EMPIRICAL EVALUATION OF BIG DATA ANALYTICS USING DESIGN OF EXPERIMENT: CASE STUDIES ON TELECOMMUNICATION DATA

Samneet Singh, Yan Liu, Wayne Ding, Zheng Li
Concordia University, Concordia University, Ericsson Canada Inc., Lund University
sam_sing@encs.concordia.ca, yan.liu@concordia.ca, wayne.ding@ericsson.com, zheng.li@eit.lth.se

Abstract
Data analytics involves the process of data collection, data analysis, and report generation. Data mining workflow tools usually orchestrate this process. The data analysis step in this process further consists a series of machine learning algorithms. There exists a variety of data mining tools and machine learning algorithms. Each tool or algorithm has its own set of features that become factors to affect both functional and non-functional attributes of the system of data analytics. Given domain-specific requirements of data analytics, understanding the effects of these factors and their combinations provide a guideline of selecting workflow tools and machine learning algorithms. In this paper, we develop an empirical evaluation method based on the principle of Design of Experiment. We apply this method to evaluate data mining tools and machine learning algorithms towards building big data analytics for telecommunication monitoring data. Two case studies are conducted to provide insights of relations between the requirements of data analytics and the choice of a tool or algorithm in the context of data analysis workflows. The demonstration also shows that our evaluation method can facilitate the replication of this evaluation study, and can conveniently be expanded for evaluating other tools and algorithms.

Keywords: Big Data Analytics, Evaluation, Data Mining Tools, Machine Learning

1. INTRODUCTION

Big data is the new reality in the telecom world. Over recent years, the mobile broadband traffic has had an explosive growth due to widespread adoption, advanced new networks, increasing penetration of smartphones, and millions of mobile applications. This growth will continue at a rapid pace as increasing deployment of Internet of Things, sharable, uploadable and findable content by mobile users, sensors, connected cars and so on. Mobile big data has proven useful for capacity and performance monitoring (e.g., during normal operation or under massive events), troubleshooting, realistic lab testing, simulation, new feature design, an architectural evolution of mobile network infrastructure products.

A telecom vendor is specialized in manufacturing of E-UTRAN Node B (a.k.a. eNodeB) or base stations. eNodeB is the hardware connected to the mobile phone network that communicates directly with mobile handsets. To manage the quality of service delivered by eNodeBs, they are monitored and observed by metrics or KPIs.

KPI measurement is frequently used and is means used by mobile operators and mobile network infrastructure vendors to search systematically and identify the system/network bottlenecks, troubleshooting, dimensioning and anomalies. For example, mobile network operators monitor KPIs to ensure an optimal dimensioning and configuration for their network with efficient use of network resources and to provide robust QoS and consistently good user experience. For wireless infrastructure vendors, the data collected from the network contains valuable information that can reveal the runtime state and quality of both hardware and software. Mining these KPI data will result in technical insights of the effectiveness and
efficiency of the service provided to the customers.

This brings increasing challenges to processing and analyzing this vast amount of data with a huge variety in a timely fashion. The monitoring data are time series data that consist of more than 200 columns with each representing one KPI. Each KPI is read every 15 minutes, leading to 96 readings per day per KPI.

The existing conventional tools and analysis algorithms of telecom services are not designed for handling data analytics at this scale. As a result, there is only a small percentage of monitoring data has been analyzed at present. To expand the data analytics scales, data mining workflow tools play an important role. A suitable tool should address the business requirements of 1) Support workflow management that scales up day to day productivity of data analytics; 2) Provide flexibility and modularity for rapid workflow modification; 3) Make use of popular languages and libraries in data science such as R and Python; 4) Provide user-friendly workflow front end; and 5) Allow scheduling workflow jobs and even back-fill missing data.

In this paper, we experimentally explore tools and methodologies for running data analysis on telecom monitoring data. According to the clarifications in (Denning, 1981), (Feitelson, 2006), using experimentation to evaluate tools and algorithms can be regulated by experimental computer science. Considering that evaluation methodology underpins all innovation in experimental computer science” (Blackburn, 2008), we utilize a set of principled techniques of Design and Analysis of Experiments (DOE) (Antony, 2014).

We consider two case studies. First, we evaluate the data mining tools on the basis of a set of requirements. Second, we evaluate the methodologies to detect outliers to find the most suitable data mining tool. We conduct the evaluation on datasets collected from base stations in a trial mobile network with two data mining tools, RapidMiner and KNIME. The dataset time frames are 1 month, 6 months, 1 year and 2 years. Additionally, we design workflows on a KPI that provides the average number of connected users per cell on base stations. We used these workflows to conduct our studies to evaluate data mining tools and outlier detection algorithms. The observations from this evaluation provide insights of each data mining tool and the outlier detection techniques in the context of data analysis workflows. This documented design of experiment facilitates telecom software engineers to replicate this evaluation study, and can be expanded for other evaluations.

2. THE DATA ANALYSIS WORKFLOW

The concept workflow architecture is depicted in Figure 1. The workflow includes three phases 1) Extract, 2) Transform and 3) Load. These phases consist of multiple tasks to complete. The Extract phase consists of two tasks, namely Collect Data and Merge Data. This phase retrieves the data from various sources and stores at the staging area 1. The staging area 1 is the file system to store data in raw form i.e.
The stored data files are merged to form a single data file for processing. The merged file is then pushed forward to the Transform phase. In the Transform phase, the Clean Data task removes inconsistency and dirty records in the data file to prevent errors. The cleaned data is stored in staging area 2 in the form of flat files. The Process Data task of the Transform phase reads the data file from staging area 2 and commences analysis. A range of analysis methods can be used to process data. For example, outlier detection techniques, such as LOC are used for analysis. Finally, the analysis results and the cleaned data are moved to the final staging area (i.e. Staging Area 3). The staging area 3 is of data storage in a relational or No-SQL database. The Load phase is responsible for generating a report or visualizing the process data. The analysis results located in the data storage are queried for different purposes. The workflow is composed and executed within a data mining workflow management framework to execute it in a periodic manner.

