Thesis

Master in Informatics Engineering

Artificial Intelligence

Automated Data Mining and Automatic Visualization

JOÃO PEDRO SILVA LEAL

Thesis Advisor

Bruno Emanuel Machado Antunes, PhD

Fernando Penousal Machado, PhD

Department of Informatics Engineering

Faculty of Sciences and Technology

University of Coimbra

2014/2015

Abstract

In the age of new technologies, where information is all around us and large amounts of data is being created at an impressive rate, there is a big amount of knowledge that is potentially important, but is yet to be discovered. There are two processes closely related to discovering new information: Information Analysis and Information Visualization.

Information Analysis, and in particular Data Mining, has the goal of discovering useful information such as previously unknown relationships in data. However, data mining is a difficult and time consuming task, that requires extensive know-how of the many techniques and procedures, such as creation of appropriate models and parameter tuning.

Information Visualization, on the other hand, does not discover useful information on its own. It provides, however, an huge amount of information through graphical representations. Yet, the creation of these visualizations is a difficult task.

This work was developed at Wizdee, a company specialized in state of the art solutions for knowledge management. In an effort to extend and improve the platform, two large systems were developed: Automated Data Mining and Automatic Visualization, where the focus is the enabling of those technologies to any business user, by removing the need for technical knowledge and expertise. The developed systems can answer questions such as *"Given this dataset what is the most appropriate visualization?"* and *"What are the causes for lost opportunities."* without needing input from the user.

The approach for Automatic Visualization is based on Case Based Reasoning. A detailed explanation of a novel procedure, named Case Mapping, for case retrieval, that is accurate and efficient, is also presented. Furthermore, details about data processing and how that can be automated into creating various types of charts is described.

As for the Automated Data Mining, this work describes an approach to design a scalable and flexible machine learning architecture, that can be used to automate various data mining tasks. For each task, a methodology was developed that automates each step of the processes.

Additionally, for each component, tests were performed using a comparative evaluation when possible, where our analysis suggest that the approaches used are fast and effective. Finally, the final implementation was integrated in the platform, showing the viability of the approaches used in a practical scenario.

Keywords: automatic visualization, automated data mining, data mining, machine learning, case based reasoning

Resumo

Na era das novas tecnologias, grandes quantidades de informação estão a ser criadas a um ritmo impressionante, no entanto, existe uma grande quantidade de conhecimento potencialmente importante que ainda está para ser descoberto. Existem dois processos importantes que podem ajudar na descoberta de novas informações: Análise de Informação e Visualização de Informação.

Análise de Informação, em particular *Data Mining*, tem como objetivo descobrir informação útil, como a descoberta de relações entre os dados que eram anteriormente desconhecidas. No entanto, *Data Mining* é uma tarefa difícil, que requer tempo e um conhecimento extensivo das diferentes técnicas e procedimentos envolvidos, tais como a criação de modelos e otimização de parâmetros.

Por outro lado, Visualização de Informação não produz, por si só, a descoberta de novas e úteis informações. No entanto, permite expor uma quantidade enorme de informação através de representações gráficas. Contudo, a criação dessas visualizações é uma tarefa complexa.

O trabalho descrito neste documento foi desenvolvido na Wizdee, uma empresa especializada em soluções que representam o estado da arte para a gestão de conhecimento. De forma a melhorar a plataforma, foram desenvolvidos dois sistemas principais: *Data Mining* Automatizado e Visualização Automática. O objetivo principal destes sistemas é permitir que todos os utilizadores de negócio sejam capazes de utilizar estas tecnologias, sem a necessidade de ter conhecimento técnico. Ou seja, o sistema desenvolvido permite responder a questões como "A partir deste conjunto de dados qual é a visualização mais adequada?" ou "Quais são as razões para perda de oportunidades?", sem necessidade de intervenção do utilizador.

A abordagem seguida para a Visualização Automática é de Raciocínio baseado em Casos. Neste documento é apresentada uma descrição detalhada de um novo procedimento desenvolvido, denominado por Mapeamento de Casos. São também descritos detalhes sobre o processamento de dados e sua automatização para diferentes tipos de gráficos.

Sobre o sistema de *Data Mining* Automatizado, é descrito em detalhe o desenvolvimento de uma arquitetura escalável e flexível, que é usada na automatização das diferentes tarefas de *Data Mining*. Para cada tarefa, foi desenvolvida uma metodologia que automatiza cada fase do processo.

É de salientar, que para cada componente desenvolvida, foram realizados diversos testes utilizando uma avaliação comparativa, sempre que possível. As análises realizadas aos resultados sugerem que as abordagens usadas são rápidas e eficazes. Por fim, é de salientar, que todos os sistemas desenvolvidos estão integrados e a ser usados na plataforma Wizdee, o que demonstra a viabilidade das abordagens desenvolvidas num cenário prático.

Palavras-Chave: visualizações automáticas, extração de conhecimento, raciocínio baseado em casos, aprendizagem automática

Acknowledgements

I find myself in need of thanking, first of all, my family, for their continued support. In particular, I would like to thank my parents for their relentless patience, encouragement and advice.

Moreover, I would like to express my gratitude to Dr. Bruno Antunes and Dr. Paulo Gomes for their tireless efforts to give me timely feedback and guide me in the right direction, especially during stressful times.

In a similar manner, I would like to thank Dr. Penousal Machado for the assistance and suggestions throughout the work.

In addition, I would like to thank Sara for all her love and encouragement. Finally, I am thankful to all my friends, whose support was much appreciated.

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence

CBR Case Based Reasoning

ML Machine Learning

DM Data Mining

TP True Positive

TN True Negative

FP False Positive

FN False Negative

SVM Support Vector Machines

SVR Support Vector Regression

ARIMA Autoregressive integrated moving average

DBSCAN Density-based spatial clustering of applications with noise

ID3 Iterative Dichotomiser 3

Chapter 1

Introduction

In the age of new technologies, where information is all around us and large amounts of data is being created at an impressive rate (Gantz and Reinsel, 2012), there is a big amount of knowledge that is potentially important, but is yet to be discovered. There are two processes closely related to discovering new information: Information Analysis and Information Visualization.

Information Analysis, and in particular Data Mining, has the goal of discovering useful information such as previously unknown relationships in data (Witten and Frank, 2005). However, this valuable technology is yet to be adopted by business users at large. This can be attributed, in part, to the nature of the data mining tasks. Data mining is a difficult and time consuming task, that requires extensive know how of the many techniques and procedures, such as creation of appropriate models and parameter tuning.

Information Visualization, on the other hand, does not discover useful information on its own. It provides, however, an huge amount of information through graphical representations (Ware, 2013). Yet, the creation of these visualizations is a difficult task. The data is not always easy to understand, due to its complexity and high dimensionality.

This work was developed at Wizdee[1], a company specialized in state of the art solutions for knowledge management. Wizdee platform is a product that changes the paradigm of the interaction enterprises have with information, by allowing users to access their data by typing a query, using a natural language search engine to make it easy and accessible for everyone. Business intelligence becomes simple, by letting the users explore their data in a intuitive way.

As expected, developing a system that allows business users to have this control over their data, is not a trivial task. In an effort to extend and improve the platform, two large systems were developed: Automated Data Mining and Automatic Visualization, where the focus is the enabling of those technologies to any business user, by removing the need for technical knowledge and expertise. This was accomplished using techniques that allow a system to make decisions on its own, without requiring user input.

It is important to note that, in the Automated Data Mining approach, the goal is not to provide automated methodologies that can produce better results than those created by experts. Instead, we wanted to achieve reasonable results with minimum effort, empowering non-expert users. Additionally, in the Automatic Visualization, the approach only concerns with back-end processing, thus, it is out of the scope of this work high-level systems, such as the user interface that presents the visualization to the end user.

Regarding research on these topics, in the field of Automated Data Mining we can find some relevant work. But it is common to find approaches bound to specific architectures and workflows, that do not generalize well to other systems or platforms. A good example

[1]http://wizdee.com/home/

of such an approach is found in (Campos et al., 2009). Another form of related work is the automation of procedures for specific algorithms, such as parameter optimization on Support Vector Machines (Gomes et al., 2012).

In the Automatic Visualization field, there has been techniques developed that attempt to automatize the process of creating visualizations, however, there are certain limitations, that we were able to surpass. An example of such limitation is a system able to identify the correct chart type, with good accuracy, but is unable to choose its axes and series (Jill Freyne, 2009). Other form of related work in this field, includes research that tackles challenges outside of the scope of this work, such as choosing colours for a chart according to the table data (Lin et al., 2013).

Finally, our research is able to surpass some of the limitations found in the existing work in both Automatic Visualization and Automated Data Mining, with the added advantage of being integrated into a commercial product and deployed into production.

1.1 Goals

As mentioned, the goal of this work is to make the processes behind Data Mining and Information Visualizations as effortless as possible. Therefore, the focus of this work, for Automatic Visualization, is answering the question of *"Given this dataset what is the most appropriate visualization?"* and, for Automated Data Mining, we want to be able to answer questions such as *"What's my sales forecast for next year?"* or *"What are the causes for lost opportunities."*. These goals can be decomposed into two main requirements:

- **Automatic Visualization:** Deciding automatically the most appropriate visualization based on the search results provided by the system. Not only the chart type must be returned but also its axes and series. Additionally, it must be capable of handling the processing for each one of chart types, including making decisions regarding specific parameters, such as the number of bins in an Histogram of the number of slices in a Pie Chart.

- **Automated Data Mining:** Use raw data to provide the ability to discover patterns, predictions or simply new knowledge. This includes the creation of automated processes for each data mining task. Additionally, there is also the creation of a central environment, that handles any data mining task in the Wizdee platform.

1.2 Approach

The approach for Automatic Visualization is based on Case Based Reasoning, as explained by (Kolodner, 1992) and (Riesbeck and Schank, 1989). An approach inspired by human reasoning and memory organization. Rather than following a set of rules, this approach reapplies previous a successful solutions in the new problem. This approach is particularly useful due to its adaptability and easily augmentation.

As for the Automated Data Mining system, the approach is based on concepts from machine learning, where important patterns and trends are extracted as an attempt to understand what the data says (Mohri et al., 2012). Those concepts are then extended so that they can be applied without users input or know-how, by finding mechanisms that can automate common steps such as *metadata extraction*, *pre-processing*, *algorithm selection* or *model evaluation*.

1.3 Contributions

This work resulted in various contributions:

- **Process for Automatic Visualization:** This work presents details about each step of the approach, including specifications and decision made regarding case representation, attribute features and similarity measure.
- **Case Mapping:** This work presents an innovative technique, labelled Case Mapping, used in the process of Automatic Visualization, that surpasses the limitations found in the related work, regarding the axes and series.
- **Chart processing techniques:** This work presents solutions to complex challenges found during chart processing, such as pagination mechanisms and chart parameters decisions.
- **Scalable and flexible machine learning architecture:** One of the important aspects to develop a system capable of doing Automated Data Mining tasks is the existence of a central machine learning component. In this work we describe the process of creating an architecture that can handle low-level details gracefully while being scalable and flexible.
- **Process for Automated Classification:** Details on how to automate the task of classification are described in detail.
- **Process for Automated Forecast:** Details on how to automate the task of forecast are described in detail.
- **Process for Automated Outlier Detection:** Details on how to automate the task of outlier detection are described in detail.
- **Process for Automated Clustering:** Details about on to automate the task of clustering are described in detail.
- **CRM Classification Model:** Using the machine learning architecture, a manually crafted model for the Salesforce industry was created and described in this work. The model shows high accuracy in the test results.
- **Commercial Product Integration:** Although the work involves research, the final implementation was integrated in the platform - showing the viability of the approaches used in a practical scenario.

1.4 Outline

The outline of this work is described below:

- **chapter 2 - Platform**: In this chapter, an introduction to the Wizdee platform was made. An overview of how the user interacts with the system is also described.
- **chapter 3 - Background Knowledge**: In this chapter some of the main concepts relevant to the work are described. It has a short introduction to visualizations and charts, followed by a detailed description of the Case Based Reasoning (CBR) approach - the chosen approach for Automatic Visualization. Furthermore, in this

section, there is a detailed overview for concepts related to Machine Learning and its algorithms, particularly relevant to the Automated Data Mining task. State of the art literature and other relevant work are described in the Relevant Work. Finally, relevant Resources and Tools are analysed.

- **chapter 4 - Competitor Analysis**: In this chapter, relevant products are analysed, where the focus regards the products features.
- **chapter 5 - Approach**: In this chapter the approach is described, starting with the functional and non-functional requirements, followed by the architecture - where the the system is described, relevant components and components to be developed, risks, and test specification.
- **chapter 6 - Methodology**: In this chapter the methodology used during this work is described.
- **chapter 7 - Implementation**: In this chapter the work developed is presented, where the techniques and procedures are detailed.
- **chapter 8 - Implementation**: In this chapter the results and analysis of the tests are described, for each one of the tasks.
- **chapter 9 - Conclusions**: Finally, in this chapter a summary and discussion is made for the work that has been developed.

Chapter 2

Wizdee Platform

In the following chapter the Wizdee platform is briefly described. There is an introduction to the relevant features in Wizdee product, from an user perspective, in section 2.1. Additionally, a description of the previous automatic visualization system is described in 2.2.1.

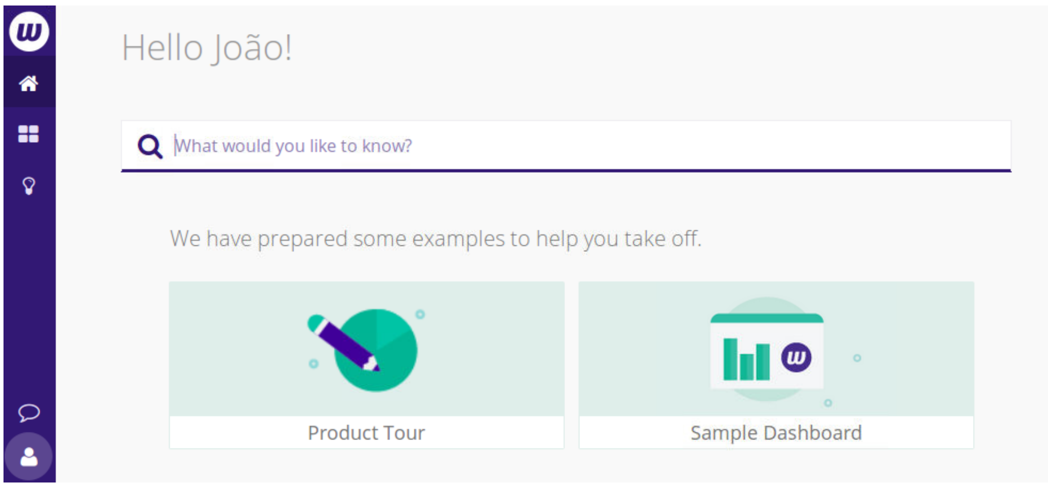

Hello João!
What would you like to know?
We have prepared some examples to help you take off.
Product Tour
Sample Dashboard

Figure 2.1: Wizdee product home page.

2.1 Wizdee Product

The home page of the Wizdee product is depicted in 2.1. The most relevant part is the search box, where an user can write a query in natural language and the search engine interprets the query and returns an answer.

To further explain, we can take a look at the mockup depicted in figure 2.2, where the query *Monthly Sales by City* was entered.

Figure 2.2: Wizdee search box mockup.

With that query, the engine will interpret and retrieve data. In this case, the data relates to the number of sales in each city by month. The end result of this query is a chart, and this is where the Automatic Visualization is used. With the data, obtained by the search engine, the Automatic Visualization system will create the most relevant chart using that data. In this case, a line chart showing the number of sales throughout the months for the available cities could be a reasonable choice, as depicted in the figure 2.3.

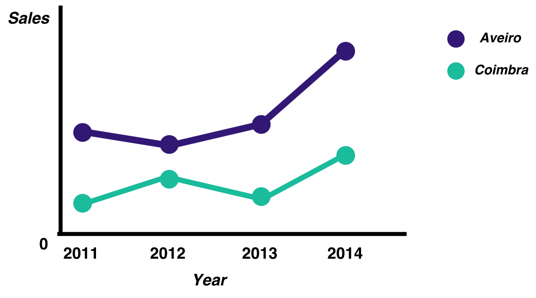


Figure 2.3: Monthly sales by city chart mockup.

Additionally, the user can also see the possible charts and click on a different chart.

Finally, regarding Automated Data Mining, no UI has been developed, however, it is on the roadmap and should be developed in the following sprints. Anyhow, these tasks are requested on-demand, through an UI button. A good example of this is with a query of *Monthly Sales*, where the user would be able to see it is possible to do an Automated Forecast task and after clicking on it, it would show something like depicted in figure 2.4.

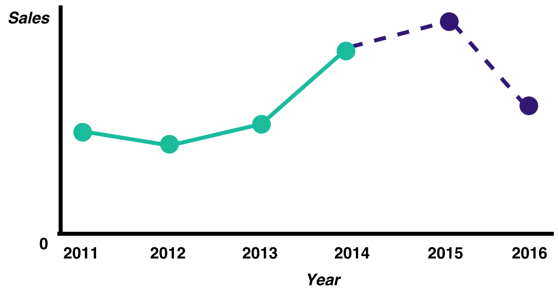


Figure 2.4: Monthly sales forecast mockup.

It is important to note, that in neither task the system requires for more input from the user.

2.2 Automatic Visualization

In this section a brief description of the previous Automatic Visualization system will be made. This previous implementation was based on rules.

2.2.1 Rule-based System

The previous system architecture is depicted in the following diagram 2.5.

All the rules were programmed in Groovy[1] and the listing shows some examples that were taken from those rules. The system has various steps, starting with a request to create

[1]A dynamic language for the Java platform with features similar to Python or Ruby - `http://groovy.codehaus.org/`.

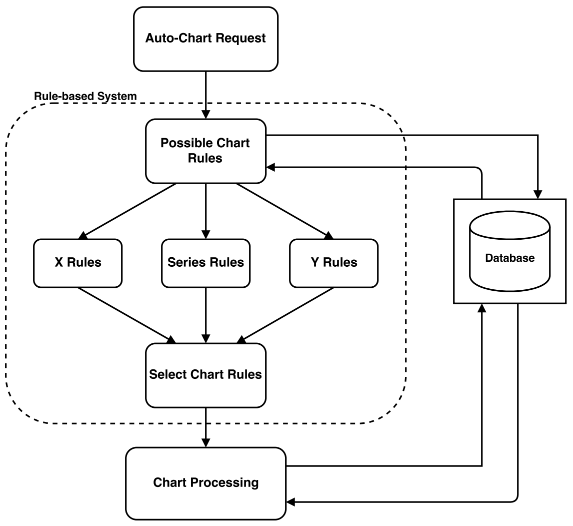


Figure 2.5: Diagram of the previous automatic visualization system, a rule based approach.

an Auto-Chart, followed by defining the axes and series according to the rule system, and finally returning an object that could be processed into a chart.

A brief explanation of the various rule components can be provided as follows:

- **Possible Chart Rules:** In this step the system attempts to figure out what the possible charts are. As an example, according to the rules, to make a Pie Chart, the system needs operators, more than one numeric attribute and an attribute that is not numeric. Writing these rules in groovy would result in the script presented in listing 1:

```
'PIE_CHART': [if(has_operators && num_numerics != 0 &&
attributes.any{a->a.is_unique == true && a.is_numeric == false}) true]
```

Listing 1: Groovy example to check the possibility of a Pie Chart.

- **X Rules:** X rules are used to define what type of attributes can be used as an X axis. For instance, according the rules defined in a *Bar Chart* the values must not be aggregated and for a Line Chart the values must be sequential.
- **Y Rules:** Similar to the X Rules, the Y Rules are used to select a possible attribute for the Y axis. According to the rules defined, a Bar Chart needs an Y Axis that is numeric and aggregated.
- **Series Rules:** These are used to define additional series, besides the main one. It checks the possibility of having more than one *series*. As an example, the Pie Chart can not have more than one series while the Bar Chart can, as long as it is numeric and different from the main series.
- **Select Chart Rules:** Finally, the selection chart rules are a set of rules that choose the best the chart from the possible charts. It follows a strategy of the first-to-be-true is the one chosen.

Chapter 3

Background Knowledge

In the following chapter the state of the art is described by approaching the relevant high-level background concepts and existing technical work. There is an introduction to Automatic Visualization, in section **??**, and Machine Learning, in section 3.2. In the Automatic Visualization section some of the main concepts, creation process and evaluation of visualizations are described, as well an highly relevant approach that is used to tackle some of the challenges found in this task.

Data Mining is often defined as the extraction of implicit and potentially useful information from data, where programs can look through databases automatically seeking patterns (Witten and Frank, 2005). However, this comes with many challenges, from uninteresting patterns to accidental coincidences. To complicate it further, real data is often imperfect, motivating the creation of robust algorithms.

Machine learning provides the technical basis of data mining (Witten and Frank, 2005), as it is used to extract information from raw data. This work approaches data mining by with a machine learning oriented mindset, introducing the tools and techniques for machine learning that are used in practical data mining, focusing on concepts, techniques and underlying ideas in the current practice.

Related work is described and discussed in the section 3.3. Finally in section 3.4, a set of resources and libraries will be briefly presented and compared.

3.1 Automatic Visualization

Visualizations have the ability to provide a huge amount of information - more than that provided by all the other senses combined (Ware, 2013). Some decades ago (W. Litte, 1972), the concept of visualization only meant building a visual image in the mind, but now it means a graphical representation of data or concepts. Information visualization was born out of this conceptualization, combining efforts from many research fields.

However, creating a graphical representation of data or concepts is a challenging task, because information does not always have a clear physical representation. Information is often abstract or complex, and obtaining a visual form that provides insights to the user is one of the main objectives of any visualization (Craik, 1967). As mentioned, visualizations should be able to convoy the meanings of the data they represent, there are some criteria that points to what a good visualization is. For instance, Edward Tufte says that visualizations should give the viewer *"the greatest number of ideas, in the shortest time, using the least amount of ink, in the smallest space"* (Tufte and Graves-Morris, 1983). Also, the visualization should not distort the data or lead the viewer into wrongful interpretations. According to (Ware, 2013), a good visualization can provide the following:

- **Big Data:** the ability to comprehend huge amounts of data by showing the important aspects immediately.
- **Emergent Properties:** the perception of properties that weren't originally anticipated - the visualization of a pattern can be the basis of new insights.
- **Apparent Problems:** problems with the original data become obviously apparent. Visualization reveals things not only about the data but also about the way it is collected. Errors and artefacts in the data often jump out and for this reason, visualizations can be an invaluable tool in quality control.
- **Data Features:** allows the understanding of both large-scale and small-scale features of the data, making it possible to see patterns linking the data.
- **Hypothesis Formation:** facilitates hypothesis formation, for instance motivate research about the significance of certain features.

Some work has been made on defining the needed steps that should be followed when creating visual representations, for instance (Ware, 2013) describes the process into four stages and some feedback loops:

- **Data:** collection and storage of data.
- **Preprocessing:** transforms the data into something easy to manipulate, may include more advanced concepts such as data reduction.
- **Mapping:** mapping the selected data into a visual representation, typically through algorithms that produce an image on the screen but not always.
- **Human:** the stage of the perceiver, of the human perceptual and cognitive system.

Other methodologies can be found in literature, for instance in the work by Riccardo Mazza (Mazza, 2009) there are five steps to follow when creating visual representations:

- **Define the problem:** identification of the needs for the visualization.
- **Nature of the data:** the data can take many forms: numerical, textual, etc.
- **Identify attributes:** data has a number of attributes, those can be dependent or independent, and dependent attributes are analysed in respect to independent attributes.
- **Data structures:** data structures can take many forms, they can be linear (e.g. vectors, tables), temporal - data that changes over time - spatial (e.g. maps), hierarchical (e.g. flowcharts, files) or networks - representations of relationships between entities.
- **Interaction type:** the interaction type can be static, where no modification by the user is possible, transformable, when the user can make modifications on the data (e.g. changing parameters) or manipulatable, where the user controls the visualization itself.

Both the stages described in (Ware, 2013) or the steps outlined by (Mazza, 2009), show a similar process for creating a visualization. The work presented in this thesis will be focused on automating those earlier stages of this process, such as data processing and mapping.

3.1.1 Concepts

Some concepts regarding information visualization are used throughout the work of this thesis, these are necessary to have a better understanding of the techniques and algorithms that follow.

Chart

A chart is a visualization technique, that conveys data better than a simple table is an useful technique because data is represented by symbols, such as bars in a bar chart or slices in a pie chart (Stacey et al., 2013).

A chart can be composed of various elements, but typically the most relevant are:

- **Item** An item or data point is a single element in a chart (e.g. a single bar in a bar chart, single point in a scatter plot) (Karsten, 1925).
- **Axis** In the context of this work, we can define an axis simply as a reference line of the visualization system. Some charts, like the bar chart, have two axes (typically titled as x and y) while others can have just one (e.g. pie chart).
- **Values** A set of measurements for each axis of an item (Karsten, 1925).
- **Series** A collection of measurements is called a series (Karsten, 1925). In a practical sense, a series is a set of data, for instance, a line graph or one set of columns.
- **Labels** Labels are an important part of a chart, whether it is the title of the chart or labels for the axes, they provide information about what is being visualized.

Chart Types

Choosing the correct chart to represent the data is considered a non-trivial task (Suda, 2010). Not only there isn't a set of rules, there are even contrasting opinions found in literature. For instance, Edward Tufte refers to pie charts as dumb in his book (Tufte and Graves-Morris, 1983), because humans are not good at estimating areas and doesn't recommend using them at all. Yet, it is a widely popular type of chart, commonly requested by users and found in business and media.

These are some popular chart types found in literature:

- **Bar Chart:** This is a chart with rectangular bars with length proportional to the values that they represent. One axis shows the specific categories of the series while the other axis show their values.
- **Line Chart:** Line charts are similar to bar charts but they are made of continuous lines which connect the data points. It is commonly used with continuous data, such as sales through time.
- **Scatter Plot:** Scatter plots are an useful tool that can reveal relationships between independent values. The data points are placed in a grid and with it is possible to observe possible correlations (e.g. age and height).
- **Pie Chart:** The pie chart is a circular graphic divided into slices to show numerical proportion.
- **Histogram:** Histogram show the distribution of the data by arranging the data into bins - dividing the range of values into intervals and counting how many values fall into each interval.
- **Gauge Chart:** Gauge charts are used to show progress levels, a goal value is defined and the data is compared against the goal using a gauge. Since this chart is more unusual than the others, a mockup of this chart is depicted in figure 3.1.

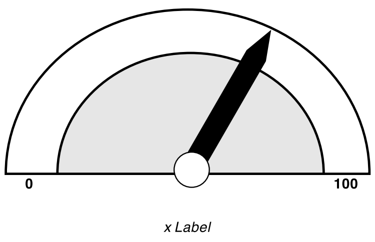


Figure 3.1: A gauge chart example.

3.1.2 Evaluation

Evaluating a system that uses visualizations is a difficult task (Mazza, 2009) because it is typically subjective. Each person has their own experiences and knowledge which can produce different reactions for the same visualization. Taking this into consideration a system that produces visualizations must be evaluated through a a specific methodology that tries to measure the effectiveness of the user interface and more importantly to this work, the quality of the information is gives to the users in terms of the knowledge that can be extracted. A wide variety of methodologies have been researched and developed, from techniques originating in Human-Computer Interaction (Dix et al., 2003), to a series of improvements over classic testing tools such as controlled experiments designed to evaluate visualization (Plaisant, 2004), to a complex approach that defines a series of steps that include extensive documentation and use of expert users (Shneiderman and Plaisant, 2006).

3.1.3 Case Based Reasoning

Developing a system that is able to decide automatically which visualization to choose, according to the data available, while being as effective as visualizations created manually, is far from a trivial task. Various approaches have been studied throughout the years and some of those have been able to produce interesting results. This work takes an approach largely based on advances made in the field of Artificial Intelligence.

Case Based Reasoning Case Based Reasoning (CBR), as explained by (Kolodner, 1992) and (Riesbeck and Schank, 1989), is inspired by the human reasoning and memory organization, where our knowledge comes from memory packets from significant events in our lives, interconnected by our expectations. This means, that if a person finds himself in a similar situation to a past event, the previous memory is obtained and the person will follow the same steps to reach a solution. Rather than following a set a rules, a person reapplies previous successful solution steps in a new context. This algorithm puts this model in practice, where previously solved problems, in the form cases, are used against new but similar problems.

Some advantages of CBR when compared to other approaches, such as rule-based systems, according to (Riesbeck and Schank, 1989), are:

- CBR doesn't require a domain model therefore knowledge aggregation becomes a task of gathering cases.
- CBR is implemented by identifying the most important features, an easier task than creating a model for a complex domain. Humans experts are much more capable of recalling experiences than of articulating internal rules.

- CBR can learn and gain new knowledge simply with the addition of new cases, furthermore, the system is changed and augmented by each additional case that is presented.

There are also a few assumptions about the world that represent the basis of the CBR approach, as defined by (Leake, 1996) :

- **Regularity:** the same actions under the same conditions will have the same or similar outcomes.
- **Typicality:** experiences tend to repeat themselves.
- **Consistency:** small changes in the situation require small changes in the interpretation and in the solution.
- **Adaptability:** when things repeat, the differences tend to be small, and the small differences are easy to compensate for.

Taking this into consideration, now we must define what a case is.

Case Representation

A case is a representation of an experience. In general, a case compromises a problem description and a problem solution. Most CBR systems are limited to store only problem-solution pair cases (Pantic, 2014).

The problem description should contain as much data about the problem as needed, for an efficient but accurate case retrieval. Other information such as problem id, retrieval statistics (e.g. number of times retrieved, average match value) can be valuable to maintain, in order to improve the system.

Cases are usually represented in some form of feature vector but they can also be represented in a more formal way such as objects, frames, predicates or rules (Pantic, 2014). Ultimately, the particular representation depends largely on the information to be stored and the complexity of the process.

CBR Cycle

The original CBR cycle, according to (Leake, 1996), can be described in four processing stages: case retrieval, case adaptation, solution evaluation and case-base updating. However, the process was later revised in (Agnar Aamodt, 1994), using a different terminology but keeping a similar meaning.

- **Retrieve:** Given a new target problem, the new problem is matched against the case base and the most similar cases are retrieved.
- **Reuse:** A solution is suggested by the matching cases, and a solution is mapped from the previous case to the target problem.
- **Revise:** With a solution, unless the match is a close or exact match, it might have to go through adaptation and therefore revisions and tests.
- **Retain:** If the solution has been successfully adapted to the target problem, a new case is retained and added to the case base.

This can be visualized in figure 3.2.

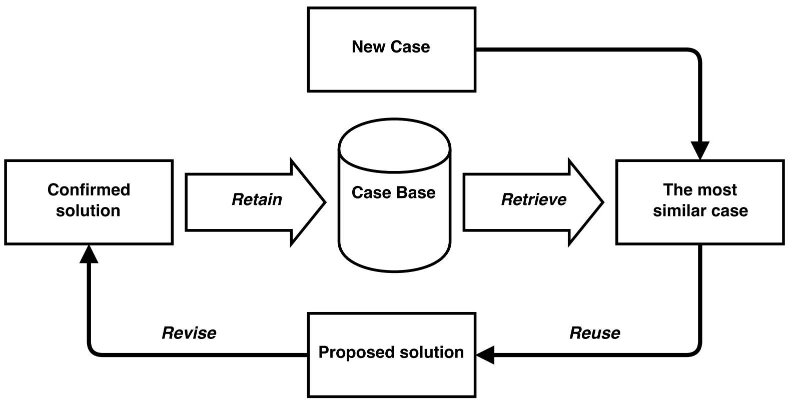


Figure 3.2: Case-base reasoning cycle.

Case Library

The storage of the cases is one of the most important aspects in a CBR system. It should provide the means to create an efficient and accurate search, however, accurate retrieval and efficient retrieval are inversely proportional(Pantic, 2014).

Considering there's no perfect solution to this problem, it is important to choose the best solution for the specific domain by optimizing the compromise between accuracy and efficiency.

In general, one can find three main approaches to CBR organization:

- **Flat Organization:** The most straightforward organization, where the case base follows a flat structure such as list. It is simple to create and maintain. Search is usually done in a case-by-case search of the whole case base. It is very accurate for this reason but it is also inefficient.
- **Clustered Organization:** In this type of organization, the cases are stored as clusters of similar cases. The advantage is that it can reduce the time needed to find the best match since they are grouped by similarity. The disadvantage is that it needs a complex algorithm to add and delete cases, which makes it harder to maintain. The quality of the cluster may also reduce the accuracy of the system.
- **Hierarchical Organization:** When cases share the same features and the domain allows categories, these cases can be grouped together. Each case has an assigned category and its category is inter-linked within the structure. It has the disadvantage of higher complexity but has the advantage of fast and accurate case retrieval. A common approach, as seen on (Barry Smyth, 2014), is to have abstract cases, offering high-level solutions and concrete design cases. This way it is possible to decompose target problems into sub-problems and reuse parts of complex problem as individual cases.

Similarity Measure

A retrieval algorithm should retrieve the cases that are most similar to the problem at hand, however, choosing the best matching cases can be a complex task. There are many ways of retrieving cases, including: nearest neighbour search, serial search, hierarchical search, parallel search, etc. Yet, the quality of the matched cases highly depends on the similarity measure. Creating appropriate similarity measure is considered one of the major challenges in CBR (Steffens, 2006).

Sometimes, the measure is domain specific and requires a domain expert. Other times it is possible to apply direct algorithms and methods on the cases problem-feature vector, such as an euclidean distance. Other algorithms and methods such as neural networks or support vector machines can also be used.

Adaptation

As previously mentioned, once a matching case is chosen, it may not correspond exactly to the same problem and may need adaptation. This is a domain specific issue but there are a few general approaches to deal with adaptation as explained in (Watson and Marir, 1994) and (Riesbeck and Schank, 1989):

- **Structural Adaptation:** Applies rules directly to the solution stored in cases. Can include modifying certain parameters and interpolate various cases to generate a good solution. Requires some understanding of the problem domain. e.g. $LargeMeatRecipe.duration = SmallMeatRecipe.duration * LargeMeat.size$
- **Abstraction Adaptation:** If a piece of the matched solution does not apply to the target problem, we abstract that part and try to specialize. e.g. explaining human diseases with animal diseases, makes it possible to theorize solutions by using abstracted solutions in another domain.
- **Critic-based Adaptation:** A solution is proposed and its features are checked for problems. Each problem has a pre-defined strategy to solve it. Side effects of each strategy must also be taken into account. e.g. taking a recipe as a solution and simply substitute the old ingredients for new ones may cause unnecessary steps.
- **Derivational Adaptation:** Reuses algorithms or methods that generated the original solution to produce a new solution. This requires knowing the planning sequence that produces a solution. This kind of adaptation is highly domain specific and can only be applied in well understood problem domains. This can mean replacing a step in the solution-generator to make it in context with the problem. e. g. solving a dispute between Egypt and Israel over Sinai by adapting the solution applied for USA and Panama over the Panama Canal where the method to reach the solution was *"divide into different parts"*, and use that method over Sinai to reach a solution.

3.2 Machine Learning

Machine Learning (ML) can be defined *"as computational methods using experience to improve performance or to make predictions"*, experiences relating to past information , typically in the form of data (Mohri et al., 2012). Furthermore, according to Tom Mitchell *"The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience"* (Mitchell, 2006). Additionally, there is a certain multidisciplinary element found in Machine Learning. This is supported in (Marsland, 2014) with *"one of the most interesting (...) is that it lies on the boundary of several different academic disciplines, principally computer science, statistics, mathematics and engineering. This has been a problem as well as an asset, since these groups have traditionally not talked to each other very much (...) "*.

Before dwelling into Machine Learning and its algorithms, it is important to understand some basic concepts and terminology. The following section will focus on a few important concepts, however, there are many more. The work found in (Alpaydin, 2004), (Witten and Frank, 2005) and (Mohri et al., 2012) provides a more comprehensive look into these concepts.

3.2.1 Concepts

The concepts are organized by high-level topics, starting with Knowledge Representation, in section 3.2.1, followed by Generalization, in section 3.2.1, and ending with Evaluation, in section 3.2.1.

Knowledge Representation

Data Data can take various form, a traditional structure for data is rows and columns, like a database table or an excel spreadsheet. Other types of data can be images, videos and text and this is called unstructured data. It is often represented or abstracted as an $n \times d$ data matrix with n rows and d columns (M.J. Zaki, 2014).

