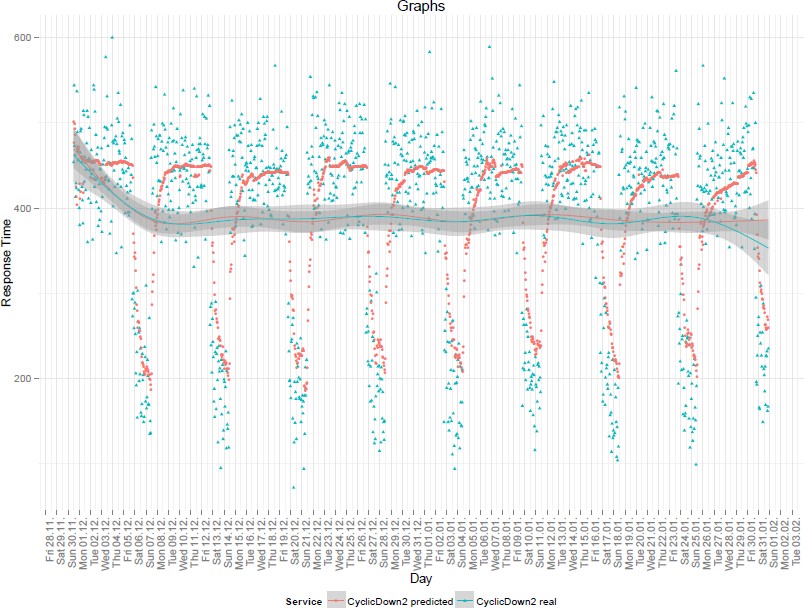
Figure 4.7: Real and predicted response times of SMOReg algorithm for GettingBetter generation strategy.

The standard deviation for algorithm M5P is presented in table 4.12.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **CyclicDown2** | **GettingWorse** | **GettingBetter** | **NVLine** |
| Set 1 | 46.92 | 25.37 | 25.78 | 24.78 |
| Set 2 | 47.37 | 25.47 | 26.58 | 24.46 |
| Set 3 | 48.51 | 24.79 | 25.77 | 24.81 |
| Set 4 | 47.43 | 25.29 | 24.92 | 25.05 |
| Set 5 | 49.19 | 24.92 | 25.47 | 24.55 |
| Set 6 | 47.44 | 25.10 | 25.86 | 25.72 |
| Set 7 | 47.49 | 25.42 | 26.09 | 26.11 |
| Set 8 | 48.29 | 24.39 | 25.03 | 25.83 |
| Set 9 | 48.78 | 25.28 | 25.67 | 25.53 |
| Set 10 | 48.13 | 26.00 | 25.52 | 25.24 |
| **Average** | **47.955** | **25.203** | **25.669** | **25.208** |

Table 4.12: Standard deviation for M5P algorithm.

On figure 4.8 and 4.9, the real and predicted response times of M5P algorithm correspondingly for CyclicDown2 and GettingBetter generation strategies are presented.



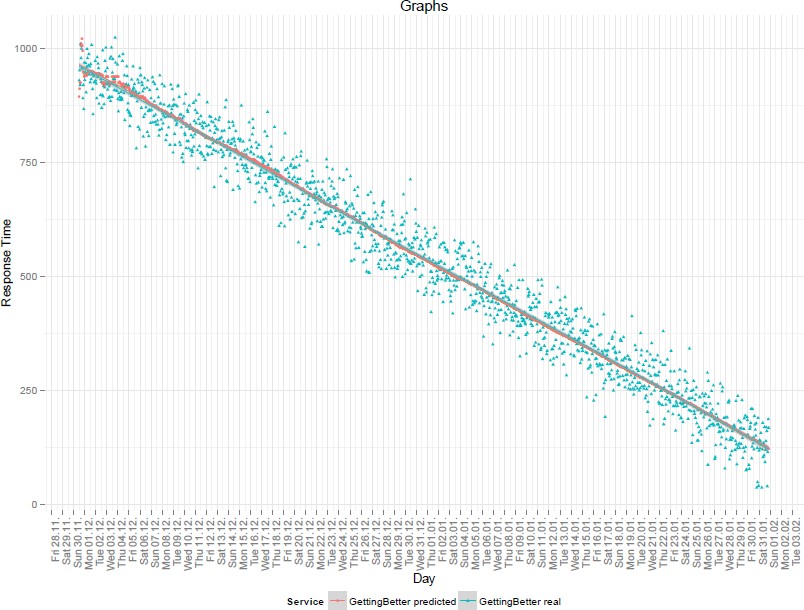
Figure 4.8: Real and predicted response times of M5P algorithm for CyclicDown2Periodic generation strategy.

Figure 4.9: Real and predicted response times of M5P algorithm for GettingBetter generation strategy.

The average values of real and predicted response times for each dataset have also been calculated and corresponding data is presented in tables 4.13 and 4.14.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **CyclicDown2** | **GettingWorse** | **GettingBetter** | **NVLine** |
| Set 1 | 389 | 451 | 541 | 350 |
| Set 2 | 390 | 449 | 540 | 351 |
| Set 3 | 394 | 449 | 541 | 350 |
| Set 4 | 391 | 450 | 542 | 350 |
| Set 5 | 391 | 451 | 539 | 351 |
| Set 6 | 391 | 451 | 539 | 349 |
| Set 7 | 392 | 449 | 540 | 350 |
| Set 8 | 391 | 451 | 539 | 351 |
| Set 9 | 389 | 448 | 541 | 349 |
| Set 10 | 389 | 451 | 540 | 350 |
| **Average** | 390.7 | 450 | 540.2 | 350.1 |

Table 4.13: The average values of real response times for generation strategies.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **CyclicDown2** | **GettingWorse** | **GettingBetter** | **NVLine** |
| Set 1 | 381 | 437 | 560 | 349 |
| Set 2 | 383 | 436 | 552 | 350 |
| Set 3 | 385 | 432 | 554 | 350 |
| Set 4 | 383 | 434 | 560 | 351 |
| Set 5 | 384 | 436 | 557 | 348 |
| Set 6 | 388 | 436 | 554 | 347 |
| Set 7 | 385 | 435 | 556 | 348 |
| Set 8 | 382 | 434 | 557 | 350 |
| Set 9 | 386 | 440 | 557 | 351 |
| Set 10 | 381 | 434 | 558 | 350 |
| **Average** | 383.8 | 435.4 | 556.5 | 349.4 |

Table 4.14: The average values of predicted response times for generation strategies.