There exist a variety of data mining workflow tools. Each has its own set of features. When processing the data analytics logic on a given infrastructure, these features become a set of factors that affect both the functional and non-functional attributes of the end data analytics product. The choice of a tool demands systematic evaluation to identify the relations among the factors, the combinations and requirements of data analytics. Likewise, machine learning algorithms need to be selected according to the desired features of the data analytics product. The evaluation of data mining workflow tools and machine learning algorithms are at different granularities: the former includes the entire workflow runtime, while the latter is a component of a workflow. The key to covering both kinds of evaluation is a systematic evaluation method.

3. The Evaluation Method

Data mining tools are a special type of software that aims for facilitating data mining jobs (Mikut, 2011). The comparison between software products is a typical evaluation practice that belongs to the field of experimental computer science (Denning, 1981). We adopt the principles of Design of Experiment (DoE) to guide evaluation implementation for selecting suitable data mining tools. Figure 2 outlines the steps of DoE in eight steps. The description of each step is as follows:

I. Requirement Recognition: Identification of evaluation requirements is the first and foremost task in DoE evaluation method. These evaluation requirements are necessary to comprehend system related problems as well as the evaluation purpose. The experts with prior knowledge of associated problems brainstorm to write down a set of evaluation requirement. At first, the experts use natural language to describe evaluation requirements. They are then scrutinized thoroughly and mapped to requirements questions. This process determines the objective of the assessment process that is to address these requirement questions.

II. Feature Identification: It standardizes the terms, concepts and their relationships within a system domain to determine a set of features. The features include both functional and non-functional features. This step utilizes the requirement questions to identify relevant features for evaluation.

III. Metrics and Benchmark Listing and Selection: Firstly, experts investigate the relevant metrics and benchmark and prepare a list constituting them. Then, the most appropriate ones are selected based on the available resources in hand, estimating the overhead of potential experiments, and judging the evaluator’s capabilities of operating different benchmarks. The selection process is a crucial task and plays an essential role in evaluation implementation. Once the metric and benchmarks are chosen, the selection of experimental factors begins.

IV. Experimental Factors Listing and Selection: The experimental factors are the parameters or variables that affect the performance features selected for evaluation. Similar to the previous step (i.e. Metrics and benchmark listing and selection), this step lists a set of potential candidate experimental factors. Later, the selection process considers the candidate factors with highest impact and relevance.

V. Experiment Design: According to DoE, the next step after careful selection of the metrics, benchmarks, and experimental factors is to design experiments for evaluation. In the beginning, simple experiments are designed based on the pilot trials. Later these experiments are modified for more complex designs using DOE techniques. The step outputs experimental instructions, experimental blueprints, and driving benchmarks. They are then used to implement these experiments.

VI. Experiment Implementation: This step is to carry out series of experiments. For instance,
observing the behavior of the system by gradually increasing the value of a selection factor and taking multiple reading for each value. The results obtained from the implementation step are moved forward for analysis.

VII. **Experiment Analysis:** In this step, evaluators comprehend the results and compare different systems on both functional and non-functional grounds.

VIII. **Conclusion and Documentation:** Finally, the requirement questions are answered based on the analysis results and documented for future reference.

We apply this evaluation method based on DoE to two aspects of the architecture depicted in Figure 1. The first aspect is evaluating and comparing data mining tools that support all tasks of the workflow. The second case study is evaluating outlier detection algorithms described as one of the sub-task i.e. Process Data in data mining workflow shown in Figure 1.

### 4. Evaluation of Data Mining Tools

A data mining workflow tool is a special type of software that aims for facilitating data mining jobs (Mikut, 2011). It is a core component of the architecture depicted in Figure 1 to analyze the monitoring data collected from base stations. The evaluation of data mining workflow tools focuses on comparing alternative tools according to both quantitative and qualitative requirements. We apply the process of evaluation in the aforementioned eight steps.

#### 4.1 Requirement Recognition

Driven by the evaluation method, the requirement recognition is both to understand the real-world problem and to achieve clear statements of the corresponding evaluation purpose. In fact, it is the recognized evaluation requirement that essentially drives the remaining evaluation steps. In this case, an engineering team with both telecom engineers and software engineers brainstormed the selection criteria on data mining tools as follows:

![Figure 2: The Steps of Design of Experiment](image)
It should be able to read data either from files (e.g., CSV or Excel files) or provide capabilities to read data from the database systems.

It should provide appropriate support for data manipulation and transformation.

It should provide a useful statistical model or Machine learning support for data processing.

The application should have interoperability.

It should accommodate a large volume of data, for example, a data set with approximately 2 million data points.

It should provide effective data exportation and visualization.

It should support other Scripting languages, such as R, Python, etc.

It should process data effectively and efficiently.

The interface of the tool should be user-friendly and easy to learn.

The main evaluation requirement becomes:

\textbf{RQ:} Given the selection criteria, what are the key differences between a pair of data mining tools on the same sets of data?

It is clear that the selection criteria of data mining tools include both quantitative (like mining job performance) and qualitative (like operational capabilities) concerns. Since the evaluation method encourages defining a set of specific requirement questions to be addressed by potential evaluation experiments, we further distinguish between those quantitative and qualitative criteria and clarify the above requirement as follows:

rq1. How fast does a tool perform data mining jobs?
rq2. How many resources does a tool need for data mining jobs?
rq3. How well does a tool match the qualitative selection criteria?

4.2 Tool Feature Identification

According to the clarified evaluation requirement and selection criteria, we identify the features of data mining tools to be evaluated. There are two quantitative performance features and eight qualitative indicators, as listed in Table 1. We mainly focus on the experimental evaluation of the performance features in the following subsections, while leaving the qualitative discussions to the end of this study.

