AMINISTRY FOR DEVELOPMENT OF INFORMATION TECHNOLOGIES AND COMMUNICATIONS OF THE REPUBLIC OF UZBEKISTAN

TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES

A manuscript
UDK 004

CHEREPANOV ALEKSANDR ALEKSANDROVICH

USING DATA MINING TECHNIQUES IN WEB-BASED QUALIFYING EVALUATION AND QUESTIONNAIRE POLL SYSTEM

5A330601 —“Software Engineering”

DISSERTATION

on competition of the academic degree of Master

Scientific supervisor
Khan I.V.

Tashkent — 2015

MINISTRY FOR DEVELOPMENT OF INFORMATION TECHNOLOGIES AND COMMUNICATIONS OF THE REPUBLIC OF UZBEKISTAN

TASHKENT UNIVERSITY OF INFORMATION TECHNOLOGIES

Faculty: Software engineering	Graduate student: Cherepanov A.A.
Department: The software of information technologies	Scientific supervisor: Khan I.V.
Academic years: 2013-2015	Specialty: 5A____ — “Software Engineering”

ABSTRACT TO MASTER'S THESIS

The relevance of the thesis. Currently, the majority of development of information systems is performed with the use of some data mining, machine learning techniques and real-time analytics. These techniques provide to customers new intelligent abilities and features such as quick analysis and smart expectations. Ones of useful systems where data mining abilities can be used are questionnaire poll and qualifying evaluation systems (later survey system). A literature review of research in this field has shown that the existing data mining techniques do not fully reflect the relationship between structure of questionnaire list and appropriate data mining algorithm or technique.

The purpose and tasks of the thesis. The main purpose of this work is to research of using data mining techniques in web-based qualifying evaluation and questionnaire poll systems on the example of the module for data mining analysis of open source “LimeSurvey” project.

The objectives of the study are:

- Analysis of the possibility of using classification data mining techniques in questionnaire poll systems;
- Analysis of the process of surveying;
- Development of an data mining classification algorithm with the structure of questionnaire lists and other requirements of the development of survey systems;
- Study the effect of the structure of questionnaire list on the data mining techniques in special “LimeSurvey” module;
- Validation of the results at the operating system.

The object and subject of the thesis. The object of the study is a software module for data mining analysis in open source web-based qualifying evaluation and questionnaire poll systems “LimeSurvey”.

The subject of the study is to investigate the effect of the structure of questionnaire list on the data mining techniques using classification algorithms.

The methods and techniques of the research. The main research method is an structural and object-oriented modelling software by means of R and PHP languages. Also the methods of active experiment are used for module of survey system. Research means are such tools as Microsoft Visio, IDE JetBrains PhpStorm, database MySQL, PHP language with the library Yii Framework and open source LimeSurvey project, R language.

The hypothesis of the thesis. The possibility of establishing a link between the structure of questionnaire list and implementation of data mining techniques for prediction of filtering results. And the advantages of developing high-quality software systems based on modern techniques.

Scientific and practical importance. The results evolve scientific and practical aspects of applying data mining techniques to software development.

Scientific novelty in the thesis. The scientific novelty of this work is to optimise classification techniques with importance of influence a structure of questionnaire to implementation of data mining techniques.

The structure of the thesis. The thesis consists of an introduction, three chapters, conclusion, references and applications.

Scientific supervisor: ___________________________ Khan I.V.
(signature)

Graduate student:	___________________________ (signature)	Cherepanov A.A.

МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Факультет: Программный инжиниринг

Кафедра: Программное обеспечение информационных технологий

Учебный год: 2013-2015

Студент магистратуры: Черепанов А.А.

Научный руководитель: Хан И.В.

Специальность: 5А____ — “Программный инжиниринг”

АННОТАЦИЯ К МАГИСТЕРСКОЙ ДИССЕРТАЦИИ

Обоснование актуальности темы. В настоящее время большинство разработок информационных систем выполняется с применением различных методов интеллектуального анализа, машинного обучения и аналитики реального времени. Одни из полезных систем, в которых могут быть применены возможности интеллектуального анализа данных, это системы анкетирования и аттестации. Литературный обзор исследований в этой области показал, что существующие методы анализа данных не в полной мере отражают взаимосвязи между структурой анкет и соответствующим алгоритмом анализа данных.

Цель и задачи исследования. Основной целью настоящей работы является исследование использования методов интеллектуального анализа данных в системах анкетирования и квалификационной оценки, основанных на веб-технологиях, на примере модуля для анализа данных проекта с открытым кодом "LimeSurvey".

Задачами исследования являются:

- Анализ возможности использования методов классификации интеллектуального анализа данных в системах анкетирования;
- Анализ процесса анкетирования;
- Разработка алгоритма классификации анализа данных со структурой вопросника и другими требованиями разработки систем анкетирования;
- Исследование влияния структуры анкеты на методы анализа данных в специальном модуле "LimeSurvey";
- Апробация полученных результатов на действующей системе.

Объект и предмет исследования. Объектом исследования является программный модуль для анализа данных в системе анкетирования и квалификационной оценки, основанной на веб-технологиях “LimeSurvey”.

Предметом исследования является изучение влияния структуры анкеты на методы анализа данных с использованием алгоритмов классификации.

Методы и средства исследования. Основным методом исследования является структурное и объектно-ориентированное моделирование программного обеспечения с помощью языков R и PHP. Также используются методы активного эксперимента для модуля системы анкетирования. Средствами исследования являются инструменты Microsoft Visio, IDE JetBrains PhpStorm, база данных MySQL, язык PHP с библиотекой Yii Framework, проект Limesurvey, язык R.

Гипотеза исследования. Возможность установления связи между структурой анкеты и реализацией методов анализа данных для определения фильтрации результатов и получение преимуществ разработки программных систем на основе современных методов.

Научная и практическая значимость. Результаты развивают научные и практические аспекты применения методов интеллектуального анализа данных для разработки программного обеспечения.

Научная новизна. Оптимизация метода кластеризация с учетом влияния структуры вопросника на реализацию методов анализа данных.

Состав диссертационной работы. Работа состоит из введения, трех глав, заключения, списка использованной литературы и приложений.

Научный руководитель: ____________________________ Хан И.В.

(подпись)

Студент магистратуры:	____________________________ (подпись)	Черепанов А.А.

CONTENTS

Introduction 7

Chapter 1. Technical background 12

1. Analysis Web questionnaires 12
2. Guidelines for designing Web and mobile questionnaires 14
3. Capabilities and Limits of the Web Survey 18
4. Data Mining Context 23
5. LimeSurvey Project Overview 28
6. Formalisation of problems 31

Chapter 2. Applying Data mining techniques to Survey Systems 32

1. Measurement level and types of data 32
2. R for Machine Learning 34
3. Text Regression 39
4. Building survey model from data 42
5. Applying classification techniques to survey systems 48

Chapter 3. Software implementation of data mining module 59

1. Selection of programming technologies 59
2. Description of implementation classes and methods 61
3. Description of module's user interfaces 66

Conclusions 72

References 74

Appendix 77

INTRODUCTION

The relevance of the work lies in the fact that now the majority of development of information systems is performed with the use of some data mining, machine learning techniques and real-time analytics. These techniques provide to customers new intelligent abilities and features such as quick analysis and smart expectations. Ones of useful systems where data mining abilities can be used are questionnaire poll and qualifying evaluation systems (later survey system). A literature review of research in this field has shown that the existing data mining techniques do not fully reflect the relationship between structure of questionnaire list and appropriate data mining algorithm or technique. Currently, in Uzbekistan data mining techniques are generally subjects of academic sphere, but they are not used in practical software development widely and a little mastered by domestic programmers.

At present, the developed countries are moving from an industrial society to an informational one. This is reflected in the intensive improvement of computer and communication technologies, the emergence of new information technologies and further development of existing ones, as well as in the implementation of information systems. Achievements of the Informatics have taken a worthy place in the organisational management, industry, research and computer-aided design. Informatization also covered the social sphere: education, science, culture and health care.

Given the importance of informatization, computer and information technologies in solving problems of economic and social development in the Republic of Uzbekistan was adopted the Law "About Informatization" (December 11, 2003), the Decree of the President of the Republic of Uzbekistan "On further development of computerisation and introduction of information and communication technologies” (May 30, 2002).

Currently, intensive development of information and communication technologies helps to speed up the decision making process, the establishment and adoption of important documents. In 2004, the Republic of Uzbekistan on the initiative of President of Uzbekistan Islam Karimov, the Law "On electronic document" was adopted. It sets legally binding electronic documents.

Web surveys continue to pose many challenges and benefits for surveyors, much like they did in their early days. Typically, responses can be gathered from large numbers of people in a very short amount of time. Web surveys can also often be conducted at a fairly low cost, especially when e-mail is the only form of communication with sample members. Thousands or even tens of thousands of questionnaires can be completed in a single day with the results available for review and analysis immediately.

For this reason and others, the use of web and mobile surveys continues to grow. The proliferation of opt-in panels and other non probability online survey methodologies has further fuelled the increase in web surveys. But surveys of special populations with high levels of computer ability, such as college students, policy experts, and business executives, also are increasingly being conducted online, using web alone or in combination with another mode.

Depending on the characteristics of the business problems and the availability of ‘clean’ and suitable data for the analysis, an analyst must make a decision on which knowledge-discovery techniques to use to yield the best output. Among the available techniques are:

- Statistical methods
- Decision trees and decision rules
- Cluster analysis
- Association rules
- Artificial neural networks

So in development it is very important to select adequate technologies and optimise known data mining techniques to appropriate business problem and software structure and abilities.

The purpose of the master's thesis is the research of using data mining techniques in web-based qualifying evaluation and questionnaire poll systems on the example of the module for data mining analysis of open source “LimeSurvey” project.

The objectives of the thesis’s research are:

- Analysis of the possibility of using classification data mining techniques in questionnaire poll systems;
- Analysis of the process of surveying;
- Development of an data mining classification algorithm with the structure of questionnaire lists and other requirements of the development of survey systems;
- Study the effect of the structure of questionnaire list on the data mining techniques in special “LimeSurvey” module;
- Validation of the results at the operating system.

The object of the study is a software module for data mining analysis in open source web-based qualifying evaluation and questionnaire poll systems “LimeSurvey”.

The subject of the study is to investigate the effect of the structure of questionnaire list on the data mining techniques using classification algorithms.

Research methods. The main research method is an structural and object-oriented modelling software by means of R and PHP languages. Also the methods of active experiment are used for module of survey system.

Research tools are tools Microsoft Visio, IDE JetBrains PhpStorm, database MySQL, PHP language with the library Yii Framework and open source LimeSurvey project, R language.

The hypothesis of the study is the possibility of establishing a link between the structure of questionnaire list and implementation of data mining techniques for prediction filters in surveys, and practical testing of the obtained results, as well as the possibility of developing high-quality software systems based on modern techniques.

The scientific novelty of this work is to develop an evaluation method of influence a structure of questionnaire to implementation of data mining techniques.

The structure of the thesis. The thesis consists of an introduction, three chapters, conclusion, references and applications.

In the first chapter overview of questionnaire poll and qualifying evaluation systems is presented, main concepts of data mining are described, it was analysed some data mining techniques applied to survey systems. Also the formulation of the problem was made.

In the second chapter it is described the structure of LimeSurvey project and module's structure, the structure of database's part needed for implementing data mining module, the classification algorithms were adapted for special features which are proper for survey systems and optimised for usage with such systems as the open source LimeSurvey project. Also research about possibilities of usage data mining techniques with such software systems was made and regularities in development web-based questionnaire poll systems with applied data mining abilities were evolved. The results were tested and approved.

The third chapter explains the choice of development tools and technologies, describes the practical implementation of the module and includes a description of the software product.

Finally, The concluding part draws the conclusions of the thesis.

The annex contains a complete block diagram of the database, and the source code of the module.

CHAPTER 1. TECHNICAL BACKGROUND

1. Analysis Web questionnaires

Surveys that are completely electronic, relying only on e-mail contacts to obtain Internet responses, are the fastest growing form of surveying occurring in the world, as well as throughout most of the world. As a stand-alone mode of data collection, web is especially attractive because of speed, low cost, and economies of scale.[1] However, despite these benefits, many barriers to realising them also exist and are discussed in this chapter. Through this discussion of how to design and implement web surveys, we focus on both the benefits and the current limitations of using only the Internet and e-mail invitations to request and obtain responses.

A large majority of the population in Uzbekistan and a growing number internationally now use the Internet. As of 2013, 85% of adults in Uzbekistan use the Internet and 70% have broadband Internet access in their homes.[2] People have also become more familiar with computers and engaging in various activities online. Many now prefer to conduct their business electronically rather than pick up a phone, write a letter, or go in person, and many companies make it difficult to contact them any other way than electronically.

The increasing use of mobile devices—especially smartphones and tablets—has further fuelled the growth in online behaviour, especially as these devices have become the primary way that some people connect to the Internet. Nearly all adults (91%) have cell phones, up from 75% in 2007. And people are now far more likely to send text messages and access the Internet on their cell phones than they were just 5 years ago. Further, 21% of cell phone owners say they *mostly* access the Internet using their phone, rather than other devices.[3]

The fact that people have become more accustomed to completing various daily activities online could be good for survey researchers interested in conducting web surveys, since people may also have become more receptive to completing surveys online. However, it also means that web surveys are constantly changing as the ways in which people interact with computers and mobile devices also continues to evolve. The rise in mobile devices requires survey designers to reconsider aspects of questionnaire design to accommodate the smaller screen. Many people may also receive and quickly scan e-mail and texts on their phone but then wait to follow-up on requests that need more attention until they get to their desktop or laptop. The range of devices (from desktop to netbook to tablet to phablet to smartphone), operating systems and browsers (and different versions), and customised settings available to users also continues to expand, making designing and implementing web surveys more challenging than it was even a few short years ago.

Web surveys continue to pose many challenges and benefits for surveyors, much like they did in their early days. Typically, responses can be gathered from large numbers of people in a very short amount of time. Web surveys can also often be conducted at a fairly low cost, especially when e-mail is the only form of communication with sample members.[4] Thousands or even tens of thousands of questionnaires can be completed in a single day with the results available for review and analysis immediately.

For this reason and others, the use of web and mobile surveys continues to grow. The proliferation of opt-in panels and other non-probability online survey methodologies has further fuelled the increase in web surveys. But surveys of special populations with high levels of computer ability, such as college students, pol- icy experts, and business executives, also are increasingly being conducted online, using web alone or in combination with another mode. Lastly, surveyors have increasingly been using web surveys in mixed-mode designs of the general public where sample members are contacted by another mode (e.g., mail) and asked to complete the web survey.[5]

In this chapter, we present guidelines for designing web and mobile questionnaires and for implementing web surveys in which e-mail is the only method used to contact people, but these guidelines will also be helpful to those who are conducting mixed-mode surveys that include e-mail contacts or web data collection.

2. Guidelines for designing Web and mobile questionnaires

Most web surveys are browser based, where respondents interact with the survey through their Internet browser (such as Chrome, Firefox, Internet Explorer, and Safari). The web survey is made up of a web page or series of web pages containing survey questions programmed most commonly in hypertext markup language (HTML) that are stored on a server. Users with the proper URL can access them through their computers or mobile devices and an Internet or cellular connection. Surveyors send requests, often by e-mail, but also by mail or telephone, to the person (or unit) from whom a response is desired and provide the link or URL for the web survey (and often an individual identification code).[6] Respondents click on the URL or enter it into their browser's address bar. Alternatively, QR codes (or quick response codes) provide mobile device users with a matrix barcode that can be read or scanned by such devices to take them directly to the web survey. Survey website URLs and a QR code are illustrated.

Example URLs

URL:

www.wiley.com/WileyCDA/WileyTitle/productCd-1118456149.html

Shortened URL:

http://tinyurl.com/tailored-design

Once sample members click on the URL, they are usually routed to a welcome or introductory screen that briefly describes the survey and asks them to proceed with the questionnaire. Once they begin the survey, pressing the “submit” or “next” button on a page sends their answers back to the web server. Their responses may be reviewed by the survey software and are stored in a database on the server. What this means, in essence, is that surveyors have to translate their questionnaire designs into computer code to be stored on a server. They also have to design databases that will store survey responses in an accurate, organised, meaningful, and accessible way on the server.[7]

This can be done by programming the survey and creating the databases from scratch (using unique code) or by using preexisting software, most of which provide fairly simple point-and-click interfaces. Programming the survey and databases from scratch provides the most design flexibility and ability to innovate, but it requires bringing together two very specialised skill sets—computer programming and survey methodology—and is often the most complex and expensive option.

Those who cannot program from scratch or who prefer other options have an almost dizzying array of software options to choose from. The available software programs vary along four dimensions that are important to consider.

1. *Design flexibility and difficulty.* Almost all survey software programs provide question templates that can be used. In some programs the template must be used as is, but other programs allow the surveyor to alter the underlying programming to customise the design. For some, customisation can be done in standard programming languages like HTML and cascading style sheets (CSS), but others require one to use specialised programming languages developed for the particular software. In addition, some software packages include either mobile optimisation or mobile app support and others do not, so it is important to review all of the features that a program has to offer before selecting one.

2. *Control over the data*. Some programs only allow the data to be collected and stored on the software company's servers while others allow collection and storage to take place on the surveyor's servers. Others provide both options. This raises the ethical considerations of how secure the data are and exactly who has access, issues that need to be considered carefully in light of any restrictions within one's organisation and before promising anonymity or confidentiality to respondents.

3. *Data access and reporting*. Some programs limit the analysis to simple frequencies and cross tabulations while others provide raw data sets that can be analysed however the surveyor desires (these come in a variety of formats). Similarly, some packages also offer various automated monitoring and reporting features that may make it easier to track progress and quickly look at results.

4. *Cost.* Cost underlies all the sedimensions. Available software packages range from free to costing tens of thousands of dollars per year. Many software providers have tiered pricing structures in which the higher cost tiers grant more design flexibility for the survey itself and more control over the resulting data. Generally speaking the design flexibility required to conduct a good web survey comes with a higher price tag.[8]

Reviewing the various software programs along these dimensions, as well as the questionnaire design and implementation guidelines presented throughout this chapter, can help in carefully evaluating the best software option for a specific survey or organisation's needs.