### 4.5.3 Third Experiment Conduction

The third experiment showed how algorithms could handle with rush hour peaks. In this experiment, the order of the evaluation procedure remained the same as in the second experiment: the model first evaluated on the i-th instance and then trained on the i-th instance. The standard deviation for algorithms for first and second datasets are presented in tables 4.15 and 4.16 correspondingly.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **MLP** | **SMOReg** | **M5P** |
| Set 1 | 87.6 | 73.36 | 64.30 |
| Set 2 | 84.51 | 72.42 | 61.02 |
| Set 3 | 78.9 | 73.92 | 62.00 |
| Set 4 | 87.21 | 68.30 | 65.73 |
| Set 5 | 88.09 | 73.70 | 64.44 |
| Set 6 | 88.38 | 76.06 | 66.72 |
| Set 7 | 83.20 | 71.09 | 65.63 |
| Set 8 | 84.20 | 70.82 | 60.96 |
| Set 9 | 85.03 | 75.67 | 61.68 |
| Set 10 | 85.36 | 81.70 | 67.96 |
| **Average** | **85.25** | **73.70** | **64.04** |

Table 4.15: The standard deviation for algorithms for the first dataset.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **MLP** | **SMOReg** | **M5P** |
| Set 1 | 74.80 | 84.46 | 58.81 |
| Set 2 | 79.03 | 81.67 | 57.72 |
| Set 3 | 78.35 | 82.46 | 56.29 |
| Set 4 | 80.41 | 83.89 | 58.63 |
| Set 5 | 77.52 | 84.67 | 58.25 |
| Set 6 | 83.25 | 88.16 | 57.88 |
| Set 7 | 79.24 | 85.82 | 59.43 |
| Set 8 | 78.27 | 78.75 | 57.71 |
| Set 9 | 79.72 | 85.08 | 59.81 |
| Set 10 | 80.86 | 83.62 | 57.54 |
| **Average** | **79.15** | **83.86** | **58.21** |

Table 4.16: The standard deviation for algorithms for the second dataset.

On figure 4.10, 4.11 and 4.12 the real and predicted response times for the first dataset for algorithms MLP, SMOReg and M5P are presented correspondingly.

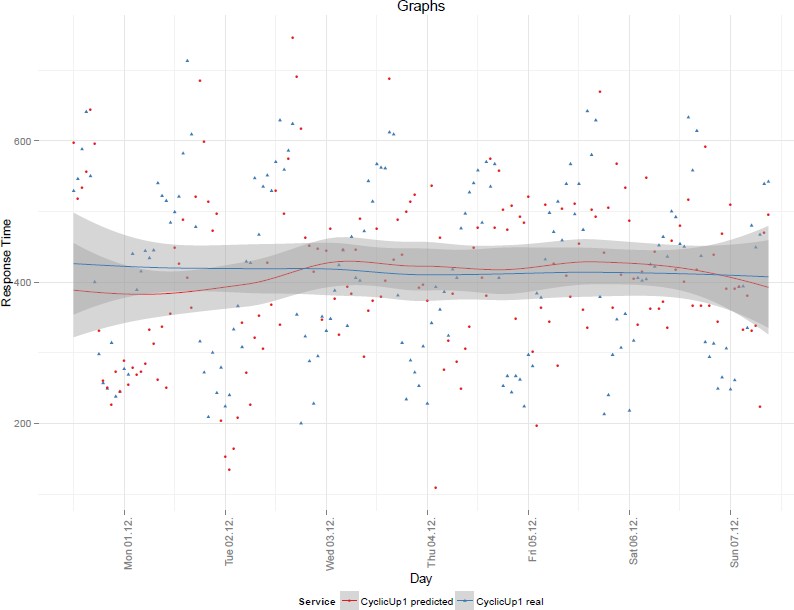


Figure 4.10: Real and predicted response times for the first dataset of MLP algorithm.

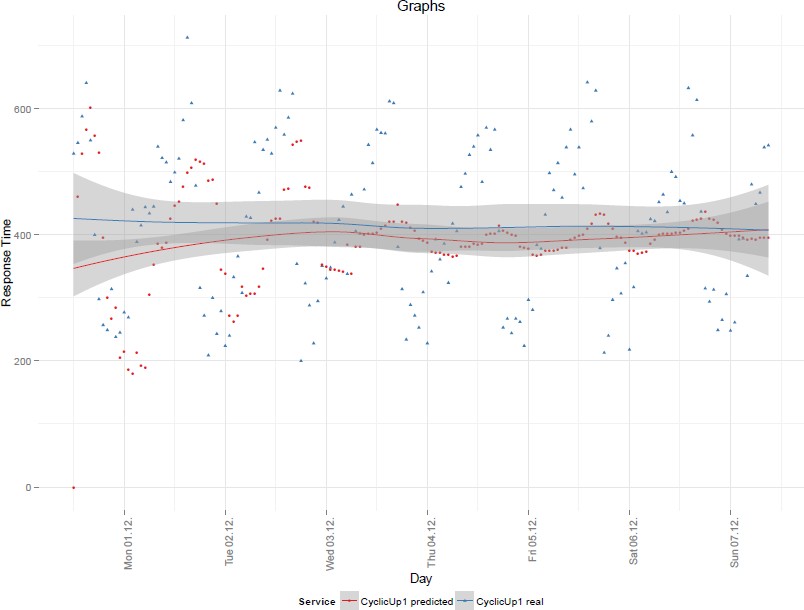


Figure 4.11: Real and predicted response times for the first dataset of SMOReg algorithm.