<table>
<thead>
<tr>
<th>Category</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantitative Performance</td>
<td>Data Mining Latency</td>
</tr>
<tr>
<td>Qualitative Indicator</td>
<td>Data Import Support</td>
</tr>
<tr>
<td></td>
<td>Data Export Support</td>
</tr>
<tr>
<td></td>
<td>Node IO Limit</td>
</tr>
<tr>
<td></td>
<td>Scripting Language Support</td>
</tr>
<tr>
<td></td>
<td>Visualization Support</td>
</tr>
<tr>
<td></td>
<td>Data Manipulation and Transformation</td>
</tr>
<tr>
<td></td>
<td>Interoperability in Batch Mode</td>
</tr>
<tr>
<td></td>
<td>Interface Usability</td>
</tr>
</tbody>
</table>

4.3 Metrics & Benchmarks Listing and Selection

We consider two types of jobs in processing telecommunication service monitoring data, namely forecasting job and clustering job. We developed solutions to benchmarking each type of jobs. For example, to evaluate data mining tools regarding forecasting, we used the scripting language R to implement a data forecasting algorithm as a baseline. The forecasting algorithm employs Seasonal Naive and BATS forecasting models and predicts the data point for next three days. As for the clustering job, we used the K-means clustering algorithm to run clustering analysis on a dataset.

The two metrics use to measure the performance of data mining tools are the data mining jobs' Execution Time and Memory Load. Execution Time is the time required for job completion, as formulated in equation below,

\[ Time_{Exe} = TimeStamp_{end} - TimeStamp_{start} \]

Memory Load is the percentage of memory used during the job execution, which is monitored every second.

4.4 Experimental Factors Listing and Selection

Knowing the relevant factors (also called parameters or variables) is a tedious but necessary task of designing proper experiments (Le Boudec, 2010). Our evaluation method distinguishes among the workload-related, resource-related and capacity-related factor types. We identified experimental factors and their levels for evaluating data mining tools, as listed below.
Workload-related factors:
  o Data Mining Jobs: Forecasting and Clustering
  o Workload Size: 1 month, 6 months, 1 year and 2 years of data

Resource-related factors:
  o Tools Brand: RapidMiner and KNIME

Capacity-related factors:
  o Latency: Execution Time
  o Memory: Memory Usage

These benchmarks for forecasting and clustering are employed to vary data mining jobs by different sizes of workload. The workload of 1 month, 6 months, 1 year and 2 years of data have 2880, 17280, 34560 and 69120 data records respectively. We regard data mining tools as service resources to be evaluated, and then treat different tool brands as different resource types.

4.5 EXPERIMENTAL DESIGN

Evaluation experiments are subsequently prepared and designed by utilizing the selected experimental factors. Here we employ the Full-Factorial Design technique (Montgomery, 2009). Since the factors Data Mining Job, Workload Size and Tool Brand have two, four and two values respectively; we have $2 \times 4 \times 2 = 16$ types of experimental trials. Note that the capacity related factor Latency in term of Execution Time plays a response role in this design. We also measure Memory Usage as another capacity related factor.

Considering that Randomization and Replication are two principles of applying Design of Experiments (Montgomery, 2009), we decide to repeat 50 times each of the 16 types of experimental trials and randomized those trials into a design matrix (please see Table 2) by using Minitab.

<table>
<thead>
<tr>
<th>Trial</th>
<th>Data Mining Job</th>
<th>Workload Size</th>
<th>Tool Brand</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Clustering</td>
<td>1 month</td>
<td>RapidMiner</td>
</tr>
<tr>
<td>2</td>
<td>Forecasting</td>
<td>1 Year</td>
<td>RapidMiner</td>
</tr>
<tr>
<td>3</td>
<td>Forecasting</td>
<td>2 Year</td>
<td>KNIME</td>
</tr>
<tr>
<td>4</td>
<td>Clustering</td>
<td>2 Year</td>
<td>RapidMiner</td>
</tr>
<tr>
<td>5</td>
<td>Forecasting</td>
<td>1 Year</td>
<td>KNIME</td>
</tr>
<tr>
<td>6</td>
<td>Clustering</td>
<td>1 Year</td>
<td>RapidMiner</td>
</tr>
<tr>
<td>7</td>
<td>Clustering</td>
<td>1 Month</td>
<td>RapidMiner</td>
</tr>
<tr>
<td>8</td>
<td>Clustering</td>
<td>6 Months</td>
<td>KNIME</td>
</tr>
<tr>
<td>9</td>
<td>Forecasting</td>
<td>1 Month</td>
<td>RapidMiner</td>
</tr>
<tr>
<td>10</td>
<td>Forecasting</td>
<td>6 Months</td>
<td>RapidMiner</td>
</tr>
<tr>
<td>11</td>
<td>Forecasting</td>
<td>2 Years</td>
<td>RapidMiner</td>
</tr>
<tr>
<td>12</td>
<td>Clustering</td>
<td>2 Years</td>
<td>KNIME</td>
</tr>
<tr>
<td>13</td>
<td>Forecasting</td>
<td>1 Month</td>
<td>KNIME</td>
</tr>
<tr>
<td>14</td>
<td>Clustering</td>
<td>6 Months</td>
<td>KNIME</td>
</tr>
<tr>
<td>15</td>
<td>Forecasting</td>
<td>6 Months</td>
<td>KNIME</td>
</tr>
<tr>
<td>16</td>
<td>Clustering</td>
<td>1 Year</td>
<td>KNIME</td>
</tr>
</tbody>
</table>

4.6 EXPERIMENTAL IMPLEMENTATION

The evaluation experiments were implemented following the full factorial design. The experimental environment is consistent for running both forecasting and clustering jobs on RapidMiner and KNIME.

In the case of the forecasting job run by RapidMiner, the workflow includes three different phases: (1) The first phase reads data from the CSV files, and then passes the data to the next phase. (2) The preprocessing phase prepares the data and extracts the training and testing sets from the entire data set. (3) The last phase is composed of three major processes, namely data processing, validation, and plotting.