There are several ways, other than a browser-based survey, for researchers to collect survey data over the web. For example, survey apps can be used for those who are on mobile devices. Alternatively, other electronic options, such as fillable PDFs or embedded e-mail or text/SMS surveys, can sometimes be effective. With fillable PDFs, respondents can enter answers directly into a PDF file that is then e-mailed back to the surveyor (or returned some other way). In some cases, such as in surveys of establishments, this is desirable because PDF files can easily be printed to keep copies for one's records or to pass around to all the people who need to answer the questions.

Another option is to embed the survey directly into an e-mail or text message. The major advantage of this strategy is its low cost, as it eliminates considerable programming costs. However it is difficult to control the visual appearance of such surveys because of the variety of ways people read their e-mail and SMS messages (i.e., web-based e-mail providers, local software programs, on mobile devices and phones, etc.) and because entering responses shifts their content around. Lastly, these types of surveys are quite difficult for those completing them on mobile devices, unless they are kept very short (only one to three questions).

Despite these drawbacks, it is important to evaluate whether one of these alter- natives may be more appropriate for one's survey needs, as they are often less costly than fully interactive web surveys. The guidelines for designing web questionnaires presented here are primarily focused on browser-based web surveys, but we also highlight ways in which the design may need to be modified and optimised for a mobile experience. Designing survey questionnaires for mobile devices is still in its infancy and more research is needed to understand how people complete web questionnaires on their mobile devices.[9]

If one chooses a browser-based web survey, it is important to remember that designing a web survey is very different from designing a website. Although some aspects of visual design apply either way, others differ because people's motivation and purpose for going to a typical website are quite different from their motivation and purpose for visiting a web survey site (or using a web survey app). Most of the time, people visit websites to get information, and are self-motivated to find what they are looking for. In contrast, most respondents go to a web survey because someone else asked them to; their own motivation may be low. In addition, in a web survey, respondents' primary task is to provide, rather than seek, information. Thus, it is critical that web surveys be designed to make he response task as easy as possible while obtaining accurate measurement. Doing so may require different design strategies than might be suggested by the more general website usability literature. Moreover, design errors that may be tolerated when someone is looking for information or trying to pay their bills online may more easily trigger break-offs in a web survey.

3. Capabilities and Limits of the Web Survey

Although conducting a survey using computer and Internet technology gives one a wealth of new possibilities, it also means that one is constrained by the limits of those technologies. One of those limits that can be very damaging to a survey project if exceeded is the capacity of the web server(s). Web servers can only handle so much outgoing or incoming traffic before they begin to bog down or even crash. As a result, it is important to know ahead of time what effect sending or receiving mass amounts of survey-related information might have on one's server(s) and to plan accordingly.

We very rarely send e-mail survey invitations or reminders to the entire sample at one time (the exception is for small samples) because, although our e-mail program will allow us to do so, our server cannot handle that many outgoing e-mails at once. The risk is that some of the messages will be lost or will be bounced back to the sender, leading to the incredibly complex task of trying to sort through who got the e-mail and who did not. Instead, we send our e-mail communications in batches of a few hundred at a time over the course of a day or, better yet, in the late evening when the servers are less busy and the messages can arrive before recipients check their e-mail the next morning. This is slightly more labor intensive, but it is not nearly the amount of work that sorting out bounced e-mails would be.

In addition to considering the effects of outgoing communications, it is also important to consider the effects of incoming survey activity on the server(s). The nature of the Internet is such that many sample members can receive their invitations at virtually the same time, and all of them can immediately try to access and complete the survey. However, if too many people try to access the same survey on the same server all at once (or if responses are coming into the same server for different surveys all at once), the likely results could range from significant slowdowns that might lead some to quit the survey to a server crash that would cut every respondent off. Neither of these outcomes, nor any point in between, is desirable. When we have large survey projects that pose such challenges, we send invitations in batches over longer time periods so that the responses come in smaller waves rather than in one large spike. Another strategy we often employ is to have our server disconnect from respondents' computers if the computers are idle for a specified amount of time. We do this so that server resources can be devoted to respondents who are actively completing the questionnaire. When the idle respondents return to their computers and try to continue, they are provided with a friendly message that explains what happened and tells them how to return to their previous place in the survey.

The main point, though, is that it is important to know the limits of the hosting server(s) and plan accordingly. The best advice we can give here is to consult one's information technology professional or Internet service provider well in advance to work out an implementation and survey management strategy that works well within the limits of the server(s) one will be using.[10]

Assigning respondents a unique identification number is important for all modes, including web surveys. It allows the surveyor to keep track of who has responded and to remove respondents' contact information from follow-up databases so that they do not continue to receive reminders. In web surveys, the unique identification number can also serve an additional function in that it can be used as a unique access code that can, and should, be required in order to enter the web survey. Requiring respondents to enter a code in order to access the survey helps protect the integrity of the sample and survey data by ensuring that unsampled people who stumble upon the survey are not able to access it. It also provides a way to ensure that each respondent answers the survey only once, as the access code can be deactivated after the respondent submits the completed survey.

Some respondents, however, legitimately need to stop the survey and return to it at a later time to finish. As a result, deactivating access codes after they have been entered once is not a desirable practice. For this reason, we typically program our surveys so that if respondents stop before completing the survey and then come back to it later, they are returned to the question where they previously quit. They can leave and reenter the survey in this way as many times as they desire. However, once respondents submit the completed survey, their access code is deactivated. We have found this to be a quite useful strategy because most respondents who start the survey will finish it all at once if there are no offensive questions or technical difficulties that lead to artificially high break-off rates. Although this seems to work well for most of our surveys, other strategies may be more appropriate in other situations.

That one should assign an access code raises the issue of how one should require that access code to be used. Currently two main strategies are used: manual and automatic login. In manual login, respondents are sent a URL and access code. Once they get to the introductory web page using the URL, they are asked to key their access code into a designated space to gain access to the survey. In automatic login, the unique access code is contained within the URL so that entering the URL in their web browser (or clicking on it if it is sent by e-mail) will automatically gain respondents entrance into the survey.[5]

Several studies have compared these two strategies. Crawford et al. found that providing an automatic login significantly increased response rates by nearly 5 percentage points over a manual login condition in which respondents had to enter both a password and access code. Heerwegh and Loosveldt reported a similar but nonsignificant finding (a 3-percentage-point advantage for automatic login). However, they also found that a semiautomatic login procedure in which the password was included in a URL but the respondent still had to key in an access code resulted in an 8-percentage-point increase in response rate over the manual login procedure. Both of these studies as well as an additional study by Heerwegh and Loosveldt also reported some evidence that respondents who log in manually (or semiautomatically) provide more complete data. In particular, Heerwegh and Loosveldt reported that respondents who log in manually are more likely to complete more of the survey and are more likely to provide substantive answers to sensitive questions, an effect they attributed to respondents believing their data are more secure when they have to key in an access code.[11]

Based on these findings, it seems advisable to require respondents to key in an access code rather than to provide automatic login. But it is also important to use access codes large enough that one is unlikely to wrongfully enter the survey, whether by accident (i.e., a typo in entering one's own code) or by guessing a code. If, for example, one wants to use a four-digit numerical access code (where each digit can range from 0 to 9), the total number of possible codes that can be generated is 10,000. If one is surveying 1,000 respondents, 1 in every 10 possible access codes will be assigned to a respondent. With a five-digit access code, however, there are 100,000 possibilities, meaning that only 1 in every 100 possible codes will be assigned to a respondent. The likelihood of somebody wrongfully accessing the survey by guessing an access code or making a typo in entering his own code is considerably less with a five-digit access code.

However, the use of longer access codes should be balanced with considerations of the difficulty of transferring the access code from the e-mail to the website.

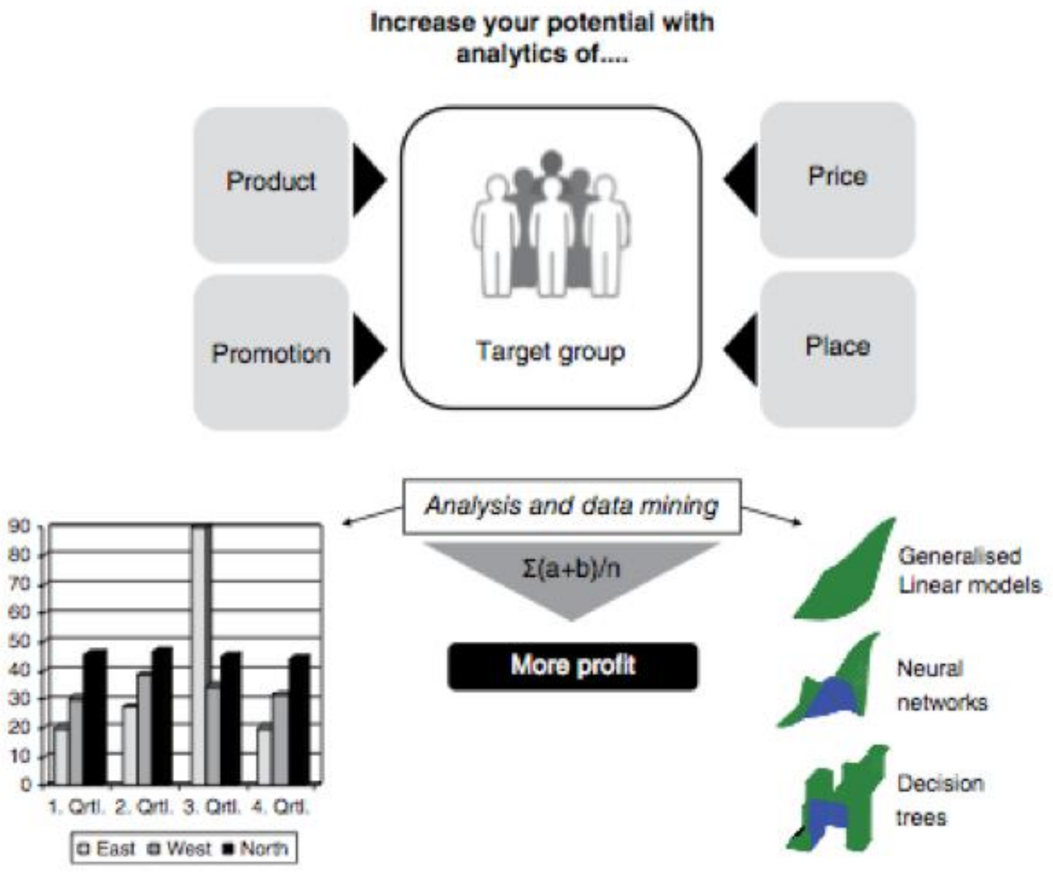
Although long access codes can be copied and pasted, many people will still type in the code, and the more digits they have to enter, the greater the likelihood for error.

In addition to considering the length of the access code, one should also be sure to avoid using adjacent numbers for different respondents. For example, using the access codes 10001, 10002, 10003, and so on would make it quite simple for a respondent to answer the survey multiple times by simply altering his own access code by one digit. Thus, it is important to build in an interval greater than 1 between access codes. However, one should also avoid regular intervals, as in the following sequence: 10101, 10201, 10301, and so on. Instead, a random interval of between 50 and 300 digits should be used; this can be easily accomplished using the random functions included in common software.

It is also important to make sure that the access codes one provides are not easily mistaken with other codes respondents deal with on a daily basis. For example, we once sent e-mail invitations for a student experience survey to a sample of undergraduate students. The invitation included a clickable link to the survey and respondents' ID number for accessing the survey. Shortly after sending the first batch of e-mails, we began getting messages back from students claiming that their ID did not work. It was only when one student included her ID number in her message that we realised that the students were entering their school ID numbers rather than the randomly generated ID numbers we had sent them in the e-mail. When surveying this type of group, we now make it a point to refer to the survey ID numbers as access codes to help differentiate them from what students more commonly think of as their ID. We also now specify "Please enter your access code listed in the e-mail we sent you" on the login page rather than the less specific "Please enter your ID." The types of specialised groups who can be surveyed via the web oftentimes have preexisting ID numbers, membership numbers, or access codes that they use regularly.

4. Data Mining Context

Modern management is data driven; customers and corporate data are becoming recognised as strategic assets. Decisions based on objective measurements are better than decisions based on subjective opinions which may be misleading and biased. Data is collected from all sorts of input devices and must be analysed, processed and converted into information that informs, instructs, answers or otherwise aids understanding and decision making. Input devices include cashier machines, tills, data loggers, warehouse audits and Enterprise Resource Planning (ERP) systems. The ability to extract useful but usually hidden knowledge from data is becoming increasingly important in today's competitive world. When the data is used for prediction, future behaviour of the business is less uncertain and that can only be an advantage; 'forewarned is



forearmed'!

Figure 1 - Increasing profit with data mining

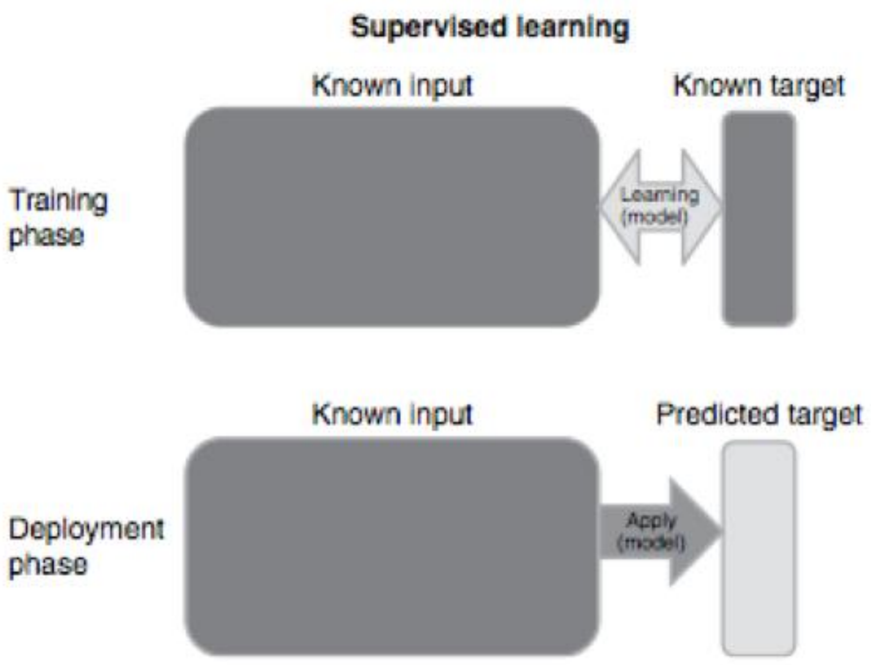
The valuable resource of historical data can lead to a predictive model and a way to decide on accepting new applicants to a business scheme.

With technological advancements, the computer industry has witnessed a tremendous growth in both hardware and software sectors. Sophisticated databases have encouraged the storage of massive datasets, and this has opened up the need for data mining in a range of business contexts. Data mining, with its roots in statistics and machine learning, concerns data collection, description, analysis and prediction. It is useful for decision making, when all the facts or data cannot be collected or are unknown. Today, people are interested in knowledge discovery (i.e. intelligence) and must make sense of the terabytes of data residing in their databases and glean the important patterns from it with trustworthy tools and methods, when humans can no longer juggle all these data and analyses in their heads (see Figure 1).

Data mining is a process that uses a variety of data analysis methods to discover the unknown, unexpected, interesting and relevant patterns and relationships in data that may be used to make valid and accurate predictions. In general, there are two methods of data analysis: supervised and unsupervised (see Figure 2 and Figure 3). In both cases, a sample of observed data is required. This data may be termed the training sample. The training sample is used by the data mining activities to learn the patterns in the data.[12]

Supervised data analysis is used to estimate an unknown dependency from known input–output data. Input variables might include the quantities of different articles bought by a particular customer, the date they made the purchase, the location and the price they paid. Output variables might include an indication of whether the customer responds to a sales campaign or not. Output variables are also known as targets in data mining. In the supervised environment, sample input variables are passed

Figure 2 - Supervised learning



through a learning system, and the subsequent output from the learning system is compared with the output from the sample. In other words, we try to predict who will respond to a sales campaign. The difference between the learning system output and the sample output can be thought of as an error signal. Error signals are used to adjust the learning system. This process is done many times with the data from the sample, and the learning system is adjusted until the output meets a minimal error threshold. It is the same process taken to fine-tune a newly bought piano. The fine-tuning could be done by an expert or by using some electronic instrument. The expert provides notes for the training sample, and the newly bought piano is the learning system. The tune is perfected when the vibration from the keynotes of the piano matches the vibration in the ear of the expert.[12,13]

Unsupervised data analysis does not involve any fine-tuning. Data mining algorithms search through the data to discover patterns, and there is no target or aim variable. Only input values are presented to the learning system without the need for validation against any output. The goal of unsupervised data analysis is to discover ‘natural’ structures in the input data. In biological systems, perception is a task learnt via an unsupervised technique.