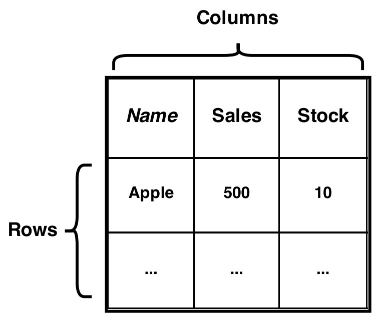


Figure 3.3: A table exemplifying data in rows and columns.

Instance An instance is an observation from the domain, in a standard database structure it is a single row of data. Each instance is characterized by the values of attributes that measure different aspects of the instance. (Witten and Frank, 2005)

Features In a standard database structure it is a single column of data. In an abstract way of reasoning it can be considered a component of an observation, also called an *attribute*. Some features may be inputs but others can be outputs, often represented as a vector. Continuing the example of spam in emails, relevant attributes for an email dataset could be length of the message, name of the sender, domain, subject, presence of specific keywords on the body and more(Mohri et al., 2012).

Training sample/set Examples or input data used to train a learning algorithm. In the spam problem the training consists of a set of examples along with their associated labels. The size of the training sample varies according to the task.(Mohri et al., 2012)

Test sample/set Examples used to evaluate the performance of the algorithms. This sample is separated from the training (and validation samples) and is not used for learning purposes. Its purpose is to assess the performance of the algorithm against un-trained data - this can be used to detect a phenomenon called *overfitting*(3.2.1). Considering a simplistic overview, in the spam problem the algorithm would try to classify a set of email examples with the label "spam" or "non-spam" based on the features available and those predictions would be compared against the labels of the test sample to measure the performance of the algorithm.(Mohri et al., 2012)

Validation sample/set While not always used, this is examples used to tune the parameters of a learning algorithm during training with labelled data. Not all algorithms have parameters but most have at least one, those parameters can influence the performance of the algorithm and the validation sample is used to select the most appropriate values for that.(Mohri et al., 2012)

Model The result of the learning process is called a model. The model can take various representations, in some cases the output can take the form of a collection of rules, trees, clusters and more but a common and very simple style of representation is a *linear model* where the output is just the sum of the attribute values with weights that are applied to each attribute. (Witten and Frank, 2005)

Generalization

Generalization To make decisions or predictions for instances that were not seen during training, a process of *generalization* is required. Generalization can be reasoned as search through an enormous, but finite, search space (Witten and Frank, 2005).

Supervised Learning This is a way of learning where the correct prediction is known and required for the process. Generally the algorithm compares the predictions by the model to the known answers and corrects the model in order to optimize the amount of correct answers. Of course, in this case for every example in the training data there is a label associated (Bell, 2014).

Unsupervised Learning A type of learning where no known results are required. This type of learning focus on the overall structure of the data and tries to find relations between the instances. In this type of learning there's usually no explicit right or wrong answer, it's a matter of finding good patterns. (Bell, 2014)

Semi-Supervised Learning Input is mixture of labelled and unlabelled data. In this type of learning the model must learn the structure as well as make predictions. Typically a small amount of labelled data with a large amount of unlabelled data. The motivation behind this is that many situations have huge amounts of data but they are not labelled and this is an expensive manual process (Witten and Frank, 2005). In the figure 3.4, the left side shows a *decision boundary* - a hyperplane that divides the classes - learnt with just two labelled points and the right side shows the decision boundary with added unlabelled points.

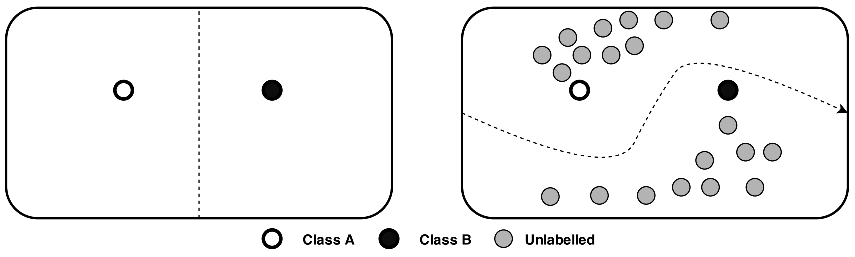


Figure 3.4: Example showing the influence of unlabelled data in semi-supervised learning.

Overfitting A central problem in any ML process, where an algorithm seems to be a very good fit in training data but results in inaccurate predictions on unseen data. The assumption that the patterns that hold the training set are valid for the entire domain is not necessarily true (Larose, 2005). This phenomenon is called *overfitting* (Fürnkranz et al., 2012). The cause for this can be many: ranging from the features not capturing the most important domain characteristics, to data noise, due to a excessively complex algorithm model e.g. too many parameters for the number of examples.

Underfitting On the other side of the spectrum of *overfitting* is *underfitting*, this happens when a model has not learned enough about the structure at hand. This can be due to various reasons from the learning process being too short, lack of data or bad parametrization (Larose, 2005)

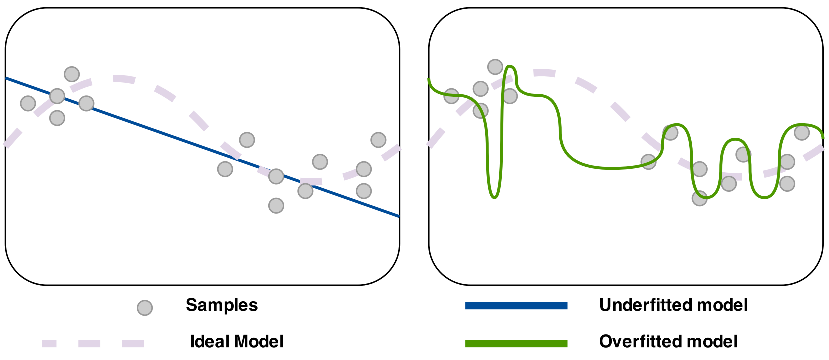


Figure 3.5: An example of overfitting and underfitting.

Bias Most models have accuracy errors since they are generalizations from observations. Bias are introduced by the generalizations made in the model and it's the limits imposed on the selected model, the errors from erroneous assumptions, making it related to under-fitting. In some extreme cases it's not possible to reduce bias such as modelling a *linear regression* model to fit a *quadratic function*. The solution would have an high bias no matter how the parameters are set. (Larose, 2005)

Variance The sensitivity of the model against the fluctuations in the training dataset is called variance (Larose, 2005). A way to reason about variance has to do about how easily the changes are made to the predictions in response to the training set, thus an high variance does not generalize well and is related to over-fitting.

Bias-Variance Tradeoff When selecting a model, there's a trade-off between the bias and variance. As model complexity increases, the bias on the training set decreases but the variance increases. (Larose, 2005) Ideally the algorithms wants to choose a model that captures the small details of the data but also generalizes well to unseen instances. With an *high bias* model more data won't solve the issue beyond a certain point of accuracy, this is a situation where more features that represent better the data can improve the score. In contrast to an *high variance* scenario where getting more training examples or even reducing the set of features can improve it.
In the figure 3.6 when the model complexity is *low*, the variance is low but the generalization

is *high* - high bias - however the error is big. Similarly when the model complexity is *high*, the variance is high, but the generalization power is *low* and the error is also high.

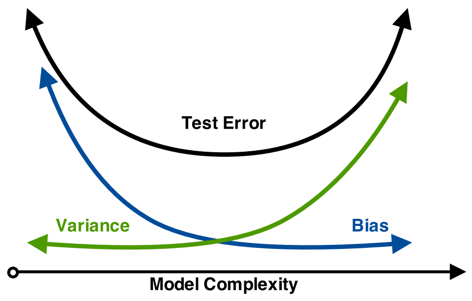


Figure 3.6: Bias and variance tradeoff.

Evaluation

Evaluation is a critical aspect on any ML problem. There are many ways to create a model but in order to determine which method or algorithm to use in a particular task a good process to evaluate how different methods work is needed to compare one with another, it is often considered to be simple task on the surface but increasingly complex for more accurate techniques (Witten and Frank, 2005). In the following paragraphs explanations will be given on the process of evaluation.

Error rate Using an error rate, where the model predicts the class of each instance, and if it is correct, that is counted as a success. If not, it is an error. With this the error rate is just the proportion of errors in the dataset. However, this way of reasoning has some limitations, if only applied to the training dataset (Witten and Frank, 2005).

What we are interested when doing classification work, is how the model behaves when presented with new instances, not the past performance on training instances.

Therefore, to truly assess the performance of the model on new data, the error rate must be retrieved from a dataset that is independent from the training instances. This is called the test dataset as described on the Concepts section, in (3.2). It is important to note that the test samples can't be used in any step of the creation of the model, such as cases where an algorithm involves parameters to optimize. The test samples can't be used to optimize those parameters. another separate dataset must be used, the validation dataset.

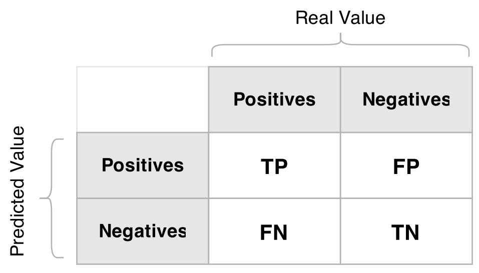
One of the challenges, regarding model evaluation, is how to split the dataset. The model improves with the quantity of data used in the training dataset and the error rate estimation improves with the quantity of data used in the test dataset. In Cross-Validation some techniques are described that attempt to minimize this issue.

Cross-Validation A simple way of estimate the error rate is called the holdout. In this method a certain amount of the dataset is reserved for testing and the remainder for training. Although there's no perfect ratio it is common to hold out one-third of the data for testing and the remaining for training (Witten and Frank, 2005). There are disadvantages in this method, for instance, the samples used for training might not be representative of the domain and, in extreme cases, certain classes can be omitted from the training set. Random sampling is a technique that may help with issue and this procedure is named stratification. However, even this might not be able to solve the issue of uneven representation, since a large percentage of the dataset is used for testing.

Finally, a statistical technique called cross-validation was created to solve this issue. In cross-validation there is a fixed number of folds, then the data is split into k equal partitions, each in turn is used for testing and the rest is used for training, this procedure is repeated k times, so that every instance has been used exactly once for testing. A common value for k is 10, since 10 has been showed to get a good estimation of error (Witten and Frank, 2005).

Cost Till now, the methods described do not take into account the cost of making wrong decisions. For instance, if a certain problem has a very unbalanced amount of one specific class, the methods above could paint an erroneous picture. For instance, if in a spam detection problem, 95% of the time the email is not spam and the model outputs *"not spam"* always, it would create a model with an impressive accuracy. Depending on the use case, this can lead to serious problems, such as in oil-slick detection, *"the cost of failing to detect an environment-threatening real slick is far greater than the cost of a false alarm"*(Witten and Frank, 2005).

If we have a two-class problem, with the classes yes or no - spam or not spam for instance - a single prediction can have four different outcomes, as show on the confusion matrix, in figure 3.7. The true positives and true negatives show the correct predictions. A false positive happens when a prediction is incorrectly predicted as *"yes"*) when it should be a *"no"*. In contrast, a false negative is when the prediction is incorrectly predicted as *"no"* and it should be *"yes"*.


Real Value
Predicted Value

	Positives	Negatives
Positives	TP	FP
Negatives	FN	TN

Figure 3.7: Confusion matrix

We can define the true positive rate as TP divided by the total number of positives, which is $TP + FN$. The false positive rate is FP divided by the total number of negatives which is $FP + TN$. Finally the *error rate* is 1 minus the number of correct classification divided by the total number of classifications(Witten and Frank, 2005):

$$errorrate = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

Measures If we have a system that locates 100 documents, 40 which are relevant and another that locates 400 documents, 80 which are relevant, which is better? This is problem proposed in (Witten and Frank, 2005) and the answer is that it depends on the cost of false positives - documents returned that aren't relevant - and false negatives - documents that are relevant but aren't returned. For this two concepts were defined, *recall* and *precision* (also known as *sensitivity* and *specificity*):

$$recall = \frac{num\_of\_documents\_retrieved\_that\_are\_relevant}{totalnumber\_of\_documents\_that\_are\_relevant} = \frac{tp}{tp + fn} \tag{3.2}$$

$$precision = \frac{num\_of\_documents\_retrieved\_that\_arerelevant}{total\_number\_of\_documents\_that\_are\_retrieved} = \frac{tp}{tp + fp} \tag{3.3}$$

Plotting one against the other can help answering the question of which is the better system. Albeit, there are also solutions that create a single measure that can characterize quality. A very common one, is the F-measure:

$$f_{measure} = \frac{2 \times recall \times precision}{recall + precision} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{3.4}$$

Number Predictions All the evaluations measure described above work in classification problems but numeric prediction problems require a different set of measures. The basic evaluation principles such as using a test dataset and training set as independent datasets are still valid. Likewise for the holdout and cross validation techniques. However, the measure that is used to obtain error rates is no longer appropriate due to the property of numeric predictions error not belonging to classes - they have ranges of values.

There are many measures that can be used to evaluate the success of a numeric prediction, this is a list of commonly used ones, where p_i is a prediction and a_i is the correct value:

- **Mean-Squared Error**

$$error = \frac{(p_1 - a_1)^2 + ... + (p_n - a_n)^2}{n} \tag{3.5}$$

- **Mean Absolute Error**

$$error = \frac{|(p_1 - a_1)| + ... + |p_n - a_n|}{n} \tag{3.6}$$

- **Relative-Squared Error**

$$error = \frac{(p_1 - a_1)^2 + ... + (p_n - a_n)^2}{(a_1 - \bar{a})^2 + ... + (a_n - \bar{a})^2} \tag{3.7}$$

  where

$$\bar{a} = \frac{1}{n} \sum_{i=0}^{n} a_i \tag{3.8}$$

- **Mean absolute percentage error**

$$error = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{a_i - p_1}{a_i} \right| \tag{3.9}$$

Applications

Applications of Machine Learning correspond to a wide variety of learning problems or tasks, this is a short list of common problems, as briefly explained by (Mohri et al., 2012) and (Alpaydin, 2004):

- **Classification:** Discriminate classes for the input data e.g. categorize images into landscape, face or animal. The number of categories, although generally small, can be large or even unbounded as in OCR, text classification or speech recognition.

- **Regression:** A prediction of a value based on the attribute values. There are many examples of this in various fields, from prediction of stock values or variations of economic variables to the prediction of weather or the value of an used car.

- **Clustering:** It's the partition of items into homogeneous regions. A common example is the attempt to identify "communities" within groups of people, the strategy behind market segmentation. This type of algorithm is often performed to analyse very large data sets.

A relevant problem to this work, Outlier Detection, can be a specific application of the problems explained above, however, there are also simpler techniques, purely based on statistical methods.

3.2.2 Algorithms

To understand machine learning it is important to study some of its algorithms, but one of the common issues is the amount of algorithms used in this field. Categorizing algorithms by their mechanism and behaviour is used to organize them which helps understanding what is an extension to methods and what is a *canonical* algorithm.

In the following section, popular and relevant ML algorithms are described. However, many more algorithms and mechanisms exist.

Regression

Regression is a process that refers to modelling the relationship variables into predictions iteratively using a measure of error. They are widely used in statistics (Witten and Frank, 2005).

Algorithms

Linear Regression Linear Regression is a popular algorithm for when the desirable outcome is numeric and all the attributes are also numeric (Witten and Frank, 2005). The main idea is to express the prediction as a linear combination of the attribute with predetermined weights, as de equation 3.10 shows:

$$x = w_0 + w_1 a_1 + w_2 a_2 + \cdots + w_k a_k \tag{3.10}$$

where x is the prediction, $a_1, \ldots, a_k$ the attributes and $w_0, \ldots, w_k$ the weights.

The weights are pre calculated using the training data, and using the difference between the predicted the actual values the method of linear regression can choose the coefficients w_j. Linear regression can be considered a simple numeric prediction algorithm but it has been used in statistical applications with success for decades (Witten and Frank, 2005). This algorithm has the disadvantage of being linear and in certain tasks the data might not fit a linear model.

Other common algorithms are:

- **Logistic Regression**
- **Stepwise Regression**

Instance-based

Instance based learning models a decision based on examples of training data that are deemed important. Typically the model creates an internal database of example data and compares with new data using a similarity measure in order to find the best match and make a prediction. They are also called winner-take all as well as case-based andmemory-based learning. Importance it put on representation of the instances and similarity measures used between instances. Once the nearest training instance has been located, its class is predicted for the test instance (Witten and Frank, 2005).

The distance function is one of the most important parts of the algorithm and the most common distance is the *Euclidean distance*, which is similar to how humans think of distance in the real world (Larose, 2005):

$$d_{Euclidan} = \sqrt{\sum_{i}^{n} (x_i - y_i)^2} \tag{3.11}$$

where x and y represent the attribute values of two instances.

Other popular distances are the *Mahalanobis distance*, which is based on the idea of measuring how many standard deviations away P is from the mean of D:

$$D_{Mahalanobis} = \sqrt{(x - \mu)^T S^{-1} (x - \mu)} \tag{3.12}$$

or *Chebyshev distance*, which is a metric where the distance between two vectors is the greatest of their difference along any dimension, if we imagine the vectors with two dimensions (a and b), their Chebyshev distance is:

$$D_{Chebyshev} = \max\left(\left|b_2 - b_1\right|, \left|a_2 - a_1\right|\right) \tag{3.13}$$

Other measurements also exist for other data types, such as the *Hamming distance* that works strings

When measuring distance certain attributes have large values and that could distort the results, giving less influence to attributes that are measured in smaller scales. To avoid this *normalization* is needed, such as.

- *Min-max normalization*

$$Z = \frac{X - \min(X)}{\max(X) - \min(X)} \tag{3.14}$$

- *Z-score standardization*

$$Z = \frac{X - \mu(X)}{\sigma(X)} \tag{3.15}$$

Algorithms

k-Nearest Neighbour The intuition behind k-NN is to consider the most similar items according to their attributes and by looking at their labels, assign the class according to a voting system. There are two major questions: how to define similarity and how many neighbours should the algorithm take a look at.

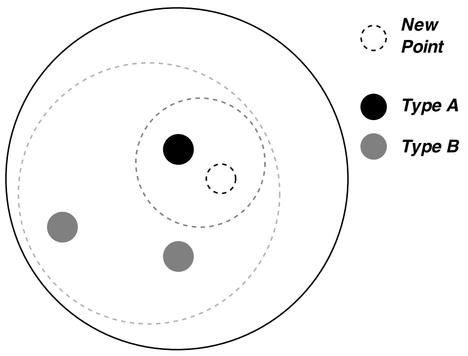


Figure 3.8: Four data points, one unlabelled point and three labelled.

Let's assume four data points, similar to the figure in 3.8, if we choose $k = 1$ for the algorithm, the result would be the label of the nearest point - *black*, but if $k = 3$ the result would be the label based on the three points closest to it and the vote system would choose *medium grey.* note that the classification assigned differed on the value we choose for k.

Choosing a value for k is not always obvious, if the k value is too small, it might get too influenced by outliers or noise, however if k is too large, the closest neighbours lose influence (Schutt and O'Neil, 2013). Therefore a balance is needed which makes it perfect scenario to use the evaluation methods explained earlier.

Other common algorithms are:

- **Radial Basis Function Networks**

Time-series analysis

A time series can be described as a sequence of points, typically made over a time interval. Time series analysis consists of methods that analyses time series data in order to extract statistics and characteristics of the data.

Algorithms

ARMA Autoregressive-moving-average (ARMA)(Box et al., 2011) models are widely used in statistical analysis of time series. They consist of two polynomials, one for the autoregression and the second for the moving average. Using a dataset, containing a time series, the ARMA model can be used to forecast future values.

The autoregressive part of the model describes a time-varying process. It specifies that the output depends linearly on the previous values. The moving-average part of the model is conceptually a model that contains a series of average of different subsets of the full data.

Other common algorithms are:

- **ARIMA**
- **ARIMAX**

Decision Tree

Decision tree algorithms create a model based on decisions based on the actual *values* of attributes in the data. Decisions fork in tree structures until a prediction decision is made for a given entry. They are generally used in classification and regression problems. Some

of advantages of them are how the model generated is usually easy to read and interpret how some of the decisions were made, how it can handle numerical or categorized information and how it handles large datasets(Bell, 2014).
Some of the limitations is how overly complex the model can get depending on the attributes and training set, making it possible to overfit the data easily.
Every tree is comprised of nodes, each node is associated with of the attributes. The edges are the total possible values of the node. Decision trees always start with a root node and end on a leaf.

Algorithms

ID3 The Iterative Dichotomiser 3 (ID3) calculates the entropy for every attribute in the dataset, it then selects the attribute with the smallest entropy and the dataset is split by the selected attribute, it continues recursively on each subset.(Bell, 2014). It uses the method of information gain, the measure of difference in entropy to decide on the root node - the node with the highest information gain.

C4.5 The C4.5 algorithm is also based on the information gain method but it enables the trees to be used for classification. At each node of the tree, it chooses the attribute of the data that most effectively splits its set of samples into subsets in one class or another (Bell, 2014). Other improvements in C4.5 over the original ID3 is the ability to work on continuous attributes, for instance, with a list of values like following:

$$a = 80, 40, 50, 60, 48, 23, 78, 122, 39, 88, 81 \tag{3.16}$$

C4.5 can work out a split point for the attribute a and give a decision criterion of:

$$a \leq 80 \vee a > 80 \tag{3.17}$$

C4.5 also has the ability to work with missing attribute values. The gain and entropy calculations are skipped when there is no data available.

Other common algorithms are listed:

- **CART**
- **J48**

Kernel

Kernel methods relate to the use of kernel functions that enables the mapping of input into a higher dimensional vector space where some classification or regressions problem are easier to model. The most common algorithm in this type of approach is Support Vector Machines.

Algorithms

Support Vector Machines (SVM) Support Vector Machines are models that, using supervised learning, create a hyperplane in a high dimensional space (Cortes and Vapnik, 1995). These models can be used for classification, regression and other tasks. A good model is defined by a hyperplane that has the largest distance to the nearest training instance of any class. The motivation behind the construction of an hyperplane, in high-dimensional space, is that in many problems, the sets to discriminate are not linearly separable in that space. Mapping the dataset into an higher-dimensional space makes the separation easier, however, to make this happen, kernel functions are used.

Therefore, some advantages of Support Vector Machines include effectiveness in low and high dimensional spaces - even where the number of dimensions is greater than the number of instances - and versatility, by specifying custom kernel functions. Finally, one of the issues of Support Vector Machines, is that it does not directly provide probability estimates.

Other common algorithms include:

- **nu-Support Vector Machines**

Clustering

Clustering algorithms are typically organized into centroid-based or hierarchical based, but there are more ways such as the distribution-based or density-based.
Independently of the type, the algorithms are concerned about using the inherent structure of the data to best model it into groups of maximum similarity. These clusters reflect some mechanism that causes some instances to have stronger resemblance to each other than they do to the remaining instances (Witten and Frank, 2005).

Algorithms

k-Means This classic method is also known as iterative distance-based clustering, first the number of clusters must be specified, this is the parameter k. After choosing a k value, k points are randomly chosen as cluster centers. The instances are then assigned to their closest cluster according to a distance metric (Witten and Frank, 2005).Next, the *centroid*, or of the instances in each cluster is calculated, these centroids are going to be the new center values for their respective clusters. Iterations continue until the cluster centers have stabilized.

This clustering method is simple and effective, once the iterations have stabilized, each point is assigned to the nearest cluster center. Obviously there's no guarantee that it is the global minimum, since the final clusters depend on the initial random choice. The common tactic to increase the chances of getting a global minimum is to run the algorithm several times and choose the best final result - the one with the smallest total squared distance (Witten and Frank, 2005).

K-means can be improved, by carefully choosing the initial cluster centers. A procedure named *k-means++* can improve both speed and accuracy over the original algorithm (Arthur and Vassilvitskii, 2007).

Other common algorithms are:

- **Expectation Maximisation**
- **DBSCAN**

3.3 Related Work

This section reviews relevant literature that can offer insights on techniques used to solve specific challenges and tasks associated to either automatic visualization in subsection

(3.3.1) or automated data mining in subsection (3.3.2). Companies that develop applications or products relevant to this thesis are discussed in Chapter 4. This section only concerns with techniques and algorithms, not on their use in existing products.

3.3.1 Automatic Visualization

In recent years a lot of literature work has been put to improve visualization systems, from creating aesthetically pleasing visualizations to developing methodologies that help building strong user experiences and more recently to the automation of certain decisions. These are some relevant works that tackle challenges in creating automatic visualization mechanisms.

Creating Visualizations: A Case-Based Reasoning Perspective

A relevant work by Jill Freyne and Barry Smyth (Jill Freyne, 2009) that explores how case based representation of datasets including simple features such as size and content types can produce recommendations of visualizations types . Although the purpose was to assist novice users in the selection of appropriate visualizations, the work presented in this thesis share some of the motivations behind this paper, for instance *"Microsoft's Excel offers 11 different types of chart and a total of 73 variations (...) the problem, of course, is that the average user is not a visualization expert and producing the right sort of visualization for a given dataset is far from trivial."* which mentions how overwhelming and often complicated of a task it is for a typical end-user to create useful visualizations from raw datasets.

Jill Freyne et al. used a social platform called Many Eyes[1] to make the visualizations, it has the advantage of having thousands of datasets and thousands of visualizations created. These visualizations have importance knowledge in terms of the decisions taken by a user about how to visually represent a given dataset. This knowledge provided them the case specification and the resulting visualization provided the case solutions. Therefore each case represent a single visualization of a single dataset, as shown in 3.18 where c_i is a case, d_i a dataset component and for visualization component, v_i.

$$c_i = \{d_i, v_i\} \tag{3.18}$$

In contrast to the work of this thesis, the specifics of the case solution is limited to just the type of visualization and more complex features such as axis placement, series and labels are not considered. In terms of features, they distinguish between text and tabular datasets and extract different features for each, for text it includes the total and unique number of terms and for tabular they extract te number of textual columns, number of numeric columns and number of data points. In the case of numeric they extract the maximum, minimum, average and standard deviations of the columns.

For the Similarity function they use the metric shown in 3.19 which is a simple weighted relative difference between features, in this case, between the number of textural and numeric columns and rows between the target and the case.

$$sim(c_t, c_i) = 1 - \sum_{f \in col_txt, col_num, rows}^{n} w_f * \frac{|c_t(f) - c_i(f)|}{\max c_t(f), c_i(f)} \tag{3.19}$$

The results showed an high accuracy for the Case Based Reasoning (CBR) recommendation strategy. CBR outperforms all other techniques on their test set.

[1]http://manyeyes.alphaworks.ibm.com

Unfortunately the findings are not based on real-user data nor it has been tested on live users, it is also limited to just the type of visualization, however they acknowledge this by suggesting the next steps for a more sophisticated CBR such as the use of semantics in the case representation or adaptation.

Automating the design of graphical presentations of relational information

A widely cited work from Jock Mackinlay from 1986 (Mackinlay, 1986) that studies the development of an application-independent tool that automatically design graphical presentations, such as bar charts, scatter plots and connected graphs) of relational information . Two problems were raised during his work, the encoding of graphic design criteria in a form that could be used by the tool and the generation of wide variation of designs that the tool could use to accommodate a wide variety of information. The algorithm created is based on a divide-and-conquer strategy, it has three steps: partition, selection and composition. The research applies artificial intelligence techniques and uses a logic programming approach where, for example, the equation 3.20 describes part of the tool bar chart rule that state that the relation must be a dependency from a non-numeric set to a numeric set. The condition limits the number of bars due to making the presentation difficult to read.

$$\begin{split} rel = x \rightarrow y \wedge \neg Numeric(x) \wedge Numeric(y) \wedge Cardinality(x) < 20 \ldots \\ \Rightarrow Presents(barchart, rel) \end{split} \tag{3.20}$$

This research described some techniques and frameworks for the development of tools that are capable of automatically design graphical presentations for a wide variety of information. A challenge of this approach is the complex logic programming.

Automated generation of graphic sketches by example

Unlike rule-based systems that couldn't address some challenges in automated generation, this work by Michelle X. Zhou and Min Chen (Zhou and Chen, 2003) from IBM, Watson Research Center was one of the first to apply Case-Base Reasoning . Their motivation was how time-consuming and skilful one how needs to be to create effective visual presentations and the many challenges that previous systems couldn't cope with.
In their approach they used a database of existing graphic examples as cases to automatically create presentations, their approach has three main contributions. The first is an augmented similarity metric with an evaluation criteria to retrieve most similar cases with a case/request decomposition when a usable case is not found. The second step is speed improvements by organizing cases into hierarchical clusters based on their similarity distances and by using selected cluster representatives. The third contribution is an adaptation technique that creates a solution based on multiple cases.

The case base works as a labelled graph structure described in XML that expresses the mapping between data hierarchy and visual hierarchy. Within each hierarchy, a node is described by a set of features. There are parent-child relations and data/visual node relationships. A user request therefore is also described as a similar case representation.
For the similarity metric it tries to match the overall graph structure by matching its nodes and attributing a distance. For the top-matched case it must be tested for inadequacy using a number of criteria since the case may not produce a good result.

The ideas presented in this work are innovative but have a complex implementation that requires an unusual structure for the case base which makes it hard to adapt to other contexts and problems.

Task-analytic approach to the automated design of graphic presentations

This work by Stephen M. Casner from 1991 (Casner, 1991) explores automated graphic design and presentations based on an analysis of the task which a graphic is intended to support . The system developed allows the user to replace demanding logical inferences with simple perceptual inferences. Perceptual inferences such as color and size determinations allow the users to obtain the same information as more demanding logical inferences such as numerical comparisons and mental arithmetic. A key feature of the approach is that it is able to design different presentations of the same information depending on the requirements of different tasks.

Not all works try to automate the complete process, there are relevant work that tackles specific challenges within the automatic process of creating charts and visualizations, some of those are presented here:

Selecting Semantically-Resonant Colors for Data Visualization

In this work by Sharon Lin et al. (Lin et al., 2013) an algorithm that automatically selects semantically-resonant colors is presented . This algorithm makes it possible to attribute the color *blue* for data about *oceans* or *pink* for *love*. Values are mapped to colors by collecting representative images, analysing the color distribution to determine a score and choosing an optimal assignment. The score takes into account how well it discriminates among data values. The results from a controlled study showed that semantically-resonant colors improved speed on chart reading tasks.

3.3.2 Automated Data Mining

Data mining is a complex task, it is typically applied manually (Campos et al., 2009), in recent years there has been much discussion on the possibility of automating data mining (e.g. on online communities [2] or in literature) but even though there are many challenges, there has been some work on this field, even if within certain limitations or for specific tasks.

Data-Centric Automated Data Mining

An highly relevant work by Marcos M. Campos et al. (Campos et al., 2009) from Oracle that approaches the automation of data mining using a data-centric focus and automated methodologies to make data mining accessible to non-experts . The approach is based on two main ideas:

- Data-centric focus
- Process automation

[2]http://www.dataminingblog.com/can-we-automate-data-mining/

The data-centric focus can be considered the main contribution but process automation is what enables good usability. The goal of this work was not to provide better results than those creating by data mining experts but allows users to achieve reasonable results with minimum effort.
One of the requirements for the approach is given a data source the user performs computations on the data, this is data-centric as it is restricted to the data - traditional data mining approaches have models as key entities and they are not tied to a data source, something that this approach considers complex - the attributes in the data should match the attributes in the model.
The type of data mining hides all the details of methodology and works through high-levels task that are goal oriented such as:

- *PREDICT* - creating predictions for classification and regression
- *EXPLAIN* - identify attributes that are important
- *GROUP* - cluster similar records together
- *DETECT* - identify records that are atypical, outliers or anomalies
- *MAP* - project data to a lower dimension

To make this work the automation of the data mining process requires different processing stages, the following steps were used:

- **Computation of statistics** in this steps information about the number of attributes, records and features like range of values is gathered.
- **Sampling** this is used to improve build times with large datasets.
- **Attribute data types** considering some algorithms cannot work with some type of data (e.g. Support Vector Machines can't work with categorical data) and other treat types differently (e.g. CART) an algorithm is needed to identify types.
- **Select Attributes** elimination of irrelevant attributes help the performance. There are methods that remove attributes before a model is built and methods that build multiple models each with a different combination of attributes. The former is the approach they used.
- **Select algorithm** depending on the characteristics of the data, it requires the use of different techniques to achieve good results. Choosing between a classification or regression may depend on the target data type for instance. The approach used takes into consideration the nature of the data to select the algorithm.
- **Transform data** Considering most algorithms require a transformation of some sort, common transformations include: binning, normalization, missing value imputation and outlier removal. This work applies these techniques according to the data type, attribute value range, cardinality and percentage of of missing values. There's also a careful planning of sequences, for instance, outlier removal before missing value replacement.
- **Select model and assess quality** model selection requires evaluation of the models from different algorithms or parameters, and for this they use a simple test set or cross validation. The model with the best score is chosen and a reliability measure of the results is also calculated.

- **Generate output** in some tasks, explanations or descriptive information is provided.

The proposed high-levels tasks and data-centric approach are good contributions to develop an autonomous data mining system and it's a good overview of the workflow needed to create this process, but the implementation itself is made specifically to work on the Oracle ecosystem therefore the procedures used to tackle the various challenges in the tasks can't be easily transferred to other contexts.

Automated model building and evaluation for data mining system

Another relevant approach to the work of this thesis, made by Burton et al. (Bloom et al., 2003), in a way similar to the previous, this work describes a method that automatically generates data mining models, compromises the following steps:

- Receiving information
- Generate a model
- Preparing data source for training and testing
- Evaluation of the built model
- Selecting a model
- Outputting the model

Considering the similarity between approaches, the most important thing to take from this work is the selection criterion used to choose a model, which is described in Equation 3.21:

$$score = \frac{w * (number\_of\_correct\_positives)}{w * (number\_of\_actual\_positives)} + \frac{number\_of\_correct\_positives}{(w+1) * (number\_of\_actual\_negatives)} \tag{3.21}$$

Where the *score* is the value attributed to each model and the model with the largest value is chosen, and w as a ratio of *"a cost of predictions that are all correct except for a portion of negative predictions being false negatives, to a cost of predictions that are all correct except for a percentage of positive predictions being false positives."* A similar equation must be developed to create a selection criterion to autonomously chose models in the work of this thesis.

Similarly to what happens within the *automatic visualization* not all works try to automate the complete process, some relevant work tackles specific challenges within the automation of data mining, such as:

Generalized Feature Extraction for Structural Pattern Recognition in Time-Series Data

This approach on structural pattern recognition by Robert T. Olszweski (Olszewski, 2001) mentions some of the challenges in hand-made feature extraction: time-consuming, inexact

and requires domain knowledge. The use of *structural detectors* removes this limit allowing the creation of solutions with complex and poorly-understood domains.
To address the problem of generalized feature extraction in a domain involving time-series, structure detectors from literature were developed.
The classification results with these structure detectors compared to statistical feature extractors were at least as good and often superior therefore showing it's possible to use general structure detectors and achieve reasonable results, comparable to hand-made features from domain experts.

Automatic recommendation of classification algorithms based on data set characteristics

Choosing an appropriate classification algorithm autonomously is a very relevant task for this thesis. In this work by Qinbao Song et al. (Song et al., 2012) they propose a method that extracts a feature vector from the data set using structural and statistical information and an evaluation is performed on the data using various classification algorithms. From the results a new vector is extracted and its *k* nearest data sets are identified. The results showed that the proposed method is effective and can be put to practice.

Automatic subspace clustering of high dimensional data for data mining applications

Automating clustering algorithms has to deal with issues of being able to handle high dimensional data, large amount of data and achieve good results. In this work by Rakesh Agrawal et al. from IBM Research Center (Agrawal et al., 1998) , it's presented CLIQUE, an algorithm that attempts to satisfy these challenges. The results shows it finds accurate clusters in large high dimensional datasets, it also has good scalability. The results shown that when compared to other algorithms designed for clustering in full dimensional space such as BIRCH or DBSCAN the accuracy is similar.

Parameters Selection of Hybrid Kernel Based on GA

While Support Vector Machine made significant achievements in pattern classification and regression estimation, two problems exist: the selection of a kernel and the selection of the parameters. In this work by Y. Y. Meng (Meng, 2009) it's proposed a method to select parameters by using genetic algorithms. The results showed improvements on the score and speed when compared to grid search.

3.4 Libraries and Resources

This section describes resources and libraries that can be used in the work of this thesi. One of the requirements defined by *Wizdee* is that these libraries and resources have a commercial use license and must be open source, without fees, which means not all libraries here described can be used, but are still relevant due to their popularity. Examples of allowed licenses are LGPL[3], BSD License[4], Apache License[5] and MIT[6], on the other hand

[3]`https://www.gnu.org/licenses/lgpl.html`
[4]`https://www.freebsd.org/copyright/license.html`
[5]`http://www.apache.org/licenses/LICENSE-2.0.html`
[6]`http://opensource.org/licenses/MIT`

licenses similar to the GPL [7] are not valid since they force the user into publishing the source code. Considering most of the systems in *Wizdee* are developed in Java, libraries developed in Java are preferred but libraries in Python are also accepted as long as they provide some advantage over their Java counterparts.