The prepared training data set is fed to the forecasting models, and the testing dataset is used to validate the forecasting results based on the validation metrics, such as root-mean-square error (RMSE) and mean absolute error (MAE) (Chai, 2014). This phase also plots the results for the purpose of visualization.

Along with the changing workloads, RapidMiner's execution time of running the forecasting job varies ranging from about 29 seconds to 107 seconds, on average, for the data sizes from 1-month to 2-year.

Unlike RapidMiner, each of the KNIME nodes supports one input node only. As a result, KNIME's forecasting workflow has four phases, namely reading data, preprocessing, data processing, and merging and plotting. In particular, the third phase provides data processing and validation, while the last phase provides data merging and plotting. We show the workflow of KNIME's forecasting job, as illustrated in Figure 3.

Given the above experimental setup, a KNIME's forecasting job takes approximately from 35 seconds for processing 1-month data to around 107 seconds for 2-year data. As for the clustering job, the corresponding workflow is divided into four phases including reading, preprocessing, clustering and writing data for both data mining tools. Similarly, here we only show the workflow of KNIME's clustering job, as illustrated in Figure 4.
Since the clustering job is simpler than the forecasting job, the two tools’ execution time changes between around 8 seconds and about 13 seconds for different sizes of workloads. We further visualize the response time given the workload size is shown in Figure 5 and Figure 6.

Likewise, we evaluate the memory usage of running jobs in both the forecasting and clustering workflows as shown in Figure 7 and Figure 8. Both jobs have a steady increase of the memory usage, indicating that the algorithms applied in each job do not have a big
size of intermediate results that lead to intensive demands of memory usage. This is an important indicator to consider for the workflow design since an intermediate step can contribute to the capacity related factors, while the workload related factors (such as the job types and workload sizes) and the resource-related remain the same.

Figure 5: Average Execution Time of the Forecasting Job

Figure 6: Average Execution Time of the Clustering Job

Figure 7: Average Memory Usage of the Clustering Job

4.7 EXPERIMENTAL ANALYSIS

Given the results from the experimental implementation, we further conduct quantitative and qualitative analyses for those features respectively.

4.7.1 ANALYSIS FOR QUANTITATIVE FEATURES

For evaluating quantitative features, DoE strongly suggests employing suitable statistical methods for experimental analysis. Note that the statistical methods do not directly prove any factor's effect (Montgomery, 2009) in the context of factorial experimental design. However, statistical analysis can add objectivity to drawing conclusions and to the potential decision-making process.

In this case, we first investigate if the effect of data mining tools on job execution could be influenced by the other factors, by statistically analyzing the interactions between different experimental factors against the response Latency. Benefiting from Minitab, we use the Interaction Plot to visualize the factor interactions, as shown in Figure 9. In an interaction plot, the greater the difference in slope between two lines, the higher the degree of interaction between the corresponding factors while parallel lines indicate no interaction. It is then clear that there is a potential interaction between the factors Data Mining Job and Workload Size but no interaction between these two factors and Tool Brand. In other words, changing the value of Data Mining Job and Workload Size will not influence the effect of Tool Brand on job execution time.
Secondly, we investigate how significant changing data mining tools would impact on job execution time, by statistically analyzing the influences of different experimental factors on the response Latency. We employ the Pareto Chart to visualize the effects of the various experimental factors and their combinations, as shown in Figure 10. In a Pareto chart, the absolute effect values are drawn with a reference line. Any effect that extends beyond this reference line indicates a potentially important factor or factor combination. As can be seen, the factors Data Mining Job and Workload Size and their combination have potentially significant effects on the execution time of data mining jobs, whereas Tool Brand seems not to be a major factor regarding the response Latency.

In terms of memory usage, the analysis in Figure 11 and Figure 12 shows two different tool brands have a clear cross trend concerning the data mining jobs, more specifically, RapidMiner uses less memory for the clustering jobs, while KNIME uses less memory for the forecasting jobs.

For the qualitative features, we directly discuss them one by one to compare those two data mining tools, and the discussions are based on our experiences of implementing the aforementioned experiments.

- **Data Import Support:** Both RapidMiner and KNIME support importing data from a wide range of file formats and databases, such as CSV files, Excel files, Sequence files, etc. In addition, they also support model import and data streaming from various databases.

- **Data Export Support:** Similar to data importation, both KNIME and RapidMiner support enhanced
data exportation support for a wide range of file formats, models, and databases. However, the open source version of RapidMiner has limitations to the support.

- **Node IO Limit:** Another desirable feature is the capability of a node to accept multiple inputs from and output flow connections to other nodes in a workflow management system. While RapidMiner allows multiple inputs and output flow capabilities in the workflow designing, KNIME has a limitation regarding input ports. Since a single node in KNIME has one or maximum two input port(s), making the workflow design complicated for complex workflows.

- **Scripting Language Support:** KNIME and RapidMiner both have efficient support for R and Python scripting languages. The advanced Node IO feature (as discussed in the above point) provides additional flexibility for external scripting languages in the workflow.

- **Visualization Support:** RapidMiner has exclusive dynamic data visualization support over KNIME. This feature provides the support to a wide range of charts for visualization, including scatter plot, heat maps, multiline plots, etc. However, this feature exacerbates its performance while visualizing a large amount of data.

- **Data Manipulation and Transformation:** KNIME and RapidMiner have a similar number of nodes to support data manipulation, statistical modeling, and machine learning algorithms. They support machine learning algorithms including Decision Trees, K-Means, Support Vector Machine (SVM), and Bayesian Networks.

- **Interoperability in Batch Mode:** KNIME and RapidMiner support batch mode processing, allowing them to run faster with loading GUI every time a workflow execution takes place.

- **Interface Usability:** KNIME has the same legacy interface of popular Eclipse IDE, making it easier for developers to interact efficiently with it. RapidMiner's interface is more user-friendly and interactive and provides effective tutorials that make it easier for a novice to learn.