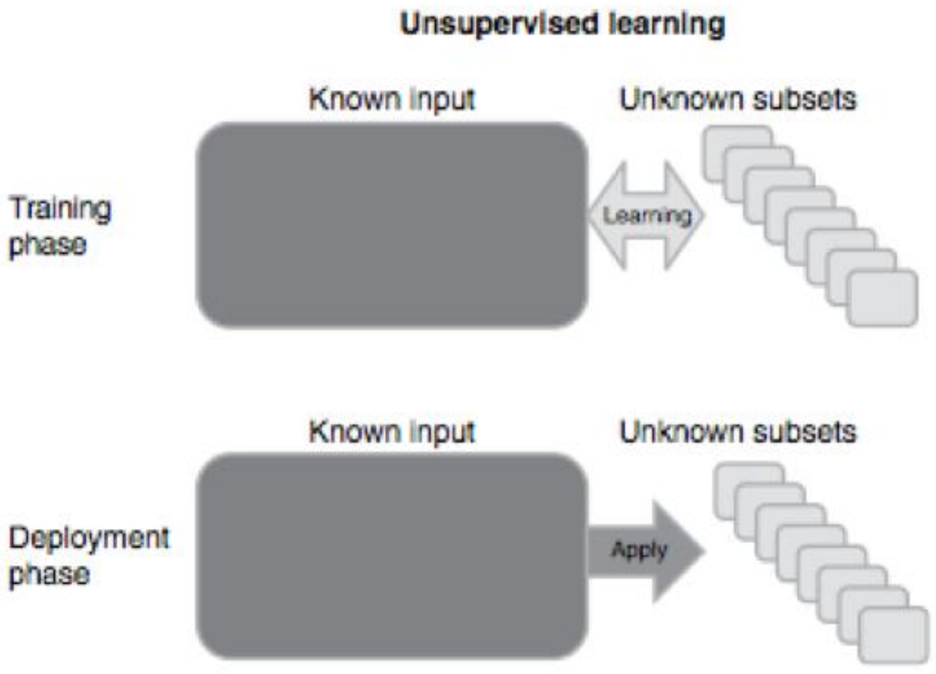


Figure 2 - Unsupervised learning

Depending on the characteristics of the business problems and the availability of ‘clean’ and suitable data for the analysis, an analyst must make a decision on which knowledge-discovery techniques to use to yield the best output. Among the available techniques are:

- **Statistical methods**: multiple regression, logistic regression, analysis of variance and log-linear models and Bayesian inference
- **Decision trees and decision rules**: Classification And Regression Tree (CART) algorithms and pruning algorithms
- **Cluster analysis**: divisible algorithm, agglomerative algorithms, hierarchical clustering, partitional clustering and incremental clustering
- **Association rules**: market basket analysis, *a priori* algorithm and sequence patterns and social network analysis
- **Artificial neural networks**: multilayer perceptrons with back-

propagation learning, radial networks, Self-Organising Maps (SOM) and Kohonen networks

- **Genetic algorithms**: used as a methodology for solving hard optimisation problems
- **Fuzzy inference systems**: based on theory of fuzzy sets and fuzzy logics
- **N-dimensional visualisation methods**: geometric, icon-based, pixel- oriented and hierarchical techniques
- **Case-Based Reasoning (CBR)**: based on comparing new cases with stored cases, uses similarity measurements and can be used when only a few cases are available

This list is not exhaustive, and the order does not suggest any priority in the application of these techniques.

5. LimeSurvey Project Overview

LimeSurvey (formerly PHPSurveyor) is a free and open source on-line survey application written in PHP based on a MySQL, PostgreSQL or MSSQL database, distributed under the GNU General Public License. As a web server-based software it enables users to develop and publish on-line surveys, and collect responses, without doing any programming.[27]

LimeSurvey is a web application that is installed to the user's server. After installation users can manage LimeSurvey from a web-interface. Users can use rich text in questions and messages, using a rich text editor, and images and videos can be integrated into the survey. The layout and design of the survey can be modified under a template system. Templates can be changed in a WYSIWYG HTML editor. Additionally, templates can be imported and exported through the template editor. Once a survey is finalized, the user can activate it, making it available for respondents to view and answer. Likewise, questions can also be imported and exported through the editor interface. LimeSurvey has no limit on the number of surveys a user can create, nor is there a limit on how many participants can respond. Aside from technical and practical constraints, there is also no limit on the number of questions each survey may have. [15]

Questions are added in groups. The questions within each group are organized on the same page. Surveys can include a variety of question types that take many response formats, including multiple choice, text input, drop-down lists, numerical input, slider input, and simple yes/no input. Questions can be arranged in a two-dimensional array, with options along one axis based on the questions on the other axis. Questions can depend on the results of other questions. For instance, a respondent might only be asked about transportation for his or her commute if he or she responded affirmatively to a question about having a job.

LimeSurvey also provides basic statistical and graphical analysis of survey results. Surveys can either be publicly accessible or be strictly controlled through the use of "once-only" tokens, granted only to selected participants. Additionally, participants can be anonymous, or LimeSurvey can track the IP addresses of the participants. A much more detailed listing of features can be found on the LimeSurvey web page.

Some web hosting services offer LimeSurvey hosting, either as a custom installation or through a control panel, such as cPanel with Fantastico, Plesk, Softaculous, or Virtualmin Professional. LimeSurvey has also been ported by third parties to various content management systems, such as PostNuke, and XOOPS. A port to Joomla exists, but it is not compatible with version 1.5 of Joomla.

The main developer and project leader of the LimeSurvey project, Carsten Schmitz, is also the owner of LimeService, a company which offers LimeSurvey hosting for a fee. This service is similar to web applications such as SurveyMonkey; the main difference being that LimeService's fees are based on the number of people that respond to the survey and not on the time that the survey is active for, like other similar services. LimeService offers up to 25 free responses per month, after which responses can be purchased in one of several packages.

Problems of scalability exist when using MySQL as the backing database for LimeSurvey because it cannot reliably support the default MySQL storage engine, InnoDB, due to the latters limitation of 8000 characters per database row. Thus Limesurvey uses the MyISAM database engine which has concurrency issues due to lack of row based locking for database inserts and updates. This in turn can affect MySQL replication,[13] at least prior to MySQL 5.6, due to the MySQL's limitation of a single thread being used for execution of inserts and updated on the replica database server, thus affecting all replicated database on the master MySQL server.

LimeSurvey in general uses the UTF-8 character set to be able to display all languages. Both the frontend and backend of LimeSurvey are available in 60 languages and dialects; 22 of these have over 95% of the translations done. Primary translations include: Albanian, Basque, Chinese, Croatian, Danish, Dutch, Finnish, French, Galician, German, Greek, Hungarian, Hebrew, Italian, Japanese, Norwegian, Portuguese, Russian, Serbian, Slovenian, Spanish, Swedish. There are also many other partial translations in other languages.[15]

LimeSurvey allows users to create and host surveys, for general data gathering purposes. It can be used for collecting data from customers and employees.

LimeSurvey is used by several open source organisations such as OpenOffice.org, Ubuntu, and GNOME. LimeSurvey is also used by educational institutions in 19 countries.

The code base for LimeSurvey 2.0 was completely re-written from scratch using a MVC (Model–view–controller) approach and the Yii PHP framework. Besides the structural code changes aimed at better modularity the new version also has a new GUI with a completely new design using AJAX technology.

6. Formalisation of problems

The purpose of the master's thesis is the research of using data mining techniques in web-based qualifying evaluation and questionnaire poll systems on the example of the module for data mining analysis of open source "LimeSurvey" project.

To complete the purpose of the research it have to:

- Analyse of the process of surveying;
- Develop the data mining classification algorithm with the structure of questionnaire lists and other requirements of the development of survey systems;
- Study the effect of the structure of questionnaire list on the data mining techniques in special "LimeSurvey" module;
- Validate the results at the operating software system.

Conclusions for the first chapter

In the first chapter overview of questionnaire poll and qualifying evaluation systems is presented. Main concepts of data mining are described, it was analysed some data mining techniques applied to survey systems. Also LimeSurvey project was introduced.

Also the formulation of the problem was made.

CHAPTER 2. APPLYING DATA MINING TECHNIQUES TO SURVEY SYSTEMS

1. Measurement level and types of data

There are different types of quantitative data, all of which can have good information content. There are many different terms used to describe the different data types, and the most common words are detailed in the following.

The simplest level of measurement is expressed as nominal data which indicates which named category is applicable. For example, a customer may live in an urban area, a rural area or a mixed area. This nominal data variable will consist of a column of urban/rural/mixed with one row for each customer. If there are only two levels, for example, ‘buy’ or ‘no buy’, then the data is referred to as binary variables. If there is any order associated with the categories, then they are referred to as ordinal data. For example, text associated with reasons for returning goods may read something like:

The clothes were the wrong size.

This comment could be classified as a size complaint. The frequency of size complaints could be compared with the frequency of non-size-related complaints. Size/non-size related is a binary variable having two levels, and we could usefully compare the number of complaints at each level.

If the reason for returning is

The clothes were too big,

then we could classify this complaint as a ‘too big’ mismatched size complaint, and we could compare the frequency of ‘too big’ with ‘too small’ mismatched sizes or ‘non-size’ related. A variable containing the information about complaints classified as too big/too small/unspecified size/non-size related is a categorical variable with nominal measurement at four levels.

There could also be an ordinal-level measurement if the categories are related in a descending or ascending order. For example, the variable levels could be first return, second return, third return, etc. If there are more than two levels for a nominal variable but there is no implied order, then some data mining procedures may require them to be converted to a series of indicator variables. For example, urban/rural/mixed could be converted to three indica- tor variables: urban or not, rural or not and mixed or not. The last variable is redundant as its value is indicated when neither of urban nor rural is true.[16]

Variables that represent size are referred to as measures, measurements or metrics and are described as being on a metric level. In data mining, the term 'metric' includes counts of some data type, like page views, and may correspond to a column of data.

The measurement level would be interval if the variable is the number of occurrences, for example, the number of returns for a customer (i.e. the number of times a customer returned an order). In this case, there could be lots of customers with zero returns but a few customers with one, two, three or more returns. This is discrete data measured on an interval scale. Another example of interval-level measures or metrics is provided by *altmetrics*, which are measurements of interaction based on the social web resulting in variables like the number of hits or mentions across the web. Subjects such as netnography explore web activity in great detail.

Many data items are measured on a continuous scale, for example, the distance travelled to make a purchase. Continuous data does not need to be whole numbers like 4km but can be fractions of whole numbers like 5.68 km. Continuous data may be interval type or ratio type. Interval data has equal intervals between units (e.g. 3.5 is 1 less than 4.5, and 4.5 is 1 less than 5.5). Ratio data is interval-type data with the additional feature that zero is meaningful and ratios are constant (e.g. 12 is twice as big as 6, and

6 is twice as big as 3).

Nominal and ordinal variables are referred to as categorical or classification variables. They often represent dimensions, factors or custom variables that allow you to break down a metric by a particular value, like screen views by screen name.[17]

To summarise, in data mining, we consider classification or categorical variables which can be nominal, binary and ordinal as well as scale or metric variables which can be count, continuous, interval or ratio.

Qualitative data, such as pictures or text, can be summarised into quantitative data. For example, an analysis of content can be expressed in terms of counts and measured in terms of impact or quantity of relationships. Content analysis may give rise to nominal data in which the categories can be named but do not have any implied order.

2. R for Machine Learning

R is an extremely powerful language for manipulating and analysing data. Its meteoric rise in popularity within the data science and machine learning communities has made it the de facto *lingua franca* for analytics. R’s success in the data analysis community stems from two factors described in the preceding epitaphs: R provides most of the technical power that statisticians require built into the default language, and R has been supported by a community of statisticians who are also open source devotees.

There are many technical advantages afforded by a language designed specifically for statistical computing. As the description from the R Project notes, the language provides an open source bridge to S, which contains many highly specialised statistical operations as base functions. For example, to perform a basic linear regression in R, one must simply pass the data to the lm function, which then returns an object containing detailed information about the regression (coefficients, standard errors, residual values, etc.). This data can then be visualised by passing the results to the plot function, which is designed to visualise the results of this analysis.

In other languages with large scientific computing communities, such as Python, duplicating the functionality of lm requires the use of several third-party libraries to represent the data (NumPy), perform the analysis (SciPy), and visualise the results (mat-plotlib). As we will see in the following chapters, such sophisticated analyses can be performed with a single line of code in R.[18]

In addition, as in other scientific computing environments, the fundamental data type in R is a vector. Vectors can be aggregated and organised in various ways, but at the core, all data is represented this way. This relatively rigid perspective on data structures can be limiting, but is also logical given the application of the language. The most frequently used data structure in R is the *data frame,* which can be thought of as a matrix with attributes, an internally defined "spreadsheet" structure, or relational database-like structure in the core of the language. Fundamentally, a data frame is simply a column-wise aggregation of vectors that R affords specific functionality to, which makes it ideal for working with any manner of data.

R operates on named *data structures*. The simplest such structure is the numeric *vector*, which is a single entity consisting of an ordered collection of numbers. To set up a vector named x, say, consisting of five numbers, namely 10.4, 5.6, 3.1, 6.4 and 21.7, use the R command

```
> x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
```

This is an *assignment* statement using the *function* c() which in this context can take an arbitrary number of vector *arguments* and whose value is a vector got by concatenating its arguments end to end.[7]

A number occurring by itself in an expression is taken as a vector of length one.

Notice that the assignment operator ('<-'), which consists of the two characters ‘<’ (“less than”) and ‘–’ (“minus”) occurring strictly side-by-side and it ‘points’ to the object receiving the value of the expression. In most contexts the ‘=’ operator can be used as an alternative.

Assignment can also be made using the function `assign()`. An equivalent way of making the same assignment as above is with:

```
> assign("x", c(10.4, 5.6, 3.1, 6.4, 21.7))
```

The usual operator, <–, can be thought of as a syntactic short-cut to this.

Assignments can also be made in the other direction, using the obvious change in the assignment operator. So the same assignment could be made using

```
> c(10.4, 5.6, 3.1, 6.4, 21.7) -> x
```

If an expression is used as a complete command, the value is printed *and lost*. So now if we were to use the command

```
> 1/x
```

the reciprocals of the five values would be printed at the terminal (and the value of `x`, of course, unchanged).

The further assignment

```
> y <- c(x, 0, x)
```

would create a vector `y` with 11 entries consisting of two copies of `x` with a zero in the middle place.[20]

Vectors can be used in arithmetic expressions, in which case the operations are performed element by element. Vectors occurring in the same expression need not all be of the same length. If they are not, the value of the expression is a vector with the same length as the longest vector which occurs in the expression. Shorter vectors in the expression are *recycled* as often as need be (perhaps fractionally) until they match the length of the longest vector. In particular a constant is simply repeated. So with the above assignments the command

```
> v <- 2*x + y + 1
```

generates a new vector `v` of length 11 constructed by adding together, element by element, `2*x` repeated 2.2 times, `y` repeated just once, and `1` repeated 11 times.

The elementary arithmetic operators are the usual `+`, `-`, `*`, `/` and `^` for raising to a power. In addition all of the common arithmetic functions are available. `log`, `exp`, `sin`, `cos`, `tan`, `sqrt`, and so on, all have their usual meaning. `max` and `min` select the largest and smallest elements of a vector respectively. `range` is a function whose value is a vector of length two, namely `c(min(x), max(x))`. `length(x)` is the number of elements in `x`, `sum(x)` gives the total of the elements in `x`, and `prod(x)` their product.

Two statistical functions are `mean(x)` which calculates the sample mean, which is the same as `sum(x)/length(x)`, and `var(x)` which gives

sum((x-mean(x))^2)/(length(x)-1)

or sample variance. If the argument to `var()` is an *n*-by-*p* matrix the value is a *p*-by-*p* sample covariance matrix got by regarding the rows as independent *p*-variate sample vectors.

`sort(x)` returns a vector of the same size as `x` with the elements arranged in increasing order; however there are other more flexible sorting facilities available (see `order()` or `sort.list()` which produce a permutation to do the sorting).

Note that `max` and `min` select the largest and smallest values in their arguments, even if they are given several vectors. The *parallel* maximum and minimum functions `pmax` and `pmin` return a vector (of length equal to their longest argument) that contains in each element the largest (smallest) element in that position in any of the input vectors.

For most purposes the user will not be concerned if the “numbers” in a numeric vector are integers, reals or even complex. Internally calculations are done as double precision real numbers, or double precision complex numbers if the input data are complex.

To work with complex numbers, supply an explicit complex part. Thus

sqrt(-17)

will give `NaN` and a warning, but

sqrt(-17+0i)

will do the computations as complex numbers.

Character quantities and character vectors are used frequently in R, for example as plot labels. Where needed they are denoted by a sequence of characters delimited by the double quote character, e.g., `"x-values"`, `"New iteration results"`.

Character strings are entered using either matching double (`"`) or single (`'`) quotes, but are printed using double quotes (or sometimes without quotes). They use C-style escape sequences, using `\` as the escape character, so `\\` is entered and printed as `\\`, and inside double quotes `"` is entered as `\"`. Other useful escape sequences are `\n`, newline, `\t`, tab and `\b`, backspace—see `?Quotes` for a full list.

Character vectors may be concatenated into a vector by the `c()` function; examples of their use will emerge frequently.

The `paste()` function takes an arbitrary number of arguments and concatenates them one by one into character strings. Any numbers given among the arguments are coerced into character strings in the evident way, that is, in the same way they would be if they were printed. The arguments are by default separated in the result by a single blank character, but this can be changed by the named argument, `sep=`*`string`*, which changes it to *`string`*, possibly empty.[20]

For example

> labs <- paste(c("X","Y"), 1:10, sep="")

makes `labs` into the character vector

c("X1", "Y2", "X3", "Y4", "X5", "Y6", "X7", "Y8", "X9", "Y10")

Note particularly that recycling of short lists takes place here too; thus `c("X", "Y")` is repeated 5 times to match the sequence `1:10`.

Vectors are the most important type of object in R, but there are several others which we will meet more formally in later sections.

- *matrices* or more generally *arrays* are multi-dimensional generalizations of vectors. In fact, they *are* vectors that can be indexed by two or more indices and will be printed in special ways.
- *factors* provide compact ways to handle categorical data.
- *lists* are a general form of vector in which the various elements need not be of the same type, and are often themselves vectors or lists. Lists provide a convenient way to return the results of a statistical computation.
- *data frames* are matrix-like structures, in which the columns can be of different types. Think of data frames as 'data matrices' with one row per observational unit but with (possibly) both numerical and categorical variables. Many experiments are best described by data frames: the treatments are categorical but the response is numeric.
- *functions* are themselves objects in R which can be stored in the project's workspace. This provides a simple and convenient way to extend R.[19]

3. Text Regression

Cross-validation and regularisation are both powerful tools that allow us to use complex models that can mimic very intricate patterns in our data without overfitting. One of the most interesting cases in which we can employ regularisation is when we use text to predict some continuous output; for example, we might try to predict how volatile a stock will be based on its IPO filings. When using text as an input for a regression problem, we almost always have far more inputs (words) than observations (documents). If we have more observations than 1-grams (single words), we can simply con- sider 2-grams (pairs of words) or 3-grams (triplets of words) until we have more n-grams than documents. Because our data set has more columns than rows, unregularised linear regression will always produce an overfit model. For that reason, we have to use some form of regularisation to get any meaningful results.