3.4.1 CBR

There are tools and resources available to develop CBR applications, that help define an architecture to design CBR systems with components required to build them. This is a list of some of the most popular ones:

jCOLIBRI

jCOLIBRI[8] provides a reference platform for developing CBR applications. The architecture has been designed to be extensible and reusable across different domains and CBR families. It includes a persistence layer, in-memory organization of the cases and retrieval methods. Reusable and revision methods and evaluation methods. The source code is released under LGPL.

myCBR

myCBR[9] is an open-source similarity-based retrieval tool and software development kit. With myCBR Workbench you can model and test knowledge-intensive similarity measures in a powerful GUI and easily integrate them into your own applications using the myCBR SDK. It has the advantage of assigning several similarity measures to each attribute. The license for this resource is GPL.

CBR*Tools

CBR*Tools[10] is an object-oriented framework. It aims to be modular, reusable and extensible. It integrates hot spots and is based on use cases to structure hot spots. In contrast to the other resources it lacks many of the features existing in other resources, it offers a set of abstract classes to model the main concepts necessary to develop applications and offers some traditional methods such as closest neighbours indexing and standard similarity measures. No license information is publicly available.

From the libraries available, jCOLIBRI is by far the one that offers the most, from the type of license to the features available. With that said, even that library suffers from an overly complex implementation that makes the flexibility this projects requires (e.g. specialized retrieval and similarity functions) an arduous task therefore removing the advantages of using a library.

[7] `http://www.gnu.org/copyleft/gpl.html`
[8] http://gaia.fdi.ucm.es/research/colibri/jcolibri
[9] http://mycbr-project.net/
[10] http://www-sop.inria.fr/axis/cbrtools/

3.4.2 Machine Learning

In contrast to resources related to case-base reasoning, there are many libraries concerning machine learning, using various license types and for different architectures. Only the most popular ones were chosen since, especially in Python where there's a wide range of available resources[11].

Mallet

Mallet[12] is a Java based package for statistical natural language processing, document classification, clustering, topic modeling, information extraction and other machine learning applications. In addiction to ML applications it includes routines for transforming text documents into numerical representations.

Apache Mahout

Apache Mahout[13] provides a scalable machine learning library for reasonably large data sets. Some of its core algorithms (clustering, classification and batch based collaborative filtering) are implemented above Apache Hadoop using the map/reduce paradigm, which allows writing applications that rapidly process vast amounts of data in parallel on large clusters of compute nodes. With that said, it still allows applications to run on a single node or on a non-hadoop cluster, because the core libraries are highly optimized for good performance in a non-distributed algorithms. It has a large community of developers and users that facilitate support, discussions and further development.

Java Data Mining Package (JDMP)

Java Data Mining Package[14] is an open source java library for data analysis and machine learning. It provides easy access to data sources, various ML algorithms such as clustering, regression, classification and optimization and other visualization modules. It has a matrix libary that allows storing and processing any kind of data, handles huge matrices even when they not fit into memory. It provides interfaces for importing and exporting data for JDBC data bases and files (e.g. TXT, CSV, Excel). Besides proving its own algorithms, it provides interfaces to other packages such as Weka, LibSVM, Mallet, Lucene and Octave. The data processing methods are separated from data sources, so algorithms and data may be distributed over several computers, which allows parallel processing. This package is still in early development which means it may not be suitable for production use.

Java-ML

Java-ML[15] is a library that contains a collection of ML algorithms and related regression, data pre-processing, ensemble learning, voting and others. It works by providing a common interface for each type of algorithm. It doesn't have a GUI so it's clearly aimed to engineers and programmers. It has a vast documentation but has a GPL license that doesn't allow it to be used in commercial software.

[11] `https://www.cbinsights.com/blog/python-tools-machine-learning/`
[12] `http://mallet.cs.umass.edu/`
[13] `http://mahout.apache.org/`
[14] `http://www.jdmp.org/`
[15] `http://java-ml.sourceforge.net/`

Weka

Weka[16] is a collection of ML algorithms that can be applied directly to a dataset by code or can be used in a GUI. It provides tools for data pre-processing, classification, regression, clustering, association rules and visualization. It is also well-suited for developing new machine learning schemes but its license is GPL and it doesn't allow commercial use.

scikit-learn

scikit-learn[17] is an open source machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, logistic regression, naive Bayes, random forests, gradient boosting, k-means and DBSCAN and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Statsmodel

Statsmodel[18] is a Python module that allows users to explore data, estimate statistical models, and perform statistical tests. An extensive list of descriptive statistics, statistical tests, plotting functions, and result statistics are available for different types of data and each estimator. It includes linear regression models, predictive modelling and functions for time series analysis and plot functions.

Gensim

Gensim[19] is a module implemented in C++ with an interface to Python. It is designed for large-scale learning for a broad range of feature types and learning settings. It offers a considerable number of machine learning models such as support vector machines, hidden Markov models, multiple kernel learning, linear discriminant analysis, and more.

From these libraries, there were some that stand out because of the features they offer. Weka would be the best library in terms of features for Java, unfortunately their license is incompatible with commercial work. Apache Mahout also offers some functionalities, however, it still pales to the libraries available in Python. scikit-learn is a widely used library that has support for many types of algorithms used in machine learning (clustering, classification, regression, feature extraction, evaluation). It also has the advantage of a detailed documentation, big community, recent updates and a compatible license. Thus, scikit-learn is the main library used, however, other libraries are being used for specific tasks such as Apache Mahout and Statsmodel.

[16]`http://www.cs.waikato.ac.nz/ml/weka/`
[17]`http://scikit-learn.org`
[18]`http://statsmodels.sourceforge.net/`
[19]`http://radimrehurek.com/gensim/`

Chapter 4

Competitor Analysis

There are a few products in the market that attempt to tackle either Automatic Visualization, in chapter 4.1, or Automated Data Mining, in chapter 4.2. In this section some of these products are briefly analysed. Most of these products do not have a detailed description of their approach available so the focus in this section regards the products features.

4.1 Automatic Visualization

Although there are many products that help users create visualization, only a few are able to automate the process in any way. Here we describe some of the most popular products that are able to create a chart without guidance from the user. When a demo was freely available, a very simple test was made to check the automation capabilities, using the following data:

Sales	**Country**
30	*Portugal*
10	*USA*
15	*Spain*
18	*France*

(4.1)

4.1.1 Infogram

Infogr.am[1] is a web application that creates infographics or charts, it is real-time, interactive and a carefully designed application, with 30 chart types and other types of objects, such as videos, images and maps. It also has saving and sharing capabilities and it allows to have live data and Google Drive integration. It has the capability to simply upload a file (e.g. *CSV* or *XLS*) and generate a chart.

Using the previous dataset4.1 as a *CSV* file:

- The process of uploading a file is straightforward, unfortunately it couldn't generate a chart with the *raw* data.
- It requires the data to be in a specific format for the chart to be generated, where the *axes* and *values* are explicit.

[1]`http://infogr.am/`

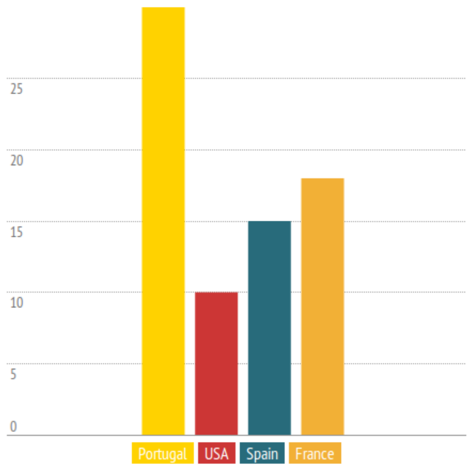


Figure 4.1: Charts generated with Infogram after adjusting the dataset.

- It requires the user to specify a chart type without having the option for the system to decide the best one.
- In case of duplicated values it simply duplicates the column and in case of many entries it attempts to load all of them causing slow downs and unreadable data.

4.1.2 Piktochart

Piltochart[2] is a web-based real-time and interactive infographic generator that also has capabilities to generate charts, it has good usability with simple steps that allow a non-designer or analyst to create useful visualizations. It has the option to directly import files, such as *XLS* or *CSV*, and generate charts from the imported data. It also has the option to customise the results.

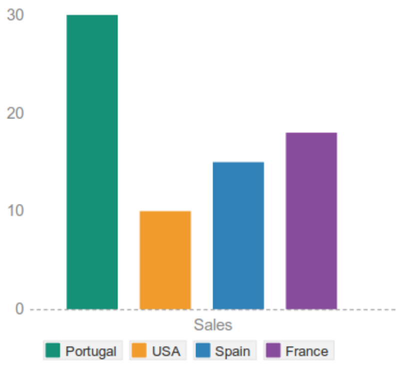


Figure 4.2: Charts generated with Piktochart after sorting the dataset columns.

Using the previous dataset4.1 as a *CSV* file:

- The process of uploading a file is straightforward and it was able to read the data correctly, however it required the first column to work as the *series*.

[2]`https://magic.piktochart.com/`

- It requires the user to specify a chart type without the option for the system to decide the best one, but it has the ability to correctly filter impossible charts and adapt when a new type of chart is chosen.
- In case of duplicated values it simply duplicates the column and in case of many entries it attempts to load all of them causing slow downs and unreadable data.

4.1.3 Watson Analytics

Watson Analytics[3], from IBM, is a natural language-based cognitive service. It was born out of the motivation that most analytical tools require a certain level knowledge and so Watson Analytics is designed to make advanced and predictive analytics easy to use for anyone. Unlike analytics offerings designed for data scientists and analysts focused on visualization, IBM Watson Analytics automates steps like data preparation, predictive analysis and visual storytelling for businesses across various disciplines such as marketing, sales, finance or human resources. Their predictive analytics automatically obtains the most relevant facts and attempts to uncover unforeseen patterns or relationships.

Using the previous dataset4.1 as a *CSV* file:

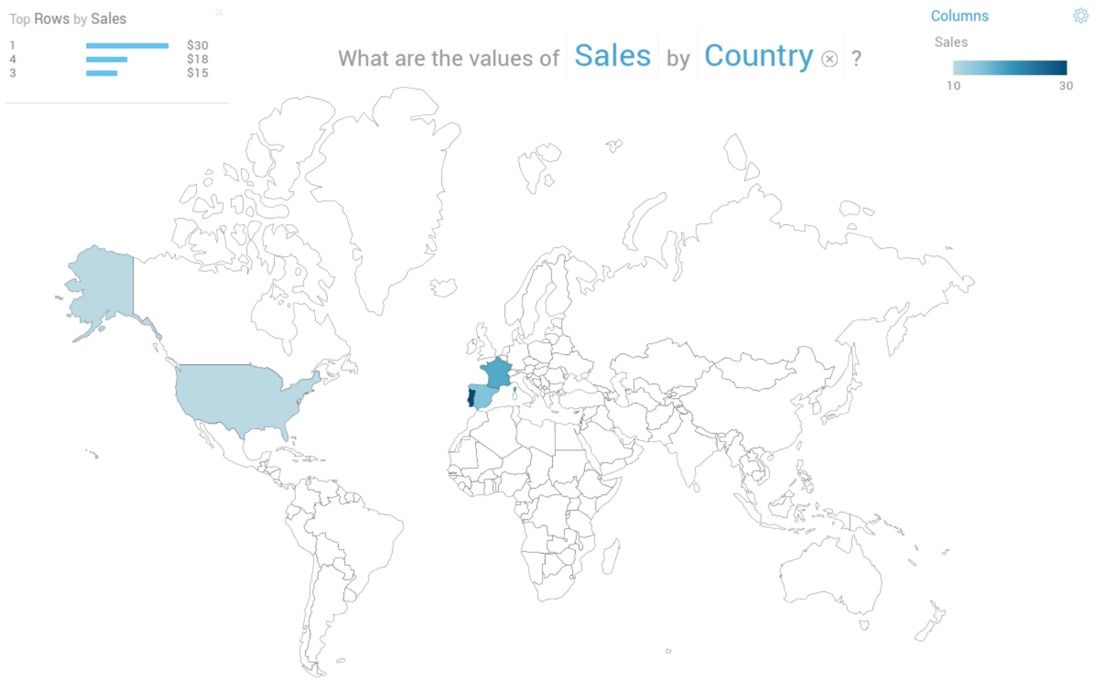


Figure 4.3: Chart generated with Watson automatically from the *raw* file.

- The process of uploading a file is straightforward and it was able to read the data correctly without any modification, it also has an useful *data quality* analysis to let the users become aware of data problems early on the process.
- It understands the type of data provided and gives highly relevant types of charts - by showing a map when talking about *countries* and by assigning the dollar sign to the *sales* column.

[3]http://www.ibm.com/smarterplanet/us/en/ibmwatson/

- It automatically adapts the data to other chart types and successfully reads the data and generates a chart.

- In case of duplicated values it automatically *aggregates* the values - in this case with a *sum* operation - and in case of many entries it creates a window mechanism where only a n number of entries are shown.

4.1.4 Charted

Charted[4] is an open sourced tool created by Medium[5]. It automatically tries to create a chart based on the data it receives, it does not store any data but attempts to fetch the data every 30 minutes. It does not transform or manipulate the data and displays only and exactly what it receives, adjustments must be made before sending the file. It's made with simplification in mind, it doesn't have many features and it *"focuses on getting from the data to the visualization with the fewest decisions possible"*. It currently only supports two types of charts: *line chart* or *stacked column chart*.

Using the previous dataset4.1 as a *Google Drive* spreadsheet:

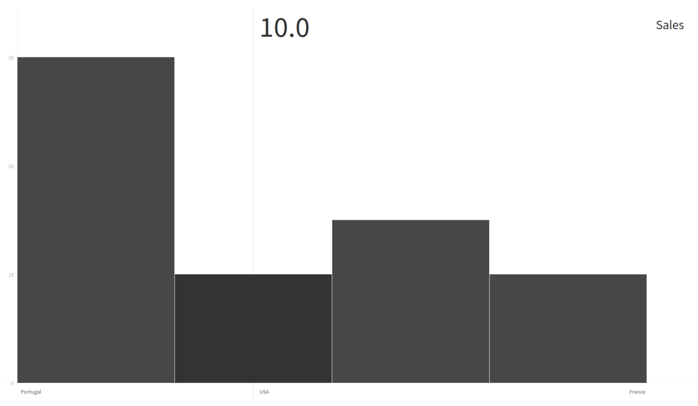


Figure 4.4: Chart generated with Charted after sorting the dataset columns.

- The process of uploading a file requires access to Google Drive or Dropbox and although it could not generate a chart with the *raw* data, it managed to read the labels correctly. Changing the columns order fixed the data reading problem.

- A bar chart was generated but there were some issues relating each country to their respective values - After making specific changes it was able to generate a correct chart.

- It requires the user to specify a chart type, without the option for the system to decide the best one.

- In case of duplicated values it simply duplicates the column and in case of many entries it attempts to load all of them causing slow downs and unreadable data.

[4]`http://www.charted.co/`
[5]`https://medium.com/`

4.2 Automated Data Mining

There are not many products that can automate data mine or apply machine learning algorithms without an expert user, but here we describe some of the popular products.

4.2.1 Watson Analytics As described in 4.1.3, Wason Analytics[6]

from IBM does not only has automatic visualization capabilities, but is also able to automatically analyse data. It has prediction features, where it attempts to find patterns or insights hidden in the data, some important features is the ability to find *predictors* for a certain attribute or what *influences* another.

4.2.2 Oracle Predictive Analytics

Predictive Analytics[7] is a technology from Oracle that captures data mining processes in routines - also known as *"one-click data mining"*. Its purpose is to automate the data mining process. Although predictive analytics uses data mining technology, knowledge about is not needed. It can be used by specifying an operation to perform on the data and it works by analysing the data and creating the mining models. These models are trained and tested and then used to generate the results for the user. There's also an *Add-In* that works within a Microsoft Excel spreadsheet. The possible operations are:

- Regression
- Classification
- Feature Selection and Extraction
- Anomaly Detection
- Clustering
- Association

4.2.3 Microsoft Azure Machine Learning

Azure Machine Learning[8] is made for people without deep data science backgrounds to start mining data for predictions. It uses a simple drag-and-drop gestures and data flow graphs to set up experiments. For most cases, not a single line of coded is needed but it also has support for seasoned data scientists with R - a popular open-source programming environment for statistics and data mining. The algorithms provided have been used by Microsoft in products such as *Bing* or *Xbox*.

Although it is possible to apply a learning algorithm with ease, the process is not fully automated, for instance the process of evaluation is not automated, neither is the algorithm

[6]`http://www.ibm.com/smarterplanet/us/en/ibmwatson/`
[7]`http://docs.oracle.com/en/`
[8]`http://azure.microsoft.com/en-us/services/machine-learning/`

chooser or feature selection - however it still hides many of the low-level concepts and in-depth knowledge is not required.

Chapter 5

Approach

Getting useful insights from the data or receiving effective visualizations automatically (without the user intervention), when dealing with complex information, are part of what makes a smart and intuitive system.

Although this work has a very large component of research, it is oriented for a real product, therefore it must follow some typical software engineering standards. In the following section, the requirements, in section 5.1, are described, giving a summary of what is expected through functional and non-functional requirements. Finally, an overview of the platforms architecture is described in section (5.2 and its components are explained in section 5.3.

5.1 Requirements

Requirements have a critical role in the development a project, as they define the guidelines of the final product. They can also directly impact the system architecture right from the start by establishing an overview of the needs. In this section, functional (5.1.1) and non-functional (5.1.2 and 5.1.3) requirements are described.

5.1.1 Functional Requirements

These are the top-level functional requirements, they describe the specific functionality that the system is supposed to accomplish. The first one, Automatic Visualization, consists on developing a system that can decide on its own the most appropriate visual representation for the results retrieved, upon a search performed by the user.

The second high level requirement, Automated Data Mining, provides the system with the capability of analysing information in an autonomous way, by applying the most appropriate machine learning algorithm with automatic pre-processing, evaluation and parameter optimization. The table 5.1 specifies each top-level requirement.

ID	**Name**	**Description**
FR.01	Automatic Visualization	The System must automatically decide the most appropriate visual representation based on the search results.
FR.02	Automated Data Mining	The System must analyse available information and provide the ability to discover patterns, predictions or simply new knowledge

Table 5.1: Top level requirements

Automatic Visualization

The following requirements represent all the fine-grained requirements needed to accomplish the functional requirement Automatic Visualization. They reflect the approach needed to analyse the structure of the data and develop a system capable of finding similar cases and use them to create new solutions.

The table 5.2 specifies each one of the Automatic Information Visualization related requirements.

ID	Name	Description
FR.01.01	Information Pre-processing	Information needs to be pre-processed due to dirty data (e.g. null values).
FR.01.02	Metadata Extraction	The system must extract data types and other relevant metadata if those exist.
FR.01.03	Case Representation	The system must be capable of representing a problem-solution pair (case) and store it.
FR.01.04	Default Cases Creation	The system must have pre-defined cases based on defined standards, best practices or relevant examples.
FR.01.05	Case Retrieval	Given a problem the system must retrieve relevant cases to solve it.
FR.01.06	Case Reuse	The system must be capable of mapping a new solution based on similar cases.
FR.01.07	Case Revise	Given a reused case the system must be capable of evaluation and revision.
FR.01.08	Case Retain	The system must store reused cases as new cases if they have been successful against the problem.
FR.01.09	Case Base Manual Modification	The system must allow a technical user to modify cases.

Table 5.2: Automatic Information Visualization Requirements

Automated Data Mining

The following requirements are for the various types of analytic tasks: Data Forecast, Data Clustering, Outlier Detection, and Data Classification. The table 5.3 specifies each one of the Automated Data Mining related requirements.

The table 5.4 specifies the overview process required for each of the data mining task requirements:

ID	Name	Description
FR.02.01	Target Table Identification	The System must allow a technical user to specify the relevant target tables with an UI.
FR.02.02	Outlier Detection	The System must choose the most appropriate outlier detection algorithm and find unusual data.
FR.02.03	Data Forecast (Time Series Analysis)	The System must can choose the most appropriate forecast algorithm and forecast new data with the least amount of error.
FR.02.04	Data Clustering	The System must choose the most appropriate cluster algorithm and group data into coherent clusters.
FR.02.05	Data Classification	The System must choose the most appropriate data classification algorithm and ouput the rules that define the target variable.
FR.02.06	Search Integration	The system must allow on-demand analysis using search operators.
FR.02.07	Specific Industry Solutions	The System must use the data mining framework to create industry specific solutions.

Table 5.3: Automatic Information Visualization Requirements

ID	Name	Description
FR.02.03.01	Information Pre-processing	The Information must be pre-processed due to dirty data (e.g. null values).
FR.02.03.02	Feature Learning and Selection	The System must capable of learning abstract features from the raw data when required and/or select relevant features for use in the model construction.
FR.02.03.03	Feature Extraction	The System must be able to reduce dimensionality, if needed.
FR.02.03.04	Model Chooser	The system must choose the appropriate model for the task.
FR.02.03.05	Model Creation	Use chosen algorithm to create models.
FR.02.03.06	Parameter Optimization	The system must optimize the algorithm parameters if they exist.
FR.02.03.07	Model Evaluation	The System must evaluate the created models.

Table 5.4: Automatic Model Creation Requirements

5.1.2 Technological Requirements

Technological requirements are non-functional requirements that relate to the technological aspects, such as programming language, that the implementation must follow. The table 5.5 specifies each technological requirement.

ID	Name	Description
TR.01	Programming Language	All source code must be written either in Java or Python.
TR.02	Licenses	All the licenses of software development, tools and resources must be open source and allow commercialization without fees. (e.g. Apache, MIT).

Table 5.5: Technical requirements

5.1.3 Performance Requirements

Performance requirements are non-functional requirements that represent performance-related aspects that the system must fulfil. Due to the real-time nature of the process, the response times are of utmost importance. Considering that performance depends on a wide range of factors (e.g. choice of database, number of requests, size of the database, type of request), it is important to define the scope of the performance requirements. For the performance requirements, only the time between the request reaching the specific handler and obtaining an answer is considered, excluding database accesses. The basic advice regarding response times has stayed similar throughout the decades, finding similar considerations in (Miller, 1968), in (Nielsen, 1993) and again in recent research on usability (Nielsen, 2014). From that research we can gather that:

- **0.1 second:** limit to make the user feels that the system is responding instantaneously.
- **1.0 second:** limit to make the user's flow of thought stays uninterrupted.
- **10 seconds:** limit to keep the user's attention focused.

The original proposed time limit was *5s* to apply the models, for the task of Automated Data Mining, however, during development, it has come to attention that while it is possible to obtain reasonable models within the time limitation, it reduces the flexibility of the system, since the training time was not being considered in the previous requirements and a the request of being able to train models on-demand was raised.

So, due to high interest in having the power to create models on-demand, it has been discussed and decided that it is preferable to have slightly longer time limits, even if it goes over 10 seconds, because it enables specific important platform features. Therefore, the new time limit is *30s*, with the previous *5s* being considered an ideal time since it would allow us to keep the user attention.

The next table 5.6 specifies each performance related requirement.

ID	**Name**	**Description**
PR.01	Automatic Visualization Response Time	The system response time to decide the best visualization must be under 5s on average, this excludes any secondary response time such as loading the page or database retrievals.
PR.02	Automated Data Mining Response Time	The system maximum time to train and apply a model must be under 30s on average, this excludes any secondary response time such as loading the page or database accesses.

Table 5.6: Performance requirements

5.2 Architecture

In this section, an overview of the product architecture is presented and described. There are three high-level layers: the *Data Layer*, the *Business Layer* and the *Presentation Layer*.

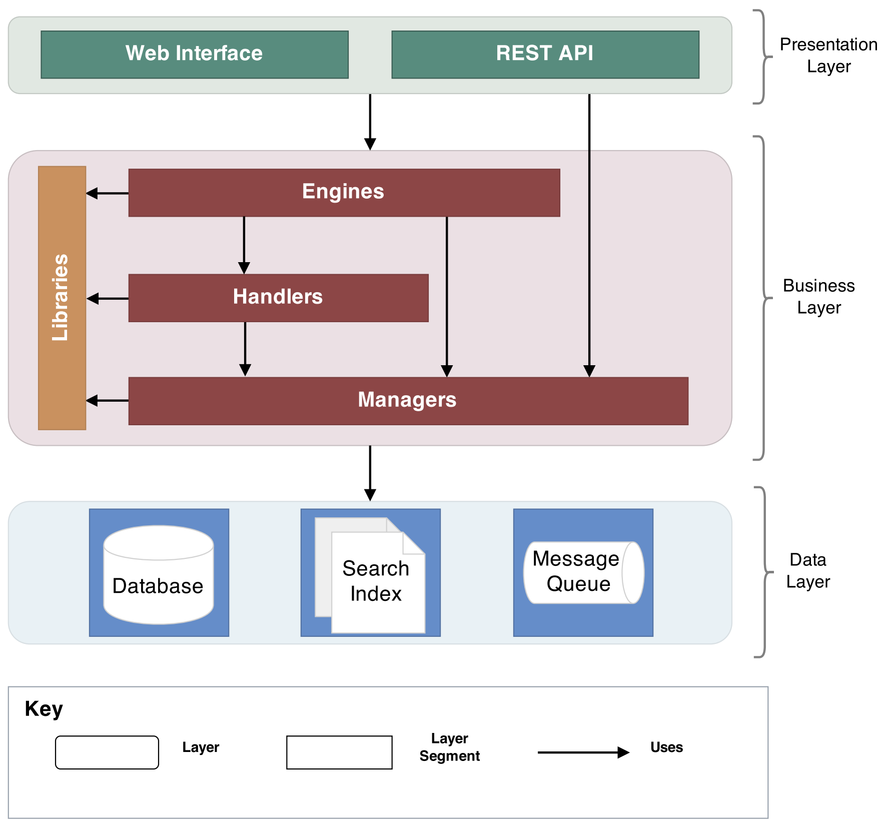


Figure 5.1: Diagram of the high-level layers of the platform.

5.2.1 Presentation Layer

The presentation layer provides the front-end, management interfaces and external APIs. There are two distinct layer segments working on the Presentation Layer:Web Interfaces and REST APIs.

Web Interfaces

Web Interfaces are applications that allow the interaction between a user and the software running on the web server - for both the managers and clients.

REST APIs

The system has many APIs that allow integration and communication with internal or third party systems. These APIs follow the REST[1] software architecture style which is a simpler alternative to SOAP and WSDL web services (Fielding and Taylor, 2002).

5.2.2 Business Layer

The Business Layer is where all the logic of the system is implemented, it has several layer segments that interact and communicate with each other.

[1]Representational State Transfer

Managers

The Manager components are responsible for the interaction with the Data Layer, they execute all the procedures related to persistent storage - CRUD[2]. The motivation behind them is to abstract the process of accessing the data, so that the persistence of data can be transparent to the other components. In the case of some managers, there are also APIs that provide access to certain features.

Handlers

Handlers have the objective of processing information and build responses from it. When the system makes a request, it can go to a specific handler (in certain cases more than one) to process the information and generate an answer (one per handler). This process is orchestrated by an engine as it invokes the intended handlers. Handlers can handle complex logic with procedures such as receiving a *query* and returning an analysis of the query, or receiving a scenario[3] and returning a visualization.

Engines

Engines aggregate and orchestrate Handlers, they are responsible for the coordination of the answer generation for a given request. For this to happen, they contain logic behind answer selection, session management and logging.

5.2.3 Data Layer

Database

The system relies on a typical Relational Database to store the large amounts of data in the system. The chosen database management system is PostgreSQL[4], it is ACID[5]-compliant, supports transactions, concurrency, it is free and open source. Widely used in the industry in many internet-facing applications with numerous concurrent users.

Search Index

One of the main characteristics of Wizdee is its natural language querying support, and this requires *text search* which is a time consuming task in common relational databases. So, the system uses Apache Solr[6] which is a fast and open source search platform built on top of Apache Lucene[7]. It provides the capabilities needed to have full-text search with high volume traffic - due to the very efficient indexing.

Message Queue

The Message Queue is a critical component used to communicate between two or more components asynchronously, which enables the producers to simply leave tasks on the queue

[2]Create, Read, Update and Delete
[3]an internal representation of a request
[4]PostgreSQL
[5]Atomicity, Consistency, Isolation, Durability
[6]Solr - http://lucene.apache.org/solr/
[7]Lucene - http://lucene.apache.org/

until a consumer executes them. The messages in this system are durable for reliability reasons. The chosen platform is Apache ActiveMQ[8], which is an open source message written in Java. It is being used to synchronize the data from external data sources into the internal system, in other words, it is used in the data importing process.

5.3 Components

In this section, an overview of some of the architecture components of the system are described. First, some relevant components are described in section 5.3.1 and the components developed are described in section 5.3.2. The relationship between these components is depicted in figure 5.2.

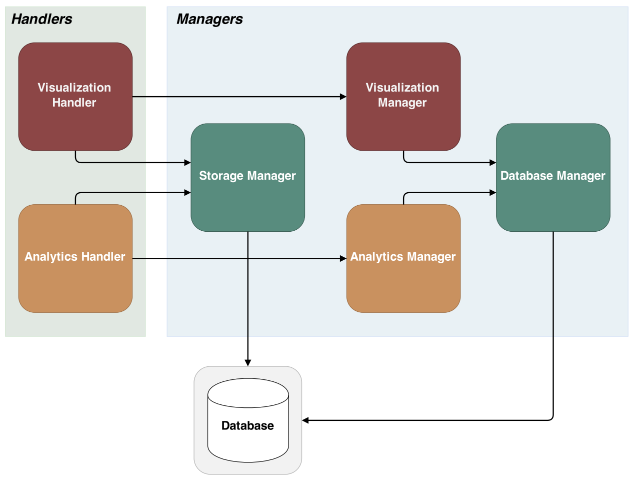


Figure 5.2: Diagram of the relationship between relevant components.

5.3.1 Relevant Components

In this section all the existing components that are relevant to this work are briefly described. Considering the type of work required, the components are quite self-contained, without interacting with many of the other existing components. The dependencies are mostly focused on data related components including the: Database Manager and the Storage Manager.

Database Manager

The Database Manager is responsible for abstracting many tasks related to using the database, such as giving a connector from a pool of connections or setting up a transaction. This manager is needed whenever other managers or handlers want to execute a command in the database.

Storage Manager

The Storage Manager abstracts all the tasks that must be performed on the data that is queryable by the user. While the Database Manager provides access to an operational

[8]http://activemq.apache.org/

data need by the system, this manager provides access to analytical data that is used to build reports and visualizations.

5.3.2 Components Developed

In this section we will give a brief overview about the components to be developed.

Visualization Manager

Because we will be using a Case Based Reasoning (CBR) approach for the Automatic Visualization, this manager has the main objective to implement the needed procedures and commands to retrieve and store cases from the case base - this doesn't include the process of finding the most similar cases. For this, the manager must obtain a connection from the Database Manager, as seen on the figure 5.2. The cases have a specific representation and that representation must be stored correctly, so that it can be retrieved as expected. This means that this manager must deal with serialization[9] and deserialization[10] of the case base. The serialization must also work so that it is machine independent.

Visualization Handler

This handler must deal with most of the logic required to accomplish the Automatic Visualization features. It requires access to the Storage Manager, to retrieve the data used to generate a visualization.

Analytics Manager

The Analytics Manager has the main objective of creating procedures to retrieve and store models created during the Automated Data Mining process. It has to deal with serialization and deserialization as well, but in some cases the tools and resources used already provide mechanisms to do these tasks.

Analytics Handler

This handler provides the logic required to automatically create a machine learning model. It requires access to the Storage Manager, to retrieve the data needed. It communicates with the Analytics Manager to retrieve and store the models. It implements processes capable of feature learning, selection and extraction, model creation, parameter selection, parameter optimization and evaluation mechanisms.

Machine Learning Library

This library is responsible for the communication with the machine learning service component. It implements the needed protocols for communication, serialization and deserialization.

[9]The process of converting objects into a format that can be stored and transmitted with the purpose of saving the state of that object and be able to reconstruct it.

[10]The reverse process of serialization

Chapter 6

Methodology and Planning

The methodology used in this work is presented and described in section 6.1 and a brief summary for the planning of the internship in section 6.2.

6.1 Methodology

This work follows the adopted methodology by *Wizdee* - the SCRUM (Schwaber, 1997) development methodology - which is an iterative and incremental development methodology used in Agile Software Developemnt (Schwaber and Beedle, 2001). Scrum assumes that the systems development process is an unpredictable, complicated process and defines this process as a loose set of activities that combines known tools and techniques that work. Scrum is a management and maintenance methodology for an existing system or production prototype.

Scrum hangs all of its practices on an iterative, incremental process skeleton. This skeleton operates by: at the start of an iteration, the team reviews what it must do. It then selects what can be done into a potentially shippable functionality by the end of an iteration. At the end of the iteration, the team presents the results so that it can be inspected. That is why the heart of Scrum is in the iteration. The team looks at the requirements, considers its own skills, tools and technology and determines how to build the functionality, modifying its approach as it encounters new complexities, difficulties and surprises. In short the team figures out what needs to be done and selects the best way to do it.

There are only three Scrum roles: the Product Owner, the Team, and the ScrumMaster. The responsibilities in a projects are divided among these three roles. The Product Owner is responsible for representing the interests of everyone and its resulting system. He creates the projects initial overall requirements, return on investment objectives and release plans. The list of requirements is called the Product Backlog, and the Product Owner is responsible for ensuring that the most valuable functionality is produced first and built upon. The team is responsible for developing the needed functionalities. The Team can be considered self-managing because they are responsible for figuring out how to use the Product Backlog into *increments of functionality* within an iteration. These iterations are also known as Sprints and the functionalities that go in each Sprint are called the Sprint Backlog, typically each Sprint last periods of two to four weeks. Team members are collectively responsible for the success of each iteration and the project as a whole. Finally, the ScrumMaster is responsible for the Scrum process, by implementing Scrum so that it fits within the companies culture and delivers the expected benefits.

An important tool in Scrum, is the measurement of progress through a Burn Down

Chart, which shows the amount of work remaining to complete a Sprint. This is also discussed in daily meetings, known as Daily Scrum Meetings, where the Team members talk about what they have done since the previous meetings and what hey are planing to do. Impediments or issues that prevent any Team member from accomplishing their goals is also discusses so that it can get resolved quickly.

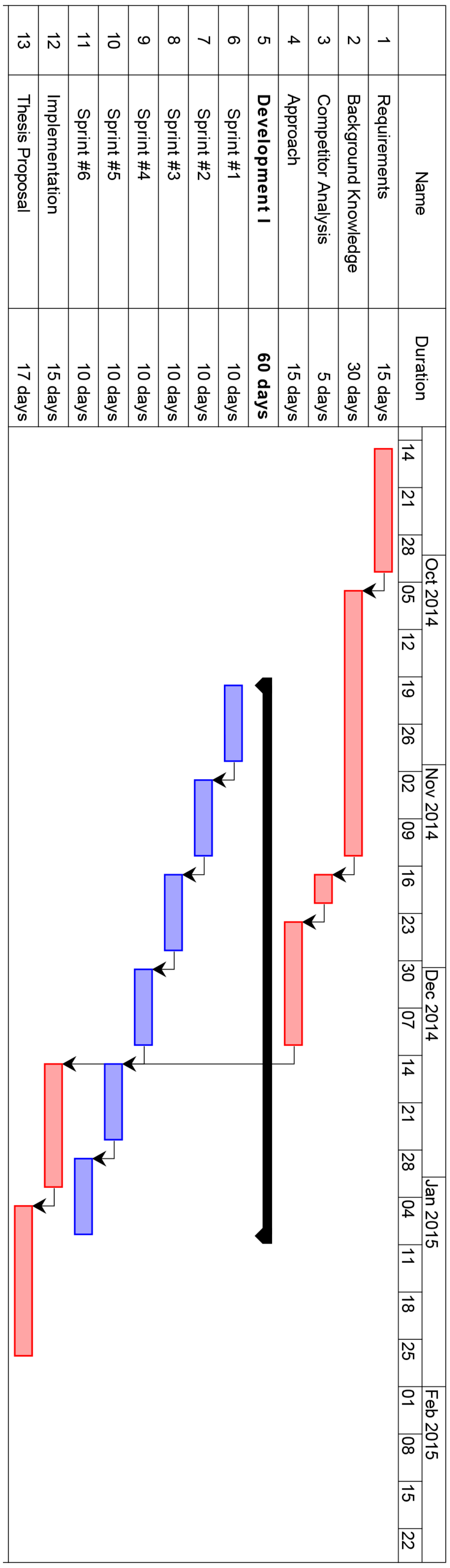
The elements that comprise the Wizdee Development Team for this specific work is composed by Paulo Gomes (Phd, CEO and Co-Founder) as a Product Owner, Bruno Antunes (Phd, CTO and Co-Founder) as the Scrum Master and João Leal as the Team, however, there are other elements that are responsible for some of the components that interact with this work. Each Sprint has the duration of two weeks, starting with a Sprint Planning Meeting. There is also the mentioned daily meetings to discuss the progress. In terms of software we use JIRA Agile[1] to manage the work that follows this Scrum process.