### 4.8 Conclusion and Documentation

As regulated by the principles of DoE, we summarize answers to the predefined requirement questions before drawing conclusions, as outlined below.

**How fast do RapidMiner and KNIME perform data mining jobs respectively?**

When running the forecasting job with different workload sizes, the execution time of KNIME varies ranging from about 35 to 107 seconds on average and of RapidMiner takes from nearly 35 to around 107 seconds (See Figure 5). As for the clustering job with different workload sizes, the latency range of KNIME is from about 9.8 to 13.4 seconds and of RapidMiner is from 8 to 9.3 seconds (See Figure ). RapidMiner shows comparatively lower latency in both the cases.

**How many resources do RapidMiner and KNIME need for data mining jobs respectively?**

Since the data size is consistent for running both data mining tools, the implementation of algorithms by each tool is a key factor on the memory consumption of each type of job. In our case, RapidMiner shows less memory utilization in both forecasting and clustering jobs, as shown in Figure 7 and Figure . RapidMiner relatively requires less number of nodes to design a workflow, which entails less memory consumption.

**How well do RapidMiner and KNIME match the qualitative selection criteria?**

Given the qualitative discussion in Section 4.7.2 Analysis for Qualitative Features, KNIME beats RapidMiner in terms of Data Import Support| and Data Export Support. RapidMiner is better than KNIME in terms of Node IO Limit, Scripting Language Support, Visualization Support and Interface Usability. KNIME and RapidMiner have the equivalent capacity with respect to Data Manipulation and Transformation and Interoperability in Batch Mode.

Overall, these answers reveal little difference between KNIME and RapidMiner regarding their quantitative features. The experimental analysis also confirms that Tool Brand is not a significant factor for executing data mining jobs, although RapidMiner runs slightly faster than KNIME in this study. Assuming the data analysis architecture has a specific emphasis on features such as Interface usability, we can select data mining tools based on their qualitative features. In this example, RapidMiner has more advantages over KNIME for interface usability. Therefore, RapidMiner can be chosen.

We have documented our evaluation logic and activities including directly reusable codes and scripts to facilitate telecom software engineers to replicate this evaluation study and evaluate other data mining tools.

### 5. Evaluation of Analysis Methods
The concept architecture in Figure 1 is further used to evaluate outlier detection algorithms in the step of Process Data. Similar to the evaluation of a data mining tool for the whole workflow, case study, we employ the eight steps of DoE to compare analysis methods.

**5.1 REQUIREMENT RECOGNITION**

After thoroughly brainstorming the requirements for the analysis method i.e. outlier detection method, we develop the following five requirements for the analysis method:

- **Distribution-independent:** Several outlier detection methods, especially statistical methods, which are based on estimating standard deviation, make assumptions about underlying data distribution. For instance, Grubb's test is an outlier detection technique that assumes the underlying dataset to be normally distributed. However, it is not possible to assume the distribution of all the features in a multivariate dataset. Such dataset contains several features having distinct data distributions. Thus, the outlier detection approach should be independent of the distribution.

- **Non-Parametric:** The outlier detection approach should also be non-parametric. Parametric models provision multiple fixed sets of parameters, which need to be set to an appropriate value to achieve efficient detection. Estimating these parameters is a tedious process, which requires adjustment for every data distribution.

- **No Explicit Training:** Patterns observed in telecommunication data constantly change due to the addition of new features or continuous upgrade. Training detection models with an obsolete data set entail inefficient results. Therefore, the outlier detection algorithm should be capable of detecting outliers without any explicit training.

- **Applicable on Multivariate Dataset:** Most of the statistical outlier detection approaches are applicable on a univariate dataset. However, our dataset is a multivariate dataset consisting of several independent features. Thus, the detection method should be applicable on a multidimensional dataset.

- **Applicable on Large Dataset:** The outlier detection technique should be capable of accommodating a large volume of data within a limited amount of available resources.

Similar to Case Study I, the main evaluation requirements are realized into the following requirement questions:

- rq1. How efficient and effective is the outlier detection method on given telecommunication data?
- rq2. How much is the resource utilization of the method?
- rq3. How well does a method match the qualitative selection criteria?

**5.2 METHODOLOGIES FEATURE IDENTIFICATION**

Based on requirements in the previous section, the selection criteria are divided into Qualitative Indicators and Quantitative Performance. The selection criteria are listed in Table 3. There is a broad range of outlier detection techniques available. However, we consider the most popular one for our analysis. Table 4 provides the outlier detection algorithms considered. We use the qualitative indicators to eliminate detection techniques that do not meet these criteria. For instance, since Grubb's test is distribution dependent, and its assumption of normal distribution contradicts the requirements listed previously, this technique is eliminated. Qualitative analysis is discussed later in section 4.7.2 Analysis for Qualitative Features.

**Table 3: Selection Criteria**

<table>
<thead>
<tr>
<th>Category</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Quantitative</strong></td>
<td>Data Mining Latency</td>
</tr>
<tr>
<td></td>
<td>Data Mining Resource Usage</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Qualitative</strong></td>
<td>Distribution Independent</td>
</tr>
<tr>
<td><strong>Indicator</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Non-Parametric</td>
</tr>
<tr>
<td></td>
<td>No Explicit Training</td>
</tr>
<tr>
<td></td>
<td>Multivariate Dataset</td>
</tr>
<tr>
<td></td>
<td>Large Data Volume</td>
</tr>
</tbody>
</table>

**5.3 METRICS & BENCHMARKS LISTING AND SELECTION**

We design a workflow to evaluate the performance and efficiency of the remaining methodologies. The process consists of several sub-processes as illustrated in Figure 1. The job retrieves specified amount of data from sources. This data is cleaned and stored in the staging area. The outlier detection methods extract the data from the staging area and process it to determine outliers in the dataset. Later, the data is labeled and stored for further analysis. Similar to the previous case
study, where we evaluate outlier detection techniques by computing the latency and memory usage of the ETL workflow. We observe the latency and memory usage under different sizes of data being processed.