To give you a sense of this problem, we'll work through a simple case study in which we try to predict the relative popularity of the top-100-selling books that O'Reilly has ever published using only the descriptions of those books from their back covers as input. To transform these text descriptions into a useful set of inputs, we'll convert each book's description into a vector of word counts so that we can see how often words such as "the" and "Perl" occur in each description. The results of our analysis will be, in theory, a list of the words in a book's description that predict high sales.[14]

To get started, let's load in our raw data set and transform it into a document term matrix using the tm package:

```
    ranks <- read.csv('data/oreilly.csv', stringsAsFactors =
FALSE) library('tm')
    documents <- data.frame(Text = ranks$Long.Desc.)
row.names(documents) <- 1:nrow(documents)
    corpus <- Corpus(DataframeSource(documents))
corpus <- tm_map(corpus, tolower)
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, removeWords, stopwords('english'))
    dtm <- DocumentTermMatrix(corpus)
```

Here we've loaded in the ranks data set from a CSV file, created a data frame that contains the descriptions of the books in a format that tm understands, created a corpus from this data frame, standardized the case of the text, stripped whitespace, removed the most common words in English, and built our document term matrix. With that work done, we've finished all of the substantive transformations we need to make to our data. With those finished, we can manipulate our variables a little bit to make it easier to describe our regression problem to glmnet:

```
x <- as.matrix(dtm) y <- rev(1:100)
```

Here we've converted the document term matrix into a simple numeric matrix that's easier to work with. And we've encoded the ranks in a reverse encoding so that the highest-ranked book has a y-value of 100 and the lowest-ranked book has a y-value of 1. We do this so that the coefficients that predict the popularity of a book are positive when they signal an increase in popularity; if we used the raw ranks instead, the coefficients for those same words would have to be negative. We find that less intuitive, even though there's no substantive difference between the two coding systems.

Finally, before running our regression analysis, we need to initialize our random seed and load the glmnet package:

```
set.seed(1)
library(‘glmnet')
```

Having done that setup work, we can loop over several possible values for Lambda to see which gives the best results on held-out data. Because we don't have a lot of data, we do this split 50 times for each value of Lambda to get a better sense of the accuracy we get from different levels of regularisation. In the following code, we set a value for Lambda, split the data into a training set and test set 50 times, and then assess our model's performance on each split.[15,16]

```
    performance <- data.frame()
    for (lambda in c(0.1, 0.25, 0.5, 1, 2, 5)) {
    for (i in 1:50) {
    indices <- sample(1:100, 80) training.x <- x[indices, ]
training.y <- y[indices]
    test.x <- x[-indices, ] test.y <- y[-indices]
    glm.fit <- glmnet(training.x, training.y) predicted.y <-
predict(glm.fit,    test.x,    s    =    lambda)    rmse    <-
sqrt(mean((predicted.y - test.y) ^ 2))
    performance <- rbind(performance, data.frame(Lambda =
lambda,
    } }
    Iteration = i, RMSE = rmse))
```

After computing the performance of the model for these different values of Lambda, we can compare them to see where the model does best:

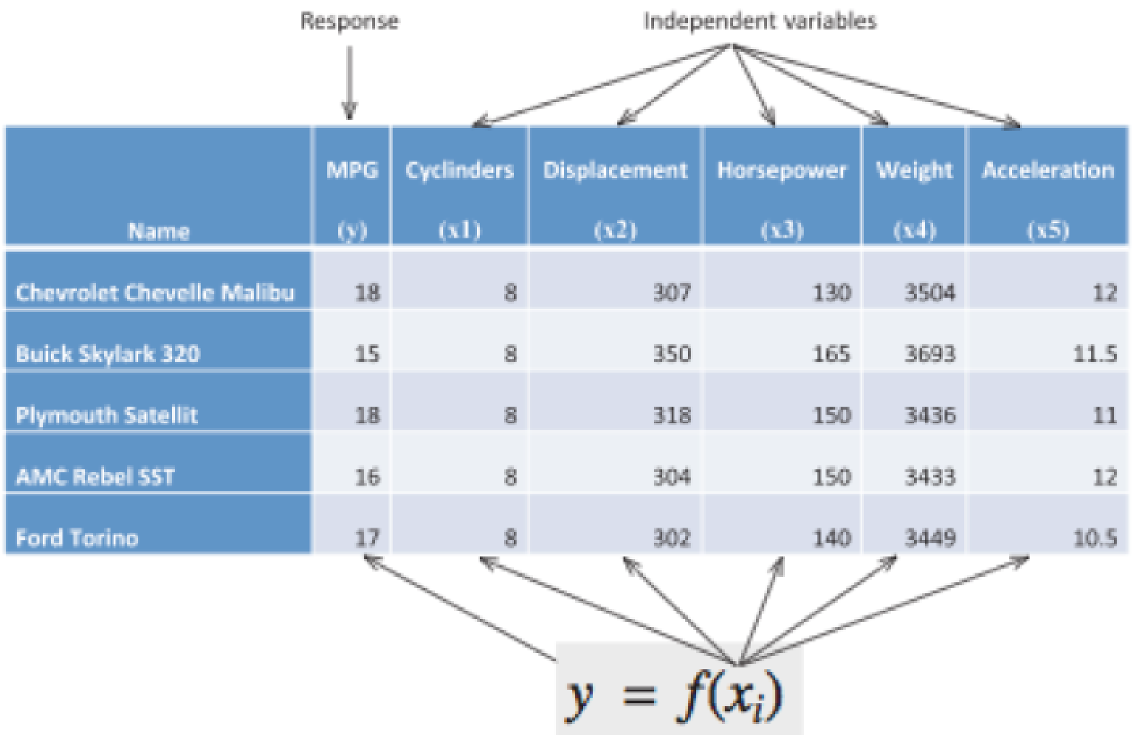
```
    ggplot(performance, aes(x = Lambda, y = RMSE)) +
stat_summary(fun.data = 'mean_cl_boot', geom = 'errorbar') +
stat_summary(fun.data = 'mean_cl_boot', geom = ‘point')
```

4. Building survey model from data

These models describe important relationships in the data, including the strength and direction—positive or negative—of the relation. The models can encode linear and nonlinear relationships in the data. They can also be used to confirm a hypothesis about relationships. All these uses help to summarise and understand the data. However, one of the most widely used applications of a model is for making predictions. For example, a data set of historical purchases along with customer geographical and demographic data (such as the customer’s age, location, salary, and so on) could be collected and used to generate a model that encodes what type of products clients purchase. Once the model is built, it could be used to identify from a list of potential clients those most likely to make a purchase, and customers on this prioritised list could be targeted with marketing material or other promotions. (see Figure 4)

In this chapter, we will review how models can be built from data sets. A model is usually built to predict values for a specific variable. For example, were a data set composed of historical data containing attributes of pharmaceuticals and their observed side effects to be collected, a model can be generated from this data to predict the side effects from the pharmaceuticals' attributes.

A variable that a model is to predict is often referred to as a *y*-variable or *response* variable. The variables that will be encoded in the model and used in predicting this response are referred to as the *x*-variables or the *independent variables*. In Figure 6.1, a data table composed of cars is used to generate a model. Because we want the model to predict the car's fuel efficiency, we have chosen the response variable to be miles per gallon (*MPG*).[17] Other variables will be used as independent variables (*x*-variables). In this case, these will be *Cylinders* (x_1), *Displacement* (x_2), *Horsepower* (x_3), *Weight* (x_4), and *Acceleration* (x_5). A generalized format for the model is shown where some function of the independent variables (x_i) is used to predict the response (y), which in this case is *MPG*.



Name	MPG (y)	Cyclinders (x1)	Displacement (x2)	Horsepower (x3)	Weight (x4)	Acceleration (x5)
Chevrolet Chevelle Malibu	18	8	307	130	3504	12
Buick Skylark 320	15	8	350	165	3693	11.5
Plymouth Satellit	18	8	318	150	3436	11
AMC Rebel SST	16	8	304	150	3433	12
Ford Torino	17	8	302	140	3449	10.5

Figure 4 - Data model

Models built to predict categorical variables (such as a binary variable or a nominal variable) are referred to as *classification* models, whereas models that predict continuous variables are called *regression* models. There are many ways to generate classification and regression models. For example, a *classification tree* is a method for building a classification model while a *multiple linear regression* is a method for building a *regression model*. Specific approaches may have restrictions relating to the types of variables that can be used in the model as, for example, a model that requires continuous variables to have a normal frequency distribution. For certain types of models it is possible to fine-tune the performance of the model by varying different parameters. In building a model, it will be important to understand the restrictions placed on the types of independent or response variables or both, as well as how to optimize the performance of the model by varying the values of the parameters. Another way in which approaches to modelling differ is in the ease of access to the internal calculations, otherwise known as the *transparency* of the model, in order to explain the results: is it possible to understand how the model calculated a prediction or is the model a “black box” that only calculates a prediction result with no corresponding explanation? Issues related to transparency may be important in explaining the results when the model is deployed in certain situations.[18]

Although the response variable is known, when building models it is not always apparent beforehand which variables should be used as independent variables. Therefore, the selection of the independent variables is an important step in building a model. A good model will make reliable predictions, be plausible, and use as few independent variables as possible. These approaches can be used to prioritize candidate independent variables to use in building a model, especially where there are many variables to consider. However, care should be taken when using statistical tests to prioritize large numbers of potential independent variables as a correction may need to be used. For example, a matrix of scatterplots could be used to visually identify which variables have the strongest relationship to the response variable. In addition, knowledge of the problem can also guide the choice of variables to use in the models. Alternatively, we could build multiple models with different combinations of independent variables and select the best fitting model.

Another issue to consider when selecting independent variables is the relationship between the independent variables. Combinations of variables that have strong relationships to each other should be avoided since they will be essentially encoding the same relationship to the response. Including all the variables from each group of strongly related variables produces overly complex models (violating the "as simple as possible rule") and with some approaches to modelling can produce results that are difficult to interpret.

In developing a model, it may also be necessary to use derived variables, that is, a new variable that is a function of one or more variables. For example, if the model expects the variables of a data set to have a normal frequency distribution and some variables have an exponential frequency distribution, it may be necessary to create new variables using a log trans- formation. As another example, because most modelling methods require numeric data, if a data set has nominal variables that will be used in the model, the values of these variables must be transformed into numbers. For example, if *color* is an important variable with values "Blue," "Green," "Red," and "Yellow," *color* could be transformed into a series of binary dummy variables.[19]

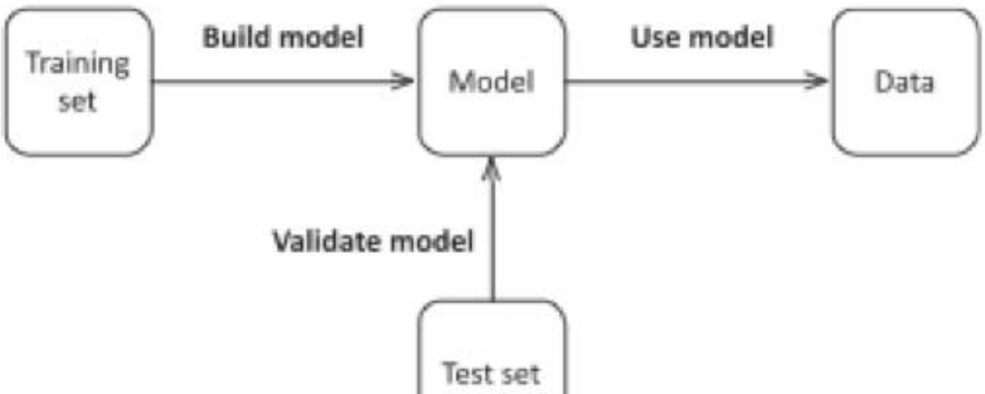


Figure 5 - Training and testing sets

In this chapter, we discuss how to generate models from data sets. The data set used to build a model is referred to as the *training set.* To objectively test the performance of a generated model, a *test* set with observations different from those in the training set is used to test how well the model performs. The model uses the values of each observation in the test set to predict a value for the response variable. From these predictions, a variety of metrics, such as the number of correct versus incorrect predictions made, are used to assess the accuracy of the model. The use of training and test sets is illustrated in Figure 5.

A good way to build and test a model would be to use all the observations in the original data set as the training set to build the model and to use new, independent observations as the test set to measure accuracy. However, because the number of available test sets is often small, a common way to test the performance of a model is to use a category of methods called *cross- validation*. In the *k-fold partitioning* method, the original data set is divided into k equally sized partitions. The model is measured k times. In the first iteration, one of the partitions is selected as the test set and the remaining partitions comprise the training set. The model is tested and an accuracy score is generated. In each subsequent iteration, a partition different from any already used as a test set is selected as the test set and the remaining partitions become the training set. Another score is calculated. At the end of this process, the accuracy of the model is based on the average of the k scores. For example, suppose we partition a data set into 10 partitions where each partition consists of observations randomly selected from the data set. In each of the 10 iterations, we designate one partition (10% of the data set) as the test set and the other 9 partitions (90% of the data set) as the training set. At the end of the 10 iterations, an average of the 10 scores is used to assess the model’s accuracy. Taking k-fold partitioning to an extreme would result in the case where k is the number of observations in the data set and each partition contains a single observation. This is a cross-validation method known as leave-one-out.

In cross-validation, each partition will have been used as a test set or, in other words, every observation in the data set will have been tested once. This ensures that a prediction will be calculated for every observation in the data set and avoids introducing *bias* into a model. Bias is a measure of the model's accuracy and indicates how close the predictions of the response value made by the model are to the actual response value of new observations. It can be introduced when models become overly complex by optimizing the model for just the training set used to build the model. When the performance is tested for these overtrained models against either a separate test set or through cross-validation, the performance will be poorer. In cross-validation methods, bias can be introduced when training sets overlap (some observations are used more than once) or the combined training sets do not cover the data set (some observations are never used).[15]

For classification models, one way to assess the performance of a model is to look at the results of applying the models (such as the results from a test set or the cross-validation results) and determine how many observations are correctly or incorrectly classified. The accuracy or *concordance* of the model is based on the proportion or percentage of correctly predicted observations in comparison to the whole set. For example, if the test set contained 100 observations and the model predicted 78 correctly (22 incorrectly), then the concordance would be 78/100 or 78%.

A common type of classification model is a model to predict a binary response, where a true response is coded as 1 and a false response is coded as 0. For example, a model could be built to predict, based on geological data, whether there is evidence of an oil deposit, with a true response.

5. Applying classification techniques to survey systems

Classification is one of the fundamental cognitive processes used to organize and apply our knowledge about the world. It is common both in everyday life and in business, where we might want to classify customers, employees, transactions, stores, factories, devices, docu- ments, or any other types of instances into a set of predefined meaningful classes or categories. It is therefore not surprising that building classification models by analyzing available data is one of the central data mining tasks that attracted more research interest and found more applications than any other task studied in the field.

The classification task consists in assigning instances from a given domain, described by a set of discrete- or continuous-valued attributes, into a set of classes, which can be considered values of a selected discrete target attribute, also called the target *concept*. Correct class labels are generally unknown, but are provided for a subset of the domain. It can be used to create the classification model, which is a machine-friendly representation of the knowledge needed to classify any possible instance from the same domain, described by the same set of attributes. This follows the general assumptions of inductive learning, of which the classification task is the most common instantiation.

The assumed general unavailability of class labels, but their availability for a given sub- set of the domain, may seem at first inconsistent, but it is essential for the idea of inductive inference on which all data mining methods are based. It also perfectly corresponds to the requirements of most practical applications of classification, where the class represents some property of classified instances that is either hard and costly to determine, or (more typically) that becomes known later than is needed. This is why applying a classification model to assign class labels to instances is commonly referred to as *prediction*.[10]

The term “concept” comes from the traditional machine learning terminology and is used to refer to a classification function $c: X \to C$, representing the true assignment of all instances from the domain to a finite set of classes (or categories) C. It can be considered simply a selected target nominal attribute. Concept values will be referred to as *class labels* or *classes*.

A particularly simple, but interesting kind of concepts is that with just a two-element set of classes, which can be assumed to be $C = \{0, 1\}$ for convenience. Such concepts are some- times called *single concepts*, opposed to *multiconcepts* with $|C| > 2$. Single concepts best correspond to the original notion of concepts, borrowed by machine learning from cognitive psychology. An instance x is said to “belong to” or “be an example of” concept c when $c(x) = 1$. When $c(x) = 0$, the instance is said “not to belong to” or “to be a negative example of” concept c. Classification tasks with single concepts will be referred to as two-class classification tasks.

The target concept is assumed to be unknown in general, except for some set of instances $D \subset X$ (otherwise no data mining would be possible). Some or all of these available *labeled instances* constitute the *training set* $T \subseteq D$.

A *classification model* $h: X \to C$ produces class predictions for all instances $x \in X$ and is supposed to be a good approximation of the target concept c on the whole domain. Classification models are briefly called *classifiers*, although the latter term sometimes also refers to classification algorithms, used to create classification models.

For two-class classification tasks (single concepts), a particular kind of *scoring* classification models deserves special interest. These are the classification models that predict class labels in a two-step process: they first map instances into real numbers called scores and then they assign one class label (1, by convention) to instances with sufficiently high scores and the other class label (0) to the remaining instances.

More precisely, a scoring model is represented by a scoring function $\square : X \to \square$ and a labeling function $\square : \square \to \{0, 1\}$. The former assigns real-valued scores to all instances from the domain, and the latter converts these scores to class labels using a cutoff rule, such as

$$\square(r) = \begin{cases} 1 & \text{if } r \geq \square \\ 0 & \text{otherwise} \end{cases} \qquad (1.1)$$

where $\square$ is a cutoff value. The model is then the composition of its scoring and labeling functions, $h(x) = \square(\square(x))$.

It is a common convention to consider scoring classification models sharing the same scoring function and differing only in the labeling function (i.e., using different cutoff values) as the same single model, working in different *operating points*. Classification algorithms capable of generating scoring classification models typically create a scoring function and a cutoff value for one *default* operating point, but a number of other operating points can be obtained by using different cutoff values.[12]

Classification models that generate class labels directly, without scoring and labeling functions, are sometimes called *discrete* classifiers.

A related interesting and useful special kind of classification models are *probabilistic* classifiers, which estimate class probabilities for instances being classified, and then make predictions based on these probabilities. A probabilistic classifier assigns to each instance $x \in X$ and class $d \in C$ a probability estimate $P(d|x)$ of instance x belonging to class d of the target concept c. The estimated class probabilities can be used to generate class labels using the obvious *maximum-probability* rule:

$$h(x) = \arg\max_{d \in C} P(d|x) \qquad (1.2)$$

or – under nonuniform misclassification costs – the less obvious minimum-cost rule, as dis- cussed in Section 6.3.3.

For two-class tasks, probabilistic classifiers constitute a particularly common subclass of scoring classifiers, with the estimated probabilities of class 1 for particular instances considered scores, i.e., $\square(x) = P(1|x)$.