6.2 Planning

The creation of a Gantt chart, in figure 6.1, is not a typical task when using a Scrum Methodology. On the other hand it is a valuable tool to identify the work done during the first semester and what sprints are planed for the seconds semester.

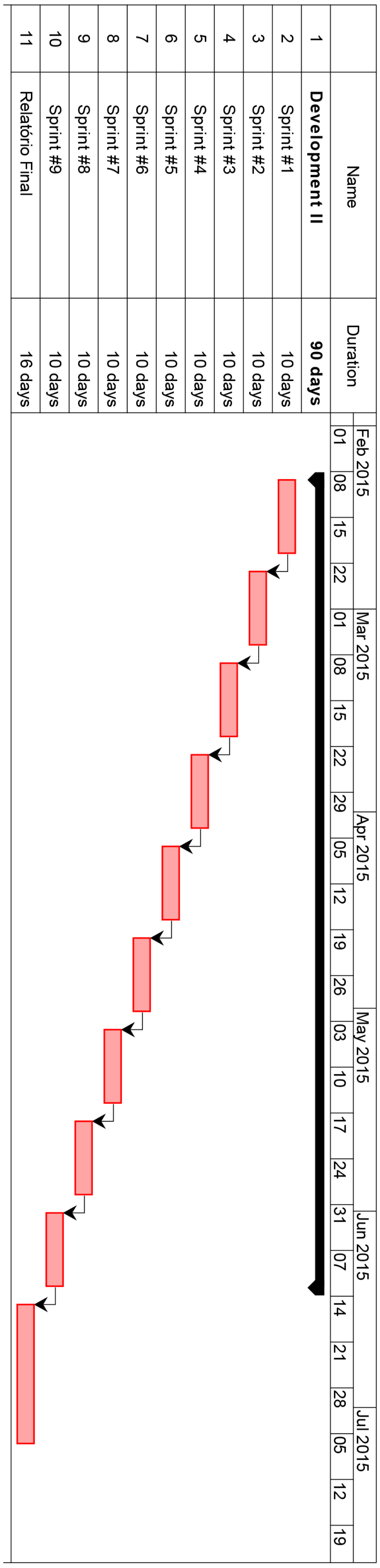
Clearly the first semester has a big component of research, this was expected, since an analysis for the main concepts in Automatic Visualization and Automated Data Mining was required and an important task in this work. Yet, a prototype for the Automatic Visualization was created during that time. In the second semester, development of all the necessary components was done.

[1]https://www.atlassian.com/software/jira/agile



	Name	Duration
1	Requirements	15 days
2	Background Knowledge	30 days
3	Competitor Analysis	5 days
4	Approach	15 days
5	**Development I**	**60 days**
6	Sprint #1	10 days
7	Sprint #2	10 days
8	Sprint #3	10 days
9	Sprint #4	10 days
10	Sprint #5	10 days
11	Sprint #6	10 days
12	Implementation	15 days
13	Thesis Proposal	17 days

Figure 6.1: Gantt chart for the first semester .



	Name	Duration
1	**Development II**	**90 days**
2	Sprint #1	10 days
3	Sprint #2	10 days
4	Sprint #3	10 days
5	Sprint #4	10 days
6	Sprint #5	10 days
7	Sprint #6	10 days
8	Sprint #7	10 days
9	Sprint #8	10 days
10	Sprint #9	10 days
11	Relatório Final	16 days

Figure 6.2: Gantt chart for the seconds semester.

6.2.1 Sprints

- **Sprint #1 (20/10/2014 - 31/10/2014)**
  - Understand current codebase for Automatic Visualization
  - Plan and discuss CBR approach and integration
- **Sprint #2 (3/11/2014 - 14/11/2014)**
  - Implementation of CBR prototype and initial integration
- **Sprint #3 (17/11/2014 - 28/11/2014)**
  - Development of chart processing abilities.
  - Initial tests of the CBR approach against real business test cases.
- **Sprint #4 (1/12/2014 - 12/12/2014)**
  - Final Case Mapping implementation and support for filters.
  - Full integration in Wizdee product.
- **Sprint #5 (9/2/2014 - 20/2/2015)**
  - Final auto-aggregation implementation.
  - Final chart processing for the most popular charts.
- **Sprint #6 (23/2/2015 - 6/3/2015)**
  - Development of a Manager that handles the case base.
  - Learn with the user behaviour automatically.
- **Sprint #7 (9/3/2015 - 20/3/2015)**
  - Explore and experiment with machine learning tools.
  - Explore and experiment with component architectural choices.
- **Sprint #8 (23/3/2015 - 3/4/2015)**
  - Prototype machine learning component architecture.
  - Integration of machine learning component architecture.
- **Sprint #9 (6/4/2015 - 17/4/2015)**
  - Explore and experiment forecast and regression algorithms.
  - Develop architecture to integrate algorithm developed in R.
- **Sprint #10 (20/4/2015 - 1/5/2015)**
  - Final Automated Forecast
  - Forecast Integration
- **Sprint #11 (4/5/201514 - 15/5/2015)**
  - Explore clustering techniques.
  - Develop Automated Clustering
- **Sprint #12 (18/5/2015 - 29/5/2015)**
  - Explore classification and outlier techniques.
  - Develop Automated Outlier Detection
- **Sprint #13 (1/6/2015 - 12/6/2015)**
  - Develop Automated Classification

Chapter 7

Implementation

As mentioned previously, there are two main components in this work. The first component is handling decision making and back-end processing of visualizations, automatically from raw data - Automatic Visualization - while the other component is to simplify and automate the creation and execution of data mining models, so that any user, without technical knowledge, can use them to discover useful patterns in data - Automated Data Mining.

In the case of Automatic Visualization the system is fully integrated in the *Wizdee* product and deployed to production. As for Automated Data Mining, the system has been integrated and tested, and is ready to be deployed in a next release. The work presented in this chapter describes in greater detail the process of both Automatic Visualization in section 7.1 and Automated Data Mining in section 7.2.

7.1 Automatic Visualization

In this section, the components related to Automatic Visualization are described, going through the various steps of the Case Base Reasoning approach in section 7.1.1 and introducing a novel technique named Case Mapping, in section 7.1.1.

It is important to note that the task of Automated Visualization only relates to the back-end process of creating visualizations, thus, it is out of the scope of this work how high level systems, such as UI, handle the result of this component. Nonetheless, the work described still concerns with making sure it is flexible enough to support the various front-end solutions that can surge in the future

7.1.1 Case Base Reasoning Approach

As mentioned in section 2.2, in the context of the Wizdee platform, the task of Automatic Visualization concerns about the development of a system that is able to provide automatic visualizations from a raw table dataset.

This section describes the specifications of the Case Based Reasoning (CBR) approach used to develop a system that is able to provide visualization automatically. The system overview is depicted in figure 7.1.

Case Representation

The representation of cases contains the following *fields*:

- *List of Attributes*: part of the case specification.

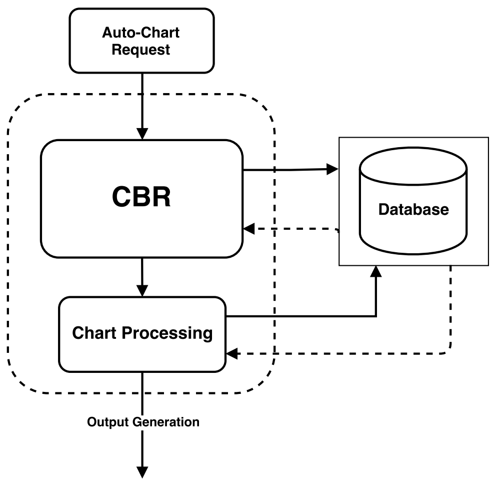


Figure 7.1: Diagram of the current automatic visualization system, using a CBR approach.

- *Type of Chart*: part of the case solution.
- *Axes (X and Y)*: part of the case solution.
- *Series*: part of the case solution.

Using the abstract representation of a case, in section 3.1.3, the type of chart, axes and series are considered as the solution part of a case and the list of attributes as the problem part of the case.

Additionally, the list of attributes are characterized by:

- *Id:* used for identification purposes.
- *Datatype:* the datatype (e.g. bigint, float, etc).
- *Label:* the name of the column.
- *Features:* Details of the features are found in the Attachment A.1

Data Transformation

Data transformation occurs during the creation of an attribute. It includes pre-processing (e.g. dealing with dirty data) and metadata extraction(e.g. extracting the features from the raw data). As mentioned, each table column can be converted to one of these types: Numerical. Date, Textual and Generic.

Case Retrieval and Retain

Serialization To store the objects containing the cases, serialization must be performed, however, in order to fulfil the requirement of making the cases modifiable by a technical user, a format that could be easily understood and editable had to be chosen. With this in mind, JSON[1] proved to be a good choice due to its human-readability, wide support and simplicity. Each case is serialized into a list of attribute-value pairs, a sample of a single attribute is shown in listing 2, in the Attachment A.1.1.

[1]http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

Retain and Retrieval The case base is using a flat hierarchy, as explained in section 3.1.3, and it is being stored in a database. All of this process is handled by a Manager and support for backup files and various languages has been implemented. Additionally, each user can have their own personalized case list besides the default case base. In case of duplicated entries, one of them is chosen. Finally, in case of conflicts such as case specifications leading to different solutions, the user cases are given priority.

Default Cases Creation A default case base of 62 cases has been created from carefully selected use cases where a good solution was known.

Case Mapping

The central component and the most innovative technique for the task of Automatic Visualization, has been labelled Case Mapping.

This technique goes beyond what has been accomplished in related works, described in section 3.3.1, by enabling the capability of introducing not only the type of chart but also its axes and series as fields of the case solution. Details of the approach are found in the Attachment A.2.

Similarity Measure

As explained in section 3.1.3, the similarity measure attempts to measure the distance between two cases. Some approaches have been described in section 3.1.3 but the current version uses a *weighted difference* between features of the attributes. Each attribute comparison starts from 0 and increases with the difference between features. In numeric features the equation is the following:

$$difference = w_i * (f_i^a - f_i^b) \tag{7.1}$$

where w represents the weight, f_i^a the feature i from the attribute a. For textual attributes, such as the label, the Levenshtein distance(Levenshtein, 1966) is used, which is a string metric for measuring the difference between two sequences. Some of the most important features include: Number of distinct rows, label distance, number of missing values and mean value.

Possible Charts A way of optimizing the amount of cases needed to compare is to know *a priori* what type of charts are even possible to apply. This avoid impossible mappings during the Case Mapping process. For this, we developed a method that analyses the data and based on a set of simple *if-statements*, it is capable of returning a list of possible charts. Those statements only specify the bare minimum data characteristics required for the processing to work, letting the CBR handle the rest of the decision making.

This method is also being used by the interface to let the user know which charts are possible to create with the results of a specific query.

Case Filters During development, the need to specify the type of chart to be returned was requested. However, considering that the Case Mapping technique attempts to find the best match without concerning with what type of chart to return, a filter system was implemented.

This filter system works by providing certain parameters during the request. A filter-parameter can be added to specify a filter (e.g. FILTER_CHART_TYPE : "BAR_CHART", meaning that the system only wants Bar Charts). The filter system filters by case and it has the option to handle various filters for a single request. It is also easily expandable to handle other type of filters (e.g. no cases with means above 5000). In the event of no suitable cases being found, an empty chart is returned.

7.1.2 Chart Model Processing

Chart Model Processing is the process of creating a chart object based on the information obtained by the CBR system above. The need for this processing comes from the higher-level systems, such as the UI, that require the charts to be transformed.

An example of this is when doing an Histogram, where a raw table can't be directly converted to an histogram without first transforming the data into bins. Thus, each chart requires a set of transformations that need to be automated.

Additionally, this process also handles data sorting and pagination mechanisms, where the data must be split into n pieces when the tables contain too many instances.

The previous rule-based system, that was in use, is described in section 2.2, had a Chart Model Processing component, however, it has been completely rewritten due to complex bugs and lack of important features such as Auto-Aggregation Mechanisms.

Auto-Aggregation Mechanism

The Auto-Aggregation mechanisms is used whenever an x-value is duplicated, which causes an ambiguity problem. If an operator is used in the query, this is quickly solved by applying the operator into the values. This means that if we have a table where the operator is a *Sum*, and there are duplicated x-values, the system would sum the values. However not every situation is so straightforward, in many cases no operator is known. For instance if we have the dataset represented in 7.2:

Sales	**Country**
30	*Portugal*
10	*Portugal*
15	*Spain*
18	*USA*

(7.2)

What sales value should the system show for Portugal in a chart?

While there is not a correct answer, there a few things to consider in a decision like this:

- The system can not produce values that do not make sense. Taking the above table as an example, if the system decides to sum the sales, it can be considered an acceptable decision since that is likely to be what the user is looking for. But as an example, if the data is dealing with probabilities (e.g. chance of closing a deal) it would not make sense to sum the values, since not only it could be wrongly interpreted it could even go above 100% which does not make sense.

- The user must be able to know what the system is deciding. This means that if it chooses to average or to sum the values, it also has to let the user know what was computed and warn the user of ambiguous data.

The chosen method to deal with ambiguous data was the creation of an Aggregation Dataset 7.3:

Measure	**Series Datatype**	**Aggregation Type**
$*$	$Date$	$Aggr_{Sum}$
$Numeric$	$Numeric$	$Aggr_{Sum}$
$Date$	$Numeric$	$Aggr_{Count}$
$Textual$	$Numeric$	$Aggr_{Count}$
$Numeric$	$Textual$	$Aggr_{Average}$
$Date$	$Textual$	$Aggr_{Count}$
$Textual$	$Textual$	$Aggr_{Count}$

(7.3)

This dataset contains the aggregation type to apply, depending on the measure data type and its series dataype. The choices were made according to use cases, albeit, these could be user-defined. Finally, the system attaches information related to the type of aggregation it made, so it can be shown in the UI.

Chart Types

As mentioned above, each chart type requires an unique procedure to create the final chart object. In this section, some innovative techniques for some of the charts are briefly described. The full list of supported charts include: Bar Chart, Histogram, Variance Chart, Pie Chart, Line Chart, Matrix Plot, Gauge Chart, Scatter Plot and Funnel Chart.

Histogram This chart has the need to create bins automatically. The chosen equation, that has shown to be adequate according to use cases, is described as:

$$number\_of\_bins = ceil(\sqrt{feature_{distinct\_rows}})$$

Next, using the number of bins, a Histogram structure is created where the *min* and *max* values are defined for each one of the bins and the data is used to fill each bin.

Pie Chart The main challenge in the Pie Chart raises from the extra element of an *others* label. The *others* label is used to limit the number of slices in a Pie Chart, since having a large number of slices would make the chart unreadable. This is challenging because if we set an hard limit of 8 slices and a query returns 9 labels, we might want to show them all. However, if we have 100 slices we might just want to show just 8. For these reasons, a dynamic system was developed where two limits are used. A soft limit defines the threshold where labels start going into the *others* label. Additionally, the size of the *others* in relation to the other slices is calculated and if the *others* label grows too much, as defined by the $perfect\_share$ 7.4, no labels are introduced into the *others*. Additionally, an hard limit is also defined, to avoid having many labels, even if it is not possible to respect the $perfect\_share$.

$$perfect\_share = \frac{100}{total_{labels} - 2} \tag{7.4}$$

Pagination Mechanisms

One complex challenge when processing the charts relates to pagination mechanisms. Typically, in real data, it is unfeasible to show a full table due to many data points causing unreadable charts or performance issues. Having a pagination system makes it possible to slice the data into manageable pieces for the UI, however, it also presents many challenges as seen by the few competitors who employ any pagination mechanisms. See section 4.1.

There are three charts that support pagination mechanisms: Bar Chart, Line Chart and Scatter Plot. Each type of chart requires an unique pagination mechanism due to different data structures. For instance, in a Line Chart, the values are continuous, while in a Bar Chart they are sorted by bar value. Many of the challenges in creating this pagination mechanisms are directly connected to the existence of various Series. Using the Line Chart as an example, we identified two approaches to implement a pagination mechanism, and we will present the main advantages and disadvantages of each approach.

Point Based Limitation Approach In point based limitation, each series has its values limited according to the number of points. This system is efficient and simple to implement since we can simply ignore all points after a certain threshold. It also has the benefit of being very useful performance wise, especially in mobile devices, since the number of points is one of the main causes for low-performance. The disadvantages come from certain use cases where the limitation might induce the user in error if more than one series exist and the limit is too small.

X-Value	Y-Value	Country
0	10	*Portugal*
1	11	*Portugal*
2	14	*Portugal*
3	12	*Portugal*
4	16	*Portugal*
5	14	*Portugal*
6	12	*Portugal*
7	4	*USA*
8	8	*USA*
9	6	*USA*

(7.5)

For instance, if we set a limitation of 5 points per series, while using the dataset 7.5, we can see the issue in figure 7.2, where the data points belonging to Portugal are not completely shown because of the 5 points limit. This problem occurs because the series USA only starts after Portugal.

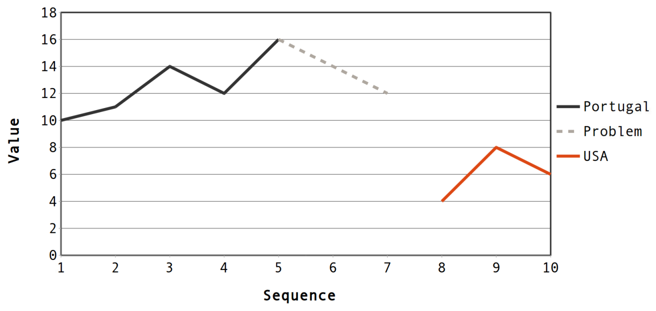


Figure 7.2: Representation of the point based limitation problem.

X-Value Limitation Approach The alternative approach is based on limiting the values based on their x-values. This approach requires a list of the possible x-values and a way to map the series into those x-values. Limiting by the x-values solves the issue in Point Based Limitation, but it also has its own disadvantages, such as worse performance and in datasets such as 7.6, where there are two series with widely different x-values, a lot of empty space is created between series.

Even so, X-Value Limitation was the chosen approach since it doesn't mislead the user.

X-Value	**Y-Value**	**Country**
0	10	*Portugal*
1	11	*Portugal*
2	14	*Portugal*
1000	16	*USA*
1001	14	*USA*

(7.6)

Finally, choosing the most relevant series is also a decision that the system must take. The approach used to handle this challenge was to pick the series with the highest average values but also create the needed components to allow an user to manually specify which series they want to see.

7.2 Automated Data Mining

In this section, the components related to Automated Data Mining are described, starting with an overview of the service developed in subsection 7.2.1 followed by a detailed description on the automation for each of the data mining tasks: Classification, Forecast, Outlier Detection and Clustering.

Due to the dynamic character of the development methodology, some features planned earlier suffered changes. While originally there was a Search Integration requirement, described in section 5.3, where these data mining tasks would start by using a specific operator on the search query, this requirement no longer applies. This modification is due to extensive changes the natural language processing engine received due to internal decisions, during the same time frame of the work developed in this thesis. The final decision is to use UI elements, similar to the ones used on the charts, instead of search operators. Nonetheless, the work described is mostly concerned with creating the necessary back-end components and making sure it is adaptable to the various front-end solutions that may be used in the future.

Conceptually, all of these data mining tasks follow a similar execution workflow. After a user makes a search query, a request for a table or chart is made and during that process the request goes through a module to decide what data mining tasks can be applied. The information regarding the possible data mining tasks is then sent to the user interface and those options become available to the user.

When the user clicks on one of the data mining tasks available, the process of Automated Data Mining takes charge of creating the models and making the necessary choices. Additionally, the user can opt to manually choose some of the parameters too. This workflow is shown on the sequence diagram presented in figure 7.3.

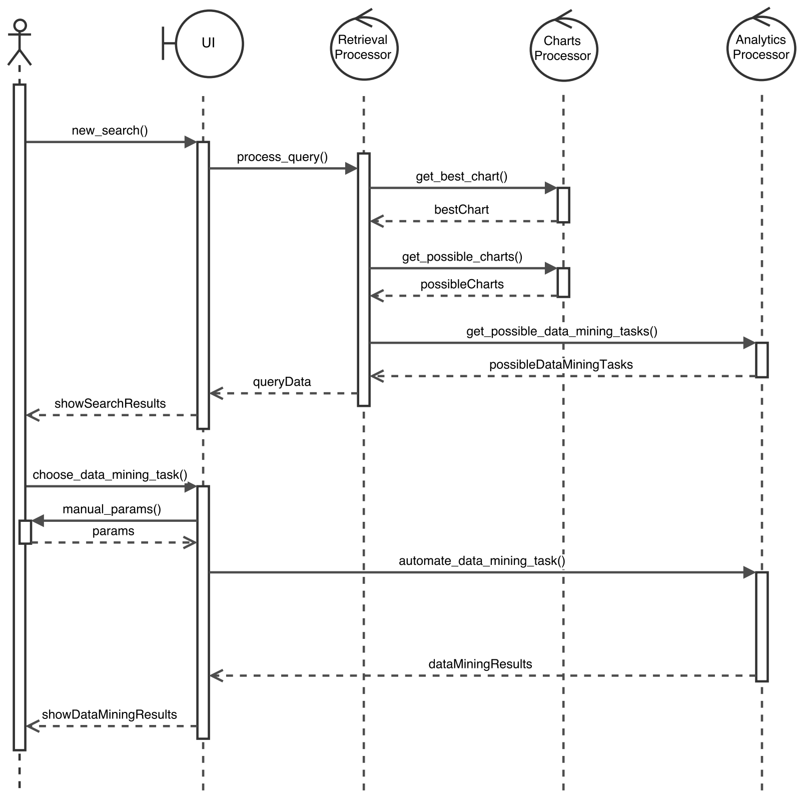


Figure 7.3: Sequence diagram for the interaction during a data mining task.

7.2.1 Machine Learning Service Component

During the development of the data mining tasks, the need to organize the machine learning components into something that was easily maintainable and flexible emerged. Furthermore, various secondary projects, outside of the work presented in this thesis, also showed interest in having a system that could be used as a central hub responsible for dealing with all the typical management and logic typically found in these type of tasks.

The answer to this was the creation of a service dedicated to all the machine learning related processes, a system with the following goals: Maintainable and consistent data model, Easy-to-use with an abstraction for low-level details, Flexible and Extensible, and scalable.

Architecture

The backbone of this system is based on the creation of a service application, capable of receiving data and doing business logic across the network. One of the first challenges was the decision regarding how the communication is handled between applications.

Communication The two main mechanisms explored to handle the data exchange were: making the service RESTful or use RPC[2] to expose the needed resources. Since both of these mechanisms work well for this type of system we chose to use an architecture style based on REST over HTTP, because many components inside Wizdee platform already use REST and we had experience implementing an architecture of the same style in the past.

- **Web Server** This service application was implemented in Python, since most of tools used for machine learning related tasks depend on Python libraries. The web application framework chosen for this task was Flask, mainly due to its simplicity.
- **Client** In order for Wizdee platform to be able to connect with the service, a client library was implemented using the Jersey JAX-RS Client. This project is already integrated in the platform and every machine learning related project is using it. The project itself handles all the client communication, including serialization, request building and response handling.

Data Model One of the most challenging aspects of this system is how the data is handled between the server and the client, and more specifically, how does each side know what data schema each request or response should have, and what to do when the data schema changes. We have chosen Protocol Buffers as the mechanism for serializing structured data and the high-level view of this process is shown in figure 7.4. Below I will explain in detail the challenges and reasons for this choice.

Considering one of the goals here was a consistent but maintainable data model, meaning that both the server and client should be synchronized to the specifications of the data, without having to manually to set it in code, the serialization and how the data schema is handled is a key aspect of this system.

A few options were considered: Protocol Buffers, Thrift and Avro. All of these support both Java and Python, albeit, both Thrift and Avro support more languages than Protocol Buffers. However, those two are tightly affiliated to their own RPC frameworks, which makes Protocol Buffers a more generalist choice - useful when developing a REST architecture. The extra features of those libraries also come at a cost, some of those features

[2]Remote Procedure Call

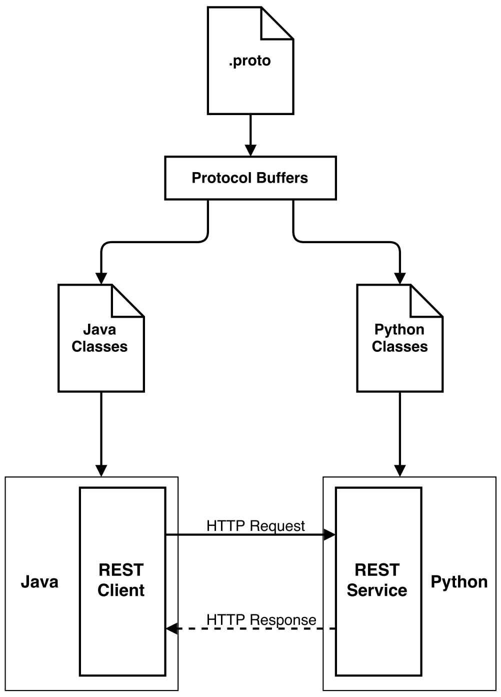


Figure 7.4: High-level view of the data model process.

are only available in certain languages (e.g. union is only provided for Ruby and Java in Thrift) which leads to a less consistent model. Regarding performance, a few benchmarks show slightly better performance[3] for Protocol Buffers in Java but worse in Python[4], which ultimately makes performance a difficult metric to base on. Finally, Protocol Buffers have more documentation and support, they have been used in production extensively for many years in other companies. In terms of code style, the Protocol Buffer uses the builder pattern which makes it easy to detect errors earlier in development, such as required fields not being set up. For these reasons we have chosen Protocol Buffers.

Using Protocol Buffers require the creation of a .proto file, that is later used to generate the classes for both Java and Python, only the .proto file should be modified and no user should modify the generated classes.

Pipeline Although more details about the architecture of each data mining task will be discussed in the subsections below, the data schema for each task shares the universal concept of a pipeline.

There are two motivations behind this pipeline. The first is to make sure the system is extensible, so that adding new components is easy and abstracted from low-level details (e.g. adding a new classification algorithm won't make a developer have to understand the underlying communication details, or how the pre-processing is handled). The other motivation is having a system that is flexible enough so that other projects, outside of the work presented in this thesis, can use it without having to reimplement the same steps repeatedly but still allowing developers to remain unconstrained on how to solve problems by giving some control over what actions they can do in the pipeline.

There are various types of pipelines in the architecture, but there are two main pipelines: the model-building pipeline and the model-appliance pipeline. The model-building pipeline

[3]https://github.com/eishay/jvm-serializers/

[4]http://floatingsun.net/articles/thrift-vs-protocol-buffers/

is show in figure B.1 in the Attachment B.1.

The model-appliance pipeline steps are described in Attachment B.2.

In short, the main idea of the pipeline is to have an organized structure of the common steps that data mining tasks require. A developer can simply choose from a list of available algorithms the ones they want to apply for each one of the steps. With the added advantage of extending the system being a matter of adding the necessary algorithms in Python and exposing them to the data schema.

Model Persistence The final step of the building pipeline is the creation of a output that can be used in a later stage and a few challenges appeared during development regarding the storage and retrieval of the models. Solving this task is essential because in many use cases you want to create a model and use it later, since training a model can take a considerable amount of time.

In terms of architecture, persistence is dealt by requiring a name for each request that indicates the model name. The model name is then stored in disk and loaded each time it is requested. It is also possible to overwrite the model if the name is duplicated. Furthermore, the storage itself also has a couple of hindrances, since not only a model of the algorithm must be created, but also a model of the whole pre-processing steps, so that it can be used later. It is important to take this effects into account on the new data or else the features might not be representative of the fitted data. This means that each pre-processing algorithm must be describable so that it can be applied without requiring the full data.

Scalability One of the concerns when developing an architecture that deals with these type of tasks is how scalable it is and, while this was not one of the focus of this thesis, some precautions were made to create an environment that could easily scale in the future. In terms of the server framework, the main concern was that it supports parallel requests and Flask was configured to make this happen. A specific example of how scalability can be handled is when doing parameter optimization. A lot of models are created during that step and it is an ideal situation to introduce multiple servers with a load balancer since each model is independent from one another. In this situation, creating a load balancer in this architecture would be simple, due to the nature of services. However, it was beyond the scope of this work, due to time constraints, but it shows how the architecture chosen can be used to create a massively scalable system.

7.2.2 Pre-processing

Most data mining tasks make use of some type of pre-processing before the data can be transferred to the machine learning algorithms, and although some pre-processing techniques are applied to specific tasks, as seen in the subsections below, there are also some techniques that are found in any task.

A particular common problem is the handling of missing values, and for this, two techniques are supported:

- *Entry Removal:* if an entry has a missing value, that entry is ignored. Besides the issue of potentially decreasing the sample size, it is also possible that it might introduce bias because the sub-sample could be unrepresentative of the original data.
- *Single Imputation:* each missing value is replaced by the mean or median of that column, it has the advantage of keeping the sample mean for that column, however it might distort the data.

Preference was given to simple, yet generic, approaches, however, there are more robust techniques that could be explored in the future, such as Multiple Imputation or Shared Parameter Model (Enders, 2011).

Another common pre-processing technique is related to feature normalization, which is useful since some machine learning algorithms assume, for instance, that all features are centered around zero and have variance in the same order (Graf and Borer, 2001). For feature normalization there are two techniques available:

- *Standardize Features:* removes the mean and scales to unit variance using the standard score.
- *Scale Features:* scales each feature to a specified range.

Additionally, a custom technique that could handle strings was developed, named as String Columns.

String Columns These are a structure similar to count vectorizers[5] or bag of words, meaning that the input is a list of strings and the output is a transformer that, given a string, produces a binary vector representative of that string. The details of this approach are described in the Attachment B.3.

Finally, there are three well known methods to reduce dimensionality implemented:

- *Variance Threshold:* a straightforward approach where every feature that doesn't meet the specified variance threshold is removed.
- *Principal Component Analysis:* requires a field saying the number of components to keep.
- *Select K Best:* Select features according to the K highest scores, using chi2.

7.2.3 Validation

Quality assessment of the created models is an important step in any supervised data mining task, as discussed in 3.2.1. In this system, three validation algorithms are supported:

- *Holdout:* the dataset is split into a train and test dataset. There's a required field to indicate the test share size and if the order is randomized before splitting. The response is a single score value.
- *k-Folds:* the data is divided into k groups and run k times. There's a single field to define the k value. The response is a k sized list of scores.
- *Stratified k-Folds:* a variation of the k-folds above, where it attempts to balance the proportions of the classes.

To calculate the validation score there are many metrics, some of those are discussed in section 3.2.1, the ones we chose to implement, taking into account the type of tasks, are the following: Accuracy, Mean Squared Error, Mean Absolute Error, Mean Absolute Percentage Error, F1 Score, F1 Weighted Score, Root Mean Square Error, Impurity index.

[5]http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

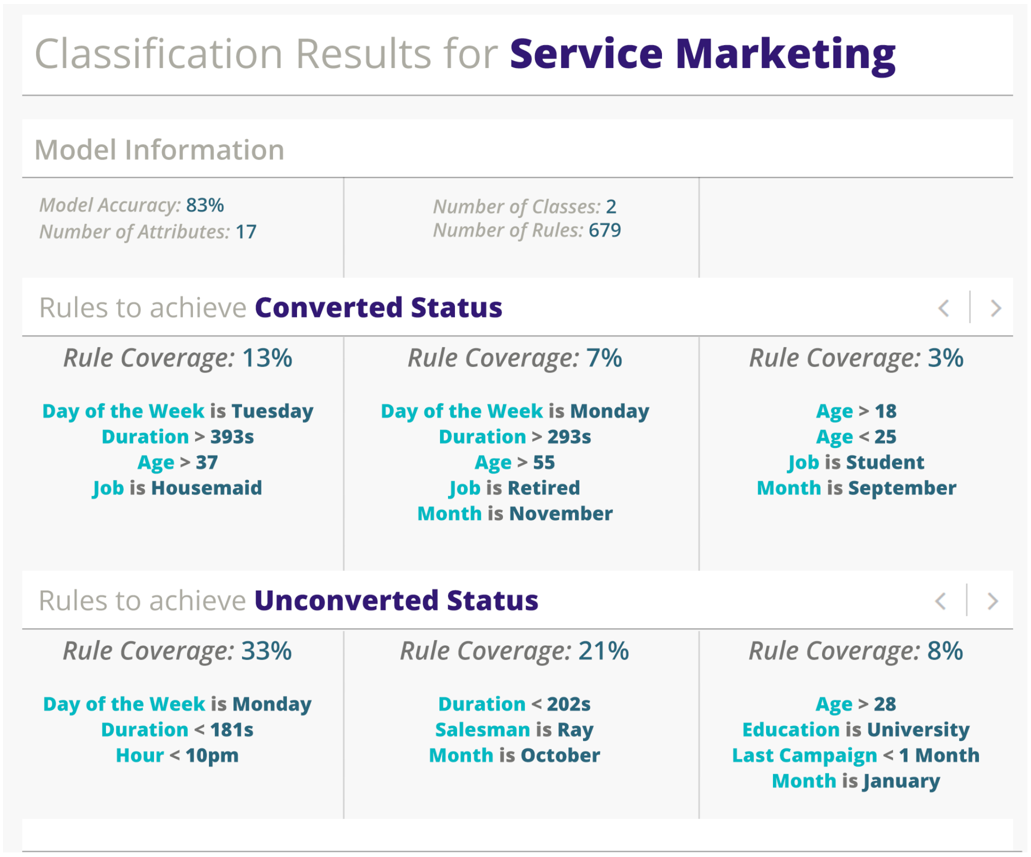
7.2.4 Automated Classification

In the context of the Wizdee platform, the task of automating classification has a very specific goal in mind, which is, in short, to understand what influences a target variable. This goal differs from the typical use case of classifying new instances with the created model.

Having this goal in mind means that we are not too concerned in using the created model to predict new instances. We are, however, interested in the underlying decisions that make the prediction. This creates the challenge of being able to describe the model and not merely obtain good evaluation results. Notwithstanding, having a good model is essential to make sure the description is apt for the problem at hand.

In order to accomplish a representation of classification models, so that non-technical users could understand, a description based on rules was chosen. Additionally, emphasis on showing the most relevant information was put forward, so that users can focus on what they are looking for and not feel overwhelmed by too many rules.

In figure 7.5 we show a mock-up of the interface presented to the user. The context of this mock-up is a request of an automated classification task on data regarding Service Marketing, where the data contains information related to market campaigns about services, and the target variable is the Status of said campaigns. Thus, this hypothetical task entails automatically finding in the Service Marketing data that causes the Status to go Converted or Unconverted automatically without user intervention.



Classification Results for **Service Marketing**

Model Information

Model Accuracy: 83%
Number of Attributes: 17

Number of Classes: 2
Number of Rules: 679

Rules to achieve **Converted Status**

Rule Coverage: 13%	*Rule Coverage:* 7%	*Rule Coverage:* 3%
Day of the Week is **Tuesday**	**Day of the Week** is **Monday**	**Age** > **18**
Duration > **393s**	**Duration** > **293s**	**Age** < **25**
Age > **37**	**Age** > **55**	**Job** is **Student**
Job is **Housemaid**	**Job** is **Retired**	**Month** is **September**
	Month is **November**	

Rules to achieve **Unconverted Status**

Rule Coverage: 33%	*Rule Coverage:* 21%	*Rule Coverage:* 8%
Day of the Week is **Monday**	**Duration** < **202s**	**Age** > **28**
Duration < **181s**	**Salesman** is **Ray**	**Education** is **University**
Hour < **10pm**	**Month** is **October**	**Last Campaign** < **1 Month**
		Month is **January**

Figure 7.5: Automated classification mock-up.

To check if it is possible to create a classification model there are two things that the system takes into consideration:

- There are at least two columns in the data table.