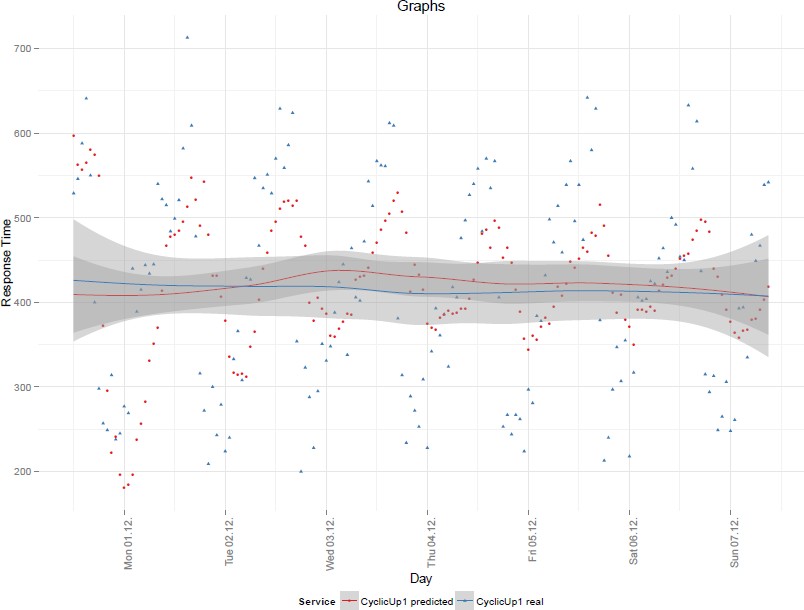


Figure 4.12: Real and predicted response times for the first dataset of M5P algorithm.

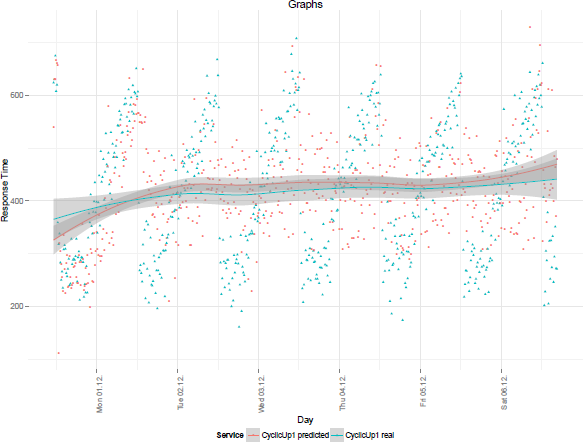
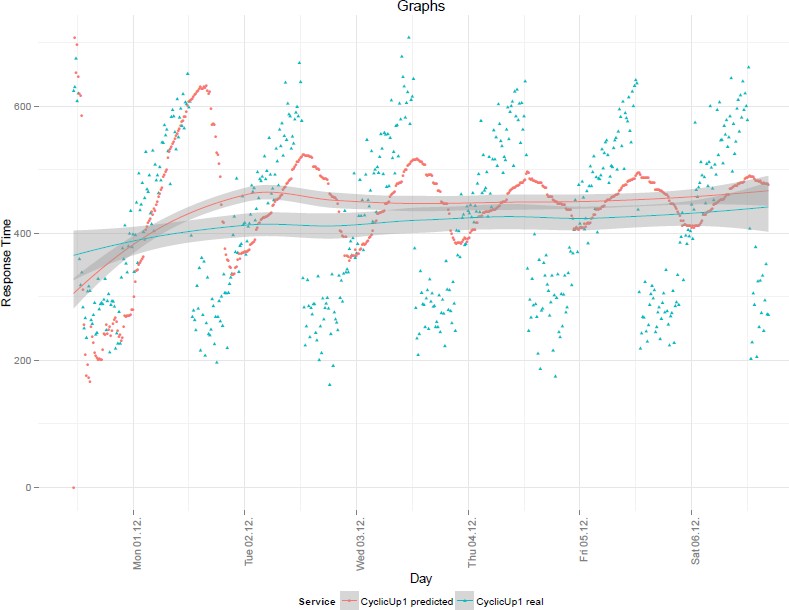
On figure 4.13, 4.14 and 4.15 the real and predicted response times for the second dataset for algorithms MLP, SMOReg and M5P are presented correspondingly.

Figure 4.13: Real and predicted response times for the second dataset of MLP algorithm.



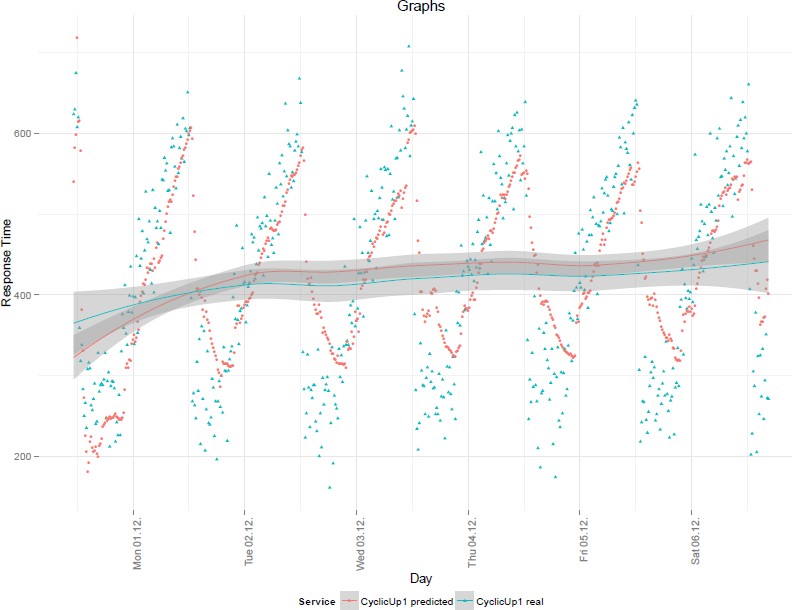
Figure 4.14: Real and predicted response times for the second dataset of SMOreg algorithm.

Figure 4.15: Real and predicted response times for the second dataset of M5P algorithm.

# Analysis

## First Experiment Analysis

Therefore, as a result, the first experiment has shown that the best accuracy has the algorithm M5P: in general, it showed 81% of accuracy according to the methodology TOP1 and 89% according to the methodology TOPGAP. It is at least 10-13% better than SMOReg and 15% better than MLP.