### 5.4 Experimental Factors Listing and Selection

We identify experimentation factors and categorized them into four groups, namely workload-related, approach-related, efficiency-related and capacity-related factors.

- **Workload-related factors:**
  - Job: Outlier detection method
  - Data size/Number of records: 10,000, 25,000, 50,000, 100,000, 800,000

- **Efficiency-related factors:**
  - Threshold
  - False Positive/Detection Rate
  - True Positive/Detection Rate

- **Capacity-related factors:**
  - Latency: Execution Time
  - Resource usage: Memory Usage

#### Table 4: Outlier Detection Techniques

<table>
<thead>
<tr>
<th>Category</th>
<th>Outlier Detection Techniques</th>
<th>Distribution</th>
<th>Non-Parametric</th>
<th>No Explicit Training</th>
<th>Multivariate Dataset</th>
<th>Large Data Volume</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Statistical Approach</strong></td>
<td>Grubb’s Test</td>
<td>x</td>
<td>x</td>
<td>φ</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Chi-Square Test</td>
<td>x</td>
<td>x</td>
<td>φ</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>Kernel Density Estimation (KDE)</td>
<td>x</td>
<td>x</td>
<td>φ</td>
<td>φ</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>General Extreme Student Estimate</td>
<td>x</td>
<td>x</td>
<td>φ</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td><strong>Classification Approach</strong></td>
<td>One Class Support Vector Machine (SVM)</td>
<td>φ</td>
<td>φ</td>
<td>x</td>
<td>φ</td>
<td>φ</td>
</tr>
<tr>
<td></td>
<td>Bayesian Naive Random forest</td>
<td>φ</td>
<td>φ</td>
<td>x</td>
<td>φ</td>
<td>φ</td>
</tr>
<tr>
<td><strong>Clustering Approach</strong></td>
<td>K-Means</td>
<td>φ</td>
<td>φ</td>
<td>x</td>
<td>φ</td>
<td>φ</td>
</tr>
<tr>
<td></td>
<td>DBSCAN</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
</tr>
<tr>
<td></td>
<td>OPTICS</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
</tr>
<tr>
<td></td>
<td>Self-Organized Mapping (SOM)</td>
<td>φ</td>
<td>φ</td>
<td>x</td>
<td>φ</td>
<td>φ</td>
</tr>
<tr>
<td><strong>Nearest Neighbor Approach</strong></td>
<td>Global KNN</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
</tr>
<tr>
<td></td>
<td>Local Outlier Factor (LOF)</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
<td>φ</td>
</tr>
</tbody>
</table>

### 5.5 Experimental Design and Implementation

The experiment design utilizes the experimental factors identified in the previous step to prepare for experimentation. Based on the metrics, benchmarks and experimental factors, two sets of experiments are designed to evaluate efficiency and performance.

For efficiency experiments, we evaluate four methods of DBSCAN, OPTICS, LOF, and GKNN. For each method, we setup three different threshold values and measure the true positive rate and false positive rate. In total, we have $4 \times 3 = 12$ trials.

For performance experiments, we also evaluate four methods of DBSCAN, OPTICS, LOF and GKNN. For
In each method, we input the data size as 10,000 records, 25,000 records, 50,000 records, 100,000 records, and 800,000 records. We measure the execution time and memory usage for each input of each method. This results in totally $4 \times 5 = 20$ trials.

Experiment implementation involves several tasks for data preparation before commencing the experimentation. These tasks are data cleaning and data labeling. The first and most important step for evaluating outlier detection techniques are Data preparation. It consists of the following tasks:

- **Data Cleaning**: It is a process that removes any sort of syntactic errors, semantic errors and coverage anomalies present in the dataset prior to its processing. For instance, in our case rows containing lexical errors such as `!DIV0!` present in the dataset is eliminated from the dataset. We removed the error by scripts programmed in R since the amount of error present was limited and known.

- **Data Labeling**: this step is a process of tagging the data record in the dataset based on domain knowledge provisioned by telecommunication experts. For instance, we know that the success rate always lies between 0 and 1. Any value less than zero and more than one is an anomaly and thus tagged.

After the data preparation, we feed the dataset into the workflow architecture to evaluate different anomaly detection techniques. The experimentation takes place by gradually increasing the amount of data points (or records).

To compare the efficiency of these outlier detection techniques, the Receiver Operating Characteristic (ROC) curves are plotted. ROC curve is a graphical evaluation method used for assessing machine learning algorithms, especially, binary classifiers. ROC curve is obtained by calculating True Positive Rate (TPR) and False Positive Rate (FPR) of the detection algorithms. Where TPR is a measure of correct positives out of all the positives detected by and algorithm. On the other hand, FPR is a measure of incorrect positives out of all the positives detected. The TPR and FPR represent the $y$-axis and $x$-axis of the ROC curve and are formulated using the following equations.

$$TPR = \frac{True\ Positive}{True\ Positive + \ False\ Negative}$$

$$FPR = \frac{False\ Positive}{False\ Positive + \ True\ Negative}$$

Each ROC curve for an outlier detection method is obtained by running it under different parameter values. The performance of each outlier detection technique depends on the value assigned to its parameters. Each parameter has a different impact on the performance. For instance, the accuracy of the DBSCAN's prediction highly depends on the value of epsilon and minimum points (i.e. the maximum distance between two samples for them to be considered as in the same neighborhood and number of data points to be considered to form a dense cluster, respectively) (Pedregosa, 2011). Therefore, only those parameters that have maximum impact on algorithms performance are selected, i.e. epsilon in case off DBSCAN. After running each outlier detection technique, we compare the results (i.e. prediction labels) produced during the detection process with the original labels to calculate TPR and FPR. Then, we plot ROC curve for each outlier detection technique and compared them through analysis.