- The system attempts to suggest a viable column to use as the target variable, this way, at least one column with no more than ten distinct values should exist. Choosing a target with too many classes might cause balancing problems in the dataset (Hoens et al., 2012)
- In terms of features, some discussions suggest having at least four times as training instances as features and, in some cases, even more (Lewis, 1992). But it is a complicated issue to deal automatically, without manual analysis. Therefore the only limitation is that there are at least as many instances as features.

As for the type of algorithm chosen for this task, the choice was based around decision trees due to their readability and performance, as discussed in section 3.2.2.

Pre-processing

Pre-processing is an important part of most classification tasks, and in this task the following pre-processing techniques are applied:

- Missing Values are pre-processed using Imputation with mean values if the column is Numeric, otherwise they are imputed with a default value, representing the notion of *other* values. Additionally, an extra feature is added, for each numeric feature that contains missing values, to signal instances where that feature is null.
- Strings, Dates and Booleans are converted to binary vector features - as String Columns, which is explained in detail in section7.2.2.
- Class values are encoded to numerical features, since the algorithms can't handle non-numeric values.

Typically, decision tree algorithms can handle categorical data as features, however, the implementation found in sklearn does not[6], which is the implementation we use.

There are other approaches when it comes to handling string based features, such as Term Frequency or TF-IDF. These were considered but were not used because decision trees would assume a continuous numeric feature, creating rules such as $fruit_{frequency} > 50$, which is not the type of information we want to give to the end users.

Model Creation

After creating the necessary features, the next step is building the model, and there various steps during this stage, represented by the outline in figure 7.6.

Algorithm There are two supported splitting criterias when it comes to decision trees on sklearn: gini impurity and entropy. Thus, choosing the algorithm is a matter of choosing the correct way of splitting the nodes.

A possible approach would be to simply do both and choose the one that performs better on validation. However, in a problem where high performance is a requirement, this is not feasible. Therefore, the decision must be made before creating the models.

Some research regarding this issue was made, however, the answer to this problem is somewhat inconclusive, since in many cases, it does not seem to make much of a difference. This is supported in (Breiman, 1996), where is said that in problems will a small number of classes should produce similar results and also in (Raileanu and Stoffel, 2004) where

[6]http://scikit-learn.org/stable/modules/tree.html

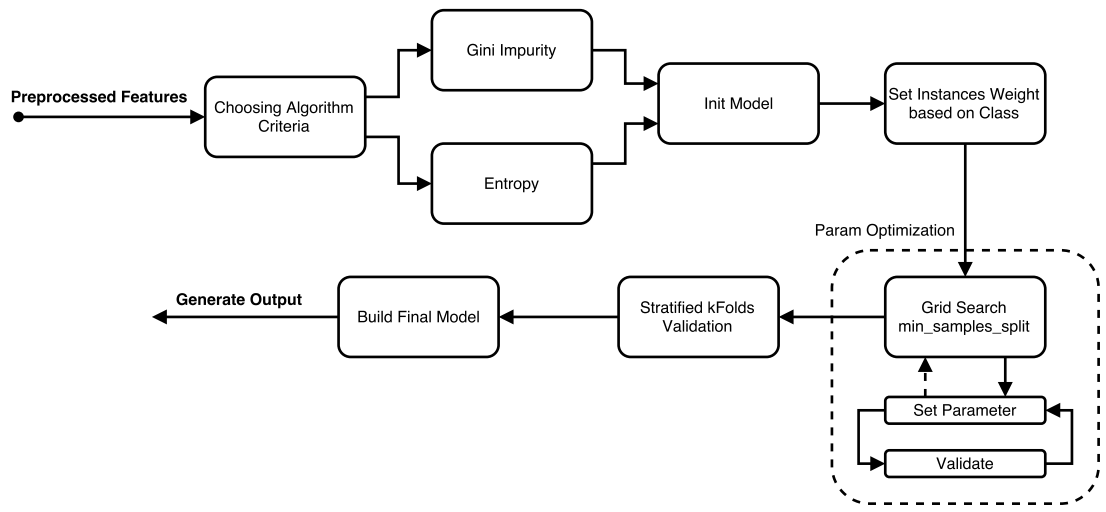


Figure 7.6: Outline of the model creation solution.

the results show that the approaches only disagree in 2% of the time. For these reasons entropy was chosen as the default criteria, with the option for the user to manually request gini.

Weighted Decision Trees Many real world problems consist of unbalanced datasets, and this can cause serious learning difficulties (Sun et al., 2009). As a consequence, creating a technique that could alleviate some of the issues related to this were explored. There are two common approaches used to tackle this problem. One is based on assigning an higher cost to misclassification to the class that has fewer instances, while the other is based on undersampling, or oversampling, the dataset until the balance is restored. Research shows that there is no clear winner (Chen et al., 2004), nonetheless, since sklearn already supports the attribution of weights for each instance, the method based on weight was chosen.

The equation to define the weights is as follows:

$$weight\_class(x) = 2 - \frac{count_{instances}(x)}{total_{instances}} \tag{7.7}$$

Parameter Optimization In the decision tree implementation there are many parameters that one can use. There is, however, one in particular that directly affects the fitting of the tree and that can help avoid overfitting - *min_ samples_ split*. This parameter controls the number of samples required to split an internal node [7]. Therefore, parameter optimization attempts to find a good value to create a model that learns from the data but still generalizes well.

The optimization process is based on grid search, where an exhaustive search is performed through a list of manually specified *min_ samples_ split* values. For each value, a validation algorithm is used in an attempt to find the best value. Grid search was chosen for the following reasons:

- Simplicity, when compared to other optimization techniques.
- Embarrassingly parallel, since the parameter settings are independent of each other, which creates the perfect setting for a scalable approach.

[7]http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

- Consistent performance in datasets with the same number of instances and features.

The validation used during parameter optimization is the Holdout, due to its speed. However, Stratified k-Folds is used to obtain the final model score. After finding the best parameter and validating, a final model is built. Still, the main concern of the task is obtaining descriptive rules for the model and not predict new instances, therefore to generate the output, some processing must be done to the model.

Generating the Output

To generate an useful output, a series of steps are required, which are outlined in figure 7.7.

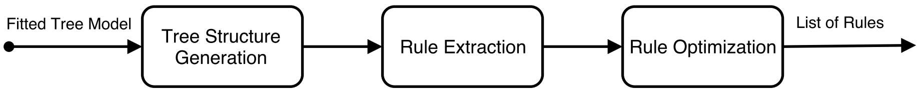


Figure 7.7: Outline of the rules creation process.

Creating a Tree One of the big challenges when extracting the rules from the model, is that while decision trees produce readable information, sklearn does not support an easy way to extract rules, nor there is documentation available to parse their internal model. Analysis of their structure shows it is based on indexes, where they have various arrays, including an array for features, left children and for right children. As an example:

$$\begin{aligned} features =& [2, -2, 3, 2, -2, -2, -2] \\ children_{left} =& [1, -1, 3, 4, -1, -1, -1] \\ children_{right} =& [2, -1, 6, 5, -1, -1, -1] \end{aligned}$$

Using the arrays, it is possible to parse this into a tree data structure, which can be easier to transverse and perform processing tasks later on. To parse the arrays an algorithm was created, and is presented in algorithm 1, in the Attachement B.4. The technique behind is to use the index in the features array to find the children features, recursively.

Extracting the Rules Finally, with the tree data structure, it is easier to extract the rules, but before extracting the rules it is important to only extract the most relevant ones. In this context, the most relevant rules are defined as the rules with the highest leaf coverage, for each one of the classes. The algorithm behind this process obtains the leafs for each class of the target variable and sorts them by coverage. The top 3 leafs with the highest coverage are selected and rules are extracted. The algorithm to extract the rules starts on a leaf and transverses the tree upwards until it reaches the root. During this process, each node checks the parent type and whether they are on their left or right side. The parent type is important, because when dealing with String features the possible operators are different from the ones used in Numeric features.

The output of this algorithm is a list of rules, albeit, duplicated rules might exist. An example of this is depicted in figure 7.8, where the rule that describes the leaf *Expensive* is:

$$Price \geq 500 \wedge Fragility = Strong \wedge Price \geq 1000$$

When it could be reduced to simply:

$$Fragility = Strong \wedge Price \geq 1000$$

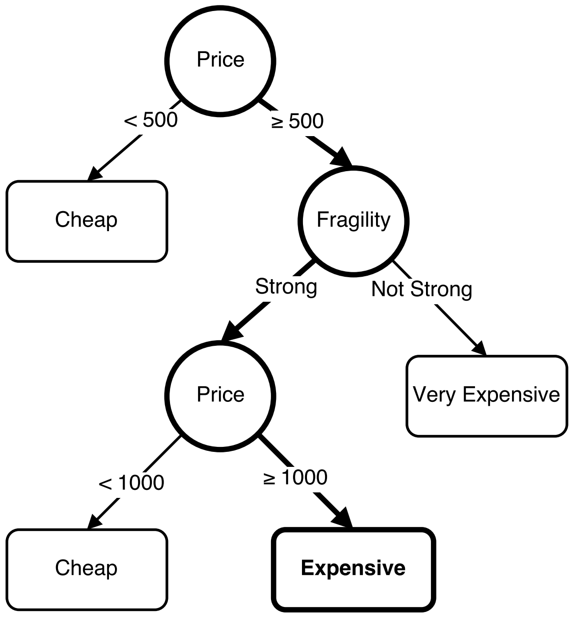


Figure 7.8: Example of duplicated rules

This process of removing duplicated rules is calling optimizing the rules.

Optimizing the Rules The algorithm used to optimize the rules is described in algorithm 2, in Attachment B.5. The algorithm finds rules with the same feature operator to reduce them.

7.2.5 Automated Outlier Detection

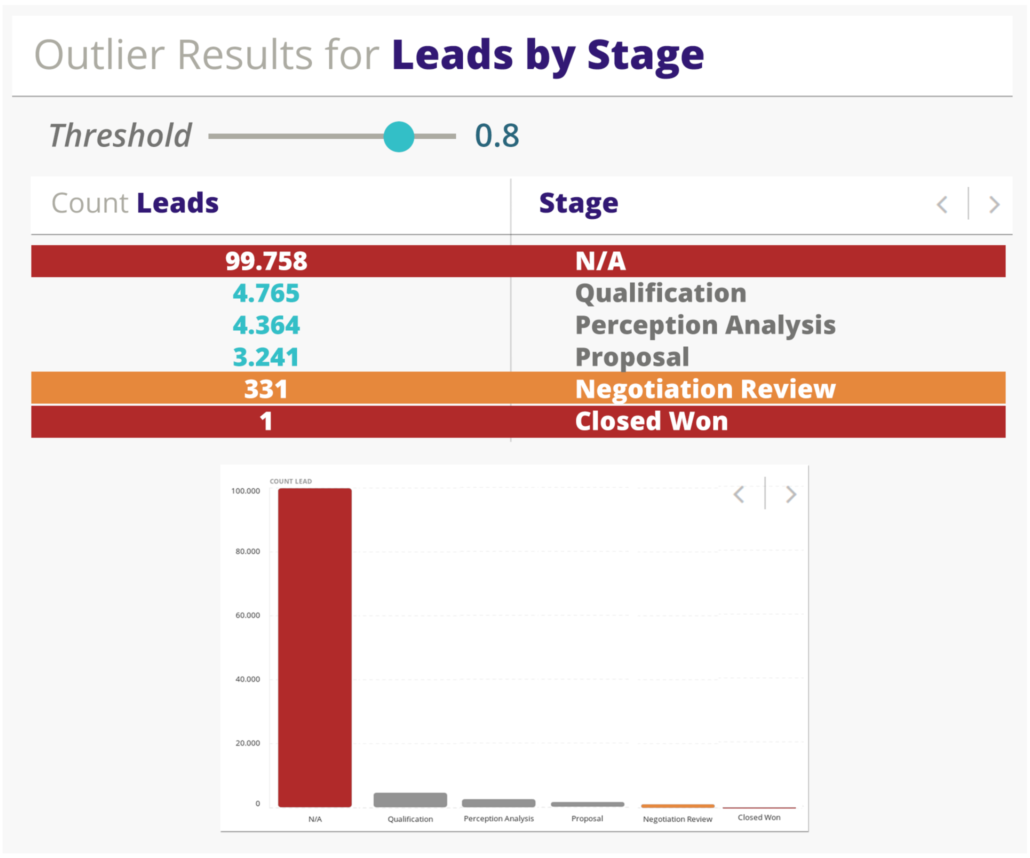
The task of Automated Outlier Detection deals with finding outliers in a dataset in an unsupervised fashion. Data often contains instances that are unusual when compared to the others of the same dataset, and those instances can have various causes, such as data entry errors. In any case, these type of situations should attract the attention of an user so that can be further analysed. For these reasons, Wizdee found it essential to have outlier detection capabilities in its platform.

In terms of the overall approach, it was important that the technique used coule be easy to understand and that the user could have some control of what makes an outlier. All this without the need of having to specifying previously what an outlier is, meaning no supervised algorithms are allowed.

In figure 7.9 we present a mock-up of the interface shown to the user, the context of this mock-up is a request of automated outlier detection on a table that has information about the number of leads by stage.

As we can see, there are a few entries that stand out from the rest, there is a value of 99.758 leads in Stage *N/A* and a value of 1 in Stage *Closed Won*. The first one might mean a data entry error due to N/A being a null or non-existent stage while the other might simply mean a strange value. In either case, it is worth notifying the user about those instances.

It is important to note that the user can control if some other instances might also be worth considering, such as the *Negotiation Review*, depending on the a threshold value. That is why the approach needs to be easy to understand, with the ability to have control over a threshold type parameter.


Outlier Results for **Leads by Stage**

Threshold 0.8

Count **Leads**	**Stage**
99.758	N/A
4.765	Qualification
4.364	Perception Analysis
3.241	Proposal
331	Negotiation Review
1	Closed Won

Figure 7.9: Automated outlier detection mock-up.

To check if it is possible to create a outlier model there are a few requirements: At least one column in the data table and more than 10 instances

As for the type of algorithm chosen for this task, the choice was based around distance metrics, due to their easiness, flexibility and by being computationally feasible (Pei et al., 2006)

Pre-processing

The following pre-processing techniques are applied:

- Missing Values are handled by imputing with a default value indicating a missing value. Other methods such as imputation with mean values are not used in this task because keeping those missing values could be very useful in finding outliers.

- String, Dates and Booleans are converted into a distribution map, where the distribution of each value is counted. This technique was chosen instead of others such as

String Column, described in section 7.2.2, because the approach is distance based, making it more efficient since it is a continuous variable.

Model Creation

After creating the necessary features, the next step is building the outlier model and there various steps during this stage, the outline is shown in figure 7.10.

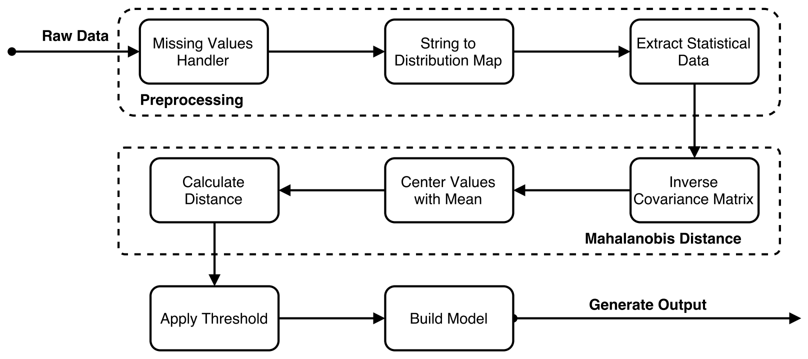


Figure 7.10: Outline of the automated outlier detection process.

Algorithm As mentioned before, the approach is based on distance metrics, a few of the popular ones include:

- Euclidean distance: an ordinary, straight line, distance between two points in Euclidean Space.
- Manhattan distance: where the distance is the sum of the absolute differences of their Cartesian coordinates.
- Chebyshev distance: defined by the greatest distance along any coordinate dimension between two vectors.
- Mahalanobis distance: the measurement of how many standard deviations a vector A is from the mean of D.

From these possible metrics, the chosen metric was the Mahalanobis distance (Mahalanobis, 1936) where instances with a large Mahalanobis distance are indicated as outliers. One of the advantages of this metric is that it accounts for different scales and variance of each one of the attributes of the dataset. It has been used in outlier detection with some success in the past, (Matsumoto et al., 2007; Stevens, 1984; Hardin and Rocke, 2005) and it is also extendable for better results (Franklin et al., 2000).

However, we are aware more robust techniques exist and that in some situations, such as high multivariate data, it might not perform well (Hadi, 1992). Nonetheless, it is still a flexible technique, that allows user control and provides reasonable results according to past academic work.

The other distance metrics such as the Euclidean distance, also suffer from various problems, especially in dimensions higher than 3, where distance might not be what it seems (Aggarwal et al., 2001).

This implementation required several steps, as shown in figure 7.10, where of interest is the calculation of the covariance matrix and its inverse. For this, Mahout was used due to its fast implementations. Finally, the formula of Mahalobis distance, as shown in equation 7.8 is put together, by centring each value by the mean and doing all the dot multiplications, followed by the square root:

$$MD = \sqrt{(x - \mu)^T \, \Sigma^{-1} \, (x - \mu)} \tag{7.8}$$

With the distances calculated, a threshold is used to define what is an outlier, a default value of 0.3 is used, however, the user can define their own.

Generating the Output

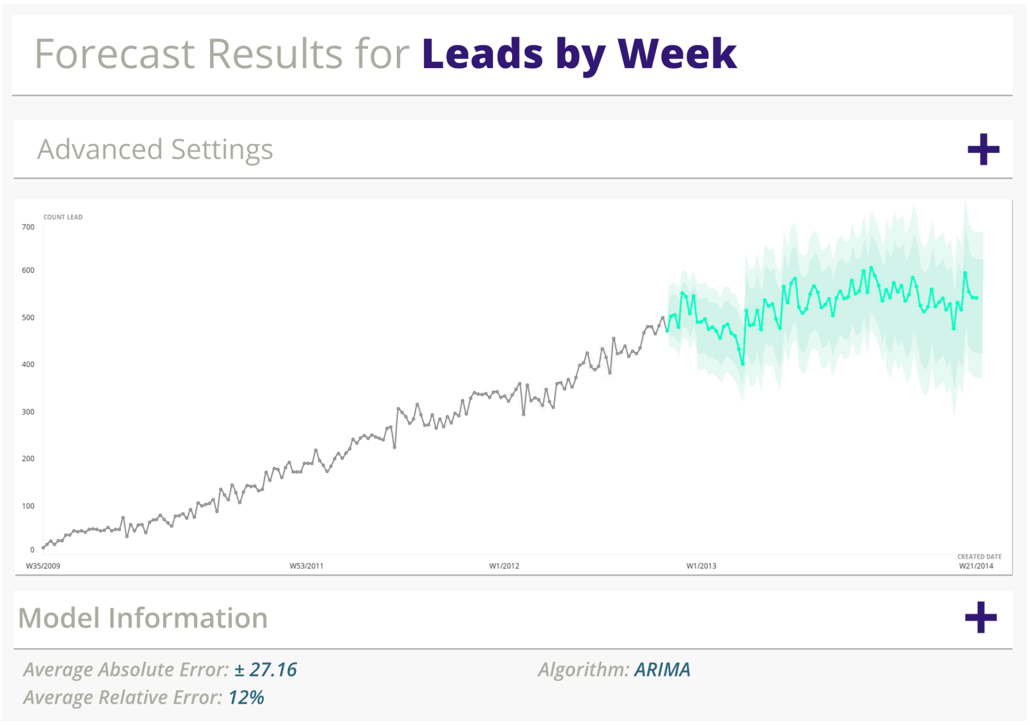
In this task, the output is simply a list of the outliers and for this, a map of the features and their respective instances is kept so it can be used to map the outliers to the original instances.

7.2.6 Automated Forecast

In the context of this work, the task of automated forecasting relates to the ability to predict new values, future values if in a date context, but also coming values in a sequence of numbers, based on past and present data.

In this type of tasks it is common to find uncertainty, especially in an automated setting. Therefore, a metric indicating this degree of uncertainty is also required. Additionally, this task must be performed under strict time limits while providing reasonable results. The sections below describe the process to create a system that responds to these challenges.

In figure in 7.11 we show a mock-up of the interface presented to the user when a request for automated forecast is made. In this case, the request was made with information about the number of Leads throughout the days. The dotted line indicates the forecast while the secondary lines indicates the uncertainty.


Forecast Results for **Leads by Week**

Advanced Settings +

Model Information +

Average Absolute Error: ± 27.16
Average Relative Error: 12%
Algorithm: ARIMA

Figure 7.11: Automated forecast mock-up.

To check if it is possible to create a forecast model there are a few characteristics that the system needs: A single numeric column, a table with two columns, where one of them is numeric and the other is a date or where one of them is numeric and the other is a sequence, and an higher amount of past points than points to predict.

As for the type of algorithm chosen for this task, the choice was based around statistical analysis models and regression models, described below.

Pre-processing

Pre-processing is an important part in most data mining tasks, however, during forecast one must be careful to avoid distorting the data. In some cases it might be a mistake to do certain pre-processing tasks such as missing values imputation or outlier removal since those might fundamental in the forecast task, such as forecasting when a failure will happen in the future. For these reasons the only pre-processing is to limit total past values: due to very strict time limitations a limit for the amount of past values is set and all the other values are discarded.

Model Creation

After creating the necessary features, the next step is building the model and there various steps during this stage, that are outlined in figure 7.12.

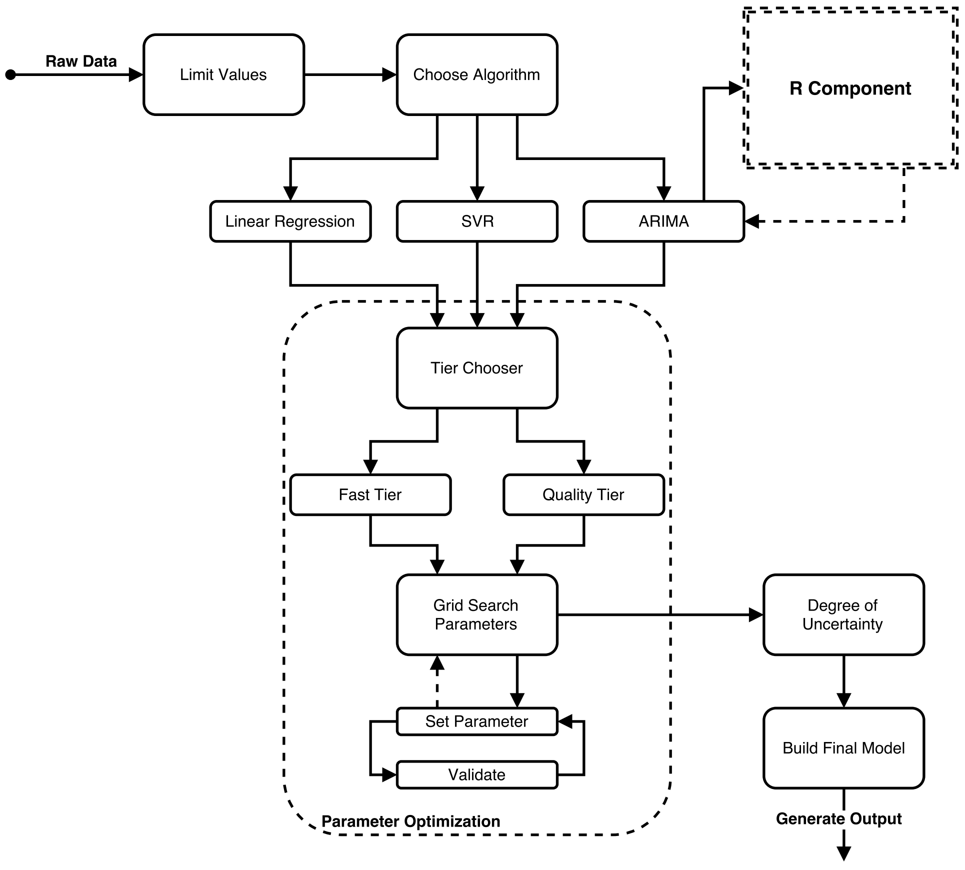


Figure 7.12: Outline of the automated outlier detection process.

Algorithm There are many methods to predict new values, however, since this work is not meant as an exploratory research, the focus was put into finding methods that are popular and also have been proven to be reasonable models in terms of quality and performance.

For these reasons, the chosen algorithms are:

- Autoregressive integrated moving average
- Support Vector Regression as proposed in (Vapnik, 2013)
- Linear Regression (as baseline)

The quality of ARIMA models is shown in various works such as (Pindyck and Rubinfeld, 1998; Contreras et al., 2003; Hyndman and Khandakar, 2007). This is a model typically chosen for its accuracy and mathematical soundness. On the other hand, ARIMA models can be badly distorted by outliers (Ledolter, 1989) and they do not support multivariate data. Thus, this process also uses Support Vector Regression, where one of the biggest advantage is to be able to handle high dimensional spaces (Smola and Vapnik, 1997). Linear Regression is used mostly as a base line algorithm.

One of the big challenges regarding this task was that both Linear Regression and Support Vector Regression are both available in the sklearn library - fitting easily into the Wizdee machine learning architecture - there is no implementation for ARIMA.

Additionally, the most popular implementation of ARIMA in Python is found in Statsmodel[8], but it proved to have implementation flaws that will be analysed below.

As a solution to this problem, integration with R was discussed due to the wide range of analytical and statistical algorithms. This, however, raised concerns regarding integration.

R ARIMA vs Statsmodel ARIMA As mentioned above, during the development of this system a few issues were found with the Statsmodel implementation, not only from a functional point of view but also from a quality one.

A model for both implementations was created using the same ARIMA parameters of $P = 2$ $D = 0$ $Q = 1$. These parameters are not optimal, but it is possible to observe in figure 7.13, that the R implementation results in a better model. In figure 7.14, a model for both implementations was created using the same ARIMA parameters of $P = 3$ $D = 0$ $Q = 4$. We can notice that the R implementation provides a good prediction model, while the Statsmodel implementation can not create a model at all.

Similar issues have been posted[9], but this problem becomes very noticeable by doing a few tests comparing their implementation with the one in R.

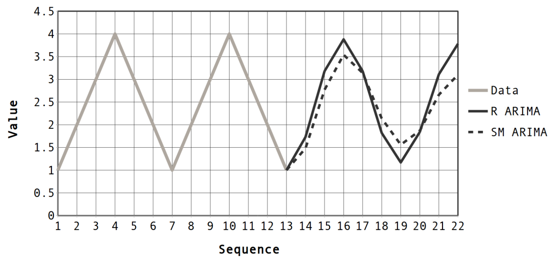


Figure 7.13: Predictions from the ARIMA implementation in R and in Statsmodel using $P = 2$ $D = 0$ $Q = 1$

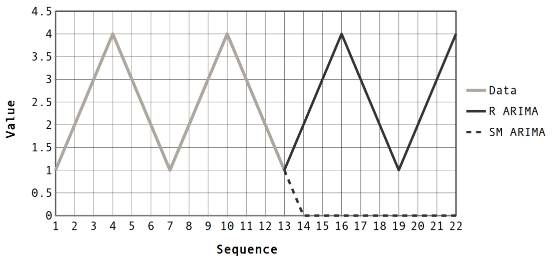


Figure 7.14: Predictions from the ARIMA implementation in R and in Statsmodel using $P = 3$ $D = 0$ $Q = 4$

[8]http://statsmodels.sourceforge.net/0.6.0/generated/statsmodels.tsa.arima_model.ARIMA.html
[9]https://github.com/statsmodels/statsmodels/issues/1155/

R Integration The integration with R from python is made using the rpy2 library[10], where R objects are exposed as instances of Python classes. Using this techniques had a few challenges, such as handling data types between processes and model serialization.

Algorithm Preference Choosing the algorithm is straightforward, by making ARIMA the first choice. However, in certain datasets where ARIMA is not able to fit a model the fallback is Support Vector Regression.

Parameter Optimization Both ARIMA and SVR have parameters that need to be optimized. The parameters of ARIMA are:

- *p:* Autoregressive order
- *d:* Degrees of differencing
- *q:* Moving Average order
- *seasonal:* specification of the seasonal part, if one exists.
- *seasonal period:* period of season
- *seasonal p,d,q order:* similar to the non-seasonal order parameters.

The parameters for SVR are:

- *C:* Penalty parameter
- *epilson:* Penalty parameter of the epilson-SVR model.

The optimization process is based on grid search, where an exhaustive search is performed through a list of manually specified values for each one of the parameters, where for each value a validation algorithm is used in an attempt to find the best value. The reasons behind choosing grid search are detailed in section 7.2.4.

Quality Tiers Considering the strict time limitations of the system, and how resource intensive this task is, two quality tiers were created. The fast tier performs grid-search in the ARIMA model ignoring seasonality, unless manually specified, while in the SVR the grid coverage is smaller. In contrast, the slower tier provides a more thorough search in the SVR parameters and attempts to find the best seasonality parameters in ARIMA.

The validation used during parameter optimization is the Holdout, using the Mean Absolute Error as score function, due to its speed. After finding the best parameter and validating, a final model is built.

Generating the Output

In the task of Automated Forecast the output is a list of pair values for the new predictions. However the user is also interested in the degree of uncertainty of the model.

Degree of uncertainty The degree of uncertainty relates to the error of the model created. There are two main metrics related to the degree of uncertainty: Mean Absolute Error and Mean Absolute Percentage Error.

[10]http://rpy.sourceforge.net/

7.2.7 Automated Clustering

In the context of the Wizdee platform, the task of automated clustering serves a very specific purpose, which is grouping similar instances together so that a user can explore their data and find interesting relationships.

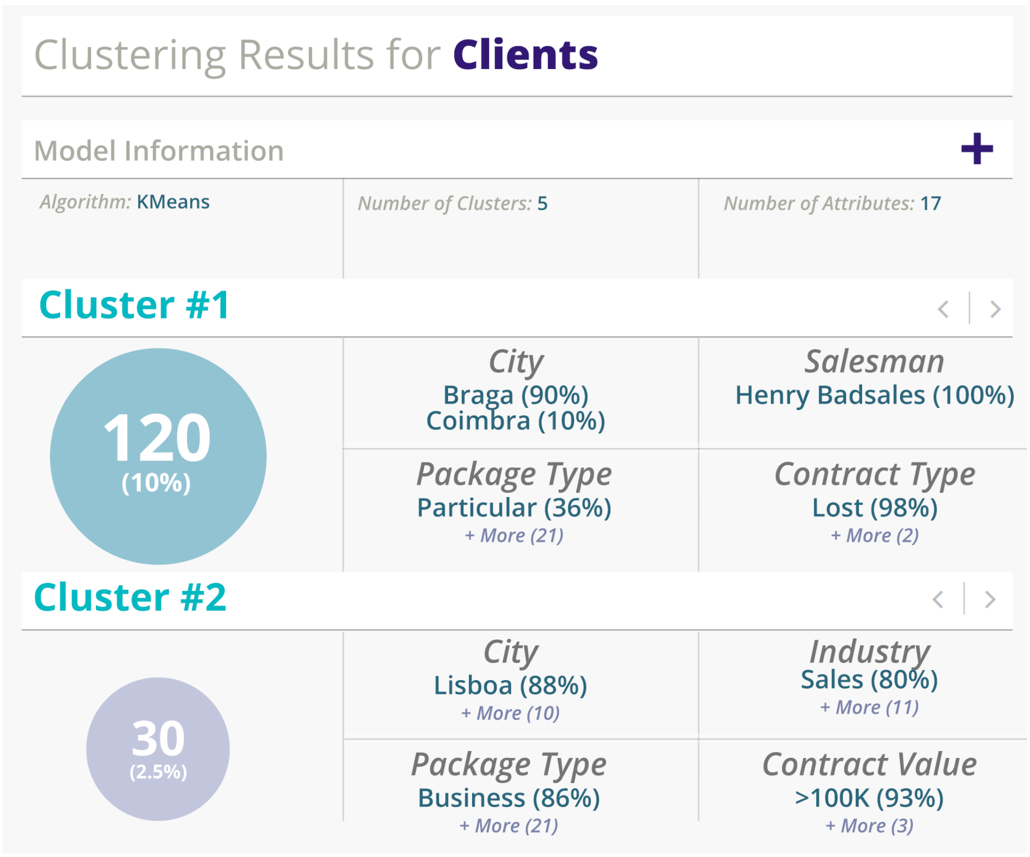
One of the greatest challenges relates to the subjective nature of clustering, where most results depend on the user and the particular problems. For this reason it is common to say that clustering is in the eye of the beholder (Estivill-Castro, 2002).

Thus, at the very least, automating the clustering task means developing all the needed tools and processes so that an end-user can explore their clusters without technical knowledge or know-how, with a comprehensive output. Anyhow, a few techniques are used to provide relevant results automatically, including the number of clusters and algorithm.

Additionally, emphasis on the output representation was put forward, so that the model results could be easily understood. This is based on describing the clusters by analysing the data on each cluster instead of merely assigning a cluster to each instance.

In figure 7.15 we show a mock-up of a clustering request regarding a dataset about clients. The problem at hand is exploring groups of those clients to see what type of patterns arise. A specific goal is not needed, since this is an exploratory task, however it is possible that users could be looking for specific groups or patterns.

Some common goals include finding high-valued clusters and see what attributes they have in common, so that they can focus their business on those clusters. But it is also frequent to find unexpected clusters, such as a cluster composed of lost contracts by a single salesman.


Clustering Results for **Clients**

Model Information +

Algorithm: KMeans | *Number of Clusters:* 5 | *Number of Attributes:* 17

Cluster #1

120 (10%)

City: Braga (90%), Coimbra (10%)

Salesman: Henry Badsales (100%)

Package Type: Particular (36%) *+ More (21)*

Contract Type: Lost (98%) *+ More (2)*

Cluster #2

30 (2.5%)

City: Lisboa (88%) *+ More (10)*

Industry: Sales (80%) *+ More (11)*

Package Type: Business (86%) *+ More (21)*

Contract Value: >100K (93%) *+ More (3)*

Figure 7.15: Automated clustering mock-up.

To check if it is possible to create a clustering model there are only two things that the system takes into consideration: At least twenty lines in the data table and at least two columns in the data table.

Pre-processing

Pre-processing is an important part of most clustering tasks and in this task the following pre-processing techniques are applied:

- Missing Values are imputed with a default value indicating a missing value.
- Strings are converted to binary vector features - as String Columns, explained in detail in 7.2.2. A similar approach has been successfully used in various works such as (Ralambondrainy, 1995).
- Booleans are converted to a binary attribute.
- Dates are converted to a millisecond representation value.
- Numeric attributes are normalized using MinMax to be between 0 and 1.
- String attributes with an high number of unique values are discarded, by using the following equation:

$$\frac{unique\_values}{total\_values} > 0.9$$

Additionally, a cluster structure is kept to map between the generated features and their original data values. This structure is used to create a comprehensive output in a later stage.

Model Creation

After creating the necessary features, the next step is building the model and there are various steps during this stage, which are outlined in figure 7.16

Algorithm There are many types of clustering and various algorithms for each one of those types, as discussed in section 3.2.2. The main motivation for choosing the algorithms was to choose robust and scalable techniques, that had proven to work well in various problems. Sklearn contains an overview of the scalability of some of the clustering algorithms[11] and the most scalable algorithms are: K-Means and DBSCAN.

Other algorithms, such as hierarchical clustering methods, can handle various problems well, however they are discouraged for clustering large data sets (Huang, 1998). These two algorithms are also good choices because they can cover different types of problems. While K-Means is a simple general purpose approach, it can't handle non-linear separable geometry well, while DBSCAN has no issue in that regard (Ester et al., 1996).

