UPTEC X 17 018
Examensarbete 30 hp
Juni 2017

Exploration of big data and machine learning in retail

Johan Edblom

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Exploration of big data and machine learning in retail

Johan Edblom

During the last couple of years, there has been an immense increase in data generation. This new data era has been referred to as the big data paradigm. More and more business areas are today realizing the power of capturing more data, and by this hope to reveal hidden patterns and gain new insights of their business. ICA is one of the largest retail business in Sweden, and saw the potential of utilizing the big data technologies to take the next step in digitalisation.

The objective of this thesis is to investigate the role of these techniques in combination with machine learning algorithms and highlights advantages and possible limitations. Two use cases were implemented and tested which reveals possible application areas and important aspects to consider.

Handledare: Lotta Silverberg
Ämnesgranskare: Kjell Orsborn
Examinator: Jan Andersson
ISSN: 1401-2138, UPTEC X 17 018

Populärvetenskaplig sammanfattning

Under de senaste åren har det skett en extrem ökning av datagenerering. Företag och organisationer samlar in mer och mer data, vilket har utmanat dagens traditionella system i form av effektiva sätt för lagring, hantering, analys och visualisering. Detta har gett upphov till en ny teknisk term, benämnd som ”big data”.

Som ett svar på ”big data” har nya tekniker tagits fram vars mål är att erbjuda effektiva sätt att både spara och lagra, hantera, analysera och manövrera dessa enorma mängder data. En av de populäraste och mest använda tekniker är Hadoop, vilket är ett *open source*-projekt det vill säga ett mjukvaruprojekt som är öppet för allmänheten att bidra till. Projektet är uppbyggt av flera underprojekt/moduler som var och en erbjuder verktyg för olika ändamål såsom skalbar dataanalys, lagring av stora datamängder, realtidsströmning av data samt avancerade analyser med maskininlärningsalgoritmer.

Många branscher är nyfikna på dessa tekniker då de erbjuder nya effektiva sätt att hantera data och skapar nya insikter, och en bransch som har mycket att vinna på detta är detaljhandeln. ICA är ett av Sveriges största företag inom detaljhandeln och är även de intresserade av den digitalisering som fortgår i samband med konceptet kring ”big data”. Målet med detta examensarbete har därför varit att djupdyka i innebörden av teknikerna samt undersöka relevanta områden som kan bidra till nya kundinsikter för ICA.

Resultaten i detta examensarbete visar på de svårigheter som kan uppstå med att använda dessa tekniker samt föreslår två användningsområden som skapar värde för ICA och detaljhandeln generellt. En utförlig diskussion presenteras där viktiga aspekter och lärdomar sätts i ett sammanhang för framtida problemställningar.

List of contents

ABBREVIATIONS

API	Application Programming Interface
CSV	Comma-Separated Value
HDFS	Hadoop Distributed File System
IDE	Integrated Development Environment
JVM	Java Virtual Machine
LHS	Left Hand Side
NoSQL	Not Only Structured Query Language
NGS	Next Generation Sequencing
OS	Operative System
RDD	Resilient Distributed Dataset
RHS	Right Hands Side
SQL	Structured Query Language

1 INTRODUCTION

More and more business areas are today conforming to the big data domain, and the retail business has a lot to earn by giving the customer data more attention. However, most retailers today make simple assumptions based on the data about their customers, which ultimately leads to non-evolving personas as the customer's needs changes throughout time.

ICA is one of the largest retailer of goods in Sweden, with over 1300 stores and a revenue of 72 624 MSEK(1). Due to the company's size, it does not come as a great surprise that there is a tremendous amount of customer data that can be utilized in big data applications and advanced analytics. To investigate this further, a project initiative was set up at ICA where the goal was to investigate innovative ways to capture customer data and by this achieve better insights and hopefully be able to perform new types of analytics.

This project investigates and utilizes the big data technologies together with suitable machine learning algorithms to provide ICA with insights in what is possible and what might be obstacles to overcome in the future. In summary, the task is to cover both technical and practical aspects of the big data paradigm and enable high-level analytics.

.

2 TECHNICAL BACKGROUND

2.1 Big data: the new paradigm

During the last couple of years, there has been an immense increase in data generation and it keeps increasing every year (3). According to IBM:s webpage 2.5 quintillion bytes, or exabytes (10^{18}), are generated each day (4).

Companies and media channels are collecting new types of data, and there is a large diversity of the data being collected. It can for example include social media posts, various kinds of interactions and clickstream data (5).

This immense new flow of data is referred to as “big data”. Big data is often described by a set of V:s, and more V:s are added over time to describe the full picture of big data. IBM lists four V:s; *Volume, Variety, Velocity* and *Veracity*.