In addition, we also profile execution time and memory usage of detection techniques and plotted the results for the experiments shown in Figure 13 and Figure 14. For this purpose, 10 readings were taken for execution time and memory usage by gradually increasing the data size, i.e. 10,000, 25,000, 50,000, 100,000 and 800,000 records, at the end of each cycle of ETL job.

### 5.6 Experimental Analysis

#### 5.6.1 Analysis of Quantitative Features

In terms of Efficiency, we plot the ROC curve for the outlier detection techniques as shown in Figure 15. The ROC curve of the technique Local Outlier Factor (LOF) shows the best performance among the evaluated detection techniques. It shows the maximum TPR of 0.7 and minimum FPR of 0.15 approximately. In contrast, OPTICS and DBSCAN have an almost same area under the curve and shows similar behavior according to the analysis.

In terms of the Execution Time and the Memory Usage, we employ Pareto Chart to visualize the effects of the different experimental factors and their combinations. Two factors are plotted in the charts. Factor A represents Technique; Factor B represents Data Size, and Factor AB represents the combinations of the two factors. We compare LOF with DBSCAN and OPTICS. From Figure 16 and Figure 17, the analysis
shows that the factor Data Size has a relatively significant influence on the execution time. As Figure 18 and Figure 19 show Technique and Data Size have no significant influence on the experimental results.

Figure 13: Average Execution Time of Outliers Detection

Figure 14: Average Memory Usage of Outliers Detection Methods

Figure 15: Efficiency Analysis Result
5.6.2 Analysis of Qualitative Features

In the case of outlier detection techniques, the quality attributes are used as selection criteria. The selection process is a subtask and forms a basis of quantitative analysis. The outlier detection technique that complies all these qualitative criteria will be assessed. Each qualitative criterion is analyzed independently (i.e. without considering its dependency on other features) for each outlier detection technique. The study results are illustrated in.

- **Distribution Independent:** Most of the statistical outliers’ detection techniques, such as Grubb’s test and Chi-Square test, are distribution dependent. These statistical approaches rely on the underlying data distribution, and most of them are only applicable on a normally distributed dataset. In our case, these techniques are inapplicable as the base stations or eNodeBs generates data with distinct data distributions. After scrutinizing the outlier detection methods in consideration, we found that Grubb’s Test, Chi Square Test, Kernel Density Estimation (KDE), General Extreme Student Estimate are distribution dependent whereas the remaining techniques are independent of underlying data distribution.

- **Non-Parametric:** Similarly, distribution independent methods are also non-parametric. Therefore, based on our analysis Grubb’s Test, Chi Square Test, Kernel Density Estimation (KDE), General Extreme Student Estimate techniques are parametric, and remaining detection techniques show non-parametric behavior.

- **No Explicit Training:** Telecommunication companies generate a large volume of data. The large volume makes training data preparation a
tedious task. Therefore, outlier detection technique should not need any prior data preparation. Clustering and classification detection techniques such as One-class support vector machine (SVM), Bayesian naive, Random forest, K-Means, Self-Organized Mapping (SOM) are effective methodologies for outlier detection but need explicit training. Training these techniques require preparation of training data not feasible in our case. However, outlier detection techniques such as Grubb’s Test, Kernel Density Estimation (KDE), DBSCAN, OPTICS, Global KNN, Local Outlier Factor (LOF) do not require any training data, thereby, they are more suitable for analysis.

- **Multivariate Data**: Statistical outlier detection techniques i.e. (Grubb’s Test, Chi Square Test, General Extreme Student Estimate techniques) are only applicable on a univariate dataset. Whereas, Kernel density estimation (KDE) has an implementation for both univariate and multivariate datasets. To the best of our knowledge, most of the machine learning algorithms apply to both univariate and multivariate data. However, they tend to perform better on multivariate data than univariate data.

- **Large Volume of Data**: As mentioned previously, the base stations generate a large volume of data. Hence, the analysis technique should be capable of processing large data. In such case, Grubb’s Test, Chi Square Test, General Extreme Student Estimate are inapplicable for large datasets, but the remaining techniques work fine on large datasets.

Based on the above observations, only four outlier detection techniques qualify for quantitative assessment. These techniques are DBSCAN, OPTICS, LOF and Global KNN.

### 5.7 CONCLUSION AND DOCUMENTATION

As regulated by the DoE principles, we summarize answers to the predefined requirement questions as drawing conclusions, as outlined below.

**How much is the resource utilization of the method?**

The analysis results indicate that Global KNN is the fastest among others outperforming them with execution time of approximately 2 seconds to process 100,000 data points. However, its execution time eventually becomes similar to LOF with increasing data size becoming 3 seconds for 800,000 data points. GKNN and LOF also consume the almost same amount of memory for analysis. In the case of memory usage, OPTICS has the minimum utilization for 800,000 data points. Although the method has maximum execution time, its memory consumption is comparatively low. DBSCAN is unable to process large data volume due to excessive memory use.

**How well does a method match the qualitative selection criteria?**

Given the qualitative discussion in Section 5.6.2 Analysis for Qualitative Features, eliminating the detection methods that do not meet our criteria led us to extract four methods out of the candidate ones. These methods are DBSCAN, OPTICS, Global KNN and Local Outlier Factor. These methods are non-parametric, distribution independent, multivariate and do not require explicit training.

### 6. RELATED WORK

Selection of suitable software applications to carry out specific tasks has become challenging, due to the rapid development and availability of software. A number of performance evaluation techniques or methods have been developed to support this selection process. Performance evaluation is a method to determine the strengths and weaknesses of the underlying architecture or design pattern. Some evaluation methods developed are discussed in (Shanmugapriya, 2012), where they are categorized into an early and late evaluation.

The early evaluation methods are software evaluation methods that can assess the software application based on its specification and description. They are employed to analyze software quality attributes such as reliability, performance, scalability and availability. Most of these evaluation methods are scenario based. Scenario based evaluation methods identify scenarios in close interaction with the stakeholders and systematically investigate the software architecture based on these scenarios. Some of the examples are Software Architecture Analysis Method (SAAM), Architecture Trade-off Analysis Method (ATAM), Family Architecture Analysis Method (FAAM), and Domain-Specific Software Architecture Comparison.
Model (DoSAM) (Ionita, 2002) (Kazman R. a., 1994), (Kazman R. a., 1998), (Bergner, 2005).