Additionally, the motivation behind having to process String attributes is that while there are some clustering algorithms that can handle them, these algorithms only work on numeric values. In K-Means the variables are measured on a ratio scale (Jain and Dubes, 1988), and while some approaches convert this data into numeric values, such as Weka[12][13],

[11]http://scikit-learn.org/stable/modules/clustering.html

[12]http://stackoverflow.com/questions/15637553/weka-simplekmeans-cannot-handle-string-attributes

[13]http://stackoverflow.com/questions/28396974/weka-simple-k-means-handling-nominal-attributes

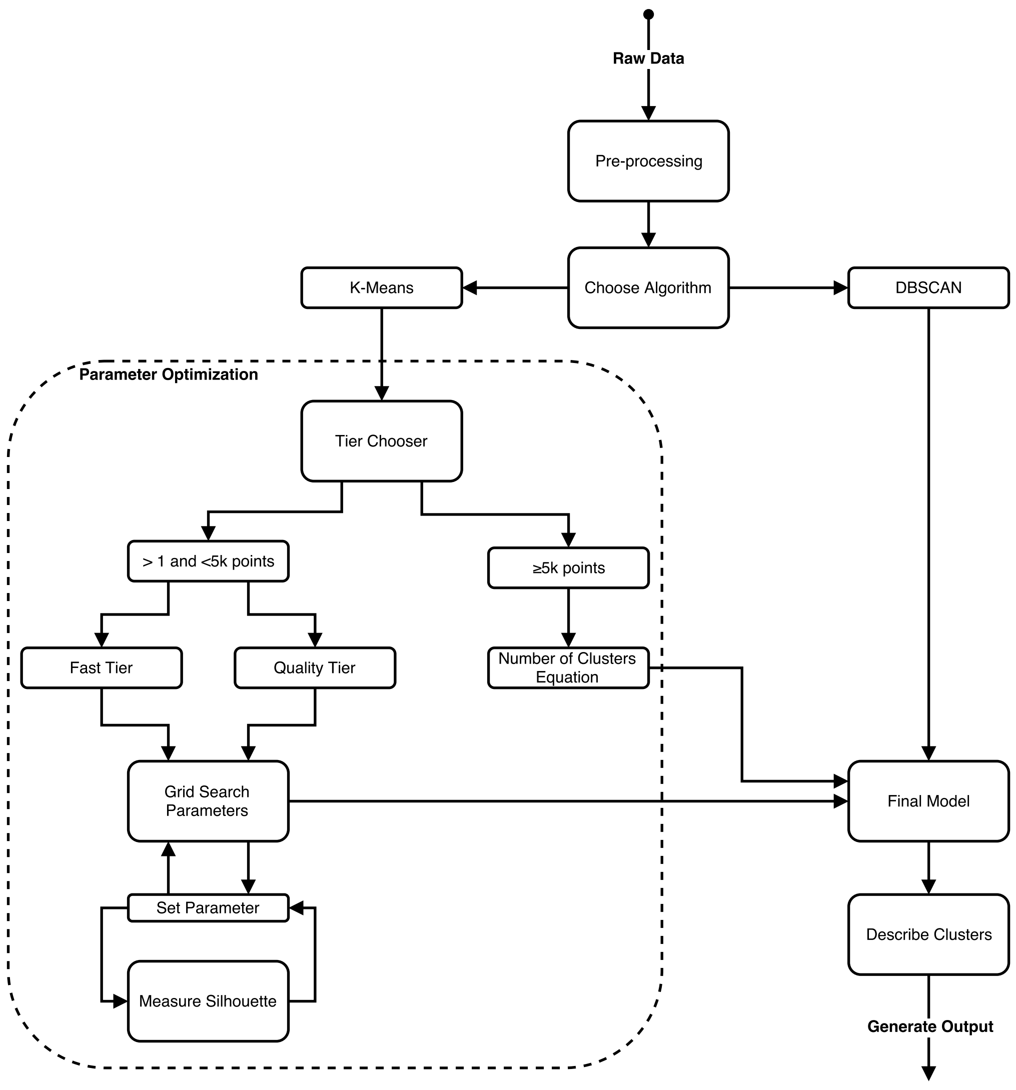


Figure 7.16: Outline of the automated clustering process.

these do not produce meaningful results in the case where categorical domains are not ordered, which is common in real data (Huang, 1998).

Algorithm Preference From an automated point of view, while DBSCAN does not require to specify the number of clusters, K-Means does, and it requires understanding of the data and its scale to choose a meaningful distance threshold ϵ, which can be a more challenging task than to specify the number of clusters.

For this reason, K-Means is the default algorithm, with the possibility of choosing DBSCAN manually. Also, to speed up convergence, a variation of K-Means is used, K-Means++ (Arthur and Vassilvitskii, 2007).

Parameter Optimization Both K-Means and DBScan have parameters that need to be optimized.

The parameter of K-Means is: Number of clusters.

The parameters for DBSCAN are:

- *epilson:* maximum distance between two samples for them to be considered in the same neighbourhood

- *min_samples:* neighbourhood density parameter

The optimization process is based on grid search, where an exhaustive search is performed through a list of manually specified values for each one of the parameters, where for each value a validation algorithm is used in an attempt to find the best value. The reasons behind choosing grid search are detailed in section 7.2.4.

Numbers of Clusters The process to find the number of clusters in an automated way is described in detail in Attachment B.6.

Generating the Output

To create a comprehensive output, instead of showing the cluster index of each instance to the user, as most clustering solutions do, a process to describe each cluster has been developed. In the description of each cluster, the information is split into the attribute type, for numerical attributes, information such as mean, min, max and standard deviation is returned while for String attributes a distribution map is created to calculate the ratio of each label inside that cluster.

7.2.8 Specific Industry Solutions

As the requirements describe in section 5.3, in the context of the Wizdee platform, the task of Specific Industry Solutions relates to the creation of models using the same machine learning components as the other tasks. However, in contrast to the other tasks, specific industry models do not have strict time requirements, nor are they automated. These are models created with manual crafted features and they are trained before a user does a query.

Taking into account Wizdee priorities and direction, this task concerns with Salesforce industry and other CRM industries. More specifically, it has a focus on classification of a specific variable highly relevant in the CRM context.

It is important to note that in contrast to the Automated Classification, the main goal is not to produce readable rules, but to create a high quality model capable of classifying new instances.

Salesforce Classification Model

As mentioned before, the chosen model has to do with a CRM solution, in particular the Salesforce Solution. What Wizdee wants to be able to predict is whether an Opportunity will be closed as Won or Lost - this is the target variable.

Salesforce[14] has several products, but it is best known for their CRM product. This model works on the data model from that product. Considering the size and complexity of the Salesforce system we will only focus on a few important concepts.

Sales Pipeline One of the important concepts of many CRMs is the existence of a sales pipeline, as shown in figure 7.17. A sales pipeline is a description of the sales process, where various steps that salespeople take are described. From a contact with a potential customer, to a lead qualified by the marketing team, to a lead qualified by the sales team, to the validation of that lead into an opportunity, until it is closed. The names

[14]http://www.salesforce.com

of the various stages may change depending on the product, however they typically mean the same thing (Stanton et al., 1991).

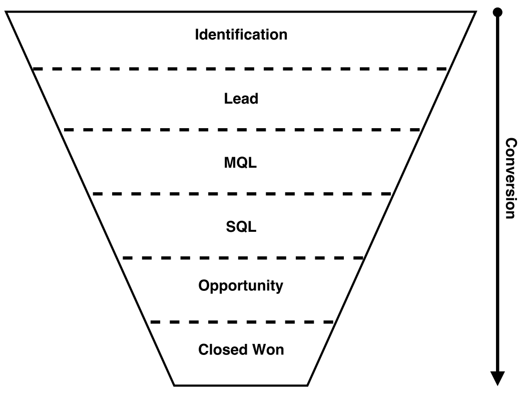


Figure 7.17: Sales pipeline.

There are a few terms that are important to distinguish, typically each company specifies more closely what each term means, but in this context we can do an overview as [15]:

- User: Represents a user inside our organization.
- Contact: A person that is qualified to professionally work with our company.
- Account: A group of persons or company that is qualified to do business with our company. There are various types such as pospect, partener, vendor, etc.
- Lead: A contact or account, typically with little information, that might qualify into an opportunity.
- Opportunity: An established contact or account with a legitimate potential to close.

When an opportunity is closed, it can either be Closed Won or Closed Lost, these two classes will be the target variable of the system.

Dataset

The data schema for the salesforce objects is available online[16], however,a modified version is used inside the Wizdee platform, where the transformed data tables is shown in figure 7.18.

One of the main motivations to manually create a dataset for this task is that we can manually join the tables and choose specific attributes.

The final manually crafted dataset has attributes from User, Account, Campaign, Lead and Opportunities and, as mentioned, the target variable is whether the Opportunity is Closed Won or Closed Lost. The dataset is split, so that it could be used during quality testing in section 8.2.5.

The high-level characteristics of the training dataset are specified in 7.1.

[15]https://developer.salesforce.com/docs/atlas.en-us.api.meta/api/data_model.htm

[16]https://developer.salesforce.com/docs/atlas.en-us.api.meta/api/sforce_api_erd_majors.htm

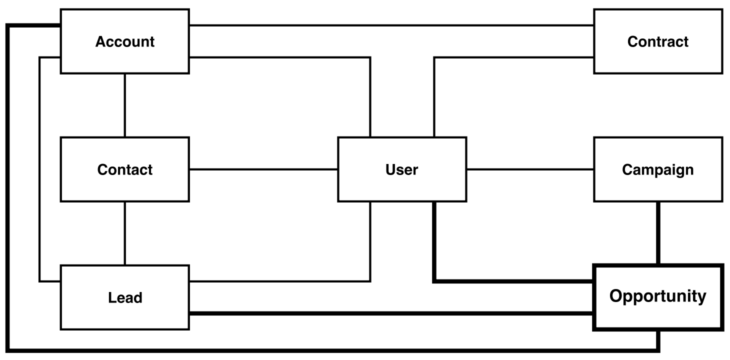


Figure 7.18: Wizdee Salesforce data tables.

N° of Instances	**N° of Attributes**	**Attribute Characteristics**	**Missing Values**	**Target Variable**
1000	36	Numeric/Categorical	Yes	Closed Won (56.7%) Closed Lost (43.3%)

Table 7.1: Salesforce Solution Dataset description.

Some of the features of that dataset are:

- Account: Number of Employees, Account Name, Billing Country, Billing City, Rating, Type
- Lead: Rating, Industry, Country, City, Annual Revenue, State
- User: Division, State, Country
- Campaign: Type, Budgeted Cost, Expected Revenue, Number of Leads, Number of Contacts, Number Sent
- Opportunity: Lead Source

Pre-processing

The same process as Automated Classification , described in section 7.2.4, is used here with added feature selection and extraction techniques: Variance Threshold, Principal Component Analysis and Select K Best.

Model Creation

After creating the necessary features, the next step is building the model. The pipeline here is also similar to the described in section 7.2.4, excluding the output generation step, since we are not concerned in building rules, but with creating a classification model.

Algorithms The algorithms chosen to classify are: Support Vector Machine, RandomForest and DecisionTree as baseline.

Parameter Optimization The method to optimize the parameters is grid search, as explained in section 7.2.4. The parameters depend on the algorithm, it includes the C, *gamma* and *kernel* for the SVMs; *n_estimators* for the RandomForest, which is the number of trees in the forest. For the decision trees, the same *min_samples_split* is used as in section 7.2.4. The validation used during parameter optimization is the Holdout, due to its speed.

Generating the Output

In the task of Specific Industry Solutions, where a model for the Salesforce domain was created, the output is the model itself since the main goal is to create a high quality model capable of classifying new instances.

Chapter 8

Tests and Experimentation

In this chapter, all the tests and experimentations are described and analysed. The tests related to Automatic Visualization are presented in section 8.1 and the tests regarding Automated Data Mining are in section 8.2.

The test environment is described in 8.1.

Specification	**Test Machine**
CPU	Intel ®Core^{TM}i7-4770 3.4Ghz (8 cores)
RAM	16GB DDR3
Operating System	Ubuntu 14.04 64-Bits
Hard Drive	Seagate Barracuda 1TB @ 7200 rpm

Table 8.1: Test environment.

8.1 Automatic Visualization

Quality assessment in visualizations is not a trivial task, as discussed in chapter 3.1. The chosen methodology to handle quality tests is based on the creation of artificial data sets and using expert users, the Quality Assurance team, to evaluate the approach.

The evaluation metric is based on whether the output of the system is considered acceptable and representative of the raw data. It does not take into consideration whether that output is the ideal response, since Wizdee considered that to be susceptible to an high amount of personal bias.

None of the cases in the case base belong to any of these datasets, since that would cause unreliable results.

8.1.1 Datasets

In this task, to assess the quality of the model, a dataset was used containing a large number of queries.

200 Visualizations Dataset This dataset is a group of 200 queries, manually created to be representative of typical business and marketing use cases.

The high-level characteristics of the dataset are specified in 8.2.

Nº of Queries	**Nº of Instances**	**Nº of Attributes**	**Attribute Characteristics**	**Missing Values**
200	100-100k	2-15	Numeric/Categorical/Dates/Boolean	Yes

Table 8.2: 200 Visualizations Dataset description.

8.1.2 Quality Tests

In the quality tests, model assessment is reviewed using the 200 Visualization dataset. As mentioned, the score metric is based on whether an automatic visualization produced an acceptable result, according to the Quality Assurance team.

Results & Analysis The evaluation results for each one of the datasets are shown in table 8.3.

Approach	**Average Score**
Case-base Reasoning	88.5% ($\sigma = 3.5$)
Rule-based System	18.7% ($\sigma = 3.05$)

Table 8.3: Automatic Visualization quality results on the 200 Visualizations Dataset.

Looking at the table 8.3 it is very apparent the difference between each system. However, it is also important to note that visualizations served as a less important feature in the previous Wizdee platform when compared to the latest version. Now, when the user performs a search, visualization for the results is automatically requested and presented to the user. Anyhow, the results are very positive and shows that the approach handles most use cases successfully.

8.1.3 Rule-Based System vs Case Based Reasoning

To take the analysis further, we can compare each system using various queries:

Count Contacts This dataset is made up of a single value containing the number of contacts. The high-level characteristics of the dataset are specified in table 8.4.

Nº of Instances	**Nº of Attributes**	**Attribute Characteristics**	**Missing Values**	**Acceptable Charts**
1	1	Numeric	No	KPI

Table 8.4: Count Contacts Dataset description.

Count Leads by Day This dataset consists of the number of leads throughout the days. The high-level characteristics of the dataset are specified in table 8.5.

Nº of Instances	**Nº of Attributes**	**Attribute Characteristics**	**Missing Values**	**Acceptable Chart**
1986	2	Numeric/Date	No	Line/Bar Chart

Table 8.5: Count Leads by Day Dataset description.

Count Opportunities by Stage This dataset has information about the number of opportunities at each stage of the opportunities pipeline. The high-level characteristics of the dataset are specified in table 8.6.

Nº of Instances	Nº of Attributes	Attribute Characteristics	Missing Values	Acceptable Chart
10	2	Numeric/Categorical	No	Pie/Bar Chart

Table 8.6: Count Opportunities by Stage Dataset description.

Count Leads by Stage by City This dataset represents a complex scenario with an amount of leads by each stage of the pipeline by city. The complexity comes from not every city having a stage, resulting in many missing values. The other challenge is related to the large amount of cities to display. The high-level characteristics of the dataset are specified in table 8.7.

Nº of Instances	Nº of Attributes	Attribute Characteristics	Missing Values	Acceptable Chart
998	3	Numeric/Categorical	Yes	Bar Chart/Matrix Plot

Table 8.7: Count Leads by Stage by City Dataset description.

Results & Analysis

- **Count Contacts** In this dataset, we can see the output from Case Based Reasoning (CBR) in figure 8.1, where the system chose to do a KPI. The figure in 8.2 shows the output from the Rule-based system, a Gauge Chart. While the Gauge Chart could be a possible choice to do manually, it does not make much sense to do it in autonomous way since a Gauge chart requires user-defined minimum and maximum values.

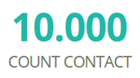


Figure 8.1: Output for the Count Contacts Dataset using CBR.

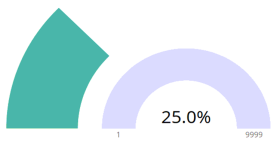


Figure 8.2: Output for the Count Contacts Dataset using the Rule system.

- **Count Leads by Day** In this dataset, we can see the output from CBR in figure 8.3, where the system chose to do a Line Chart. The figure in 8.4 shows the output from the Rule-based system, a Histogram, which is not an acceptable chart. The CBR system also chose the correct axes, with the dates in the X axis and the Count Lead in the Y axis.

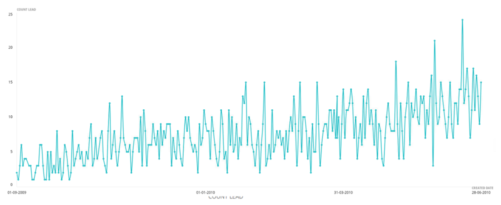


Figure 8.3: Output for the Count Leads by Day Dataset using CBR.

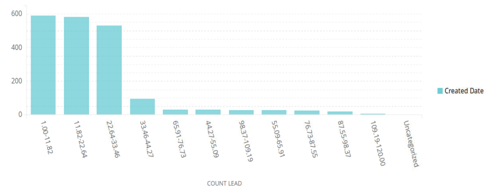


Figure 8.4: Output for the Count Leads by Day Dataset using the Rule system.

- **Count Opportunities by Stage** In this dataset, we can see the output from CBR in figure 8.5, where the system chose to do a Pie Chart. The figure in 8.6 shows the output from the Rule-based system, also a Pie Chart. The difference between the results is related to the chart model processing, where the chart model processing in the rule-based system did not use the Stage column to create the labels. Additionally, this is one of those situations where the number of slices is big and goes over the threshold. However, it is small enough that users still want to see all the slices and the dynamic slice technique decided to avoid having an *others* slice.

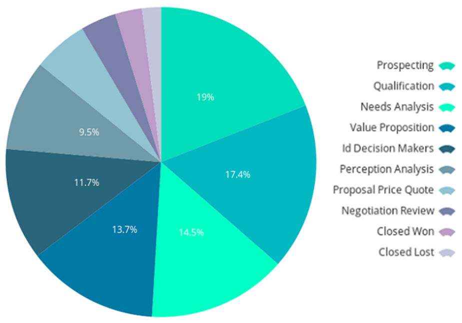


Figure 8.5: Output for the Count Opportunities by Stage Dataset using CBR.

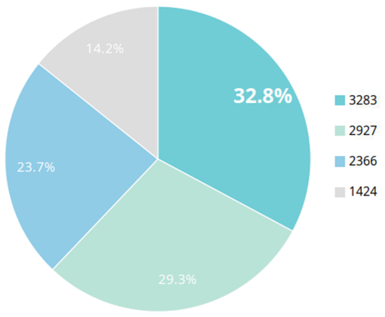


Figure 8.6: Output for the Count Opportunities by Stage Dataset using the Rule system.

- **Count Leads by Stage by City** In this dataset, we can see the output from CBR in figure 8.7, where the system chose to do a Bar Chart. The figure in 8.8 shows the output from the Rule-based system, however, the output was not created automatically since that would have resulted in an Histogram. In this example the axes and chart type were given manually so we could compare how both system handle this complex scenario. The main difference is the amount of cities shown in the chart, while the chart model processing in the rule based system did not limit the number of cities, the CBR approach limits using the most relevant cities, creating a cleaner chart.

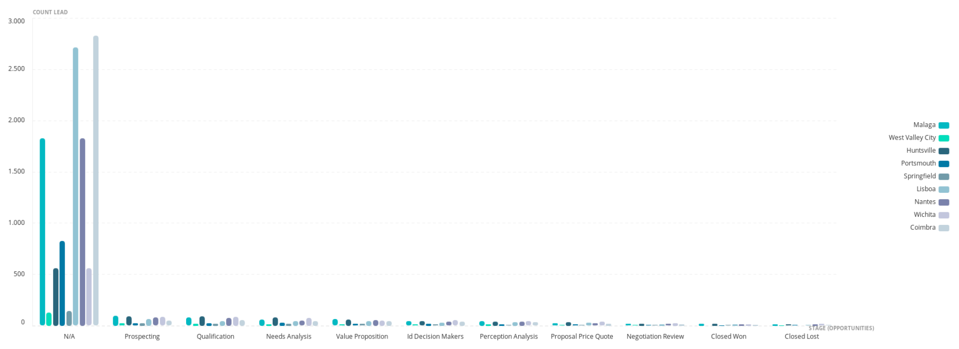


Figure 8.7: Output for the Count Leads by Stage by City Dataset using CBR.

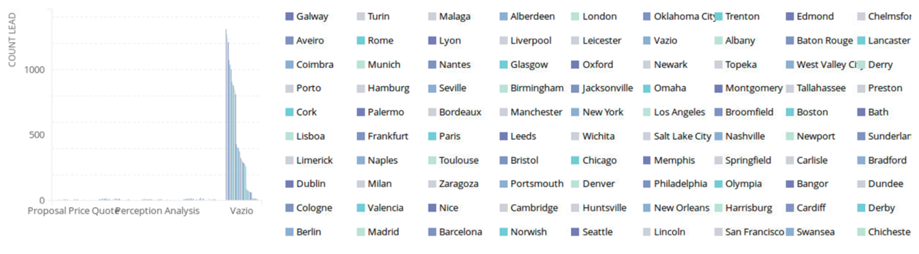


Figure 8.8: Output for the Count Leads by Stage by City Dataset using the Rule system.

Pagination A test using the pagination system is shown with the query *Leads by Country by Week*, where a sample of the page 1 results is shown in 8.9 and a sample of page 2 results is shown in 8.10. In this case we can see that it is capable of pagination with various data types, while keeping track of the series.

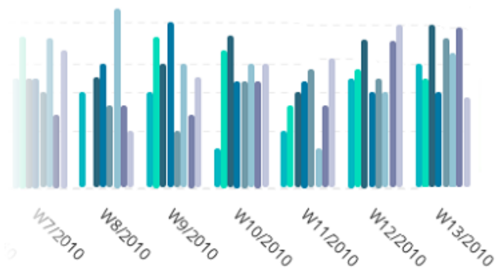


Figure 8.9: Page 1 of Leads by Country by Week using a Bar Chart.

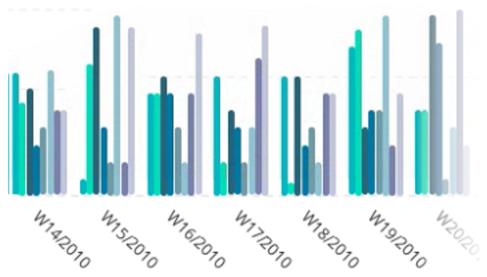


Figure 8.10: Page 2 of Leads by Country by Week using a Bar Chart.

8.1.4 Performance Tests

For the performance tests, 30 runs were executed on the 200 Visualizations Dataset and various performance metrics were extracted for the various steps. It is important to note that the requirement describes a time limitation of *5s.* excluding database accesses. However, total time includes that database accesses and other secondary time expenses.

Results & Analysis This sections analyses the performance results, an overview of the performance is shown on the table 8.8.

Dataset	Total Time	Metada Extraction	Case Mapping	Apply Chart
200 Visualizations	3.65s ($\sigma = 4.13$)	0.0007 ($\sigma = 0.00$)	0.004s ($\sigma = 0.01$)	2.7s ($\sigma = 3.6$)

Table 8.8: Automatic Visualization performance results.

Looking at the table 8.8, the results are quite reasonable and within the time limitations. It is possible to see that the majority of time is spent on applying the chart model.

Additionally, we can analyse the performance by taking a look at the relationship between instances and the total time. Of interest, is that the performance varies widely depending on how many dimensions it has. In figure 8.11, a dataset with two dimensions is used, and it is possible to observe that the total time does not reach $5s$ even with 4M instances, which is above most tested use cases. In contrast, taking a look at figure 8.12, it is possible to see that after 250k instances it goes above $5s$.

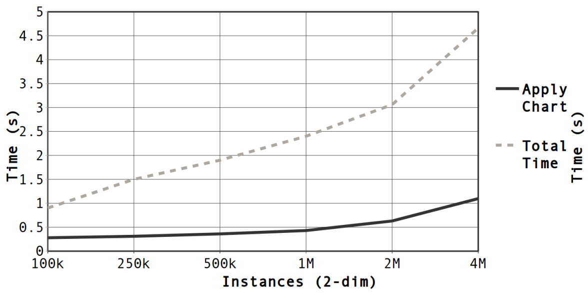


Figure 8.11: Relationship between number of instances and performance in Automatic Visualization, under 2 dimensions.

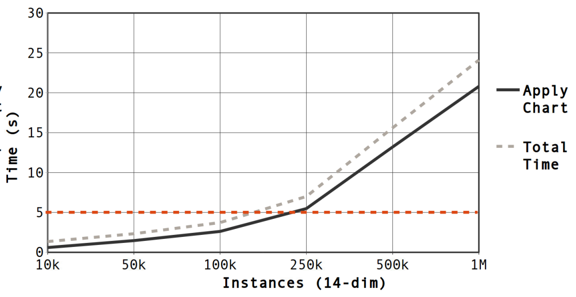


Figure 8.12: Relationship between number of instances and performance in Automatic Visualization, under many dimensions.

8.1.5 Remarks

For the task of Automatic Visualization, both the quality and performance of the system are shown to be quite sensible. The quality results are of subjective nature but using the help of the Quality Assurance team, it was possible to obtain a quantitative metric that shows how much of an improvement this system was. Nonetheless, it is important to note that the previous approach did not receive as much attention and there could be ways to improve it. Anyhow, while there is room to improvement, the solution already produces very capable visualizations under strict time limits, which is why it was integrated into the Wizdee platform, even while at prototype stage.

8.2 Automated Data Mining

In this section, the tests and experiments done regarding automated data mining are described and analysed for the following tasks: Classification, Outlier Detection, Forecast, Clustering and Specific Industry Solutions.

It should be noted that the main goal of these tasks is not to compete against manually crafted models - typically done by experts - in terms of model quality. The focus is on the ability to create good models in an automated way, so that a layman user can use these techniques and take valuable information from them without requiring extensive knowledge about the subject.

Additionally, as described in the performance requirements 5.6, the time limit is *30s* while the ideal time limit is *5s* to keep the users' attention and in the sections below, solutions to reach the ideal time limit will be discussed.

8.2.1 Automated Classification

In the task of automated classification, in section 8.2.1, it is of most importance the quality of the created models, the usefulness of the outputted rules and how fast they perform. Since this is a task that is typically performed by highly technical professionals with carefully created features and with loose time requirements, the goal is to obtain reasonable models under strict time limits. In this subsection, a few popular test datasets will be used to test the quality and performance.

Datasets

Bank Marketing Dataset This dataset(Moro et al., 2011) has to do with marketing campaigns of a Portuguese banking institution. These campaigns were based on phone calls and the target variable concerns whether that contact resulted in a subscription. The high-level characteristics of the dataset are specified in table 8.9.

Nº of Instances	Nº of Attributes	Attribute Characteristics	Missing Values	Target Variable
41188	20	Numeric/Categorical/Dates	Yes	Yes (11%) No(89%)

Table 8.9: Bank Marketing Dataset description.

The dataset includes the following features:

- Personal Data: age, job, marital status, education, housing, loan, credit in default
- Economic Indicators: consumer price index, consumer confidence index, euribor 3 month, number employed, employment variation rate
- Context: contacts during this campaign, number of days after last contact, previous contacts, outcome of previous contacts, contact
- Contact Time: month, day of week, duration

Breast Cancer Dataset This dataset(Wolberg and Mangasarian, 1990) relates to the diagnostic of breast cancer with features based on a digitized image describing characteristics of a cell nuclei. The high-level characteristics of the dataset are specified in table 8.10.

Nº of Instances	Nº of Attributes	Attribute Characteristics	Missing Values	Target Variable
699	10	Numeric	Yes	Benign (65.5%) Malignant(34.5%)

Table 8.10: Breast Cancer Dataset description.

The dataset includes the following features:

- Nuclei Data: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial size, bare nuclei, sample code, bland chromatin, normal nucleoli, mitoses

Adult Dataset This dataset(Kohavi, 1996) is based on census data with the target variable as the prediction of whether income exceeds \$50k in the USA. The high-level characteristics of the dataset are specified in table 8.11.

Nº of Instances	Nº of Attributes	Attribute Characteristics	Missing Values	Target Variable
32561	14	Numeric/Categorical	Yes	$>$50k (23.93%) $\leq$50k(76.07%)

Table 8.11: Adult Dataset description.

The dataset includes features such as:

- Personal Data: age, marital-status, relationship, race, sex, native-country, weight
- Job: workclass, occupation, hours-per-week
- Education: education, education-num
- Economic: capital-gain, capital-loss

Quality Tests

In the quality tests, model assessment is reviewed with the various datasets. The test uses 10-folds validation, using the average score of the folds and its standard deviation, using three metrics: Accuracy, F1 Score and Area under the curve.

Results & Analysis The 10-folds validation results for each one of the datasets are shown in table 8.12:

Metric	Bank	Breast Cancer	Adult
Accuracy	90.73% ($\sigma = 0.38$)	94.82% ($\sigma = 0.20$)	85.04% ($\sigma = 0.11$)
F1 Score	63.25% ($\sigma = 1.17$)	92.86% ($\sigma = 0.29$)	68.76% ($\sigma = 0.17$)
AUC	82.05% ($\sigma = 0.93$)	94.81% ($\sigma = 0.22$)	79.44% ($\sigma = 0.13$)

Table 8.12: Automated Classification quality results.

From the table above we can see that the results look very positive across all the datasets, furthermore, the low standard deviation, shows that the results are consistent across each fold. Nonetheless, it is possible to observe that in unbalanced and more complex problems, such as the Bank Dataset or the Adult Dataset, the F1 Score is lower, which is to be expected. Additionally, we can find other articles using these datasets, which is useful to compare the results:

- **Bank Dataset** There are various articles using the Bank Dataset, some of the most popular ones are shown in table 8.13. The first article was made by the original creators, where they used various algorithms, including decision trees. Using the AUC metric, it is possible to see a small negative difference between the results, yet, when compared to the other articles, also using tree-based algorithms, the approach here obtained higher results. Considering the model was created in an automated way, against real world data, the results are considered very good.

Article	Metric	Algorithm	Results	Difference
Using data mining for bank direct marketing: An application of the crisp-dm methodology(Moro et al., 2011)	AUC	DT	86.8%	**-4.75**
A Comparison of Different Classification Techniques for Bank Direct Marketing (Wisaeng, 2013)	Accuracy	J48	75.52%	**+14.21**
Bank Direct Marketing Analysis of Data Mining Techniques (Elsalamony, 2014)	Sensitivity	C5.0	59.06%	**+11.84**

Table 8.13: Results from other reports using the bank dataset.

- **Breast Cancer Dataset** The Breast Cancer Dataset is an old but widely popular dataset. To compare the results, some of the most popular and relevant works were chosen and shown in table 8.14. Of particular interest there is the first article which was made by the original creators, where a negative difference exists, albeit, it is small and does not use a tree-based algorithm. The other highly relevant article is the last, where it shows how two different tree-based algorithms perform. Overall, the difference between these articles and results obtained in an automated manner are not significant, creating satisfactory results according to Wizdee.

Article	Metric	Algorithm	Results	Difference
Multisurface method of pattern separation for medical diagnosis applied to breast cytology. (Wolberg and Mangasarian, 1990)	Accuracy	Multisurface	95.9%	**-1.08**
Selecting typical instances in instance-based learning(Zhang, 1992)	Accuracy	TIBIL	93.4%	**+1.42**
Decision tree construction via linear programming (Bennett, 1992)	Accuracy	C4.5	96.2%	**-1.38**
	Accuracy	CART	94.7%	**+0.12**

Table 8.14: Results from other reports using the bank dataset.

- **Adult Dataset** The Adult Dataset is a complex but also popular dataset since it deals with a large amount of attributes and instances, some of the most relevant and popular articles are shown in table 8.15. Against the first article there is a slightly positive difference, this also happens in the last article when used the C4.5 algorithm. Using a custom-made solution shows a slight negative difference, however, when considered this was obtained using an automated process, the results are satisfactory.

Article	Metric	Algorithm	Results	Difference
Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid(Kohavi, 1996)	Accuracy	C4.5	84.46%	**+0.48**
Comparing Bayesian Network Classifiers (Cheng and Greiner, 1999)	Accuracy	BAN	85.82%	**-0.78**
A simple, fast, and effective rule learner (Cohen and Singer, 1999)	Accuracy	C4.5	84.00%	**+1.04**
	Accuracy	SLIPPER	85.30%	**-0.26**

Table 8.15: Results from other reports using the adult dataset.

Another interesting aspect is what the parameter optimization process finds as the best minimum node splitting size. For each one of the datasets this is found in the table 8.16 where it shows the Bank Dataset, which has the largest amount of instances, using a large value while the Breast Cancer, a smaller dataset, to use a much smaller value - as expected.

Dataset	min_samples_split
Bank	400
Breast Cancer	5
Adult	250

Table 8.16: Automated Classification best splitting sizes.

To complement the model quality analysis, a test showing the relationship between instances and scores was made in figure 8.13, 8.14 and 8.15. From the results it is possible to see that in all of the datasets the results remain reasonable even with a low percentage of the dataset, even in the Breast Cancer dataset which has a lower number of instances. Thus, it is possible to observe that, in these datasets, it is possible to obtain a representative sample of the original dataset, in terms of results, with a smaller percentage of instances. This insight can be useful for performance reasons.

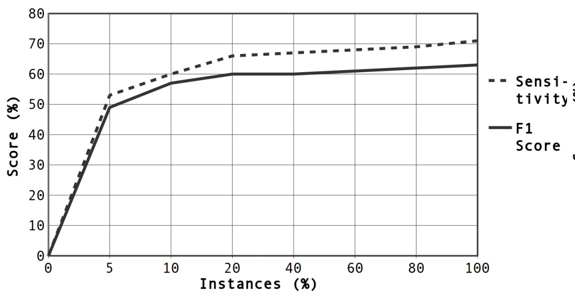


Figure 8.13: Relationship between instances and score on the bank dataset.

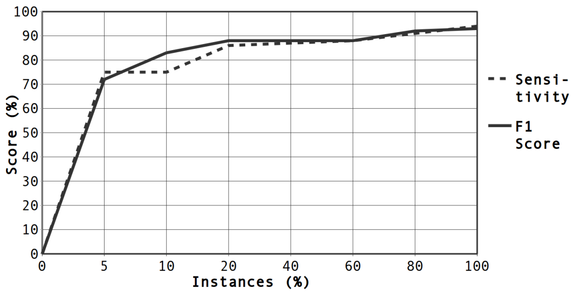


Figure 8.14: Relationship between instances and score on the breast cancer dataset.

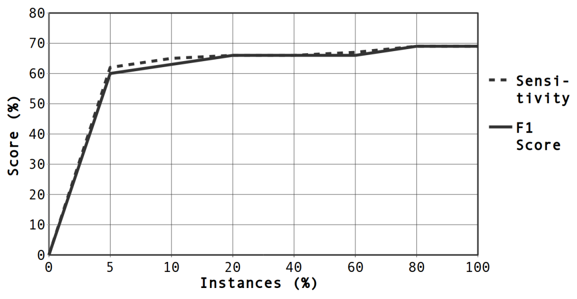


Figure 8.15: Relationship between instances and score on the adult dataset.

Additionally, the most significant rules for each one of the datasets are represented as:

- **Bank Dataset** In the Bank Dataset, the most relevant rules are shown in table 8.17, where it is possible to determine that the duration is an important attribute, used in both of the most relevant rules. This is also supported in the article made by the original creators(Moro et al., 2011). Anyhow, to understand what causes a contact to convert might require looking at more rules to have a better coverage. In contrast, for unconverted contacts this might not be needed.

Target	Coverage	Most Relevant Rule
Yes	10%	duration > 198.5 number_days_last_contact > 16 euribor_3_month < 5087
No	35%	duration < 181.5 consumer_confidence_index > -46.65

Table 8.17: Most significant rules for the bank dataset.

- **Breast Cancer Dataset** The most relevant rules for the Breast Cancer Dataset, shown in table 8.18, express that in this dataset a large number of attributes is not used to obtain the most relevant rules. The uniformity of the cell size is present in both rules which may indicate it is an important attribute. This is also supported in (Elsayad and Elsalamony, 2013). The coverage of either rule is very high, demonstrating that these rules alone are representative to assess a malignant or benign classification.

Target	Coverage	Most Relevant Rule
Malignant	53%	Uniformity_of_cell_size > 4.5 Bland_Chromatin > 4.5
Benign	88%	Uniformity_of_cell_size < 2.5 Bare_nuclei < 3.8 Clump_thickness < 7.5

Table 8.18: Most significant rules for the breast cancer dataset.

- **Adult Dataset** In the Adult Dataset, the most relevant rules, described in table 8.19, show that this might be a more complex dataset due to the fact that the most relevant rules have more attributes than what appears in the other datasets. The most important attributes to asses whether income exceeds 50k seem to be marital-status, education and age.

Target	Coverage	Most Relevant Rule
>50k	8%	marital-status ≠ Married-civ-spouse education-num > 12.5 capital-gain > 5095.5 age < 62.6
≤50k	12%	marital-status = Married-civ-spouse capital-gain < 7073 education-num < 12.5 age < 21.5

Table 8.19: Most significant rules for the bank dataset.