To support the defined claims, the statistical test was conducted, as described in resource [44]. There is a need to prove, that M5P algorithm has better accuracy comparing to two other algorithms. This can be done by proving that the difference between the average values of three algorithms accuracy is statistically significant. For that, the P-value of statistical test should be lower than or equal to 0.05. Because there are three values, which should be compared, the T-test has been conducted first for the pair MLP- M5P and then for the pair SMOReg-M5P. The data for calculations was be taken from table A.1. in the appendix. By using the online tool [45], the P- value has been calculated and it is less than 0.0001 for both cases so that it can be claimed, that there is a difference between obtained accuracy values for pairs MLP-M5P and SMOReg-M5P in the first experiment. Therefore, algorithm M5P can be claimed as the algorithm with the best accuracy.

However, despite the fact, that this experiment simulates the real user behaviour, it is very difficult to retrieve more additional information, which can give a better understanding of how good the selected algorithms work on presented dataset. In addition, the user strategy works here not so well, because the gap between user choices is too big.

## Second Experiment Analysis

According to the results of the second experiment, the following points can be highlighted:

* According to the deviation results, which have shown how the values may be different from the average, the SMOReg and M5P

algorithms cope well with linear data. Deviations of predicted data are between 25.203 and 25.748. This amounts to approximately 4.6% to 6% of the average value. At this point, a T-test has been conducted to prove that difference between two values is statistically significant. By using the online tool [45], the P-value for pair SMOReg-M5P has been calculated and it is 0.414320. In this case, it can be said, that there is no significant difference between average values and both algorithms SMOReg and M5P have the same average deviation on linear data. On the contrary, MLP has slightly worse results. Deviations of predicted data are between 27.533 and 32.017. This corresponds to approximately

5.7% to 7.9% of the average value. Calculating the P-values for corresponding pairs MLP-SMOReg and MLP-M5P the values less than 0.0001 were obtained. Therefore, for the linear dataset, the MLP can be claimed as the worst algorithm out of three while algorithms SMOReg and M5P showed significantly equal results.

* According to the deviation results, all three algorithms have the

higher value of deviation for cyclic data comparing to the linear data. Algorithm M5P has the best results among other algorithms: 47.955 and 12,5% from average and this has been confirmed by statistical testing: the P-value for the pairs MLP-M5P and SMOReg-M5P is less than 0.0001 for both cases

* According to the accuracy results, the algorithm M5P has again the

best results (P-value is less than 0.0001 for both pairs MLP-M5P and SMOReg-M5P). However, now comparing to the first experiment, there is a clear explanation of this evidence: M5P algorithm copes better with cyclic data rather that other three algorithms and better with linear data rather than MLP.

## Third Experiment Analysis

As a conclusion for the third experiment, it can be stated, that the algorithm M5P shows the best performance on the periodical set of data. This is demonstrated by the lowest values of deviation for the first data set (64.04) and the second (58.21). This is a significant difference comparing to the results of MLP and SMOReg, which has been confirmed by statistical testing: the P-value for the pairs MLP-M5P and SMOReg-M5P is less than 0.0001 for both cases and both datasets: the 15-minute and 60-minute datasets. In addition, on the decrease of the parameter deviation from 64.04 to 58.21, can be concluded that the increase of entities number in the dataset had a positive impact on the work of the algorithm. At the same time, as it can be observed from the graphs on figure 4.11 and 4.14 as well as from deviation values, the increase of the entities number in the dataset have had a negative impact for SMOReg algorithm. According to the graphs on figure 4.11 and

4.14 it can be observed, that over time the predicted by algorithm response time eliminates periodic peaks and tends to a linear distribution. Therefore, over time, the efficiency of this algorithm on a periodic data set will deteriorate.

High values of deviation for the multilayer perceptron show that from three selected algorithms this has the worst efficiency on processing the periodic data. The distribution of predicted response times can be observed in figure 4.13 and except from one peak, which has occurred on Monday, the remaining predicted by this algorithm response times is grouped neither around peaked nor linear segments of the real response times of service, but

rather lie between the mean values of these two segments. This is a clear demonstration of the fact that this algorithm has such high deviation.

## 5.4 Reliability

To provide the consistency of obtained results, for each experiment several datasets were prepared and used. After that, the overall results were presented as mean values, which then were used to make suggestions and conclusions. To prove the reliability of conclusions, which are based on obtained results, the statistic testing has been done.

In addition, considering the fact that the input data for the experiment (response time of each Web service) has not been received in real conditions but generated, a question of matching this data to real conditions can be raised. In general, the response time of a Web service is quite an arbitrary value: no clear laws or formulas have been found, according to which the exact value of service response time can be identified and calculated. However, there are some patterns that can affect the final response time of Web service. Some of these patterns have been mentioned in this work: increase of response time of Web service while increasing the distance between the Web service and the end user and also periodic variation of the response time during a day. Generation of input dataset for the experiments with respect to these characteristics allows simulating actual conditions of individual cases.

In addition, it is worth mentioning that the model has been implemented with the help of the framework Weka. This framework is sufficiently powerful and popular: updated versions of framework are regularly added to the resource Sourceforge; the book «Data Mining: Practical Machine Learning Tools and Techniques» [47] is written, where practical tasks and exercises are implemented using this framework; the development of the framework is supported The University of Waikato, New Zealand. Due to these facts, it can be argued that the implementation of machine learning algorithms in this framework is made at a high level and carefully tested.

# Discussion

According to the results of three conducted experiments, it can be concluded that the algorithm M5P is the most effective of the three observed algorithms. In the first two experiments, this algorithm had the highest accuracy according to the approach TOP1 – around 81% and also according to the approach TOPGAP – around 89%. In addition, in the third experiment, the analysis of graphs and parameter deviation demonstrated that this algorithm had the best results on the periodic dataset. All these claims were also supported by the results of statistical testing: t-Test was conducted several times for appropriate pairs of algorithms and their result values: either accuracy or standard deviation. Therefore, from three machine learning algorithms, the algorithm M5P should be used in the model of automated service selection, since this algorithm is the most efficient of the three considered.