SAAM was first introduced in 1993 as a scenario-based early evaluation method (Dobrica, 2002). The main advantage is this evaluation method is its adaptive design (Kazman R. a., 1993) that allows the modification of SAAM's rationale design to develop evaluation methods for specific requirements. The method includes the six steps or activities of 1) scenario development, 2) System Architecture (SA) description, 3) Scenario classification and prioritization, 4) individual scenario evaluation, 5) scenario interaction, and 6) overall evaluation (Babar, 2004). ATAM is another evaluation method for assessing quality attributes such as modifiability, portability, extensibility, and integrality.

DoSAM is another scenario-based evaluation method designed to assess software quality attributes like performance, scalability, and availability (Bergner, 2005).

The late evaluation methods are employed where the software application is prone to changes. An approach is introduced in (Tvedt, 2002) to evaluate software applications that undergo modification during the implementation process. It avoids system degeneration by actively and systematically detecting and correcting deviations from the planned design as soon as possible, based on explicit and implicit architectural guidelines. It has the following seven steps as follows: 1) Select the perspective for evaluation; 2) Define and select guidelines, and establish metrics to be used in the evaluation; 3) Analyze the planned architectural design in order to define architectural design goals; 4) Analyze the source code in order to reverse-engineer the actual architectural design; 5) Compare the actual design to the planned design in order to identify deviations; 6) Formulate change recommendations in order to align actual and planned designs; and 7) Verify that the design goal violations have been corrected by repeating steps 4 through 6. Other late evaluation methods are discussed in (Lindvall, 2003), (Fiutem, 1998), (Murphy, 1995), (Sefika, 1996).

Most of these evaluation methods are intended for the evaluation of a single architecture at a particular point in time. In a case of comparing tools, they mainly focused on comparing their results against a certain analysis job, for example, the accuracy of classification (Bernardino, 2013), (Al-Shawakfa, 2011). Inspired by the property “Velocity” of Big Data analytics, we are more concerned with the performance of different tools. More importantly, to our knowledge, there lacks a systematic study of evaluating data mining tools that are driven by requirements derived from a data analysis context. Thus, our study can also be viewed as an experience report on evaluating data mining tools by following a relatively rigorous methodology and applying principles of Design of Experiment (DOE) techniques.

DOE emerged traditionally for agriculture, chemical, and process industries. Considering its natural relationship with experimental activities, suitable DOE techniques have also been employed in experimental computer science. When it comes to the software engineering field, the main interest of applying DOE seems to be in software testing from the developer's perspective (Iannino, 1997), (Reilly, 2002), (Zevallos, 1998). Our study essentially extends the applicability of DOE to software comparison from the end user's perspective.

7. Conclusion
In this paper, we present our design of experiment to evaluate data mining tools and outlier detection approaches into two different case studies. The aim of this evaluation is to observe them both quantitatively and qualitatively when running data analytic jobs. The analytics is to help understand the number of connected users per cell on the base stations in the trial mobile network. Since the data are constantly generated and accumulated, the workload size becomes one experimental factor in both the cases during the period of time when datasets are collected. The evaluation could become complicated because of the combination of factors. Therefore, applying DOE principles to our evaluation study clearly make this practice in a structured eight-step procedure. At the end, the evaluation study allows us to address the evaluation requirements. The threats of validation are mainly in the evaluation environment. We used default settings in each study. Each case can be further optimized to achieve a better response time and memory utilization. Our purpose is not to compare any data mining tools or outlier detection approach in an absolute measurement. Instead, we focus on the evaluation method that allows us to observe the features and metrics of these tools or approaches, respectively. The insights can help practitioners make the decision on how a data mining tools or outlier detection approaches should be fit into the big data analysis architecture. We believe this evaluation study based on the design of experiment can be extended to
evaluating other data mining tools or outlier detection approaches.

8. REFERENCES


Chai, T. a. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? Geoscientific Model Development Discussions, 7, 1525–1534.


Authors

Samneet Singh is a master student of applied science at Concordia University pursuing his research work under the supervision of Dr. Yan Liu. His major interest lies in cloud computing, distributed systems, big data analytic, and machine learning. Currently, he is working as a data analytics intern at Ericsson Canada Inc.

Yan Liu is an Associate Professor at the Electrical and Computer Engineering Department, Concordia University, Canada. Her research interests are data stream processing, cloud software architecture, distributed computing, software performance engineering, and adaptive systems. Her research is funded by Natural Sciences and Engineering Research Council of Canada (NSERC), Fonds Nature et technologies, Quebec, Canada (FRQNT) and Mitacs (Canadian Network of Centres of Excellence). Prior to her academic career, She was a senior scientist at the Pacific Northwest National Laboratory in Washington State, leading US Department of Energy funded projects. She graduated her Ph.D. from The University of Sydney, and worked as senior researcher at National ICT Australia before moving to North America.

Zheng Li received his Ph.D. degree and M.E. by Research degree from the Australian National University (ANU) and the University of New South Wales (UNSW) respectively. During the same time, he was a graduate researcher with the Software Systems Research Group (SSRG) at National ICT Australia (NICTA). Before
studying abroad, he had around four-year industrial experience in China after receiving his M.Sc.Eng. degree from the Beijing University of Chemical Technology and the B.Eng. degree from the Zhengzhou University. His research interests include Cloud computing, performance engineering, empirical software engineering, software cost/effort estimation, and Web service composition.

Wayne Ding is a senior specialist in Ericsson, Canada. He is involved in 4G and 5G wireless system R&D. His work focus on mobile big data analytics, machine learning and wireless network KPIs. He received the BS and MS degrees from Dalhousie University, Canada.