Performance Tests

For the performance tests, 30 runs were executed for each one of the datasets and various performance metrics were extracted for the various steps. It is important to note that preprocessing includes the time needed to load the data due to optimizations made that apply the preprocessing during that step. Finally, as mentioned before, the original requirement describes a time limitation of *30s*, with the ideal time being under *5s*.

Results & Analysis This sections analyses the performance results, an overview of the performance is shown on the table 8.20.

Dataset	Total Time	Preprocessing	Grid-Search (Multithread)	Grid-Search (Single-thread)	Validation
Bank	12.63s ($\sigma = 0.37$)	1.53 ($\sigma = 0.22$)	4.51s ($\sigma = 0.12$)	10.87s ($\sigma = 0.09$)	5.58s ($\sigma = 0.07$)
Breast Cancer	0.08s ($\sigma = 0.01$)	0.01s ($\sigma = 0.00$)	0.03s ($\sigma = 0.01$)	0.16s ($\sigma = 0.01$)	0.03s ($\sigma = 0.01$)
Adult	8.88s ($\sigma = 0.15$)	1.05s ($\sigma = 0.04$)	2.58s ($\sigma = 0.07$)	8.50s ($\sigma = 0.07$)	4.66s ($\sigma = 0.10$)

Table 8.20: Automated Classification performance results.

Looking at the table 8.20, the results are quite reasonable and well within the time limitations. To note, only the Breast Cancer Dataset is below the ideal time limit, which was expected due to a much smaller sample size. Of particular importance is the difference between multithreaded grid-search and single-thread grid search, reaching a decrease of 368% in time on average across the datasets.

In the table 8.21 we can take a look at the performance of the various high level tasks and observe that the majority of the time is spent during the model training as expected.

Dataset	Train Model	Apply Model	Create Rules
Bank	11.79s ($\sigma = 0.44$)	0.54s ($\sigma = 0.07$)	0.01s ($\sigma = 0.00$)
Breast Cancer	0.07s ($\sigma = 0.00$)	0.01s ($\sigma = 0.00$)	0.01s ($\sigma = 0.00$)
Adult	8.30s ($\sigma = 0.14$)	0.52s ($\sigma = 0.67$)	0.01s ($\sigma = 0.00$)

Table 8.21: Automated Classification performance of training the model versus applying the model.

To reach the ideal time of *5s* for the two datasets above in table 8.21, there are two approaches proposed: Reduce the grid search vector size or reduce the number of instances through sampling.

- **Bank Dataset** Looking at figure 8.16 and 8.17, it is possible to observe that both can influence greatly the performance by decreasing any of its values. However, decreasing the number of instances would be preferable since it is possible to achieve the ideal time with 40% of the instances leading to a small decrease in quality according to figure 8.13.

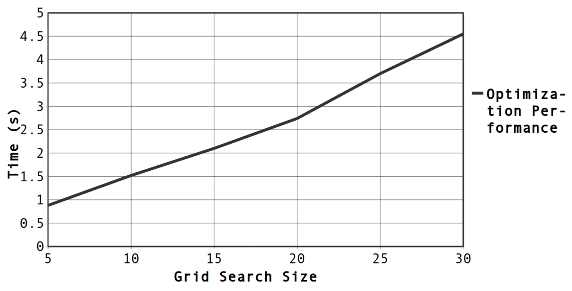


Figure 8.16: Relationship between grid-search size and performance on the bank dataset.

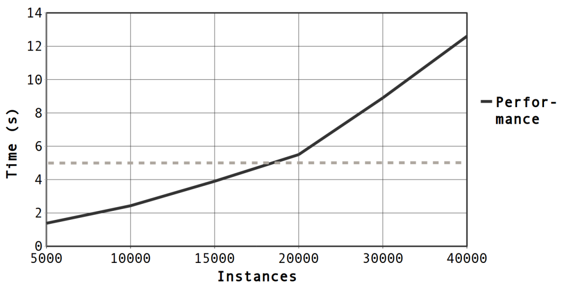


Figure 8.17: Relationship between the number of instances and performance on the bank dataset.

- **Adult Dataset**

  According to figures in 8.18 and 8.19, a similar situation to the Bank Dataset occurs where it's more sensible to decrease the number of instances, in this case to 50%, leading to a small decrease in quality according to figure 8.15.

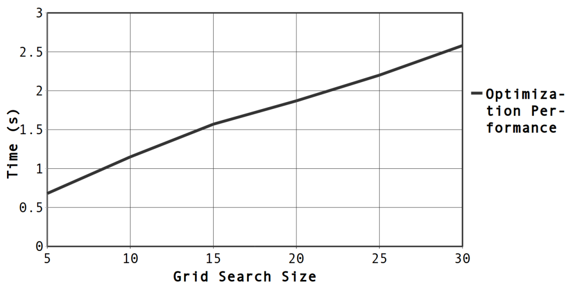


Figure 8.18: Relationship between grid-search size and performance on the adult dataset.

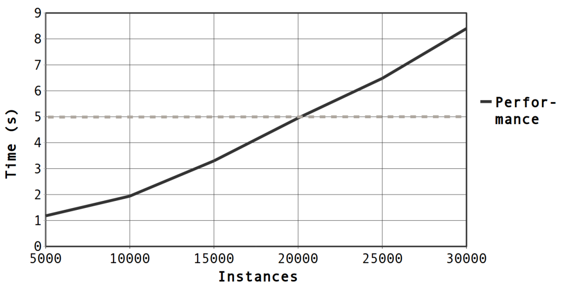


Figure 8.19: Relationship between the number of instances and performance on the adult dataset.

Remarks Finally, for the task of Automated Classification, performance of the system is shown to respect the requirements, however, the approach also produces similar results to those created by hand-crafting features in certain situations, which was above expectations, therefore, making this a good solution to handle the task of Automated Classification in a production setting.

8.2.2 Automated Outlier Detection

Assessing the quality and performance of the models created during automated outlier detection in section 8.2.2 is an important task. However, this evaluation is not an easy task because it suffers from subjective interpretation. This evaluation issue has been presented many times in literature (Aggarwal, 2013). A common technique is to use case studies in order to provide an intuitive and qualitative evaluation. This will be the main technique used to assess the quality of the models, where a few datasets were created with the help of the quality assurance team.

Datasets

Basic 2D Dataset This dataset is an abstract dataset manually created to show a scenario where the outliers clearly standout from the data. The high-level characteristics of the dataset are specified in table 8.22.

Nº of Instances	**Nº of Attributes**	**Attribute Characteristics**	**Missing Values**	**Outliers**
38	2	Numeric	No	Outlier (10.5%) Non-Outlier(89.5%)

Table 8.22: Basic 2D Dataset description.

Contacts by Week This dataset was created with real world data, it describes the number of contacts throughout the weeks, from 2009 to 2015. The data contains missing values and finding those with least amount of false positives it the main goal of this dataset. The high-level characteristics of the dataset are specified in table 8.23.

Nº of Instances	**Nº of Attributes**	**Attribute Characteristics**	**Missing Values**	**Outliers**
284	2	Numeric/Date	Yes	Outlier (3.1%) Non-Outlier(96.9%)

Table 8.23: Contacts by Week Dataset description.

Expected Revenue by Amount As the one above, this dataset was created with real world data, it describes the expected revenue from an opportunity and the real amount. The goal of this dataset is to check situations where the expected revenue is far off from the supposed true amount. The high-level characteristics of the dataset are specified in table 8.24.

Nº of Instances	**Nº of Attributes**	**Attribute Characteristics**	**Missing Values**	**Outliers**
156	2	Numeric	No	Outlier (5.1%) Non-Outlier(94.9%)

Table 8.24: Expected Revenue by Amount Dataset description.

Quality Tests

As mentioned, since the evaluation of outlier models is complex, use cases were created using a review process with the quality assurance team, outliers were marked in those datasets and using that as tests, the results are presented below, using these metrics: True positives and False positives.

Results & Analysis The results for each one of the datasets are shown in table 8.25:

Metric	Basic 2D	Contacts by Week	Expected Revenue by Amount
True Positives	4	9	5
False Positives	0	4	8
Dataset Marked Outliers	4	9	8

Table 8.25: Automated Outlier Detection quality results.

From the table 8.25 we can see that the approach is quite good at finding what has been marked as an outlier. However, it also found some false positives. Using these results by themselves is not enough to draw conclusions about the approach quality, nonetheless, since the tests use two dimensions it is possible to create visualization so we can take a better look at the data and what is being considered an outlier:

- **Basic 2D Dataset** In this visualization, shown in figure 8.20, we can see the outliers being displayed in orange. According to the results in the table 8.25, no false positives have been found and all the outliers were found, this is expected, since those points are far from all the others.

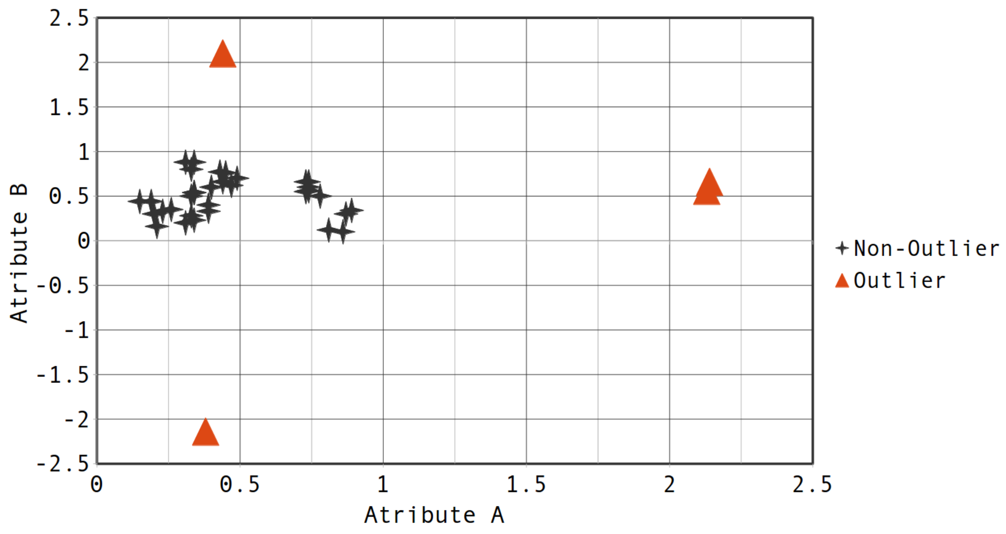


Figure 8.20: Basic 2D Dataset and its predicted outliers.

- **Contacts by Week Dataset** In this visualization, shown in figure 8.21, the outliers are displayed in orange. According to the results in the table 8.25, a few false positives have been found and all the outliers were found. The outliers represent missing information, their value is zero, and the approach found them all. However, since the data varies throughout the weeks, the highest values near the final weeks have also been considered outliers. Ultimately, it is subjective whether those points should be considered outliers, since those points do deviate from the rest of the data.

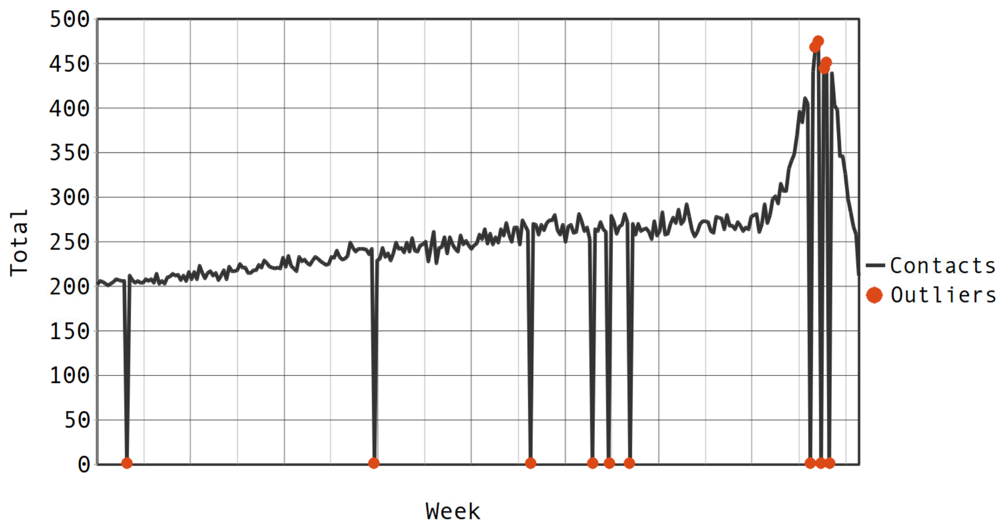


Figure 8.21: Contacts by Week Dataset and its predicted outliers.

- **Expected Revenue by Amount** This dataset relates to the relationship between expected revenue and the true amount of an opportunity. The outliers marked manually by Wizdee is shown in figure 8.22 while the predicted outliers are shown in figure 8.23. We can see that the algorithm has chosen a few different outliers. However, considering the unsupervised nature of the algorithm and the subjective nature of the task, upon discussion of the results, most of these were found to also be viable outliers.

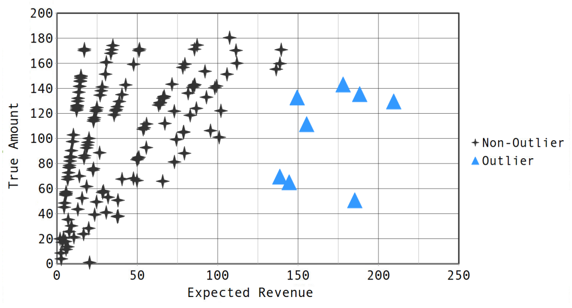


Figure 8.22: Expected Revenue by Amount Dataset and its marked outliers.

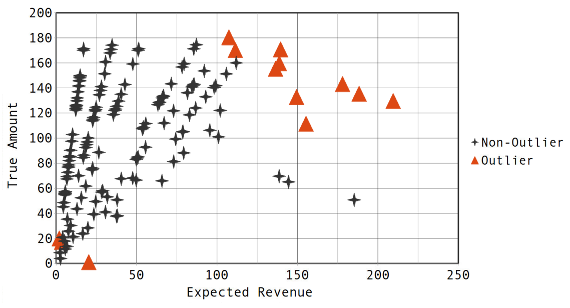


Figure 8.23: Expected Revenue by Amount Dataset and its predicted outliers.

Performance Tests

For the performance tests, 30 runs were executed for each one of the datasets and various performance metrics were extracted for the various steps. Additionally, as mentioned before, the requirement describes a time limitation of *30s*, with the ideal time being *5s*.

Results & Analysis This sections analyses the performance results, an overview of the performance is shown on the table 8.26.

Dataset	Total Time	Preprocessing	Statistics	Distance
Basic 2D	0.12s ($\sigma = 0.01$)	0.01 ($\sigma = 0.00$)	0.01s ($\sigma = 0.01$)	0.02s ($\sigma = 0.01$)
Contacts by Week	0.16s ($\sigma = 0.01$)	0.02s ($\sigma = 0.00$)	0.01s ($\sigma = 0.01$)	0.03s ($\sigma = 0.01$)
Expected Revenue by Amount	0.13s ($\sigma = 0.01$)	0.01s ($\sigma = 0.00$)	0.01s ($\sigma = 0.01$)	0.02s ($\sigma = 0.01$)

Table 8.26: Automated Classification performance results.

Looking at the table, the results are very good and well within the time limitations. Furthermore, we can conclude that most of the time is spent on outside factors, such as data retrieval from the database, since the time spent in the various stages accounts for a small percentage of the total time.

Considering the datasets used were of small size, a detailed analysis was made using larger samples sizes in figure 8.24. From the figure it is possible to see that even in large datasets, such as 50000 instances, the time remains under one second which is under the ideal time limit of 5s.

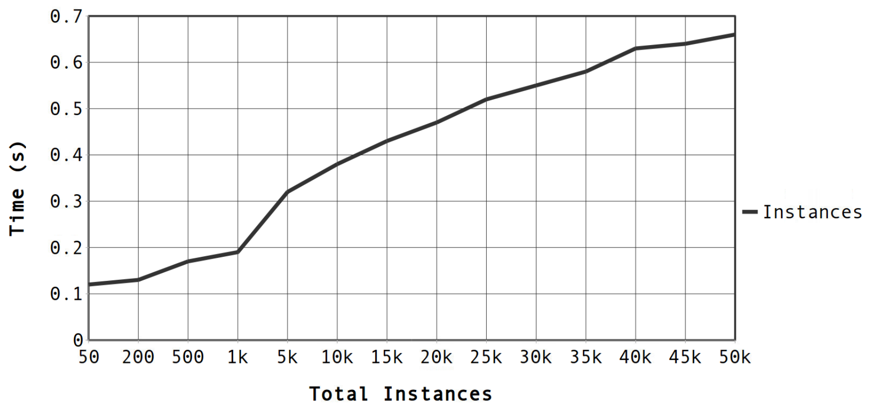


Figure 8.24: Relationship between the number of instances and the execution time.

Remarks Finally, for the task of Automated Outlier Detection, while a relatively simple technique was chosen, the results are reasonable according to Wizdee, with the added benefit of it being a flexible approach, where the user can control the distance threshold. In terms of performance it excels, handling very large datasets with ease which is a critical feature for this task.

8.2.3 Automated Forecast

In the task of automated forecast, in section 7.2.6, the quality and performance are critical elements of this system, considering the strict time limits. In this subsection, a few popular test datasets will be used to test the quality and performance.

Datasets

Internet Traffic Dataset This dataset(Cortez et al., 2012) relates to internet traffic data (in bits) from a private ISP. It was collected from 7 June to 31 July. The original data was collected at five minute intervals, however, this work uses a processed version that contains one point per hour. The high-level characteristics of the dataset are specified in table 8.27.

Instances	**Training Instances**	**Test Instances**	**Missing Values**
1230	1206	24	No

Table 8.27: Internet Traffic Dataset description.

The distribution between test and training datasets was set to be similar to other works. A sample of the dataset in shown in figure 8.25.

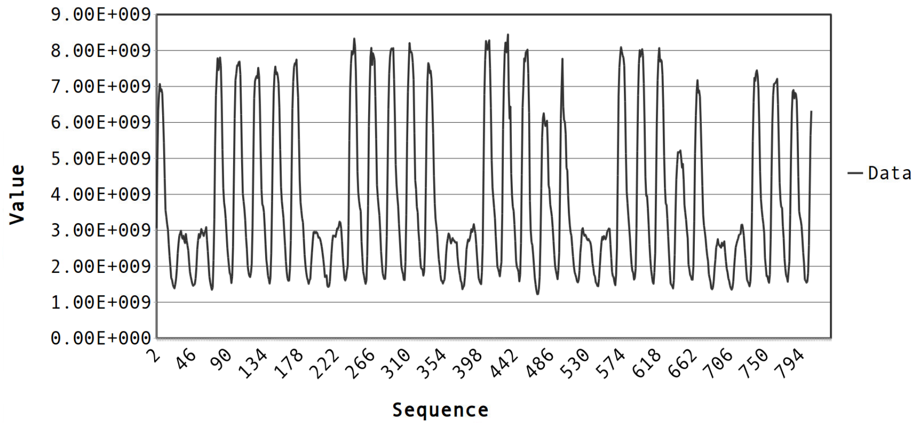


Figure 8.25: Internet Traffic Dataset

Gas Furnace Dataset This dataset(Box et al., 2011) relates to data about a gas furnace. It is a commonly used dataset in time series analysis. The high-level characteristics of the dataset are specified in table 8.27.

Instances	**Training Instances**	**Test Instances**	**Missing Values**
296	150	146	No

Table 8.28: Gas Furnace Dataset description.

The distribution between test and training datasets was set to be similar to other works. A sample of the dataset in shown in figure 8.26.

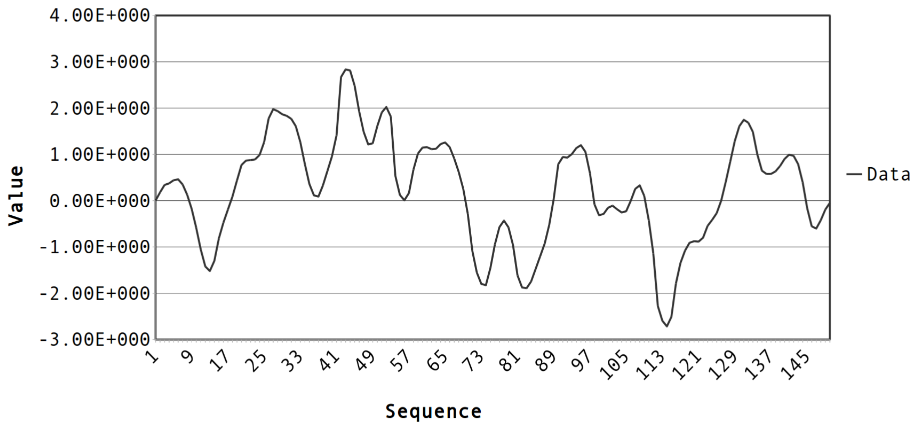


Figure 8.26: Gas Furnace Dataset

Quality Tests

In the quality tests, model assessment is reviewed with the various datasets. The test uses a separate test dataset, where the following metrics are used:

- Internet Traffic Dataset: Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE).
- Gas Furnace Dataset: Mean Absolute Error (MAE) and Root Mean Square Error (RMSE).

Additionally, in the Gas Furnace Dataset, the evaluation is performed as in (Thomakos and Guerard, 2004), where the training samples are increased by one and tested on the same model by forecasting one-step-ahead, iterating all the values. However, in the Internet Traffic Dataset, the model is fitted once and all the test indexes are forecasted with the same model.

Results & Analysis The validation results for each one of the datasets are shown in table 8.29 for the Internet Traffic Dataset and in table 8.30 for the Gas Furnace Dataset.

Algorithm	Tier	Metric	Results
ARIMA	Fast Tier	MAPE	31.59%
		MAE	7.8×10^8
	Quality Tier	MAPE	29.70%
		MAE	7.5×10^8
SVR	Fast Tier	MAPE	77.36%
		MAE	3.08×10^9
	Quality Tier	MAPE	45.63%
		MAE	1.55×10^9

Table 8.29: Automated Forecast quality results for the Internet Traffic Dataset.

From the table 8.29 we can see that there are large differences between the ARIMA and SVR approach, but the difference between Fast Tier and the Quality Tier in ARIMA is minimal while in SVR is large.

Algorithm	Tier	Metric	Results
ARIMA	Fast Tier	RMSE	0.2173
		MAE	0.027
	Quality Tier	RMSE	0.1347
		MAE	0.010
SVR	Fast Tier	RMSE	1.1161
		MAE	0.0311
	Quality Tier	RMSE	0.1645
		MAE	0.013

Table 8.30: Automated Forecast quality results for the Gas Furnace Dataset.

From the table 8.30 the results vary more than in table 8.29. Here, the ARIMA approach is still superior but there is a large difference between the Fast Tier and the

Quality Tier. Additionally we can further analyse these results by plotting the data against their real results and the predictions.

- **Internet Traffic Dataset** Comparing figure 8.27 against figure 8.28, where both approaches use ARIMA, we can see that while their results from table 8.29 are not significant, the Quality Tier ARIMA appears to predict the real values more closely. As for the SVR approach, depicted in 8.29 and 8.30, we can see that they do not follow as closely the real values, as expected from the results on the table8.29.

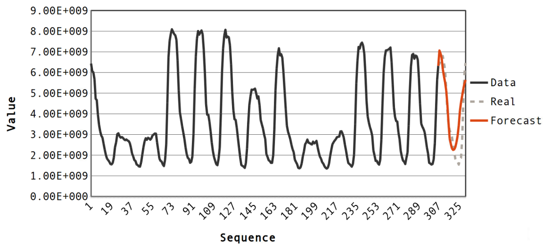


Figure 8.27: Prediction results for the Internet Traffic Dataset using the Fast Tier grid-search in ARIMA.

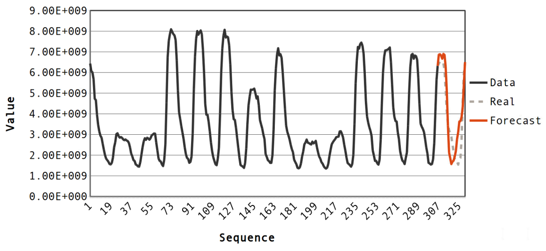


Figure 8.28: Prediction results for the Internet Traffic Dataset using the Quality Tier grid-search in ARIMA.

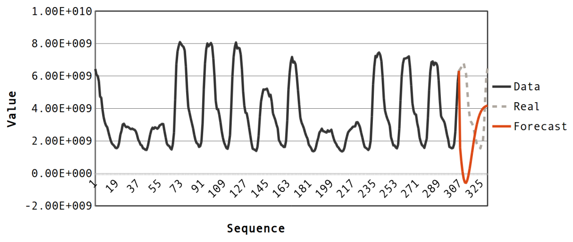


Figure 8.29: Prediction results for the Internet Traffic Dataset using the Fast Tier grid-search in SVR.

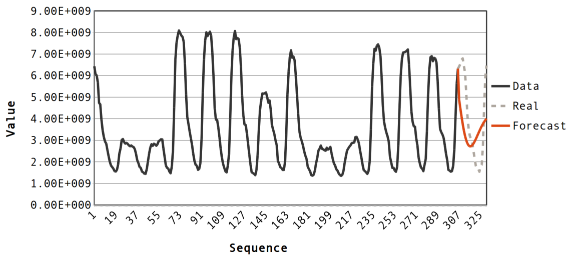


Figure 8.30: Prediction results for the Internet Traffic Dataset using the Quality Tier grid-search in SVR.

- **Gas Furnace Dataset** As we can see from the results, even in the best model, depicted in figure 8.32, the predicted values are not as closely matched to the real values as it happens in the other dataset. The lack of clear periodic values and trends make this dataset a more difficult one to forecast. Furthermore, in this dataset the forecast is performed with rolling-sample, as mentioned in (Thomakos and Guerard, 2004), and that might explain the rapid variations between points in figure 8.32.

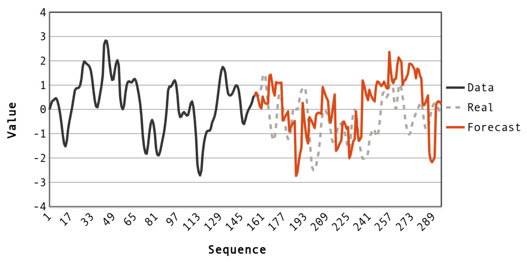


Figure 8.31: Prediction results for the Gas Furnace Dataset using the Fast Tier grid-search in ARIMA.

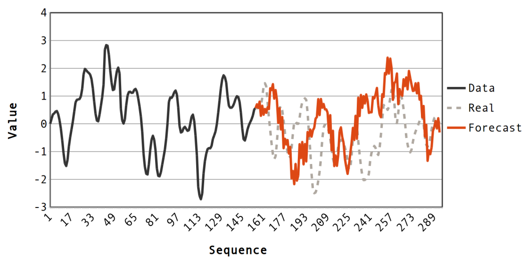


Figure 8.32: Prediction results for the Gas Furnace Dataset using the Quality Tier grid-search in ARIMA.

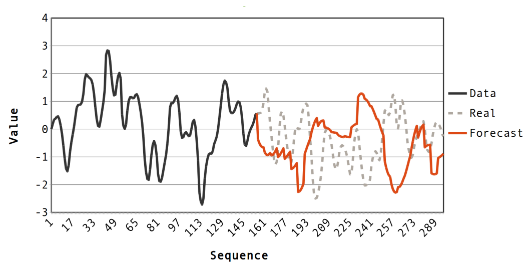


Figure 8.33: Prediction results for Gas Furnace Dataset using the Fast Tier grid-search in SVR.

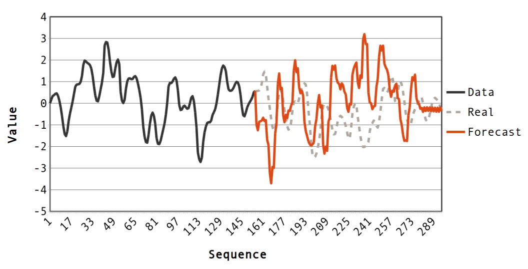


Figure 8.34: Prediction results for the Gas Furnace Dataset using the Quality Tier grid-search in SVR.

Additionally, we can find other articles using these datasets, which is useful to compare the results:

- **Internet Traffic Dataset** An exploratory article regarding methods to forecast is shown in table 8.31. It introduces a novel neural network ensemble approach, ARIMA, Holt-Winters and Naïve Bayes as approaches. It used the MAPE metric and the table compares the same dataset and the same number of test samples. From the results we can observe that our approach provides better predictions than both Naïve Bayes and Holt-Winters and is not too far off from ARIMA and their novel neural network ensemble. Taking into consideration that those models were created manually by experts, the results Wizdee obtained are very good.

Article	Metric	Algorithm	Results	Difference
Internet traffic forecasting using neural networks(Cortez et al., 2006)	MAPE	Naïve Bayes	65.67%	**-35.97**
		Holt-Winters	50.60%	**-20.9**
		ARIMA	26.96%	**+2.74**
		NNE	23.48%	**+6.22**

Table 8.31: Results from other reports using the Internet Traffic dataset.

- **Gas Furnace Dataset** Some work on forecasting values in this dataset has been done, in the table 8.32 we can see an exploratory work using various approaches such Naïve Bayes and ARIMA. From the results we can see that even though those models were created manually, this approach managed to obtain slightly better results which is very satisfactory.

Article	Metric	Algorithm	Results	Difference
Naïve, ARIMA, nonparametric, transfer function and VAR models: A comparison of forecasting performance (Thomakos and Guerard, 2004)	RMSE	Naïve Bayes	0.773	**-0.638**
		ARIMA	0.406	**-0.271**

Table 8.32: Results from other reports using the Gas Furnace dataset.

Performance Tests

For the performance tests, 30 runs were executed for each one of the datasets and various performance metrics were extracted for the various steps. Additionally, as mentioned before, the original requirement describes a time limitation of *30s*, with the ideal time being *5s*.

Results & Analysis This sections analyses the performance results, an overview of the performance is shown on the table 8.33.

Dataset	**Algorithm**	**Tier**	**Total Time**	**Grid-Search (Multithread)**	**Grid-Search (Single-thread)**[*]
Internet Traffic	ARIMA	Fast Tier	12.59s ($\sigma = 0.31$)	9.50s ($\sigma = 0.28$)	35.80s ($\sigma = 0.87$)
		Quality Tier	548.5s ($\sigma = 28.47$)	541.8s ($\sigma = 33.22$)	2070s ($\sigma = 57$)
	SVR	Fast Tier	3.50s ($\sigma = 0.07$)	3.43s ($\sigma = 0.06$)	4.85s ($\sigma = 0.18$)
		Quality Tier	4.94s ($\sigma = 0.04$)	4.8s ($\sigma = 0.03$)	7.01s ($\sigma = 0.11$)
Gas Furnace	ARIMA	Fast Tier	11.27s ($\sigma = 0.12$)	9.91s ($\sigma = 0.02$)	37.33s ($\sigma = 1.22$)
		Quality Tier	648.07s ($\sigma = 80.83$)	642.5s ($\sigma = 71.12$)	2260s ($\sigma = 116$)
	SVR	Fast Tier	3.93s ($\sigma = 0.04$)	3.83s ($\sigma = 0.06$)	7.63s ($\sigma = 0.20$)
		Quality Tier	5.16s ($\sigma = 0.03$)	5.10s ($\sigma = 0.01$)	11.44s ($\sigma = 0.28$)

[*] Run 4 times, instead of 30, due to large execution times.

Table 8.33: Automated Forecast performance results.

Looking at the table 8.33, the results are quite reasonable. The Quality Tier goes over the time limit as expected due to the large grid search size. That is also the main motivation behind the creation of a Fast Tier. Nonetheless, the Quality Tier will be available to users who wish to obtain better results at the cost of waiting longer times. Additionally, to reach the ideal time limit of $5s$ the SVR approach should be given preference.

Of importance is the difference between multithreaded grid-search and single-thread grid search, reaching a decrease of 288% in time on average across the datasets. This is of considerable importance because without multithreaded grid-search, the ARIMA approach couldn't be used unless manually specified, resulting in poorer results. In the table 8.34 we can take a look at how much time is spent on applying the highest quality model.

Dataset	**Algorithm**	**Apply Model**
Internet Traffic	ARIMA	2.79s ($\sigma = 0.13$)
Gas Furnace	ARIMA	4.14s ($\sigma = 0.45$)

Table 8.34: Automated Forecast performance of applying the model.

Additionally, looking at the figure in 8.35 we can see the relationship between the MAX_LIMIT and the performance in applying and training the best quality models. In ARIMA the time increases slightly while in SVR it starts faster but quickly increases to unreasonable numbers.

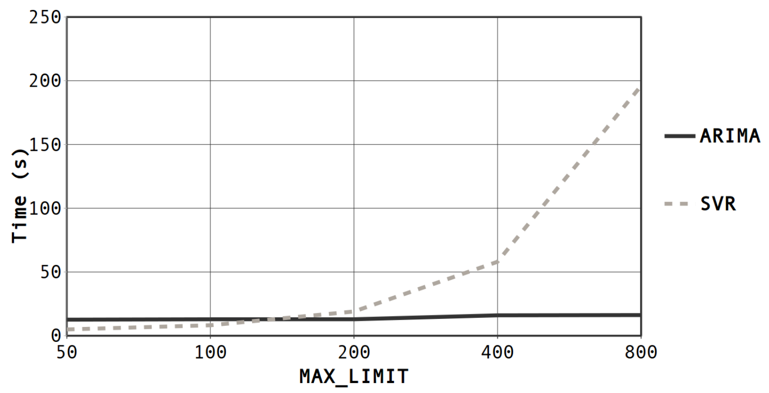


Figure 8.35: Relationship between MAX_LIMIT and performance.

Remarks Finally, for the task of Automated Forecast, both the quality and performance of the system are shown to be good, obtaining results comparable to models created manually while in strict time requirements.

8.2.4 Automated Clustering

Quality assessment in clustering is a complex task. Verifying that the created structure is correct is subjective and for these reasons, a number of approaches have been suggested, from using a evaluation metric such as silhouette (Rousseeuw, 1987) to the construction of artificial data sets (Milligan, 1980). For these reasons, metrics that take into consideration the class and metrics that evaluate the overall structure of the clusters, were both used.

Datasets

Simple Clustering Dataset This dataset is an abstract dataset manually created to show a scenario where the clusters clearly standout from the data. The high-level characteristics of the dataset are specified in table 8.35.

Nº of Instances	Nº of Attributes	Attribute Characteristics	Missing Values	Number of Clusters
66	2	Numeric	No	3

Table 8.35: Simple Clustering Dataset description.

Aggregation Dataset This dataset created by (Gionis et al., 2007) shows a scenario where there are various shaped clusters, including clusters that are aggregated to each other. The high-level characteristics of the dataset are specified in table 8.36.

Nº of Instances	Nº of Attributes	Attribute Characteristics	Missing Values	Number of Clusters
787	2	Numeric	No	7

Table 8.36: Aggregation Dataset description.

Mushroom Dataset This dataset created by (Schlimmer, 1981) shows descriptions of mushroom species, identified with them being edible or poisonous. The high-level characteristics of the dataset are specified in table 8.37.

N° of Instances	N° of Attributes	Attribute Characteristics	Missing Values	Number of Classes
8124	22	Categorical	Yes	2

Table 8.37: Mushroom Dataset description.

Quality Tests

In the quality tests, model assessment is reviewed with the various datasets. The test uses two metrics: Silhouette and Impurity Index. In datasets where the class is known, if a clustering contains k clusters with sizes $s_1, \ldots, s_k$ and the sizes of majority class in each cluster are $m_1, \ldots, m_k$ then we can use the impurity index(Gionis et al., 2007) as defined in:

$$I = \frac{\sum_{i=1}^{k}(s_i - m_i)}{n}$$

For the quality results, both K-Means and DBSCAN metrics are shown, however, K-Means is the main algorithm and the only automated one. To note, DBSCAN doesn't have the silhouette metric because it assumes convex clusters (Moulavi et al., 2014).

Results & Analysis The evaluation results for each one of the datasets are shown in table 8.38.

Algorithm	Metric	Simple Clustering	Aggregation	Mushroom
K-Means	Silhouette	0.90	0.47	0.15
	Impurity	0%	40.1%	34.19%
DBSCAN	Impurity	0%	4.6%	21.4%

Table 8.38: Automated Clustering quality results.

Looking at the table 8.38 it is possible to observe that, in the aggregation dataset, K-Means has worse results than DBSCAN, this is due to the data being a complex structure as we will see below. Finally, the results for the mushroom dataset are somewhat similar between approaches, to note is that the silhouette score is quite low even though the impurity is below 50%, which might mean that the boundaries between clusters is complex and not well defined. The number of clusters for each one of the datasets are shown in table 8.39.