In addition, the usage of this algorithm in the model of automated service selection allows achieving the goal of this thesis: improve the reliability of service selection with the efficiency ratio of service response time and cost indicators. According to the user strategy, the accuracy of selecting an efficient Web service was about 51.3%, while the usage of an algorithm M5P improved the accuracy to 81% according to the approach TOP1 and to 89% according to the approach TOPGAP. This also leads to the answer to the stated research question:

*In what way can the most efficient Web service be selected automatically with at least 80% accuracy?*

Therefore, in this work, the model has been built, which is based on M5P algorithm and consequently can provide the automatic service selection with 80% accuracy or more.

An approximate comparison of the results with the results obtained in

[9] can also be conducted. According to this work, the most efficient algorithm is FIMT-DD. Its accuracy according to the strategy TOP1 was 69%. At the same time, the results of this study had showed that the algorithm M5P was the most efficient – 81% accuracy according to strategy TOP1. Although there is no possibility to make an unambiguous conclusion that M5P is the most efficient choice for the selection model of Web services among the algorithms presented in this paper and in [9] (to evaluate the accuracy different sets of data have been used), the probability of such an outcome can be estimated as extremely high. At the same time, it should also be noted that the accuracy of the algorithm FIMT-DD according to the approach TOP2 is about 93%, which is higher than the accuracy of the algorithm MP5 according to the approach TOPGAP (89%). As it turned out during the design of TOPGAP approach, TOP2 approach may have one major drawback: it may be a situation where the difference between response time of the most efficient Web service and the second most efficient Web

service is 10% or more. From a user perspective, this difference is significant, and the second Web service cannot be considered as effective one. The approach TOPGAP sticks to exactly such logic. However, in this case, the approach TOP2 will define both services as effective ones. As a result, in some cases, the accuracy of TOP2 approach can be higher than the accuracy of approach TOPGAP.

# Conclusion

The aim of this work was to improve the reliability of service selection with the efficiency ratio of service response time and cost indicators that best fitted the requirements of the user. To achieve the following goal, the set of three experiments were conducted. To provide that, first, the analysis of currently existing service selection methods was performed. As a result of this procedure, the model-based approach was selected, based on machine learning algorithms. The decision to split the model into background and foreground submodels had been made and this gave a possibility to make the time constraints of machine learning algorithms not a decisive factor in further research. After that, the non-functional requirements for machine learning algorithms had been defined and upon these requirements, the algorithms were evaluated. As a result, three machine learning algorithms were selected: multilayer perceptron, Support Vectors Machine (SVM) and decision trees. Moreover, the framework Weka was selected for the implementation of the model on the Java program language.

The selection model was implemented, the input data was generated, the accuracy measurement techniques were defined and finally, the experiments were conducted and as a result of that work, from three considered algorithms one was selected as the most efficient according to previously defined criteria. Algorithm M5P demonstrated the highest accuracy according to the approach TOP1 – 81% and according to the approach TOPGAP – 89%. Also, to verify the achievement of the goal of the thesis: improve the reliability of service selection with the efficiency ratio of service response time and cost indicators, a strategy of web service selection from the user side, called BEST\_FOR\_TERM, was formulated. This strategy described the order of Web service selection from the user perspective and enabled to use the obtained accuracy to evaluate the effectiveness of using the model for automated Web service selection. According to the BEST\_FOR\_TERM strategy, the accuracy of selecting a Web service was about 51.3%, while the usage of an algorithm M5P would improve the accuracy to 81% according to the approach TOP1 and to 89% according to the approach TOPGAP. Therefore, it can be argued that the goal of this thesis has been achieved. Moreover, the model has been built, which is based on M5P algorithm and consequently can provide the automatic service selection with 80% accuracy or more. Finally, the information system has been implemented, which is based on developed model

## Further Research

Further work can be divided into three logical partitions:

* **The dataset for the experiment.** The one good thing here to implement is to move from the generated data to the collection of

the real data. In this work, it was impossible due to time and resources constraints. However, with the real data, there is a possibility to find out new patterns, by which the response time of the services can change over the time. In this work, only two patterns were presented: cyclic and linear. Moreover, the availability of technically identical services in different geographic locations would allow calculating the “delay time” and to separate it from the total response time of Web service for the user in more accurate and realistic way. In this work, the delay time has been calculated theoretically, based on formulas, which may differ from the real life data.

* **Machine learning algorithms.** The future work in this area can be

composed by conducting the experiments with a larger number of machine learning algorithms. There is a possibility of such outcome that some machine learning algorithms show the best results on a linear data while other algorithms perform better on a periodic data or data of another pattern. With the different data, patterns in one data set an optimal algorithm for each data pattern can be used to increase the overall efficiency of the model. However, for this it is necessary to develop a mechanism that would detect the pattern of data and automatically select an algorithm, which can process the data of such pattern in the most efficient way.

* **The strategy of selecting the optimal service by the user.** The strategies of counting machine learning algorithms accuracy can be considered as satisfactory. However, further research can be conducted in the field of selection the optimal Web service from the side of the user. This will allow comparing the results of the machine learning algorithms with more realistic real data.

In general, the partitions mentioned in this section have not been implemented in the work due to lack of resources and time constraints.