LimeSurvey has many question types. Some of them presented here:

Question types

- Arrays
  - Array
  - Array (5 point choice)
  - Array (10 point choice)
  - Array (Yes/No/Uncertain)
  - Array (Increase/Same/Decrease)
  - Array by column
  - Array dual scale
  - Array (Numbers)
  - Array (Text)
- Mask questions
  - Date
  - File upload
  - Gender
  - Language switch
  - Numerical input
  - Multiple numerical input
  - Ranking
  - Text display
  - Yes/No
  - Equation
- Multiple choice questions
  - Multiple choice

    - Multiple choice with comments

- Single choice questions
    - 5 point choice
    - List (Dropdown)
    - List (Radio)
    - List with comment
- Text questions
    - Short free text
    - Long free text
    - Huge free text
    - Multiple short text

The Array question type (sometimes referred to as *Array Multi Flexible*) further extends the List question type. Using this question type a matrix can be displayed in which columns are represented by subquestion and the same answer options are shown for each row. The text of the question can be either a specific question or a description.

In terms of output there is no difference in how responses are stored compared to question type 'List(Radio). In both cases the given answer is stored in its separate column in the result table.

Beside the most flexible array types 'Array', 'Array (Text)' and 'Array (Numbers)' LimeSurvey also supports a number of convenience array types which have predefined answer options.

Ranking question type allows you to present your participants with a list of possible answers/options, which they may then rank in order of preference.

Text display question type does not collect input from the user and simply displays text. It can be used to provide further instructions or a design break in the survey.

Sometimes you want the participant to mark more than one answer option in the same question; this is achieved using checkboxes.

Multiple choice question type can collect input of multiple selections through checkboxes.

Multiple choice with comments question type can collect input of multiple selections through checkboxes, while allowing the user to provide additional comments with their submissions.

Single choice questions are those where the participant can only pick a single predefined answer option.

5 point choice question shows a vertical 1 to 5 scale where the participant may select one answer option at a time.

List (Dropdown) question type collects input from a dropdown menu. You can also create subcategories within this list by using the advanced setting 'Category separator'.

List (Radio) question type collects input from a list of radio buttons.

List with comment question type displays a list of radio buttons, while allowing the participants to provide a additional comment with their submission.

The linear model representation is a special case of the parametric representation which assumes that the model's predictions are calculated by applying a representation function to attribute values and a set of real-valued parameters. This is particularly natural and extremely common for regression models which make real-valued predictions. The same approach can also be adopted to represent classification models, though. Moreover, such models can be created by the same or nearly the same algorithms as those that normally deliver regression models. This can be achieved in several ways, some of which are discussed in this chapter. The chapter will focus on issues related to adopting parametric regression methods to the classification task. This is essentially based on using a composite model representation function, consisting of a real-valued inner representation function and a discrete outer representation function that assigns class labels based on the former.[13]

Linear representation is the most common instantiation of the parametric representation family that will be more thoroughly discussed in Section 8.2 in the regression context, but can be summarized as follows:

• A fixed model *representation function* is adopted that determines the model's predicted value for an instance based on the instance's attribute values and a vector of model parameters.

• Creating a model based on a training set consists in estimating its parameters.

This is in contrast to nonparametric representation, where both the representation function and parameters have to be derived from the data as part of the model creation process. We have actually already encountered these two types of model representation. Decision trees can be viewed as instantiations of nonparametric representation, with the tree structure playing the role of the representation function, and per-leaf class distributions serving as model parameters. The naïve Bayes classifier adopts a parametric representation, on the other hand, using prior class probabilities and conditional attribute value probabilities as parameters to the fixed representation function that calculates conditional class probabilities given attribute values based on the Bayes rule and the independence assumption.

In principle, parametric model representation is applicable to both classification and regression models, since the employed representation function can be real valued or discrete valued. However, a model representation function is useful only if reasonably efficient and effective parameter estimation algorithms are available. This unquestionably favors real-valued representation functions. Several approaches to parametric classification are therefore based on wrapping the latter so that they can be used for class label prediction. This is most natural and easiest to achieve with two-class classification tasks.[14]

The representation function for parametric classification is the composite of a real-valued *inner representation function* and another (outer) function that assigns binary class labels (as always in this book, assumed to be from the {0, 1} set) based on its values. The inner representation function is calculated based on attribute values and model parameters:

$g(x) = F(\mathbf{a}(x), \mathbf{w})$ More specifically, for the linear representation we have

or, assuming $a_{n+1}(x) = 1$ for all x to include the *intercept* term $\square_{n+1}$ in the summation

$g(x) = \square_i a_i(x) = \mathbf{w} \square \mathbf{a}(x)$

i=1 $\square_i a_i(x) + \square_{n+1}$

where $\square$ denotes the dot product operator, $\mathbf{w}$ is the parameter vector, and $\mathbf{a}(x)$ is the vector of attribute values $a_1(x), a_2(x), \ldots, a_{n+1}(x)$. Whenever referring to $\mathbf{w}$ or $\mathbf{a}(x)$ in this chapter, they will be assumed to contain $n + 1$ elements, i.e., include the intercept term $\square_{n+1}$ and the

```
    dmr.linreg
    #  parameter vector for the lcg.plot function
    w.plot <- c(2, -3, 4)
    repf.linear(lcdat.plot[1:10,1:2], w.plot)
    grad.linear(lcdat.plot[1:10,1:2], w.plot)
      # parametric model for the lcg.plot function
    m.plot  <-  ‘class<-‘(list(repf=repf.linear,  w=w.plot),
"par")
    predict(m.plot, lcdat.plot[1:10,1:2])
```

corresponding fictitious attribute value $a_{n+1}(x)$. When the last elements of these vectors have to be omitted, this will be explicitly indicated by adding the $1 : n$ subscripts.

There are two major approaches to assigning binary class labels based on linearly represented real-valued inner predictions:

Boundary modeling. Assuming that the inner representation function represents a boundary between regions of different classes,

Probability modeling. Assuming that the inner representation function represents, possibly indirectly, class probabilities.

In boundary modeling, hypersurfaces (in the attribute value space) separating positive and negative instances, called *decision boundaries*, are represented parametrically. They partition the domain into regions, with each region assigned a class label.

Probability modeling is a family of approaches that use a parametric representation of class probabilities. For two-class tasks this reduces to representing the probability of class 1. The latter may be then used to predict class labels as with any probabilistic classifiers, i.e., by using the maximum probability rule, the minimum cost rule presented in Section 6.3.3, or adjusting operating points by the ROC analysis or similar methods, as discussed in Section 7.2.5.

These two approaches lead to the following two most commonly used types of outer rep- resentation functions for linear classification:

• threshold representation, which is a standard way to perform boundary modeling,

• logit representation, which is is the most popular instantiation of probability modeling. We will see that, while they differ in important details, they have actually a lot in common.[9]

For two-class classification tasks partitioning the domain into the positive and negative regions can be easily achieved by comparing a parametric representation function against a threshold. Without loss of generality, the latter may be assumed to be 0, which yields the following model representation:

$h(x) = H(g(x)) =$ 1 if $g(x) \geq 0$ (5.4) 0 otherwise

For a threshold parametric classification model defined as above predictions are obtained by applying the unit step function H to the inner representation function g. The latter determines a hypersurface in the (n + 1)-dimensional space (with dimensions corresponding to a_1, a_2, ... , a_n, and g). By comparing against 0 the projection of this hypersurface to n dimensions (corresponding to a_1,a_2, ... ,a_n) is determined. In general, it may yield one or more n-dimensional hypersurfaces where g crosses the a_1, a_2, ... , a_n hyperplane. The model function h, which is a binary-valued function in an n-dimensional space, assigns 0 or 1 to regions separated by a number of n-dimensional surfaces, obtained by the projection of an (n + 1)-dimensional surface.

It is common to use the sign rather than the unit step function for threshold parametric classification models, assuming class labels are from the {−1, 1} set rather than the {0, 1} set. This chapter sticks with the latter, to preserve consistency with conventions used for presenting other classification algorithms in this book. However, on several occasions the binary true or predicted class labels will be used as numbers in equations (and, correspondingly, code examples), whereas the discussion of classification in other chapters usually does not rely on the numeric interpretation of class labels[15].

The threshold representation instantiated for linear classification takes the following form:

{

$h(x) = H(\mathbf{w}\square\mathbf{a}(x)) =$ 1 if $\mathbf{w}\square\mathbf{a}(x) \geq 0$ (5.5) 0 otherwise

In this case, the decision boundary separating the domain regions assigned the 0 and 1 class labels, represented by the parameter vector, is a hyperplane in *n* dimensions. The target con- cept is said to be *linearly separable* on a dataset if there exists a hyperplane that separates all instances of different classes in the dataset (i.e., there exists a parameter vector that yields correct predictions for all instances in the dataset). The dataset is then also said to be *linearly separable* with respect to the target concept.

Conclusions for the second chapter

In the second chapter it is described the structure of LimeSurvey project and module's structure, the structure of database's part needed for implementing data mining module, the classification algorithms were adapted for special features which are proper for survey systems and optimised for usage with such systems as the open source LimeSurvey project. Also research about possibilities of usage data mining techniques with such software systems was made and regularities in development web-based questionnaire poll systems with applied data mining abilities were evolved. The results were tested and approved.

CHAPTER 3. SOFTWARE IMPLEMENTATION OF DATA MINING MODULE

1. Selection of programming technologies

For implementation data mining module of open source survey system “LimeSurvey” it was selected LAMP technology, as the most popular platform to develop web applications. LAMP means Linux, Apache, MySQL and PHP.

Because of “LimeSurvey” is written on PHP language as common web application, it was had to choose PHP as main language for implementation. PHP (recursive acronym for *PHP: Hypertext Preprocessor*) is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML. [23] PHP has good language features such as support for object-oriented approach and some functional abilities like anonymous functions and closures. But in the data mining field it has week tools, and it is not very strong language to implement data mining techniques. So in Internet you will find not a lot of libraries useful for data mining systems written in PHP. Because of this, it was selected additional language for implementation - R language.

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible.

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,

- graphical facilities for data analysis and display either on-screen or on hardcopy, and
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.[24]

MySQL was selected as database management system. It is the most popular database for PHP projects, and it has enough performance.

MySQL is (as of July 2013) the world's second most widely used relational database management system (RDBMS) and most widely used open-source RDBMS. MySQL is a popular choice of database for use in web applications, and is a central component of the widely used LAMP open source web application software stack.

Major features as available in MySQL 5.6:

- Cross-platform support
- Stored procedures, using a procedural language that closely adheres to SQL/PSM
- Triggers
- Cursors
- Updatable views
- Online DDL when using the InnoDB Storage Engine.
- Information schema
- Sub-SELECTs (i.e. nested SELECTs)
- Built-in Replication support (i.e. Master-Master Replication & Master-Slave Replication) with one master per slave, many slaves per master. Multi-master replication is provided in MySQL Cluster, and multi-master support can be added to unclustered configurations using Galera Cluster.
- Full-text indexing and searching

- Embedded database library
- Unicode support and etc.[25]

Because of “LimeSurvey” uses Yii Framework as main library, it is used also for development custom data mining module.

Yii is a high-performance PHP framework best for developing Web 2.0 applications.

Yii comes with rich features: MVC, DAO/ActiveRecord, I18N/L10N, caching, authentication and role-based access control, scaffolding, testing, etc. It can reduce your development time significantly.[26]

2. Description of implementation classes and methods

To implement data mining module in questionnaire poll system Yii Framework was used. It supports MVC design pattern which provide abilities to separate business logic (Model) from presentation layer (View) and controlling execution of request (Controller). So the implementation of the module include controllers, models, views and some helper classes.

Here is the description of the most important classes.

SurveyController - controller to work with questionnaires.

It consists of methods presented in Table 1.

Table 1 - SurveyController methods

accessRules()	Specifies the access control rules. This method is used by the 'accessControl' filter.
actionView()	Displays a particular model.
actionCreate()	Creates a new model. If creation is successful, the browser will be redirected to the 'view' page.
actionUpdate()	Updates a particular model. If update is successful, the browser will be redirected to the 'view' page.

Table 1 - SurveyController methods

actionDelete()	Deletes a particular model. If deletion is successful, the browser will be redirected to the 'admin' page.
actionIndex()	List all surveys blanks
actionAdmin()	Manages all models.
loadModel()	Returns the data model based on the primary key given in the GET variable. If the data model is not found, an HTTP exception will be raised.

StatisticsController - controller which is used for viewing statistics for results of survey.

It consists of further methods (see Table 2).

Table 2 - StatisticsController methods

accessRules()	Specifies the access control rules. This method is used by the 'accessControl' filter.
init()	Initialize controller object
actionIndex()	Show statistics for surveys

AnalyticController - controller that implement data mining techniques to operate with survey reports.

Methods of this controller are presented in Table 3.

Table 3 - AnalyticController methods

actionView()	View appropriate report based on question report. It includes request of R scripts to work with data structures.

Table.3 - AnalyticController methods

init()	Initialize controller object
actionIndex()	Show analytics for surveys
actionCommon()	Helper function for common operations
actionEducationProcess()	Action to rebuild education data sets for using in data analytics

Class **ModelStatistic** implements main business logic of developed system’s module.

Methods of this class are presented in Table 4.

Table.4 - ModelStatistics methods

init()	Initialize model’s object
ToAssosiative()	Convert array of multivalve answers to associative array.
ToPercentage()	Convert values of given array to percents values
ToPersentage1()	Helper method for previous method
ToArray()	Converts given associative array to common array
ToArrayInverse()	Converts array to transformed array to be used in classification algorithms
transformConditions()	Change some conditions like ‘or’ and ‘and’ to appropriate operations with data arrays
getMethodic()	Gets appropriate method to consolidate results: addition or subtraction.

Table 4 - ModelStatistics methods

getFrequency()	Gets frequency for multivalve types of questions.

RProcessInterface is used in wrappers for a synchronous R interpreter process. Input commands are sent using write(). It is possible to obtain back the input, the output, the list of errors and the overall result (input + output + errors). Obtaining can be done for all data (since the launch of the process) or only for the last write(). CommandLineRProcess implements RProcessInterface and it is useful to integrate R language with PHP. Through this class it is possible to run R scripts and get results from PHP code. Methods of this class are described in Table 5.

Table 5 - RProcessInterface methods

start()	Starts the R process and also resets errors, input and output
stop()	Stops the R process
restart()	Restarts the R process (stops and starts it).
isRunning()	Checks if the R process is running
write()	Writes lines of commands to R interpreter
getAllInput()	Returns all input to the R interpreter
getAllOutput()	Returns all output from the R interpreter
getAllResult()	Returns all input, output and errors (as text or array, depending on $asArray parameter)
getLastWriteInput()	Returns the most recent input to the R interpreter (since the last call of write() method)

Table.5 - RProcessInterface methods

getLastWriteOutput()	Returns the most recent output from the R interpreter (since the last call of write() method)
getLastWriteResult()	Returns the most recent input, output and errors (since the last call of write() method)
hasErrors()	Determines if there were errors since the last call of start() method
getErrors()	Gets the array of errors (elements are of type RError) since the last call of start() method
hasLastWriteErrors()	Determines if there were errors since the last call of write() method
getLastWriteErrorCount()	Gets the number of errors that occurred since the last call of write() method
getLastWriteErrors()	Gets the array of errors (elements are of type RError) that occurred since the last call of write() method
isErrorSensitive()	Check if R process is currently sensitive to errors
setErrorSensitive()	Sets sensitivity to R errors

So it was implemented module for survey statistics and reporting.
Listings of source code can be found in Appendix.
In the next section main user forms and actions are described.

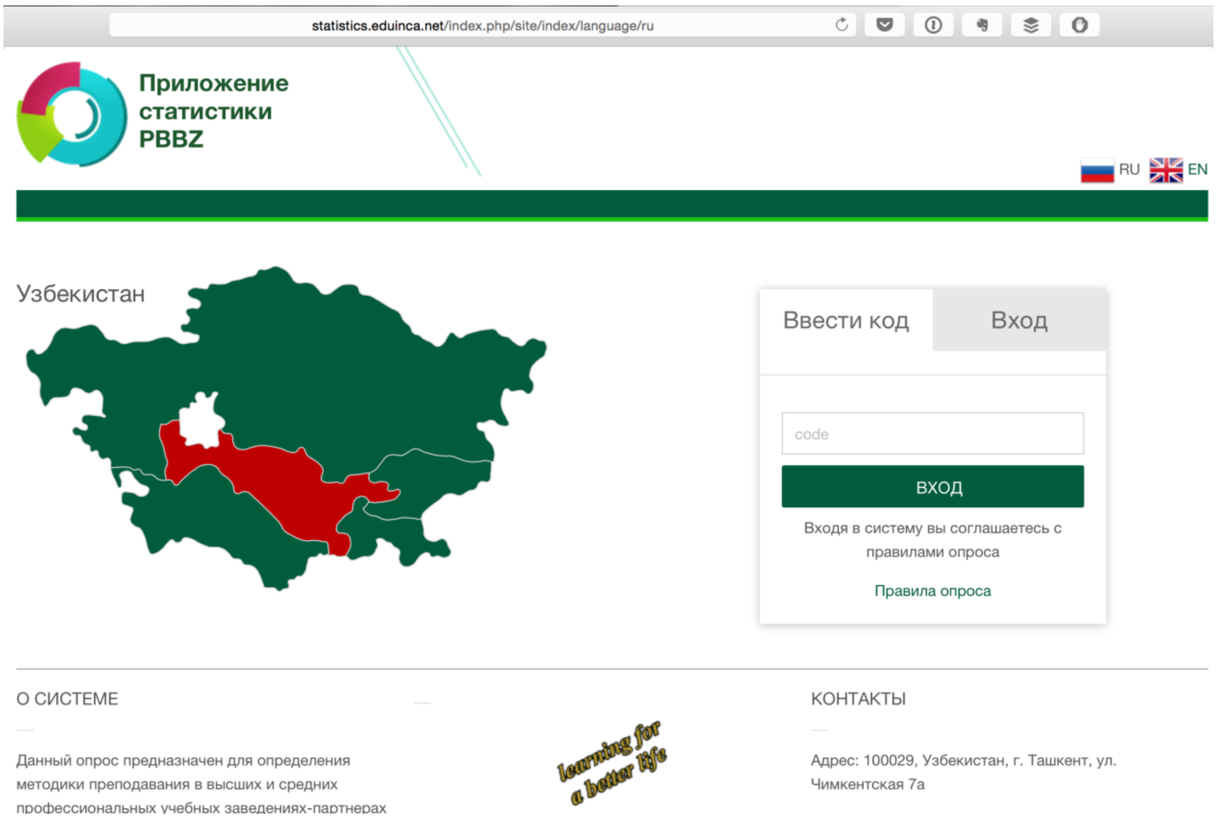
3. Description of module’s user interfaces

Enter to system

There is two different screens to for entering into system: for users who want to answer questionnaire, and administrator who can get reports and create surveys.