Algorithm	Dataset	Number of Clusters
K-Means	Simple Clustering	3
	Aggregation	7
	Mushrooms	2
DBSCAN	Simple Clustering	3
	Aggregation	7
	Mushrooms	36

Table 8.39: Automated Clustering number of clusters results.

We can see that in most datasets the optimal number of clusters is very similar to the marked clusters, with the exception of the Mushroom Dataset using DBSCAN. Additionally we can further analyse these results by plotting the clusters and their marked classes. This is not easily accomplished in the Mushroom dataset due to its large number of attributes.

- **Simple Clustering** In this simple dataset, both K-Means, in figure 8.36, and DBSCAN, in figure 8.37, obtained a perfect score. From the figures we can see that the clusters are far away from each other and each cluster is highly dense, which is an easy scenario for most clustering algorithms.

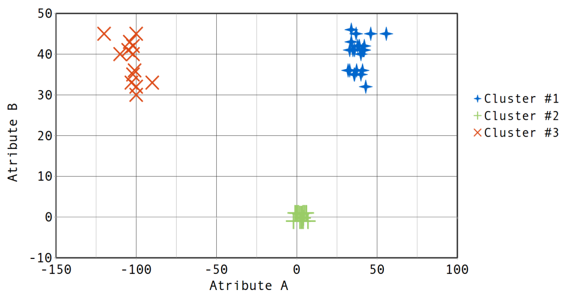


Figure 8.36: Simple Clustering Dataset and its clusters using K-Means.

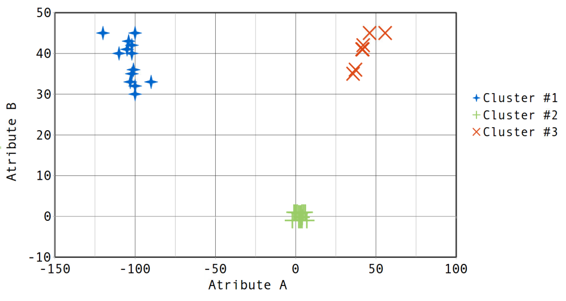


Figure 8.37: Simple Clustering Dataset and its clusters using DBSCAN.

- **Aggregation** This dataset represents a complex scenario where the marked classes, in figure 8.40, have connections between them, with varying size and shape. As expected, K-Means, in figure 8.38, has some difficulty in defining the clusters for the most difficult shapes, while DBSCAN, in figure 8.39, being a density based method, manages to create a cluster correctly around most of the shapes. These results are similar to the ones obtained by the original creators (Gionis et al., 2007).

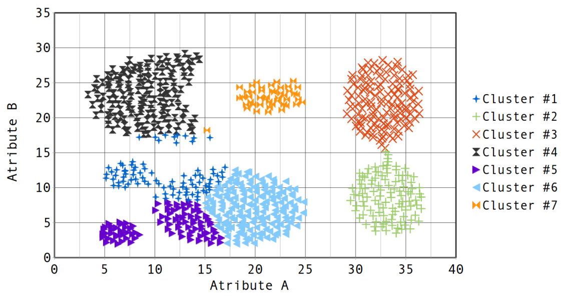


Figure 8.38: Aggregation Dataset and its clusters using K-Means.

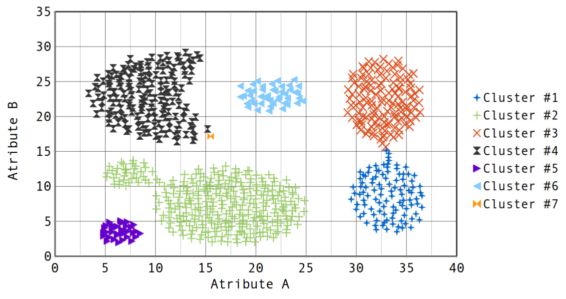


Figure 8.39: Aggregation Dataset and its clusters using DBSCAN.

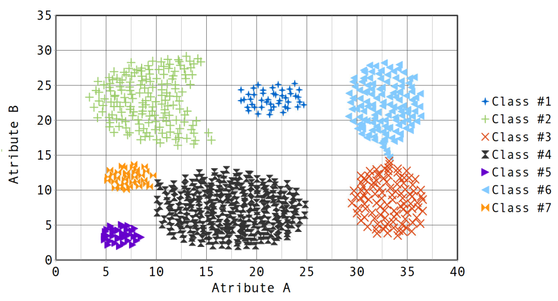


Figure 8.40: Aggregation Dataset and its assigned classes.

Furthermore, it is also interesting to see how the approach describes the clusters for the datasets.

- **Aggregation Dataset** This dataset contains many clusters, using Cluster 1 and Cluster 3 from the K-Means results and looking at the table 8.40, we can confirm that the descriptions of each cluster corresponds to what is seen on the figure. Ultimately, while in this dataset, it is possible to represent this information as a figure, there are many real-world datasets where visualization is impossible and this type of description is valuable.

Cluster Number	Attribute	Description
Cluster 1	Attribute A	$\mu = 11.0$ ($\sigma = 3.43$) $max = 17.0$ $min = 5.15$
	Attribute B	$\mu = 11.70$ ($\sigma = 2.39$) $max = 17.5$ $min = 8.25$
Cluster 3	Attribute A	$\mu = 32.69$ ($\sigma = 1.87$) $max = 36.35$ $min = 29.15$
	Attribute B	$\mu = 22.13$ ($\sigma = 3.13$) $max = 28.2$ $min = 15.5$

Table 8.40: Description of two of the clusters in the Aggregation Dataset.

- **Mushroom Dataset** This dataset contains many attributes, which makes the plotting of this dataset a complex task. Thus, looking at the description of some of the attributes from the clusters in the K-Means approach we can understand the common aspects between the samples inside each cluster without having to plot it. A few attributes were chosen and described in table 8.41.

  One of the most noticeable differences, is that all the samples in Cluster 1 are bruised while none of the samples in Cluster 2 are. On the other hand, characteristics such as Pink Gill-Color are not good to distinguish between clusters because they are distributed similarly in both clusters.

Cluster Number	Attribute	Description
Cluster 1	Bruises	100% True
	Gill-Color	28% White 21% Brown 20% Pink
	Habitat	54% Woods 19% Grasses 8% Meadows
Cluster 2	Bruises	100% False
	Gill-Color	36% Blue 17% Pink 13% Chocolate
	Habitat	88.4% Grasses 11.6% Paths

Table 8.41: Description of the two clusters in the Mushroom Dataset.

Finally, we can use the Mushroom dataset and the K-Means results, obtained in an automated way, and compare to results against other approaches:

- **Mushroom** There are many articles using the Mushroom Dataset. Thus, popular articles using the same metric as this work are compared in table 8.42. We can see that the approach used in this work gives better impurity scores than ROCK or BestClustering, nonetheless, the Agglomerative algorithm still provides the best results. To note is that the number of clusters is not the same and that approach might be more comparable to DBSCAN where the difference is just 10.3%. Anyhow, considering the results were obtained in an automated way, Wizdee feels the results are very reasonable.

Article	Metric	Algorithm	Clusters	Results	Difference
ROCK: A robust clustering algorithm for categorical attributes(Guha et al., 1999)	Impurity	ROCK	2	48.2%	**-9.01**
Clustering aggregation(Gionis et al., 2007)	Impurity	BestClustering	5	35.4%	**-1.21**
		Agglomerative	7	11.1%	**+23.09**

Table 8.42: Results from other reports using the bank dataset.

Performance Tests

For the performance tests, 30 runs were executed for each one of the datasets and various performance metrics were extracted for the various steps in the automated clustering task. Additionally, as mentioned before, the original requirement describes a time limitation of *30s*, with the non-requirement of an ideal time being *5s*.

Results & Analysis This sections analyses the performance results, an overview of the performance is shown on the table 8.43.

Dataset	**Algorithm**	**Tier**	**Total Time**	**Grid-Search (Multithread)**	**Grid-Search (Single-thread)**
Simple Clustering	K-Means	Fast Tier	1.31s ($\sigma = 0.07$)	1.04 ($\sigma = 0.03$)	1.69s ($\sigma = 0.05$)
		Quality Tier	2.24s ($\sigma = 0.05$s)	1.95s ($\sigma = 0.02$)	3.02 ($\sigma = 0.26$)
Aggregation	K-Means	Fast Tier	2.46s ($\sigma = 0.06$)	1.99 ($\sigma = 0.04$)	4.01s ($\sigma = 0.07$)
		Quality Tier	4.81s ($\sigma = 0.04$s)	4.32s ($\sigma = 0.02$)	10.30 ($\sigma = 0.17$)
Mushroom	K-Means	Fast Tier	26.8s ($\sigma = 0.12$)	14.20 ($\sigma = 0.38$)	50.31s ($\sigma = 0.14$)
		Quality Tier	51.29s ($\sigma = 1.22$)	40.57s ($\sigma = 1.37$)	174.41s ($\sigma = 4.08$)

Table 8.43: Automated Clustering performance results.

Looking at the table 8.43, the results are quite reasonable and within the time limitations. The Mushroom Dataset is above the ideal time limit in the Quality Tier, which supports the decision of introducing a SOFT_LIMIT as described in chapter 7.2.7. Of particular importance is the difference between multithreaded grid-search and single-thread grid search, reaching a decrease of 256% in time on average across the datasets.

In the table 8.44 we can take a look at the performance performing training and applying the model. Where it is possible to see that K-Means is faster, it also only requires to grid-search the number of clusters, which is typically small, while in DBSCAN there are two parameters that can cover a wide range of values. This supports the decision of using K-Means automatically but allowing DBSCAN as a manual algorithm.

Dataset	**Algorithm**	**Train & Apply Model**
Simple Clustering	K-Means	0.29s ($\sigma = 0.03$)
Simple Clustering	DBSCAN	0.89s ($\sigma = 0.11$)
Aggregation	K-Means	0.43s ($\sigma = 0.02$)
Aggregation	DBSCAN	1.01s ($\sigma = 0.04$)
Mushroom	K-Means	12.84s ($\sigma = 1.09$)
Mushroom	DBSCAN	20.98s ($\sigma = 0.97$)

Table 8.44: Automated Forecast performance of applying the model.

Additionally, looking at the figure in 8.41, we can see that after 5000 points it goes above the suggested $5s$, which is why an equation is used to define the number of clusters, as described in chapter 7.2.7.

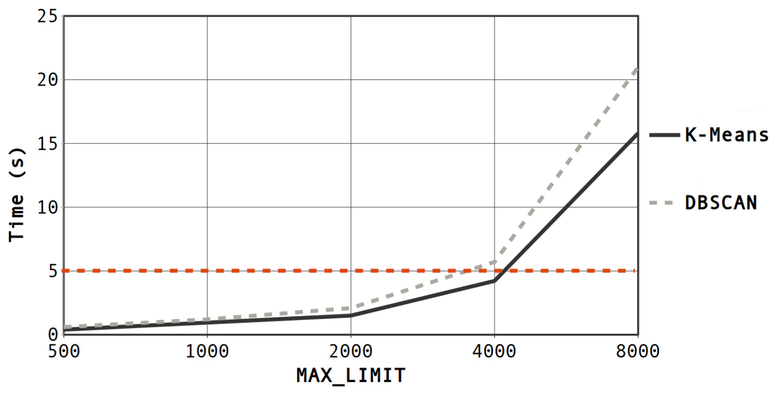


Figure 8.41: Relationship between the number of instances the time it takes to train and apply a model.

Remarks For the task of Automated Clustering, the performance of the system is shown to respect the requirements. The quality results are of subjective nature but using common evaluation metrics they are shown to be reasonable and comparable to other works.

8.2.5 Specific Industry Solutions

For the task of specific industry solutions, a process to create a model specific in the Salesforce context was created, this process is described in section 7.2.8.

Datasets

Salesforce Opportunities Dataset The target variable of this dataset is whether the Opportunity is Closed Won or Closed Lost. The features are specified in section 7.2.8. To note that the dataset was split so that it could be used in this test. The high-level characteristics of the test dataset are specified in table 8.45.

Nº of Instances	Nº of Attributes	Attribute Characteristics	Missing Values	Target Variable
204	36	Numeric/Categorical	Yes	Closed Won (61.8%) Closed Lost (38.2%)

Table 8.45: Salesforce Solution Test Dataset description.

Quality Tests

In the quality tests, we use a 10-folds validation on the training dataset but also test the model on a separate set as described above 30 times. Using two metrics: Accuracy and F1 SCore.

Results & Analysis The 10-folds validation and test results are shown in table 8.46:

Approach		Accuracy		F1 Score	
Algorithm	**Parameters**	**Validation**	**Test**	**Validation**	**Test**
SVM	*C*=2 *gamma*=0.05 *kernel*=rbf	76.90% (σ = 4.06)	85.78%	83.03% (σ = 2.57)	89.53%
Random Forest	*n_instances*=100 *min_sample_size*=4	76.02% (σ = 5.13)	80.07% (σ = 7.4)	79.13% (σ = 4.55)	84.30% (σ = 6.43)
Decision Tree	*min_sample_size*=3	74.65% (σ = 3.01)	76.78%	77.81% (σ = 2.31)	81.77%

Table 8.46: Quality results of the models created.

From the table above we can see that the results are high across every metric. Additionally it is possible to observe that while the scores between Validation and the Test dataset do not deviate too much, they are consistently higher on the test dataset. The best approach is based on SVM and we can take a look at how the C and Gamma influence the accuracy by analysing the figure 8.42, where it's possible to see the bluest point at *C=2* and *gamma=0.05*..

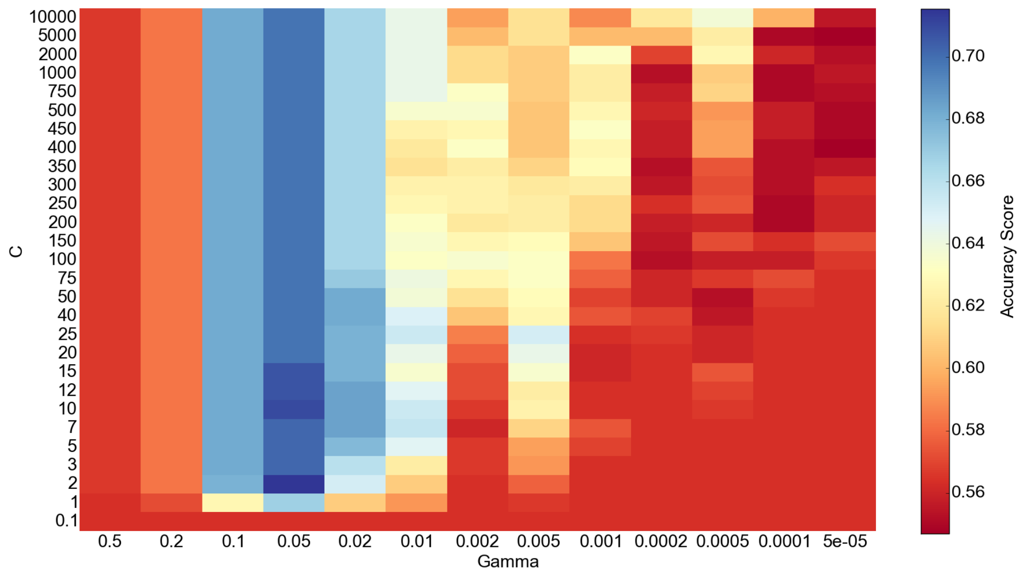


Figure 8.42: Relationship between C and Gamma during grid search.

Finally, the use of feature extraction and feature selection techniques such as PCA cause an increase of **14** points in F1 Score and **10** in Accuracy.

Performance Tests

For the performance tests, 30 runs were executed for the best approach above and various performance metrics were extracted for the various steps of this task. To note, that preprocessing includes the time needed to load the data due to optimizations made that apply the preprocessing during that phase. Finally, in this task there's no time limits regarding the time it takes to train, only to apply the model.

Results & Analysis This sections analyses the performance results, an overview of the performance is shown on the table 8.47.

Algorithm	**Total Time**	**Preprocessing**	**Grid-Search (Multithread)**	**Grid-Search (Single-thread)**	**Validation**
SVM	19.45s ($\sigma = 0.63$)	0.19s ($\sigma = 0.01$)	9.61s ($\sigma = 0.30$)	56.12s ($\sigma = 0.83$)	8.58s ($\sigma = 0.04$)

Table 8.47: Automated Classification performance results.

Looking at the table above, we can see that while there are not as many instances as in some of the datasets in Automated Classification it still takes longer. This was expected since the grid search goes more in-depth and ends up taking a large portion of the time. Of particular importance is the difference between multithreaded grid-search and single-thread grid search, reaching a decrease of 583% in time on average across the datasets, showing the value of creating a scalable architecture.

In the table 8.48 we can take a look at the performance for training and testing and observe that the majority of the time is spent during the model training as expected. While the training performance is not of much importance in this task, the model appliance time is still important but it is still under the $5s$ limit.

Algorithm	**Train Model**	**Apply Model**
SVM	18.40s ($\sigma = 0.24$)	1.03s ($\sigma = 0.03$)

Table 8.48: Automated Classification performance of training the model versus applying the model.

Remarks For the task of Specific Industry Solutions, a model specific for the Salesforce context was created. While it is not possible to compare the quality of the model against other works, the results are very good considering it deals with real world data. It is reasonable to say that the machine learning architecture developed is fundamental in the creation of the various Automated tasks but it is also very capable when doing manually crafted models.

Chapter 9

Conclusions

In the age of new technologies, where information is all around us and large amounts of data is being created at an impressive rate, there is a big amount of knowledge that is potentially important, but is yet to be discovered. There are two processes closely related to discovering new information: Information Analysis and Information Visualization. Automating those two processes were the main tasks of this thesis.

The goals of this work were to make the processes behind Data Mining and Information Visualization as effortless as possible. In Automated Data Mining, using raw data we wanted to be able to do various data mining tasks, such as: classification, outlier detection, forecast and clustering. Additionally, we also created a central machine learning environment in the Wizdee platform that could handle data mining tasks, whether they were automated or manually crafted. In Automatic Visualization the goal was to decide the most appropriate visualization based on raw data and perform the needed chart processing, so that an higher-level component could use it.

In this work, research was presented through the creation of the Background Knowledge. This gives an overview of the many needed concepts required to create a system that deals with data analysis and decision making, including an overview for the various popular machine learning algorithms and its applications in real scenarios. From the Related Work, we have shown the state-of-the-art and current limitations, such as, an approach for Automatic Visualization, that was able to identify a chart type from raw with good accuracy but was unable to choose the axes and series. This was something that we were able to surpass by creating an innovative technique labelled Case Mapping. From the competitor analysis we were able to see some of the limitations on the competitor products, such as the lack of pagination in the Automatic Visualization competitors. The number of competitors in Automated Data Mining is much smaller, which supports the difficulty and complexity of the problem.

The approach for Automatic Visualization is based on Case Base Reasoning. An approach inspired by human reasoning and memory organization. Rather than following a set of rules, this approach reapplies previous a successful solutions in the new problem. We have presented the details for each step of the CBR approach, including details about the case representation, features, similarity measure and the Case Mapping technique. Additionally, the approach also required chart processing for each chart type. This processing was responsible for handling parameter decisions, such as number of slices or bins, and handle complex chart mechanics, such as pagination, ambiguous values and series limitation.

As for the Automated Data Mining system, the approach is based on concepts from machine learning, where important patterns and trends are extracted as an attempt to understand what the data say. Those concepts are then extended so that they can be applied without users input or know how, by finding mechanisms that can automate common steps such as metadata extraction, pre-processing, algorithm selection, parameter optimization, or model evaluation. Since each data mining task is different, it requires slightly different approaches. Nonetheless, all of those follow the same baseline pipeline model. One of most challenging elements of this task was staying under the specified performance requirement, due to this task being run as requested. Other challenges are dependent on the specific task, such as having to parse the internal tree structure of sklearn tree, without documentation, or dealing with strange results from the ARIMA implementation in statsmodel, causing us to develop an integration with R.

In the tests chapter, the results and analysis were presented for each task. In Automatic Visualization the results were shown to be very positive, surpassing, by far, the previous approach. In the Automated Data Mining, a detailed test and analysis was made for each of the tasks. In Automated Classification, the performance is shown to respect the requirements, and solutions were given to reach the ideal time of $5s$. In terms of quality, it was comparable to other manually crafted works. For the task of Automated Outlier Detection, the results were reasonable in terms of quality but impressive in terms of performance. In Automated Forecast, the quality is shown to be comparable to models created manually and the performance is within time limitations. For the task of Automated Clustering, the quality, while subjective, is shown to be quite good and comparable to other manually crafter works. Furthermore, for the model created in the CRM Solution, the performance is within requirements and the quality, while incomparable due to being custom made, is shown to be very good. Finally, the importance of implementing multithreaded grid-search was also shown by presenting the difference between multithreaded grid-search and single-threaded.

This work resulted in various contributions:

- **Process for Automatic Visualization:** This work presents details about each step of the approach, including specifications and decision made regarding case representation, attribute features.
- **Case Mapping:** This work presents an innovative technique, labelled Case Mapping, used in the process of Automatic Visualization, that surpasses the limitations found in the related work, regarding obtaining the correct axes and series.
- **Chart processing techniques:** This work presents solutions to complex challenges found during chart processing, such as pagination mechanisms and chart parameters decisions.
- **Scalable and flexible machine learning architecture:** One of the important aspects to develop a system capable of doing Automated Data Mining tasks is the existence of a central machine learning component. In this work we describe the process of creating an architecture that can handle low-level details gracefully while being scalable and flexible.
- **Process for Automated Classification:** Details on how to automate the task of classification are described in detail.
- **Process for Automated Forecast:** Details on how to automate the task of forecast are described in detail.
- **Process for Automated Outlier Detection:** Details on how to automate the task of outlier detection are described in detail.
- **Process for Automated Clustering:** Details on how to automate the task of clustering are described in detail.

- **CRM Classification Model:** Using the machine learning architecture, a manually crafted model for the Salesforce industry was created and described in this work. The model shows high accuracy results.
- **Commercial Product Integration:** Although the work involves research, the final implementation was integrated in the platform - showing the viability of the approaches used in a practical scenario.

Future Work

As future work in the Automatic Visualization, we plan to introduce more charts types and add more cases to the case base. We also plan to create mechanisms to make the system scalable, so that it can handle large quantities of high-dimensionality data gracefully. In terms of personalization, it is possible in the current solution, to have personalized cases, however, this could be further expanded by giving some amount of control to end-users.

In terms of the machine learning architecture, there are many more algorithms we can support, so that developers can have a wide range of options. Furthermore, while the system is scalable, this could be further improved by introducing multiple servers and a load balancer. The decrease in time, when using multithreading, in parameter optimization is already very significant, but this could be even further reduced by distributing the load to various servers.

For the Automated Classification task, considering the importance of having a human-readable output, future work includes having a comprehensible output, even for black-box algorithms, such as SVMs, using rule extraction techniques.

In order to improve the results of outlier detection, we plan to use more robust techniques, including supervised techniques from anomaly detection, in situations where a clean dataset is known and we wish to monitor outlier insertions.

In Automated Forecast we plan to implement our own version of ARIMA to avoid using R, due to license related reasons. Additionally, we plan to use more complex approaches such as ARIMAX, that can combine linear regression and ARIMA.

Finally, in clustering we plan to introduce more types of clustering, such as hierarchical clustering. Having various methods to handle categorical values is also of great interest.

References

Aggarwal, C. C. (2013). An introduction to outlier analysis. In *Outlier Analysis*, pages 1–40. Springer.

Aggarwal, C. C., Hinneburg, A., and Keim, D. A. (2001). *On the surprising behavior of distance metrics in high dimensional space.* Springer.

Agnar Aamodt, E. P. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communication.*

Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. *SIGMOD Rec.*, 27(2):94–105.

Alpaydin, E. (2004). *Introduction to machine learning.* MIT press.

Anton, H. (2010). *Elementary linear algebra.* John Wiley & Sons.

Arthur, D. and Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics.

Barry Smyth, Padraig Cunningham, M. K. (2014). Hierarchical case-based reasoning.

Bell, J. (2014). *Machine Learning: Hands-On for Developers and Technical Professionals.* Wiley.

Bennett, K. P. (1992). *Decision tree construction via linear programming.* Center for Parallel Optimization, Computer Sciences Department, University of Wisconsin.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy).* Oral Presentation.

Bloom, B., Bhagwat, C., and Stengard, P. (2003). Automated model building and evaluation for data mining system. US Patent App. 10/383,641.

Box, G. E., Jenkins, G. M., and Reinsel, G. C. (2011). *Time series analysis: forecasting and control*, volume 734. John Wiley & Sons.

Breiman, L. (1996). Technical note: Some properties of splitting criteria. *Machine Learning*, 24(1):41–47.

Campos, M., Milenova, B., and Stengard, P. (2009). Data-centric automated data mining. US Patent 7,627,620.

Casner, S. M. (1991). Task-analytic approach to the automated design of graphic presentations. *ACM Trans. Graph.*, 10(2):111–151.

Chen, C., Liaw, A., and Breiman, L. (2004). Using random forest to learn imbalanced data. *University of California, Berkeley.*

Cheng, J. and Greiner, R. (1999). Comparing bayesian network classifiers. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 101–108. Morgan Kaufmann Publishers Inc.

Cohen, W. W. and Singer, Y. (1999). A simple, fast, and effective rule learner. In *Proceedings of the National Conference on Artificial Intelligence*, pages 335–342. JOHN WILEY & SONS LTD.

Contreras, J., Espinola, R., Nogales, F. J., and Conejo, A. J. (2003). Arima models to predict next-day electricity prices. *Power Systems, IEEE Transactions on*, 18(3):1014–1020.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Cortez, P., Rio, M., Rocha, M., and Sousa, P. (2006). Internet traffic forecasting using neural networks. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 2635–2642. IEEE.

Cortez, P., Rio, M., Rocha, M., and Sousa, P. (2012). Multi-scale internet traffic forecasting using neural networks and time series methods. *Expert Systems*, 29(2):143–155.

Craik, K. (1967). *The Nature of Explanation.* Cambridge University Press.

Dix, A., Finlay, J., E., J., Gregory, D., Abowd, and Beale, R. (2003). *Human-Computer Interaction.* Prentice-Hall Inc.

Elsalamony, H. A. (2014). Bank direct marketing analysis of data mining techniques. *Network*, 5:0.

Elsayad, A. M. and Elsalamony, H. (2013). Diagnosis of breast cancer using decision tree models and svm. *International Journal of Computer Applications*, 83(5):19–29.

Enders, C. K. (2011). Analyzing longitudinal data with missing values. *Rehabilitation Psychology*, 56(4):267.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231.

Estivill-Castro, V. (2002). Why so many clustering algorithms: a position paper. *ACM SIGKDD explorations newsletter*, 4(1):65–75.

Fielding, R. T. and Taylor, R. N. (2002). Principled design of the modern web architecture. *ACM Transactions on Internet Technology (TOIT)*, 2(2):115–150.

Franklin, S., Thomas, S., and Brodeur, M. (2000). Robust multivariate outlier detection using mahalanobis' distance and modified stahel-donoho estimators. In *Proceedings of the Second International Conference on Establishment Surveys*, pages 697–706. American Statistical Association Buffalo, NY.

Fürnkranz, J., Gamberger, D., and Lavrač, N. (2012). *Foundations of rule learning.* Springer.

Gantz, J. and Reinsel, D. (2012). The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. *IDC iView: IDC Analyze the Future.*

Gionis, A., Mannila, H., and Tsaparas, P. (2007). Clustering aggregation. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):4.

Gomes, T. A., Prudêncio, R. B., Soares, C., Rossi, A. L., and Carvalho, A. (2012). Combining meta-learning and search techniques to select parameters for support vector machines. *Neurocomputing*, 75(1):3 – 13.

Graf, A. B. and Borer, S. (2001). Normalization in support vector machines. In *Pattern Recognition*, pages 277–282. Springer.

Guha, S., Rastogi, R., and Shim, K. (1999). Rock: A robust clustering algorithm for categorical attributes. In *Data Engineering, 1999. Proceedings., 15th International Conference on*, pages 512–521. IEEE.

Hadi, A. S. (1992). Identifying multiple outliers in multivariate data. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 761–771.

Hardin, J. and Rocke, D. M. (2005). The distribution of robust distances. *Journal of Computational and Graphical Statistics*, 14(4).

Hoens, T. R., Qian, Q., Chawla, N. V., and Zhou, Z.-H. (2012). Building decision trees for the multi-class imbalance problem. In *Advances in Knowledge Discovery and Data Mining*, pages 122–134. Springer.

Huang, Z. (1998). Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data mining and knowledge discovery*, 2(3):283–304.

Hyndman, R. J. and Khandakar, Y. (2007). Automatic time series for forecasting: the forecast package for r. Technical report, Monash University, Department of Econometrics and Business Statistics.

Jain, A. K. and Dubes, R. C. (1988). *Algorithms for clustering data.* Prentice-Hall, Inc.

Jill Freyne, B. S. (2009). Creating visualizations : a case-based reasoning perspective. *Artificial Intelligence and Cognitive Science.*

Jin, R., Jin, R., Supervisory, C., Dr, C., and Hauptmann, A. G. (2003). Statistical approaches toward title generation.

Jolliffe, I. (2005). *Principal component analysis.* Wiley Online Library.

Karsten, K. G. (1925). *Charts and graphs: An introduction to graphic methods in the control and analysis of statistics.* Prentice-Hall.

Ketchen, D. J. and Shook, C. L. (1996). The application of cluster analysis in strategic management research: an analysis and critique. *Strategic management journal*, 17(6):441–458.

Kohavi, R. (1996). Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, page to appear.

Kolodner, J. L. (1992). An introduction to case-based reasoning. *Artificial Intelligence Review*, 6(1):3–34.

Larose, D. T. (2005). *k-Nearest Neighbor Algorithm*, pages 90–106. John Wiley and Sons, Inc.

Leake, D. B. (1996). *Case-based reasoning: Experiences, lessons and future directions.* MIT press.

Ledolter, J. (1989). The effect of additive outliers on the forecasts from arima models. *International Journal of Forecasting*, 5(2):231–240.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Lewis, D. D. (1992). *Representation and learning in information retrieval.* PhD thesis, University of Massachusetts.

Lin, S., Fortuna, J., Kulkarni, C., Stone, M., and Heer, J. (2013). Selecting semantically-resonant colors for data visualization. *Computer Graphics Forum (Proc. EuroVis).*

Mackinlay, J. (1986). Automating the design of graphical presentations of relational information. *ACM Trans. Graph.*, 5(2):110–141.

Mahalanobis, P. C. (1936). On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)*, 2:49–55.

Mardia, K. V., Kent, J. T., and Bibby, J. M. (1979). *Multivariate analysis.* Academic press.

Marsland, S. (2014). *Machine learning: an algorithmic perspective.* CRC press.

Matsumoto, S., Kamei, Y., Monden, A., and Matsumoto, K.-i. (2007). Comparison of outlier detection methods in fault-proneness models. In *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*, pages 461–463. IEEE.

Mazza, R. (2009). *Introduction to information visualization*, volume 149. Springer.

Meng, Y. (2009). Parameters selection of hybrid kernel based on ga. In *Intelligent Systems and Applications, 2009. ISA 2009. International Workshop on*, pages 1–4.

Miller, R. B. (1968). Response time in man-computer conversational transactions. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pages 267–277. ACM.

Milligan, G. W. (1980). An examination of the effect of six types of error perturbation on fifteen clustering algorithms. *Psychometrika*, 45(3):325–342.

Mitchell, T. M. (2006). *The discipline of machine learning.* Carnegie Mellon University, School of Computer Science, Machine Learning Department.

M.J. Zaki, W. M. J. (2014). *Data Mining and Analysis: Fundamental Concepts and Algorithms.* Cambridge University Press.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of machine learning.* MIT press.

Moro, S., Laureano, R., and Cortez, P. (2011). Using data mining for bank direct marketing: An application of the crisp-dm methodology. In *Proceedings of European Simulation and Modelling Conference-ESM'2011*, pages 117–121. Eurosis.

Moulavi, D., Jaskowiak, P. A., Campello, R., Zimek, A., and Sander, J. (2014). Density-based clustering validation. In *Proceedings of the 14th SIAM International Conference on Data Mining (SDM), Philadelphia, PA*.

Nielsen, J. (1993). Response times: The 3 important limits. *Usability Engineering*.

Nielsen, J. (2014). Response time limits. `http://www.nngroup.com/articles/response-times-3-important-limits/`.

Olszewski, R. T. (2001). *Generalized Feature Extraction for Structural Pattern Recognition in Time-series Data*. PhD thesis, Pittsburgh, PA, USA. AAI3040489.

Pantic, M. (2014). Introduction to machine learning and case-based reasoning. Imperial College London.

Pei, Y., Zaiane, O. R., and Gao, Y. (2006). An efficient reference-based approach to outlier detection in large datasets. In *Data Mining, 2006. ICDM'06. Sixth International Conference on*, pages 478–487. IEEE.

Pindyck, R. S. and Rubinfeld, D. L. (1998). *Econometric models and economic forecasts*, volume 4. Irwin/McGraw-Hill Boston.

Plaisant, C. (2004). The challenge of information visualization evaluation. In *Proceedings of the working conference on Advanced visual interfaces*, pages 109–116. ACM.

Raileanu, L. E. and Stoffel, K. (2004). Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, 41(1):77–93.

Ralambondrainy, H. (1995). A conceptual version of the k-means algorithm. *Pattern Recognition Letters*, 16(11):1147–1157.

Riesbeck, C. K. and Schank, R. C. (1989). *Inside Case-Based Reasoning*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Schlimmer, J. (1981). Mushroom records drawn from the audubon society field guide to north american mushrooms. *GH Lincoff (Pres), New York*.

Schutt, R. and O'Neil, C. (2013). *Doing Data Science: Straight Talk from the Frontline*. " O'Reilly Media, Inc.".

Schwaber, K. (1997). Scrum development process. In *Business Object Design and Implementation*, pages 117–134. Springer.

Schwaber, K. and Beedle, M. (2001). *Agile Software Development with Scrum*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

Shlens, J. (2014). A tutorial on principal component analysis. *CoRR*, abs/1404.1100.

Shneiderman, B. and Plaisant, C. (2006). Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies. In *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaluation methods for information visualization*, pages 1–7. ACM.

Smith, L. I. (2002). A tutorial on principal components analysis. *Cornell University, USA*, 51:52.

Smola, A. and Vapnik, V. (1997). Support vector regression machines. *Advances in neural information processing systems*, 9:155–161.

Song, Q., Wang, G., and Wang, C. (2012). Automatic recommendation of classification algorithms based on data set characteristics. *Pattern Recognition*, 45(7).

Stacey, M., Salvatore, J., and Jorgensen, A. (2013). *Visual Intelligence: Microsoft Tools and Techniques for Visualizing Data.* John Wiley & Sons.

Stanton, W. J., Buskirk, R. H., and Spiro, R. L. (1991). *Management of the sales force.* Irwin Homewood, IL.

Steffens, T. (2006). *Enhancing similarity measures with imperfect rule-based background knowledge*, volume 302. IOS Press.

Stevens, J. P. (1984). Outliers and influential data points in regression analysis. *Psychological Bulletin*, 95(2):334.

Suda, B. (2010). *A Practical Guide to Designing with Data.* Five Simple Steps.

Sun, Y., Wong, A. K., and Kamel, M. S. (2009). Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(04):687–719.

Thomakos, D. D. and Guerard, J. B. (2004). Naïve, arima, nonparametric, transfer function and var models: A comparison of forecasting performance. *International Journal of Forecasting*, 20(1):53–67.

Tufte, E. R. and Graves-Morris, P. (1983). *The visual display of quantitative information*, volume 2. Graphics press Cheshire, CT.

Vapnik, V. (2013). *The nature of statistical learning theory.* Springer Science & Business Media.

W. Litte, H. W. F. (1972). *Shorter Oxford English dictionary.* Oxford University Press.

Ware, C. (2013). *Information visualization: perception for design.* Elsevier.

Watson, I. and Marir, F. (1994). Case-based reasoning: A review. *The Knowledge Engineering Review*, 9:327–354.

Wisaeng, K. (2013). A comparison of different classification techniques for bank direct marketing. *International Journal of Soft Computing and Engineering (IJSCE)*, 3(4):116–119.

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann.

Wolberg, W. H. and Mangasarian, O. L. (1990). Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the national academy of sciences*, 87(23):9193–9196.

Zhang, J. (1992). Selecting typical instances in instance-based learning. In *Proceedings of the ninth international conference on machine learning*, pages 470–479.

Zhou, M. X. and Chen, M. (2003). Automated generation of graphic sketches by example. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 65–71, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.