# References

1. Netcraft. (2014, Apr. 2) April 2014 Web Server Survey. [Online]. Available: <http://news.netcraft.com/archives/2014/04/02/april-2014-web-> server-survey.html
2. T. Erl, *Service oriented architecture*. Upper Saddle River, NJ: Prentice Hall, 2008.
3. Microsoft. (2006) Chapter 1: Service Oriented Architecture (SOA). [Online]. Available: https://msdn.microsoft.com/en-us/library/bb833022.aspx
4. W3C. (2004, Feb. 11) Web Services Architecture. [Online]. Available: <http://www.w3.org/TR/ws-arch/#whatis>
5. G. Vadivelou, E. Ilavarasan, “QoS-based ranking model for web service selection considering user requirements” *Journal of Theoretical and Applied Information Technology*, vol. 59, pp. 673–682, 2014
6. L. Zhang, J. Zhang and H. Cai, *Services computing*. Beijing: Tsinghua University Press, 2007.
7. IBM. (2002, Jan. 1) Understanding quality of service for Web services. [Online]. Available: <http://www.ibm.com/developerworks/library/ws-quality/>
8. S.W. Yang, M.L. Shi, “A model for Web service discovery with QoS Constraints” *Chinese Journal of Computers*, vol. 28, no. 4, pp. 589–594, 2005
9. J. Kirchner, P. Karg, A. Heberle, and W. Löwe, “Appropriate Machine Learning Methods for Service Recommendation Based on Measured Consumer Experiences Within a Service Market” in *SERVICE COMPUTATION 2015, The Seventh International Conferences on Advanced Service Computing*, 2015, pp. 41–48
10. Linnaeus University. (2016, Jan. 11) Articles & Database search. [Online]. Available: <http://lnu.se/the-university-library/search/articles--> databases-?l=en
11. S. Rosario, A. Benveniste, S. Haar and C. Jard, “Probabilistic QoS and Soft Contracts for Transaction-Based Web Services Orchestrations”, *IEEE Transactions on Services Computing*, vol. 1, no. 4, pp. 187-200, 2008.
12. D. Schuller, U. Kampe, J. Eckert and R. Steinmetz, “Cost-driven Optimization of Complex Service-based Workflows for Stochastic QoS Parameters”, *IEEE 19th International Conference on Web Services*, pp. 66- 73, 2012.
13. W. Wiesemann, “A Stochastic Programming Approach for QoS-Aware Service Composition”, *Cluster Computing and the Grid, 2008. CCGRID '08. 8th IEEE International Symposium on*, pp. 226-233, 2008.
14. G. Cafora, M.D. Penta, R. Esposito and M.L. Villani, “A lightweight approach for QoS-aware service composition” in *Proc of the 2nd International Conference on Service Oriented Computing*, 2004, pp. 36–47
15. Wikipedia, The Free Encyclopedia. (2016) Genetic algorithm. [Online]. Available: https://en.wikipedia.org/wiki/Genetic\_algorithm.
16. W. Rong, K. Liu, and L. Liang, “Personalized Web Service Ranking via User Group Combining Association Rule” in *Proc. Int’l Conf. Web Services*, 2009, pp. 445–452
17. L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei, “Personalized QoS Prediction for Web Services via Collaborative Filtering” in *Proc. Int’l Conf. Web Services*, 2007, pp. 439–446
18. R.M. Sreenath and M.P. Singh, “Agent-Based Service Selection” *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 1, no. 3, pp. 261-279, 2004.
19. Z. Zheng, H. Ma, M.R. Lyu, and I. King, “WSRec: A Collaborative Filtering Based Web Service Recommendation System” in *Proc. Int’l Conf.*

*Web Services*, 2009, pp. 437–444

1. Wikipedia, The Free Encyclopedia. (2016) Collaborative filtering. [Online]. Available: https://en.wikipedia.org/wiki/Collaborative\_filtering
2. X. Chen, Z. Zheng, X. Liu, Z. Huang and H. Sun, “Personalized QoS- Aware Web Service Recommendation and Visualization”, *IEEE Transactions on Services Computing*, vol. 6, no. 1, pp. 35-47, 2013.
3. F. Peng, X. Zeng, H. Deng and L. Liu, “The QoS Prediction of Web Service with Location Information Ensemble”, *Journal of software*, vol. 9, no. 5, 2014.
4. J. Kirchner, A. Heberle, and W. Löwe, “Classification vs. Regression - Machine Learning Approaches for Service Recommendation Based on Measured Consumer Experiences” in *Services (SERVICES), 2015 IEEE World Congress on*, 2015, pp. 278–285.
5. Wikipedia, The Free Encyclopedia. (2016) Multilayer perceptron. [Online]. Available: <http://en.wikipedia.org/wiki/Multilayer_perceptron>
6. Dekel O., ”From Online to Batch Learning with Cutoff-Averaging”, in

*Neural Information Processing Systems*, 2008, pp. 377–384.

1. Kdnuggets. (2014) What Analytics, Data Mining, Data Science software/tools you used in the past 12 months for a real project Poll. [Online]. Available: <http://www.kdnuggets.com/polls/2014/analytics-data-mining-data-> science-software-used.html
2. Wikipedia, The Free Encyclopedia. (2016) Online machine learning. [Online]. Available: <http://en.wikipedia.org/wiki/Online_machine_learning>
3. Kdnuggets. (2015) Top 20 Python Machine Learning Open Source Projects. [Online]. Available: <http://www.kdnuggets.com/2015/06/top-20-> python-machine-learning-open-source-projects.html
4. Pylearn2. (2015) Pylearn2 dev documentation. [Online]. Available: <http://deeplearning.net/software/pylearn2/>
5. P. Wasserman and T. Schwartz, “Neural networks. II. What are they and why is everybody so interested in them now?”, *IEEE Expert*, vol. 3, no. 1, pp. 10-15, 1988.
6. S. Trenn, “Multilayer Perceptrons: Approximation Order and Necessary Number of Hidden Units”, *IEEE Trans. Neural Netw.*, vol. 19, no. 5, pp. 836- 844, 2008.
7. H. Drucker, C. J.C. Burges, L. Kaufman, A. Smola and V. Vapnik, “Support Vector Regression Machines” in *Advances in Neural Information Processing Systems 9*, 1996, pp. 155-161.
8. S. P. Boyd, L. Vandenberghe, and Lieven, *Convex optimization.*

Cambridge, UK: Cambridge University Press, 2004.

1. The University of Adelaide. (2012, Oct. 24) Linear Regression and Support Vector Regression. [Online]. Available: [http://cs.adelaide.edu.au](http://cs.adelaide.edu.au/)