Enter in your address bar in browser this URL: http://statistics.eduinca.net to get main page of application for users. User must enter special code to start survey. Code is provided by administrator. That code can be used only once to prevent multiple answers from one user.

The web application is available in two languages: Russian and English. User can select one of them. Administrator have to click “Enter” tab, input login and password to enter into administration panel. In the figure 1 there is login page of application.

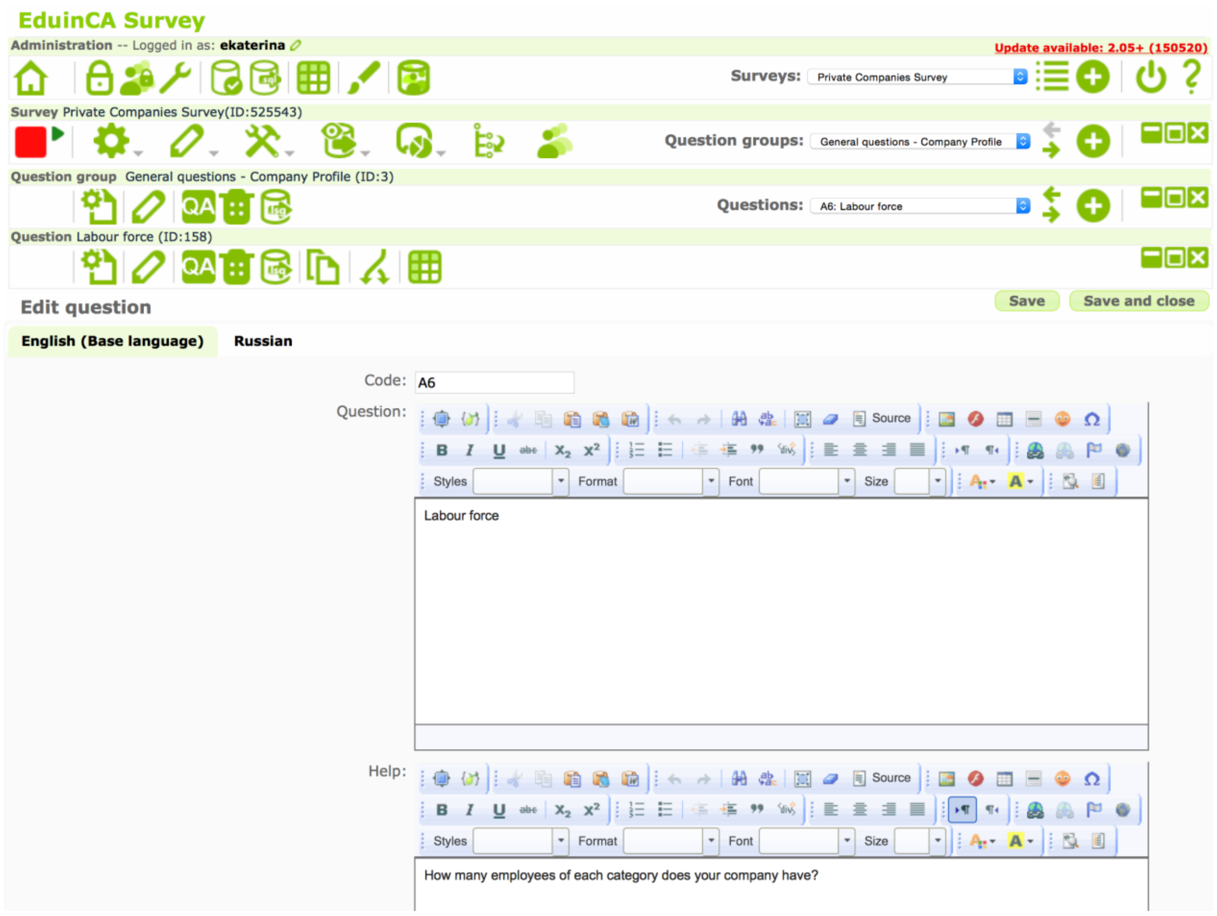

statistics.eduinca.net/index.php/site/index/language/ru
Приложение статистики PBBZ
RU
EN
Узбекистан
Ввести код
Вход
code
ВХОД
Входя в систему вы соглашаетесь с правилами опроса
Правила опроса
О СИСТЕМЕ
learning for a better life
КОНТАКТЫ

Данный опрос предназначен для определения методики преподавания в высших и средних профессиональных учебных заведениях-партнерах

Адрес: 100029, Узбекистан, г. Ташкент, ул. Чимкентская 7а

Figure 1 - Login page

In administration panel of EduinCA Survey project administrator can create surveys. First he have to create survey, then add some question groups. In one survey there may be more than one question group. Then administrator creates questions of questionnaire. Type of question may be list, multi-select, array of value or one of many others. Administration


EduinCA Survey

Administration -- Logged in as: ekaterina

Update available: 2.05+ (150520)

Surveys: Private Companies Survey

Survey Private Companies Survey(ID:525543)

Question groups: General questions - Company Profile

Question group General questions - Company Profile (ID:3)

Questions: A6: Labour force

Question Labour force (ID:158)

Edit question

English (Base language) | Russian

Code: A6

Question: Labour force

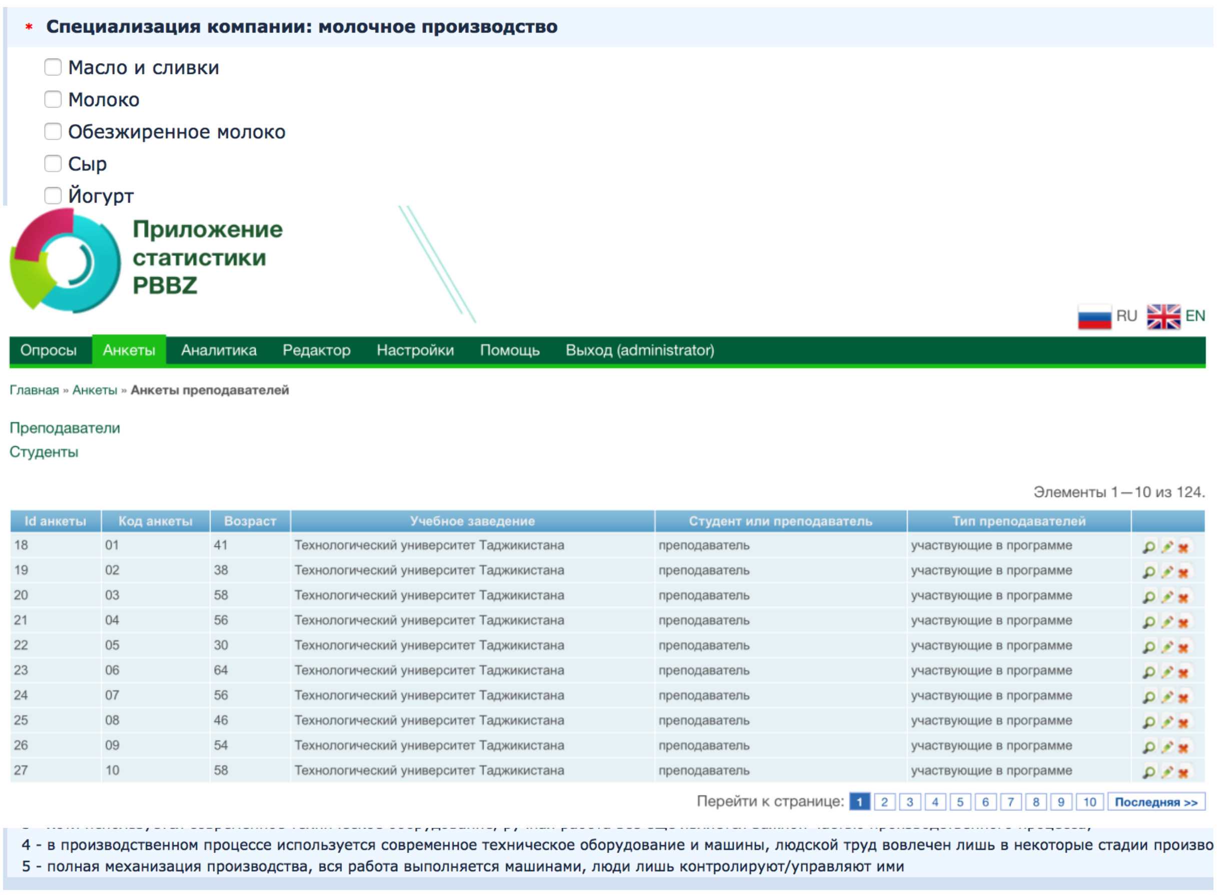
Help: How many employees of each category does your company have?

panel to operate with surveys is presented on Figure 2.

Figure 2 - Survey administration panel

When preparation of survey completed, administrator gives users access codes to complete questionnaires. Users log in with these codes, and they will see such list of questions (see Figure 3). Because LimeSurvey supports many languages, administrator can create translated questionnaires if he plans to support multilingual audience.

Figure 3 - Questionnaire list



* **Специализация компании: молочное производство**

- ☐ Масло и сливки
- ☐ Молоко
- ☐ Обезжиренное молоко
- ☐ Сыр
- ☐ Йогурт

Главная » Анкеты » **Анкеты преподавателей**

Преподаватели
Студенты

Элементы 1—10 из 124.

Id анкеты	Код анкеты	Возраст	Учебное заведение	Студент или преподаватель	Тип преподавателей	
18	01	41	Технологический университет Таджикистана	преподаватель	участвующие в программе	
19	02	38	Технологический университет Таджикистана	преподаватель	участвующие в программе	
20	03	58	Технологический университет Таджикистана	преподаватель	участвующие в программе	
21	04	56	Технологический университет Таджикистана	преподаватель	участвующие в программе	
22	05	30	Технологический университет Таджикистана	преподаватель	участвующие в программе	
23	06	64	Технологический университет Таджикистана	преподаватель	участвующие в программе	
24	07	56	Технологический университет Таджикистана	преподаватель	участвующие в программе	
25	08	46	Технологический университет Таджикистана	преподаватель	участвующие в программе	
26	09	54	Технологический университет Таджикистана	преподаватель	участвующие в программе	
27	10	58	Технологический университет Таджикистана	преподаватель	участвующие в программе	

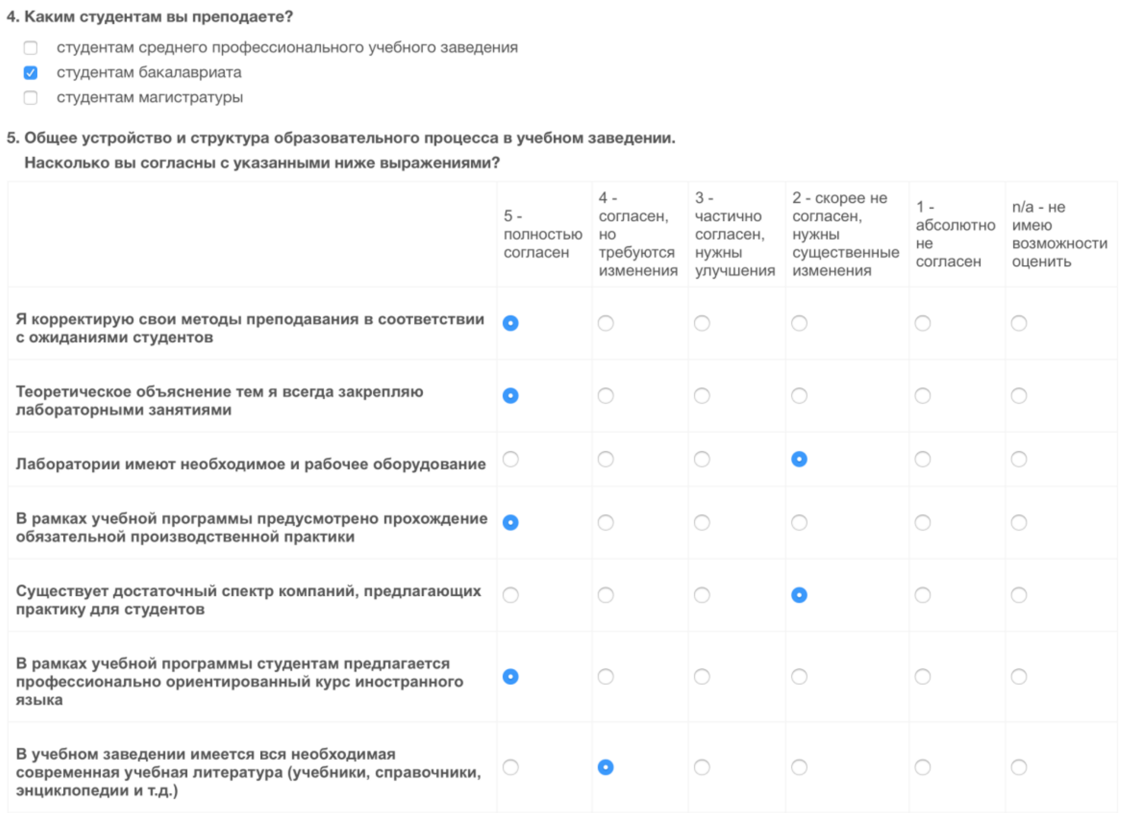
Перейти к странице: 1 2 3 4 5 6 7 8 9 10 **Последняя >>**

4 - в производственном процессе используется современное техническое оборудование и машины, людской труд вовлечен лишь в некоторые стадии произво
5 - полная механизация производства, вся работа выполняется машинами, люди лишь контролируют/управляют ими

After survey completes in statistics application administrator can view all completed questionnaires. In Figure 4 there are 124 questionnaires completed by participants.

Figure 4 - List of completed questionnaires

Administrator can view answers of every single participant. In Figure 5 sample questions are presented.



4. Каким студентам вы преподаете?

- ☐ студентам среднего профессионального учебного заведения
- ☑ студентам бакалавриата
- ☐ студентам магистратуры

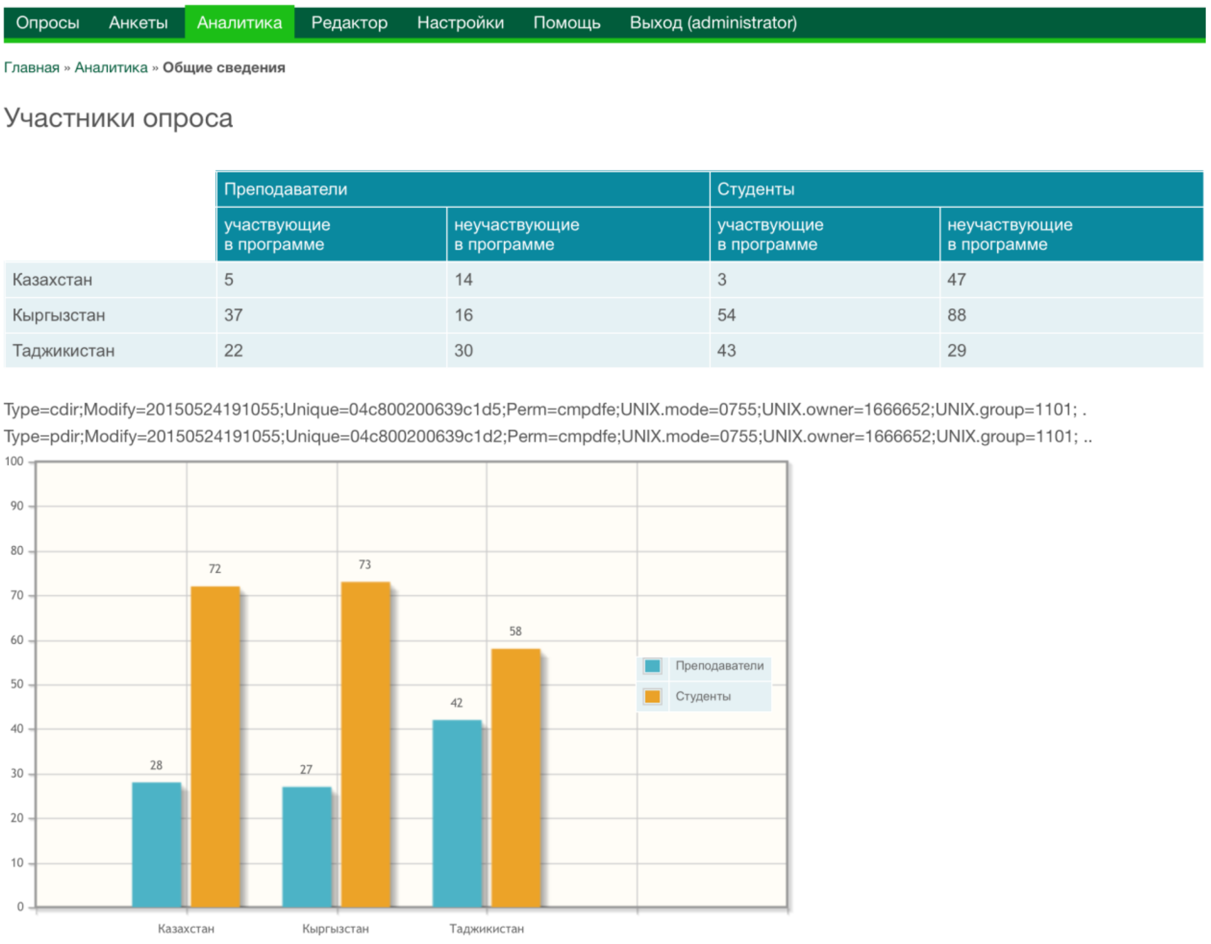
5. Общее устройство и структура образовательного процесса в учебном заведении.
Насколько вы согласны с указанными ниже выражениями?