/~chhshen/teaching/ML\_SVR.pdf

1. Tripod. (2003) Support Vector Machine Regression. [Online].

Available: <http://kernelsvm.tripod.com/>

1. Weka.sourceforge.net. (2016) Class SMOreg. [Online]. Available: <http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/SMOreg.html>
2. L. Breiman, *Classification and regression trees*. Belmont, Calif., Wadsworth International Group, 1984.
3. Opentox community site. (2016) M5P – OpenTox. [Online]. Available: <http://www.opentox.org/dev/documentation/components/m5p>
4. Matin Fowler. (2015) GUI Architectures: Model View Controller. [Online]. Available: <http://martinfowler.com/eaaDev/uiArchs.html#Model> ViewController
5. M. David, “Evaluation: from precision, recall and f-measure to roc, Informedness, markedness & correlation”, *Nature*, vol. 2, issue 1, pp. 37–63, 2011.
6. J. Andersson, A. Heberle, J. Kirchner and W. Löwe “Service Level Achievements: Distributed Knowledge for Optimal Service Selection” in *Proceedings, Ninth IEEE European Conference on Web Services*. Los Alamitos, Calif.: IEEE Computer Society, 2011, pp. 125–132
7. Encyclopedia of Laser Physics and Technology. (2015, Feb. 22) Fibers. [Online]. Available: https:/[/www.rp](http://www.rp-photonics.com/fibers.html)-[photonics.com/fibers.html](http://www.rp-photonics.com/fibers.html)
8. Dykes, Sandra G., Kay A. Robbins, and Clinton L. Jeffery, ”An empirical evaluation of client-side server selection algorithms”, in *INFOCOM 2000, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*, 2000, vol. 3, pp. 1361-1370.
9. Linnaeus University. (2015, Oct. 28) Statistical Testing. [Online]. Available: https://coursepress.lnu.se/subject/thesis-projects/statistical-testing/
10. Vassarstats. (2015, Oct. 28) t-Test for Independent or Correlated Samples. [Online]. Available: <http://vassarstats.net/tu.html>

[47] I. Witten, E. Frank and M.Hall, *Data mining: Practical machine learning tools and techniques, Second edition*, 2nd ed. San Francisco, CA: Morgan Kaufmann Publishers In, 2005.

# Appendix 1. Tables with raw data of experiments

## Table with accuracy values for 100 datasets for each observed machine learning algorithm

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **MLP**  **TOP1** | **MLP**  **TOPGAP** | **SMOReg**  **TOP1** | **SMOReg**  **TOPGAP** | **M5P**  **TOP1** | **M5P**  **TOPGAP** |
| 63.19444 | 74.30556 | 72.22222 | 75.69444 | 84.02778 | 89.58333 |
| 65.97222 | 75 | 68.05556 | 75 | 81.25 | 90.27778 |
| 67.36111 | 77.77778 | 68.05556 | 74.30556 | 80.55556 | 88.19444 |
| 66.66667 | 74.30556 | 69.44444 | 77.08333 | 81.25 | 90.97222 |
| 63.88889 | 70.13889 | 72.22222 | 75 | 85.41667 | 90.27778 |
| 67.36111 | 77.77778 | 70.83333 | 75.69444 | 81.94444 | 88.19444 |
| 68.75 | 74.30556 | 70.83333 | 76.38889 | 82.63889 | 91.66667 |
| 64.58333 | 75 | 70.83333 | 77.08333 | 80.55556 | 90.27778 |
| 70.13889 | 79.16667 | 73.61111 | 77.77778 | 86.11111 | 91.66667 |
| 67.36111 | 75 | 72.91667 | 77.77778 | 83.33333 | 90.27778 |
| 65.27778 | 80.55556 | 72.22222 | 78.47222 | 84.02778 | 90.97222 |
| 70.13889 | 77.08333 | 72.91667 | 76.38889 | 84.02778 | 90.97222 |
| 67.36111 | 78.47222 | 70.83333 | 77.08333 | 81.25 | 91.66667 |
| 66.66667 | 77.08333 | 65.97222 | 74.30556 | 78.47222 | 88.19444 |
| 68.05556 | 72.22222 | 69.44444 | 75 | 79.86111 | 90.27778 |
| 67.36111 | 71.52778 | 67.36111 | 74.30556 | 78.47222 | 88.19444 |
| 67.36111 | 72.22222 | 70.13889 | 73.61111 | 83.33333 | 89.58333 |
| 66.66667 | 81.94444 | 72.22222 | 77.77778 | 83.33333 | 91.66667 |
| 68.05556 | 76.38889 | 70.13889 | 77.08333 | 81.94444 | 93.05556 |
| 61.80556 | 74.30556 | 68.05556 | 75 | 78.47222 | 88.88889 |
| 65.97222 | 71.52778 | 73.61111 | 77.08333 | 83.33333 | 89.58333 |
| 67.36111 | 78.47222 | 70.13889 | 76.38889 | 82.63889 | 89.58333 |
| 68.05556 | 73.61111 | 68.75 | 79.16667 | 79.16667 | 92.36111 |
| 65.97222 | 73.61111 | 68.75 | 75 | 81.25 | 89.58333 |
| 65.27778 | 72.91667 | 72.22222 | 75.69444 | 81.94444 | 88.19444 |
| 64.58333 | 72.22222 | 69.44444 | 75.69444 | 80.55556 | 91.66667 |
| 61.80556 | 76.38889 | 70.13889 | 75.69444 | 80.55556 | 87.5 |
| 63.88889 | 75 | 70.83333 | 75.69444 | 83.33333 | 91.66667 |
| 66.66667 | 77.77778 | 68.05556 | 75.69444 | 79.16667 | 90.27778 |
| 70.13889 | 77.08333 | 71.52778 | 76.38889 | 84.72222 | 90.97222 |
| 63.88889 | 70.83333 | 70.13889 | 74.30556 | 78.47222 | 86.80556 |
| 65.97222 | 72.22222 | 67.36111 | 74.30556 | 79.16667 | 87.5 |
| 63.88889 | 74.30556 | 68.05556 | 73.61111 | 78.47222 | 88.19444 |
| 65.27778 | 72.22222 | 70.13889 | 75.69444 | 81.25 | 87.5 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 71.52778 | 77.08333 | 72.91667 | 78.47222 | 83.33333 | 92.36111 |
| 63.19444 | 72.91667 | 71.52778 | 75 | 83.33333 | 88.88889 |
| 66.66667 | 74.30556 | 69.44444 | 74.30556 | 80.55556 | 87.5 |
| 67.36111 | 76.38889 | 70.13889 | 77.08333 | 81.94444 | 91.66667 |
| 65.27778 | 74.30556 | 67.36111 | 74.30556 | 80.55556 | 88.19444 |
| 63.88889 | 72.91667 | 70.83333 | 76.38889 | 80.55556 | 89.58333 |
| 68.05556 | 75 | 71.52778 | 77.08333 | 80.55556 | 90.27778 |
| 68.75 | 76.38889 | 69.44444 | 75.69444 | 78.47222 | 88.19444 |
| 64.58333 | 71.52778 | 69.44444 | 76.38889 | 80.55556 | 90.97222 |
| 64.58333 | 80.55556 | 74.30556 | 78.47222 | 83.33333 | 88.88889 |
| 65.27778 | 73.61111 | 70.13889 | 77.08333 | 81.25 | 91.66667 |
| 67.36111 | 75 | 70.83333 | 75 | 80.55556 | 88.19444 |
| 65.27778 | 73.61111 | 70.83333 | 76.38889 | 83.33333 | 90.27778 |
| 67.36111 | 72.22222 | 71.52778 | 75 | 81.25 | 88.88889 |
| 63.88889 | 71.52778 | 66.66667 | 74.30556 | 77.77778 | 88.19444 |
| 64.58333 | 73.61111 | 70.13889 | 73.61111 | 80.55556 | 89.58333 |
| 65.27778 | 71.52778 | 71.52778 | 75.69444 | 83.33333 | 90.97222 |
| 61.11111 | 73.61111 | 66.66667 | 72.91667 | 78.47222 | 88.19444 |
| 66.66667 | 75 | 68.05556 | 74.30556 | 78.47222 | 87.5 |
| 62.5 | 71.52778 | 65.97222 | 71.52778 | 79.16667 | 85.41667 |
| 65.27778 | 73.61111 | 70.13889 | 77.77778 | 82.63889 | 91.66667 |
| 62.5 | 70.13889 | 64.58333 | 72.91667 | 76.38889 | 86.80556 |
| 69.44444 | 72.22222 | 66.66667 | 74.30556 | 76.38889 | 87.5 |
| 67.36111 | 75 | 70.13889 | 76.38889 | 81.94444 | 90.97222 |
| 68.75 | 77.08333 | 73.61111 | 77.77778 | 84.72222 | 90.97222 |
| 72.91667 | 78.47222 | 71.52778 | 78.47222 | 82.63889 | 91.66667 |
| 68.05556 | 76.38889 | 68.75 | 74.30556 | 79.86111 | 88.19444 |
| 65.97222 | 75.69444 | 71.52778 | 79.16667 | 81.25 | 90.27778 |
| 69.44444 | 76.38889 | 71.52778 | 75.69444 | 82.63889 | 89.58333 |
| 70.13889 | 82.63889 | 73.61111 | 78.47222 | 82.63889 | 92.36111 |
| 68.75 | 81.25 | 70.83333 | 77.08333 | 80.55556 | 89.58333 |
| 63.19444 | 75 | 66.66667 | 73.61111 | 77.77778 | 87.5 |
| 61.11111 | 75.69444 | 71.52778 | 76.38889 | 79.86111 | 90.27778 |
| 60.41667 | 71.52778 | 63.88889 | 72.91667 | 76.38889 | 87.5 |
| 66.66667 | 73.61111 | 69.44444 | 75 | 81.25 | 89.58333 |
| 70.83333 | 76.38889 | 72.91667 | 76.38889 | 83.33333 | 90.97222 |
| 62.5 | 70.13889 | 65.27778 | 72.22222 | 76.38889 | 85.41667 |
| 72.22222 | 75.69444 | 71.52778 | 77.08333 | 81.94444 | 88.88889 |
| 63.88889 | 77.08333 | 70.83333 | 73.61111 | 83.33333 | 88.19444 |
| 67.36111 | 76.38889 | 73.61111 | 78.47222 | 84.02778 | 92.36111 |
| 68.75 | 77.77778 | 69.44444 | 74.30556 | 81.25 | 88.88889 |
| 63.88889 | 70.13889 | 66.66667 | 73.61111 | 79.86111 | 87.5 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 65.27778 | 77.77778 | 68.75 | 75.69444 | 79.16667 | 90.27778 |
| 63.19444 | 70.83333 | 67.36111 | 72.91667 | 80.55556 | 88.88889 |
| 66.66667 | 73.61111 | 68.75 | 77.08333 | 80.55556 | 90.97222 |
| 62.5 | 74.30556 | 68.75 | 75 | 81.25 | 90.27778 |
| 63.88889 | 74.30556 | 66.66667 | 71.52778 | 79.86111 | 86.80556 |
| 68.05556 | 75.69444 | 70.13889 | 76.38889 | 82.63889 | 91.66667 |
| 69.44444 | 75.69444 | 69.44444 | 75 | 80.55556 | 89.58333 |
| 68.05556 | 79.16667 | 73.61111 | 77.08333 | 85.41667 | 89.58333 |
| 63.19444 | 71.52778 | 71.52778 | 76.38889 | 84.72222 | 90.97222 |
| 60.41667 | 69.44444 | 65.27778 | 74.30556 | 77.77778 | 88.88889 |
| 65.27778 | 73.61111 | 72.22222 | 77.08333 | 84.02778 | 90.97222 |
| 64.58333 | 77.77778 | 69.44444 | 78.47222 | 81.25 | 91.66667 |
| 66.66667 | 72.91667 | 70.13889 | 74.30556 | 79.86111 | 88.19444 |
| 68.75 | 79.86111 | 73.61111 | 79.86111 | 84.02778 | 91.66667 |
| 69.44444 | 73.61111 | 69.44444 | 75.69444 | 84.02778 | 90.27778 |
| 62.5 | 69.44444 | 63.19444 | 72.22222 | 74.30556 | 86.80556 |
| 63.88889 | 75 | 70.83333 | 75.69444 | 81.94444 | 88.88889 |
| 66.66667 | 73.61111 | 69.44444 | 75 | 81.25 | 90.27778 |
| 65.27778 | 76.38889 | 69.44444 | 76.38889 | 81.94444 | 90.27778 |
| 65.27778 | 76.38889 | 73.61111 | 76.38889 | 81.94444 | 88.88889 |
| 61.11111 | 70.13889 | 66.66667 | 72.22222 | 77.77778 | 86.11111 |
| 68.05556 | 73.61111 | 70.83333 | 75 | 81.25 | 86.80556 |
| 64.58333 | 72.91667 | 65.97222 | 73.61111 | 78.47222 | 87.5 |
| 65.97222 | 72.91667 | 68.05556 | 75.69444 | 79.16667 | 89.58333 |