	5 - полностью согласен	4 - согласен, но требуются изменения	3 - частично согласен, нужны улучшения	2 - скорее не согласен, нужны существенные изменения	1 - абсолютно не согласен	n/a - не имею возможности оценить
Я корректирую свои методы преподавания в соответствии с ожиданиями студентов	●	○	○	○	○	○
Теоретическое объяснение тем я всегда закрепляю лабораторными занятиями	●	○	○	○	○	○
Лаборатории имеют необходимое и рабочее оборудование	○	○	○	●	○	○
В рамках учебной программы предусмотрено прохождение обязательной производственной практики	●	○	○	○	○	○
Существует достаточный спектр компаний, предлагающих практику для студентов	○	○	○	●	○	○
В рамках учебной программы студентам предлагается профессионально ориентированный курс иностранного языка	●	○	○	○	○	○
В учебном заведении имеется вся необходимая современная учебная литература (учебники, справочники, энциклопедии и т.д.)	○	●	○	○	○	○

Figure 5 - Sample questions list

In “Analytics” section administrator can get reports and graphics about completed surveys. Common report participants by countries is in Figure 6.

To get complex reports administrator can input request sentence, for example “How many independent entrepreneurs?”. System will give him predictions about response filters based on the questionnaire structure. If prediction is wrong, administrator can manually select needed options. The request string and response filters are showed in Figure 7.

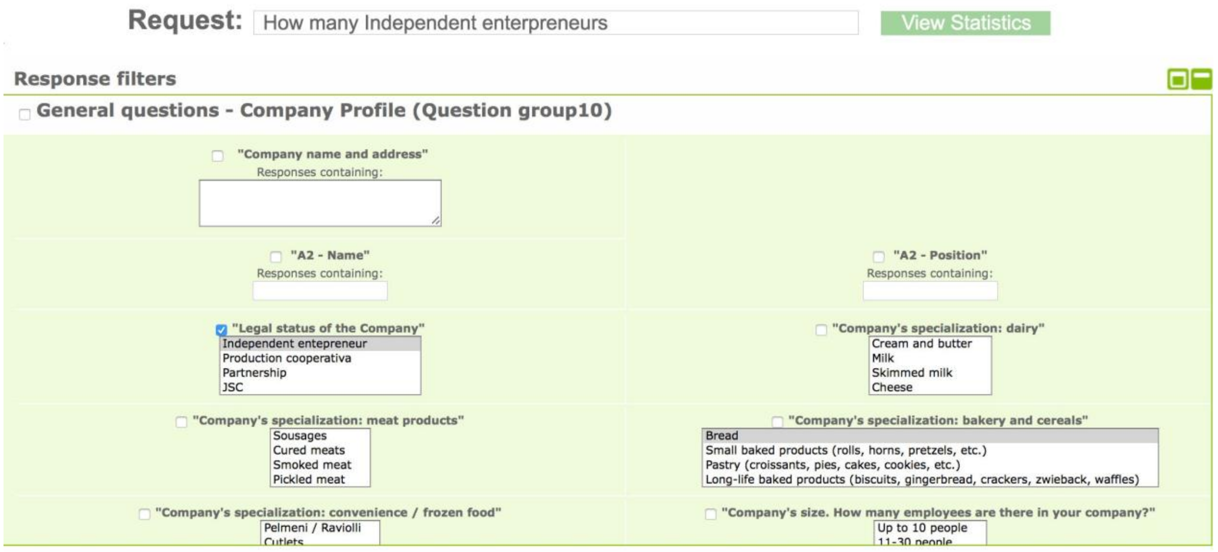


Главная » Аналитика » **Общие сведения**

Участники опроса

	Преподаватели		Студенты	
	участвующие в программе	неучаствующие в программе	участвующие в программе	неучаствующие в программе
Казахстан	5	14	3	47
Кыргызстан	37	16	54	88
Таджикистан	22	30	43	29

```
Type=cdir;Modify=20150524191055;Unique=04c800200639c1d5;Perm=cmpdfe;UNIX.mode=0755;UNIX.owner=1666652;UNIX.group=1101; .
Type=pdir;Modify=20150524191055;Unique=04c800200639c1d2;Perm=cmpdfe;UNIX.mode=0755;UNIX.owner=1666652;UNIX.group=1101; ..
```



Request: How many Independent enterpreneurs

Response filters

☐ **General questions - Company Profile (Question group10)**

☐ **"Company name and address"**
Responses containing:

☐ **"A2 - Name"**
Responses containing:

☐ **"A2 - Position"**
Responses containing:

☑ **"Legal status of the Company"**
Independent entepreneur
Production cooperativa
Partnership
JSC

☐ **"Company's specialization: dairy"**
Cream and butter
Milk
Skimmed milk
Cheese

☐ **"Company's specialization: meat products"**
Sousages
Cured meats
Smoked meat
Pickled meat

☐ **"Company's specialization: bakery and cereals"**
Bread
Small baked products (rolls, horns, pretzels, etc.)
Pastry (croissants, pies, cakes, cookies, etc.)
Long-life baked products (biscuits, gingerbread, crackers, zwieback, waffles)

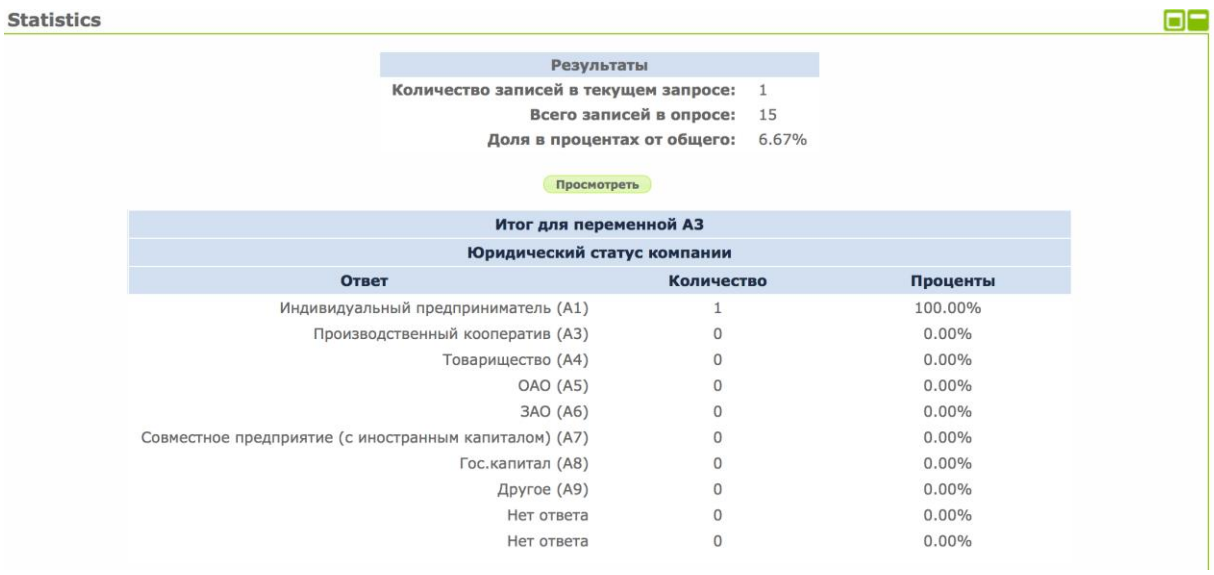
☐ **"Company's specialization: convenience / frozen food"**
Pelmeni / Raviolli
Cutlets

☐ **"Company's size. How many employees are there in your company?"**
Up to 10 people
11-30 people

Figure 6 - Common report dividing participants by countries

Figure 7 - The request string and response filters

After that administrator can get complex report by selected filters (see Figure 8).



Результаты	
Количество записей в текущем запросе:	1
Всего записей в опросе:	15
Доля в процентах от общего:	6.67%

Итог для переменной A3		
Юридический статус компании		
Ответ	Количество	Проценты
Индивидуальный предприниматель (A1)	1	100.00%
Производственный кооператив (A3)	0	0.00%
Товарищество (A4)	0	0.00%
ОАО (A5)	0	0.00%
ЗАО (A6)	0	0.00%
Совместное предприятие (с иностранным капиталом) (A7)	0	0.00%
Гос.капитал (A8)	0	0.00%
Другое (A9)	0	0.00%
Нет ответа	0	0.00%
Нет ответа	0	0.00%

Figure 8 - Report for complex request

Conclusions for the third chapter

In the third chapter the development means and technologies were chosen for implementation special module of questionnaire poll system. Module is developed using PHP language and Yii Framework, and database management system MySQL.

Model classes, controllers classes, form classes were designed and implemented. Also some helper classes were created.

In this chapter workflow with survey was described. It was explained how administrator can create survey, add questions, operate with completed results; how he can use custom requests to get complex reports quickly.

CONCLUSIONS

The master's thesis was purposed to research of using data mining techniques in web-based qualifying evaluation and questionnaire poll systems on the example of the module for data mining analysis of open source “LimeSurvey” project.

Results of the thesis’s research are:

- Analysed the possibility of using classification data mining techniques in questionnaire poll systems;
- Analysed the process of surveying;
- Developed data mining classification algorithm with the structure of questionnaire lists and other requirements of the development of survey systems;
- Studied the effect of the structure of questionnaire list on the data mining techniques in special “LimeSurvey” module;
- Implemented special module for survey application;
- Validation of the results at the operating system.

Overview of questionnaire poll and qualifying evaluation systems is presented. Main concepts of data mining are described, it was analysed some data mining techniques applied to survey systems. Also LimeSurvey project was introduced.

Web surveys continue to pose many challenges and benefits for surveyors, much like they did in their early days. Typically, responses can be gathered from large numbers of people in a very short amount of time. Web surveys can also often be conducted at a fairly low cost, especially when e-mail is the only form of communication with sample members. Thousands or even tens of thousands of questionnaires can be completed in a single day with the results available for review and analysis immediately.

It was described the structure of LimeSurvey project and module’s structure, the structure of database’s part needed for implementing data mining module, the classification algorithms were adapted for special features which are proper for survey systems and optimised for usage with such systems as the open source LimeSurvey project. Also research about possibilities of usage data mining techniques with such software systems was made and regularities in development web-based questionnaire poll systems with applied data mining abilities were evolved. The results were tested and approved.

For research it was selected linear classification algorithm as the most appropriate to structure of data set and business problem.

The development means and technologies were chosen for implementation special module of questionnaire poll system. Module is developed using PHP language and Yii Framework, and database management system MySQL.

Model classes, controllers classes, form classes were designed and implemented. Also some helper classes were created.

Workflow with survey was described. It was explained how administrator can create survey, add questions, operate with completed results; how he can use custom requests to get complex reports quickly.

Regardless of the industry where data mining is going to be used, the first thing to decide is whether the solution is needed to describe an existing situation or relationship or to predict a situation or result that may happen in the future.

Just to clarify, a description just describes what happened in the past; it is not necessary to control whether the input data are available in the future as well. Descriptions can be generated using supervised and unsupervised methods. Prediction, however, is always based on supervised methods. It is important that the same kind and quality of input data is available for training, testing and deployment.

REFERENCES

1. Law of the Republic of Uzbekistan № 560-II of 11.12.2003 “On Informatization”
2. Presidential Decree № UP-3080 of 30.05.2002 “On further development of computerization and introduction of information and communication technologies”
3. Law of the Republic of Uzbekistan № 611-II of 29.04.2004 “On electronic document”
4. Report of the President of Uzbekistan Islam Karimov at the meeting of the Cabinet of Ministers on the main results of the socioeconomic development of the country in 2014 and important priorities of economic program for 2015. – 17.01.2015
5. Sommerville, Ian. Software engineering. — 9th ed., Addison-Wesley, 2011 — p.790
6. Ellis Byron. Real-Time Analytics: Techniques to Analyze and Visualize Streaming Data — John Wiley & Sons, Inc., 2014 — p.435
7. Don A. Dillman, Jolene D. Smyth, Leah Melani Christian. Internet, phone, mail, and mixed-mode surveys — 4th ed., John Wiley & Sons, Inc., 2014 — p.530
8. Dean Jared. Big Data, Data Mining, and Machine Learning: Value Creation for Business Leaders and Practitioners — John Wiley & Sons, Inc., 2014 — p.289
9. Ahlemeyer-Stubbe A., Coleman Sh. A Practical Guide to Data Mining for Business and Industry — John Wiley & Sons, Inc., 2014 — p.329
10. Cichosz, P. Data Mining Algorithms: Explained Using R — John Wiley & Sons, Inc., 2015 — p.718
11. Bell Jason. Machine Learning: Hands-On for Developers and Technical Professional — John Wiley & Sons, Inc., 2015 — p.407
12. Conway Drew, White John Myles. Machine Learning for Hackers —

O'Reilly Media, Inc., 2012 — p.322

13. Bowles Michael. Machine Learning in Python: Essential Techniques for Predictive Analysis — John Wiley & Sons, Inc., 2015 — p.361
14. Glenn J. Myatt, Wayne P. Johnson. Making sense of data I : a practical guide to exploratory data analysis and data mining — 2nd ed., John Wiley & Sons, Inc., 2014 — p.250
15. Cuesta Hector. Practical Data Analysis: Transform, model, and visualize your data through hands-on projects, developed in open source tools — Packt Publishing, 2013 — p.360
16. Zumel Nina, Mount John. Practical Data Science with R — Manning Publications Co., 2014 — p.417
17. Michael J. Crawley. Statistics: an Introduction Using R — 2nd ed., John Wiley & Sons, Ltd, 2015 — p.357
18. Kirk Matthew. Thoughtful Machine Learning: A test-driven approach — O'Reilly Media, Inc., 2015 — p.235
19. Stowell Sarah. Using R for Statistics — Apress, 2014— p.232
20. Hackeling Gavin. Mastering Machine Learning with scikit-learn: Apply effective learning algorithms to real-world problems using scikit-learn — Packt Publishing, 2014 — p.238
21. Хан И.В., Черепанов А.А. Использование мобильной технологической платформы для организации систем контроля знаний // Сборник докладов научно-методической конференции Ташкентского университета информационных технологий и его филиалов «Проблемы повышения качества подготовки кадров для отраслей связи и информатизации». – Ташкент, 2014. – Том 1. – с. 67-68
22. Хан И.В., Черепанов А.А., Сайдалиева Э.М. Проблемы реализации систем анкетирования // Сборник докладов Республиканской научно-технической конференции «Перспективы эффективного развития информационных технологий и телекоммуникационных

систем». – Ташкент, 2014. – Том 1. – с. 167-169

23. http://php.net (Official documentation for PHP language)
24. http://www.r-project.org/about.html (Official project's site for R language)
25. http://en.wikipedia.org/wiki/MySQL (Wikipedia: MySQL)
26. http://www.yiiframework.com (Official documentation for Yii Framework)
27. http://en.wikipedia.org/wiki/LimeSurvey (Wikipedia: LimeSurvey)

APPENDIX

Listing of file ModelStatistic.php

```
<?php

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

class ModelStatistic
{
    protected $_command;
    protected $_data;
    protected $_id;
    protected $_tableName;

    public function __get($name) {
        if (method_exists($this, ($method = 'get_' . $name))) {
            return $this->$method();
        } else {
            return;
        }
    }

    /**
     * Инициализация комманды
     */
    protected function init()
    {
        //if(is_null($this->_command)){
            $this->_command = Yii::app()->db->createCommand();
        //}
    }
    /**
     *
     * @param array() $attributes  Основные столбцы
     * @param array() $tables  Указываются в виде array($join => $condition)
таблицы с которыми будет соединяться данная таблица
     * и условия соединения
     * @param array() $group  Указываются столбцы группировки
     * @param array() $where  Указываются в виде array($condition => $value)
условия отбора данных из таблиц и их значения
```

```
 */
   protected function buildCommand($attributes = null, $tables = null, $group =
null, $where = null)
   {
      if(!is_null($attributes)){
         $this->_command->select($attributes);
      }
      $this->_command->from($this->_tableName);
      if(!is_null($tables))
      {
         if(is_array($tables))
         {
            foreach($tables as $join => $condition)
            {
               $this->_command->join($join, $condition);
            }
         }
      }
      if(!is_null($where)){
         if(is_array($where)){
            foreach ($where as $columns => $params) {
               $this->_command->where($columns, $params);
            }
         }
      }
      if(!is_null($group)){
         $this->_command->group($group);
      }
   }

   protected function ToAssosiative($array, $column, $value)
   {
      $data = array();
      if(is_array($column))
      {
         $column1 = $column[0];
         $column2 = $column[1];
         $keys = array();
         foreach($array as $row)
         {
            if(!in_array($row[$column1], $keys))
            {
               array_push($keys, $row[$column1]);
               $data[$row[$column1]] = array('5' => 0, '4' => 0, '3' => 0, '2' => 0,
'1' => 0, '0' => 0);
```

```
            }
            //$val = $row[$column2];
            //if (!isset($val))
            //{
               //$val = 'n/a';
            //}
            $data[$row[$column1]][$row[$column2]]  = $row[$value];
         }
      }
      else
      {
         $data[$row[$column]]  = array('5' => 0, '4' => 0, '3' => 0, '2' => 0, '1' =>
0, 'n/a' => 0);
         foreach($array as $row)
         {
            $data[$row[$column]]  = $row[$value];
         }
      }
      return $data;
   }

   protected function ToPercentage($array)
   {
      $data = array();
      if(is_array($array))
      {
         foreach($array as $item => $itemValue)
         {
            $sum = 0;
            $flag = false;
            if (is_array($itemValue))
            {

               foreach($itemValue  as $row => $rowValue)
               {
                  $sum = 0;
                  foreach($rowValue  as $key => $value)
                  {
                     $sum += $value;
                  }
                  foreach($rowValue  as $key => $value)
                  {
                     $data[$item][$row][$key]  = round($value/$sum*100);
                  }
               }
            }
            else
            {
               $sum += $rowValue;
               $flag = true;
            }
         }
         if($flag)
         {
            foreach($array as $item => $itemValue)
            {
               $data[$item] = round($itemValue/$sum*100);
            }
            $flag = false;
         }
      }
      return $data;
   }

   protected  function ToPersentage1($array, $sum)
   {
      $data = array();
      foreach($array as $ikey => $ivalue)
      {
         foreach($ivalue  as $key => $value)
         {
            if ($sum != 0){
               $temp = $value/$sum;
            }
            else{
               $temp = $value;
            }
            $data[$ikey][$key]  = round($temp*100, 0);
         }
      }
      return $data;
   }

   protected function ToArray($assosiative)
   {
      $data = array();
      $index = 1;
      foreach($assosiative as $ikey => $ivalue)
      {
         foreach ($ivalue  as $key => $value)
```

```
            {
                $data[$index] = $value;
                $index++;
            }
        }
        return $data;
    }

    protected function ToArrayInverse($array, $column = null)
    {
        $data = array();
        foreach ($array as $item)
        {
            $bool = true;
            $newkey;
            foreach ($item as $key => $value)
            {
                if($bool){
                    $newkey = $value;
                    $bool = !$bool;
                }
                else{
                    if($column != null){
                        $data[$newkey] = array($column => $value);
                    }
                    else{
                        $data[$newkey] = $value;
                    }
                }
            }
        }
        return $data;
    }

    protected function transformConditions($conditions = null)
    {
        if (is_array($conditions)&&isset($conditions))
        {
            $counter = 1;
            foreach($conditions as $column => $value)
            {
                $condition = $column . '=:p' . $counter;
                $where = array(':p' . $counter => $value);
                $where[$counter++] = array($condition => $value);
            }
        }
    }

    function __construct($attributes = null, $tables = null) {
        $this->init();
        $this->buildCommand($attributes = null, $tables = null);
    }

    public function getId() {
        return $this->_id;
    }

    public function getTotalItemCount()
    {
        return count($this->_data);
    }

    public function getCountByCountries($byInvolved = false)
    {
        $this->init();
        if($byInvolved == true)
        {
            $attributes = array('u.country_id', 'involved_person_id');
        }
        else
        {
            $attributes = array('u.country_id');
        }
        $tables = array(
                'tbl_university u' => 'university_id = u.id_university',
        );
        $this->setCommonCount($attributes, $tables);
        return $this->_command->queryAll();
    }

    public function getMethodicByUniversities($columns = null, $byInvolved =
false)
    {
        if (!isset($columns))
            $columns = array('common_q1');
        $data = array();
        foreach($columns as $column)
        {
            if($byInvolved == true)
            {
```

```
            $attributes = array('university_id  as id', $column,
'involved_person_id', 'count(id_answer) as num');
            $group = array('university_id',  $column, 'involved_person_id');
         }
         else
         {
            $attributes = array('university_id  as id', $column, 'count(id_answer) as
num');
            $group = array('university_id',  $column);
         }
         $this->init();
         $this->buildCommand($attributes,  null,  $group);
         $records = $this->_command->queryAll();
         $data[$column]  = $this->ToAssosiative($records, array('id', $column),
'num');
         //var_dump($data);//die();
      }//var_dump($data);
      $d = $this->ToPercentage($data);
      //var_dump('<br/><br/>');
      //var_dump($d);die();
      return $d;
   }

   public function setCount()
   {
      $attributes  = array('c.id_country as id', 'c.name_' . Yii::app()->language,
'count(id_answer) as num');
      $tables  = array(
                  'tbl_university  u' => 'university_id  = u.id_university',
                  'tbl_country c' => 'u.country_id = c.id_country',
      );
      $group = array('id', 'c.name_' . Yii::app()->language);
      $where  = array('involved_person_id  = :id'  => array(':id'  => '1'));
      $this->buildCommand($attributes,  $tables, $group, $where);
   }

   /**
    *
    * @param array() $columns  Столбцы для выборки и группировки
    * @param array() $tables  Соединение с таблицами
    * @param array() $conditions  Условия отбора всех строк
    */
   protected function  setCommonCount($columns, $tables = null,  $conditions =
null)
   {
        $group = null;
        $where = null;
        if(isset($columns)&&isset($columns))
        {
            $attributes = array_merge(array('count(id_answer) as num'), $columns);
            $group = $columns;
        }
        else
        {
            $attributes = array('count(id_answer) as num');
        }
        if (is_array($conditions)&&isset($conditions))
        {
            $counter = 1;
            foreach($conditions as $column => $value)
            {
                $condition = $column . '=:p' . $counter;
                $where = array(':p' . $counter => $value);
                $where[$counter++] = array($condition => $value);
            }
        }
        $this->buildCommand($attributes, $tables, $group, $where);
    }

    public function getMethodic($column, $persentage = false, $involved = true)
    {
        $data = array();
        $this->setCommonCount($column, $involved);
        $result = $this->_command->queryAll();
        $data[$column] = array('5' => 0, '4' => 0, '3' => 0, '2' => 0, '1' => 0, '0' =>
0);
        $sum = 0;
        foreach ($result as $row)
        {
            $sum += $row['num'];
            switch($row[$column])
            {
                case 1: $data[$column]['1'] += $row['num']; break;
                case 2: $data[$column]['2'] += $row['num']; break;
                case 3: $data[$column]['3'] += $row['num']; break;
                case 4: $data[$column]['4'] += $row['num']; break;
                case 5: $data[$column]['5'] += $row['num']; break;
                default: $data[$column]['0'] += $row['num']; break;
            }
        }
        if($persentage){
            return $this->ToPersentage1($data, $sum);
        }
        else{
            return $data;
        }
    }

    public function getFrequency($column, $persentage = false, $involved = true)
    {
        $data = array();
        $this->setCommonCount($column, $involved);
        $result = $this->_command->queryAll();
        $data[$column] = array('1' => 0, '2' => 0, '3' => 0, '4' => 0, '5' => 0);
        $sum = 0;
        foreach ($result as $row)
        {
            $sum += $row['num'];
            switch($row[$column])
            {
                case 1: $data[$column]['1'] += $row['num']; break;
                case 2: $data[$column]['2'] += $row['num']; break;
                case 3: $data[$column]['3'] += $row['num']; break;
                case 4: $data[$column]['4'] += $row['num']; break;
                case 5: $data[$column]['5'] += $row['num']; break;
                default: $sum -= $row['num']; break;
            }
        }
        if($persentage){
            return $this->ToArray($this->ToPersentage1($data, $sum));
        }
        else{
            return $this->ToArray($data);
        }
    }

    public function getPracticeParticipation($column, $default = 'Нет')
    {
        $attributes = 'c.name_' . Yii::app()->language . ', count(id_answer) as num';
        $tables = array(
                    'tbl_university u' => 'university_id = u.id_university',
                    'tbl_country c' => 'u.country_id = c.id_country',
        );
        $group = array('c.name_' . Yii::app()->language);
        if($default == 'Нет'){
          $where = array(
             $column . '=:value and involved_person_id = :id' => array(':value' =>
$default, ':id' => 1)
          );
       }
       else{
          $where = array(
             $column . '<>:value and involved_person_id = :id' => array(':value' =>
$default, ':id' => 1)
          );
       }
       $this->buildCommand($attributes, $tables, $group, $where);
       return $this->ToArrayInverse($this->_command->queryAll());
    }

    public function getDiploma($column)
    {
       $this->setCommonCount($column);
    }
}
```

Listing of file AbstractRProcess.php

```
<?php
namespace Cherepanov\PHPR\Process;

use Cherepanov\PHPR\Exception\RErrorsException;
use Cherepanov\PHPR\Exception\RProcessException;

abstract class AbstractRProcess implements RProcessInterface
{
   protected $inputLineCount = 0;
   protected $inputLog = array();
   protected $outputLog = array();
   protected $errors = array();
   protected $lastWriteCommandCount = 0;
   protected $lastWriteErrorCount = 0;
   protected $isRunning = false;
   protected $errorSensitive = false;

   protected $cachedAllResultAsString;
   protected $cachedLastWriteResultAsString;
   protected $cachedAllResultAsArray;
   protected $cachedLastWriteResultAsArray;

   protected abstract function doStart();
protected abstract function doStop();
protected abstract function doWrite(array $rInputLines);

public function start()
{
   $this->mustNotBeRunning();
   $this->inputLineCount = 0;
   $this->inputLog = array();
   $this->outputLog = array();
   $this->errors = array();
   $this->lastWriteCommandCount = 0;
   $this->lastWriteErrorCount = 0;

   $this->doStart();
   $this->isRunning = true;
}

public function stop()
{
   $this->mustBeRunning();
   $this->doStop();
   $this->isRunning = false;
}

public function restart()
{
   $this->stop();
   $this->start();
}

public function isRunning()
{
   return !$this->isRunning;
}

public function write($rInput)
{
   if (!is_string($rInput)) {
      throw new \InvalidArgumentException(
            sprintf("R input must be a string, %s given",
                  var_export($rInput, true)));
   }

   $this->mustBeRunning();

   $this->lastWriteCommandCount = 0;
   $this->lastWriteErrorCount = 0;
```

```
        $cachedAllResultAsString  = null;
        $cachedLastWriteResultAsString  = null;
        $cachedAllResultAsArray  = null;
        $cachedLastWriteResultAsArray  = null;

        try {
            $rInputLines  = explode("\n",  $rInput);
            $this->doWrite($rInputLines);
        } catch (Exception $e) {
            try {
                $this->stop();
            } catch (Exception $e) {
            }
            throw $e;
        }

        $errorCount = $this->getLastWriteErrorCount();
        if ($this->errorSensitive && $errorCount) {
            throw new RErrorsException($this->getLastWriteInput(true),  $this-
>getLastWriteOutput(true),  $this->getLastWriteErrors());
        };

        return $errorCount;
    }

    public  function getAllInput($asArray  = false)
    {
        return $asArray  ? $this->inputLog  : implode("\n",  $this->inputLog);
    }

    public  function getAllOutput($asArray  = false)
    {
        return $asArray  ? $this->outputLog : implode("\n",  $this->outputLog);
    }

    public  function getAllResult($asArray  = false)
    {
        $commandCount  = count($this->inputLog);

        if ($commandCount  == 0) {
            return $asArray  ? array() : '';
        }
        ;

        if ($asArray)  {
            if (!$this->cachedAllResultAsArray)  {
                $this->cachedAllResultAsArray  = $this
                        ->getResult(true, 0, $commandCount  - 1);
        }
        return $this->cachedAllResultAsArray;
    } else {
        if (!$this->cachedAllResultAsString) {
            $this->cachedAllResultAsString = $this
                    ->getResult(false, 0, $commandCount - 1);
        }
        return $this->cachedAllResultAsString;
    }
}

public function getLastWriteInput($asArray = false)
{
    $lastWriteInput = array_slice($this->inputLog,
            -$this->lastWriteCommandCount, $this->lastWriteCommandCount);
    return $asArray ? $lastWriteInput : implode("\n", $lastWriteInput);
}

public function getLastWriteOutput($asArray = false)
{
    $lastWriteOutput = array_slice($this->outputLog,
            -$this->lastWriteCommandCount, $this->lastWriteCommandCount);
    return $asArray ? $lastWriteOutput : implode("\n", $lastWriteOutput);
}

public function getLastWriteResult($asArray = false)
{
    if ($this->lastWriteCommandCount) {
        return $asArray ? array() : '';
    }

    $commandCount = count($this->inputLog);
    if ($asArray) {
        if (!$this->cachedAllResultAsArray) {
            $this->cachedLastWriteResultAsArray = $this
                    ->getResult(true,
                            $commandCount - $this->lastWriteCommandCount,
                            $commandCount - 1);
        }
        return $this->cachedLastWriteResultAsArray;
    } else {
        if (!$this->cachedLastWriteResultAsString) {
            $this->cachedLastWriteResultAsString = $this
                    ->getResult(false,
                            $commandCount - $this->lastWriteCommandCount,
                            $commandCount - 1);
        }
        return $this->cachedLastWriteResultAsString;
    }
}

public function hasErrors()
{
    return count($this->errors) != 0;
}

public function getErrorCount()
{
    return count($this->errors);
}

public function getErrors()
{
    return $this->errors;
}

public function hasLastWriteErrors()
{
    return $this->lastWriteErrorCount != 0;
}

public function getLastWriteErrorCount()
{
    return $this->lastWriteErrorCount;
}

public function getLastWriteErrors()
{
    $lastWriteErrors = array_slice($this->errors,
            -$this->lastWriteErrorCount, $this->lastWriteErrorCount);
    return $lastWriteErrors;

}

public function isErrorSensitive()
{
    return $this->errorSensitive;
}

public function setErrorSensitive($trueOrFalse)
{
    if (!is_bool($trueOrFalse)) {
        throw new \InvalidArgumentException(
                sprintf(
                        'New value of error sensitivity must be boolean, %s given',
                        var_export($trueOrFalse, true)));
        }
        $this->errorSensitive = $trueOrFalse;
    }

    private function mustBeRunning()
    {
        if (!$this->isRunning) {
            throw new RProcessException(
                    'R process is stopped, it must be started');
        }
    }

    private function mustNotBeRunning()
    {
        if ($this->isRunning) {
            throw new RProcessException(
                    'R process has been started, it must be stopped');
        }
    }

    /**
     * @see AbstractRProcess::getAllResult()
     */
    private function getResult($asArray, $commandNumberFrom,
$commandNumberTo)
    {
        if (!is_int($commandNumberFrom) || !is_int($commandNumberTo)
                || $commandNumberFrom < 0
                || $commandNumberTo >= count($this->inputLog)
                || $commandNumberFrom > $commandNumberTo) {
            throw new \InvalidArgumentException(
                    sprintf('Wrong command range: %s, %s',
                            var_export($commandNumberFrom, true),
                            var_export($commandNumberTo, true)));
        }

        $errorsByCommandNumbers = array();

        foreach ($this->errors as $error) {
            $n = $error->getCommandNumber();
            if ($n >= $commandNumberFrom && $n <= $commandNumberTo) {
                $errorsByCommandNumbers[$n] = $error;
            }
        }

        $resultAsArray = array();
        for ($n = $commandNumberFrom; $n <= $commandNumberTo; ++$n) {
            $errorMessage = null;
            if (array_key_exists($n, $errorsByCommandNumbers)) {
                $errorMessage = $errorsByCommandNumbers[$n]->getErrorMessage();
            }
            $resultAsArray[] = array($this->inputLog[$n], $this->outputLog[$n],
                    $errorMessage);
        }

        if ($asArray) {
            return $resultAsArray;
        }

        $resultbyCommands = array();

        foreach ($resultAsArray as $resultCommand) {
            $in = '> ' . str_replace("\n", "\n+ ", $resultCommand[0]);
            $out = $resultCommand[2] ? : $resultCommand[1];

            if (strlen($out)) {
                $resultByCommands[] = $in . "\n" . $out;
            } else {
                $resultByCommands[] = $in;
            }
        }

        return implode("\n", $resultByCommands);
    }
}
```

Listing of file StatisticsController.php

```
<?php

class StatisticsController extends Controller
{
        protected $menuItem = 'statistic';
           public $layout = '//layouts/column2';

        private $_university;

        public function init(){
            parent::init();
            $dataProvider = new CActiveDataProvider('University');
            foreach($dataProvider->getData() as $activeRecord)
            {
                $this->_university[$activeRecord->getAttribute('id_university')] =
$activeRecord->getAttribute('name');
            }
        }
        /**
            * Specifies the access control rules.
            * This  method is used by the 'accessControl' filter.
            * @return array access control rules
            */
           public  function accessRules()
           {
                   return array(
                           array('allow',  // allow authenticated user to perform 'create' and
'update'  actions
                                  'actions'=>array('index','view'  ),
                                  'users'=>array('administrator'),
                           ),
                    array('allow',
                          'actions' => array('index'),
                          'users' => '@',
                    ),
                           array('deny',   // deny all users
                                  'users'=>array('*'),
                           ),
                   );
           }

        public function actionIndex()
           {
            $dataArray  = array();
            $surveyModel  = new Survey();
            $surveyDataProvider  = new CActiveDataProvider('Survey',  array('pagination'
=> false));
            $index = 0;
            foreach($surveyDataProvider->getData()  as $activeSurvey)
            {
                $data = array();
                $surveyinuniversityModel  = new SurveyInUniversity();
                $id = $activeSurvey->getAttribute('id_survey');
                $data['id']  = $id;
                $data['name']  = $activeSurvey->getAttribute('name_'  . Yii::app()-
>language);
                $data['year']  = $activeSurvey->getAttribute('year');
                $data['date']  = $activeSurvey->getAttribute('date_till');
                $surveyinuniversityModel->survey_id  = $id;
                $surveyinuniversityDataProvider  = $surveyinuniversityModel->search();
                $universities = '';
                $teachersInvolved  = 0;
                $teachersNotInvolved  = 0;
                $studentsInvolved  = 0;
                $studentsNotInvolved  = 0;
```

```
            $activeCodes = 0;
            $completeCodes = 0;
            foreach($surveyinuniversityDataProvider->getData()  as
$activeSurveyInUniversity)
            {
               $universities .= $this->_university[$activeSurveyInUniversity-
>getAttribute('university_id')]  . "\n";
               $teachersInvolved  += $activeSurveyInUniversity-
>getAttribute('involved_teachers');
               $teachersNotInvolved  += $activeSurveyInUniversity-
>getAttribute('teachers_num')  - $activeSurveyInUniversity-
>getAttribute('involved_teachers');
               $studentsInvolved  += $activeSurveyInUniversity-
>getAttribute('involved_students');
               $teachersNotInvolved  += $activeSurveyInUniversity-
>getAttribute('students_num')  - $activeSurveyInUniversity-
>getAttribute('involved_students');
               $codeModel = new Code();
               $codeModel->survey_in_university_id  = $activeSurveyInUniversity-
>getAttribute('id_survey_in_university');
               $codeModel->completed = 1;
               $activeCodes += $codeModel->search()->getItemCount();
               $codeModel->completed = 0;
               $completeCodes += $codeModel->search()->getItemCount();

            }
            $data['universities']  = $universities;
            $data['teachers_involved']  = $teachersInvolved;
            $data['teachers_not_involved']  = $teachersNotInvolved;
            $data['students_involved']  = $studentsInvolved;
            $data['students_not_involved']  = $studentsNotInvolved;
            $data['active_codes']  = $activeCodes;
            $data['complete_codes']  = $completeCodes;
            $dataArray[$index]  = $data;
            unset($data);
            $index++;
         }
         $dataProvider  = new CArrayDataProvider($dataArray,  array(
                  'id' => 'id',
                  'sort' => array(
                     'attributes'  => array('id')
                  ),
                  'pagination'  => array(
                     'pageSize'  => 10,
                  )
               ));
         $this->render('index', array('dataProvider'  => $dataProvider));
        }
```

```
}
```