Volume describes the size of the data that can exceed petabytes for a dataset. **Variety** describes the different forms of data. This includes different types of formats as well as the structure of the data, which can include both structured and unstructured. **Velocity** describes the stream of data, including how fast it’s collected and how fast it’s processed. **Veracity** describes the uncertainty of the data, such as at which level it can be trusted. This can be seen as a measurement of how true the data is, for example if the data still is valid (6). These definitions give a good insight in the complexity of big data and why it has been labelled as a buzzword and a quite vague term, since these definitions alone are very broad. However, they are all aspects that are important to take into consideration when evaluating the big data model and what kind of data that is being collected for its purpose (7).

2.2 Modelling big data

With the increasing data quantity that the concept of big data brings, it is essential to understand that there is a need to investigate the current system architecture and evaluate which parts that are directly affected by these concepts. This since it might be worth to consider a remodelling of the architectural design to harvest the full potential of the big data concepts. For example, most systems today rely on storage solutions based on RDBMS, but with new types and quantities of data being captured those types of systems experience challenges with capturing, storing, searching, analysing and visualizing data in effective manners. The concept of big data also challenges traditional RDBMS with the concept of NoSQL, which has the ability to store unstructured data. RDBMS is only deployable for structured data and cannot handle semi-structured or unstructured data in large volumes and heterogeneity, which big data entails (8).

2.3 Apache Hadoop: A suitable framework for big data

A well-suited framework for the big data concept is Apache Hadoop. Hadoop is one of the most adopted open source operating system architectures for processing and storing large data sets (9). It offers the user distributed computing that is both reliable and horizontally scalable in terms of computer clustering through simple models. It also removes the need for high performance hardware since Hadoop itself is designed to detect failures and is built on commodity hardware (9) (10). The framework offers several subprojects or modules that enables a high-performance system, suitable for a big data model and most application areas (10).

2.3.1 *Data storage in big data*

One of the biggest challenges in a big data system is data storage, which is resolved in the infrastructure of the subproject Apache Hive. One of the most powerful characteristics of Apache Hive is that it offers a hybrid solution between SQL and NoSQL; data is queried in SQL-like language and stored in a NoSQL variable database, resembling the one seen in traditional RDBMS where data is put in tables. This enables ease of use for users accustomed to the traditional SQL standard. The NoSQL variable of Apache Hive is that the rows in a table consists of a specific number of columns, where each column has an associated type, which can be either complex or primitive. This is the characteristic of Apache Hive that enables a storage solution that is not sensitive to increasing data volume (11).

The tables in Apache Hive are stored in HDFS, which is a Java-based file system framework enabling scalable and reliable storage (12). HDFS is explicitly designed to be run on commodity hardware, making it highly fault-tolerant through replication of file blocks (13), and the files can exceed several petabytes in size and are designed to be stored across a cluster of machines (12).

2.3.2 *Data management with Hadoop YARN*

While HDFS offers data storage through a large number of different data access applications, *Yet Another Resource Negotiator* (YARN) manages the coordination of these applications (12). YARN is designed to manage resources in the cluster and schedules job with the help of different dameons. The ResourceManager (RM) together with the NodeManager (NM) constitutes the data-computation framework. The RM has the highest authority in the hierarchy and handles the application resources in the system. The NM handles the containers on each node in the cluster and reports on resource usage to the RM. A third party, the ApplicationMaster (AM), constitutes a framework that negotiates resources between the RM and NM (14).

When a user-submitted application is passed to the cluster, YARN initiates the AM which in turn checks and requests for available resources (15). Containers are then started, which can be seen as reservations of resources in the cluster (16). Once the containers have been allocated, the job can be started and a continuous communication between YARN and the AM is held. Once the application is finished, all resources are released back to the cluster and combined to form the final result of the initiated application (15).

2.3.3 *Cluster computing with Apache Spark*

Apache Spark is a framework well suited for distributed computing and especially for data analytics. One of the strongest characteristics and factors that makes Apache Spark so superior is since it provides optimised memory computation resulting in increased process speed (17). Apache Spark also replaces and surpasses the famous MapReduce, also a part of Hadoop, which enables scalability across nodes in a Hadoop cluster (18). The superior aspect of Apache Spark is that it includes the Resilent Distributed Datasets (RDD) API, which is built with a similar concept as MapReduce, but RDD has been shown to be better suited for most batch jobs (19). In short, data sharing between jobs in MapReduce is slow since it stores the results in HDFS and then retrieves it, meaning that resources are needed continuously throughout the job. The RDD API resolves this by storing the state of the memory as objects across the jobs in the cache, which in turn makes data sharing between nodes possible and no need of physical storage in HDFS (20). Another strong characteristic of Apache Spark is that users can write applications in Python, Java, Scala and R. This means that developers can pick the programming language they are most accustomed to, resulting in an ease of use when writing applications. Since Apache Spark also supports data streaming, SQL-querying and machine learning libraries, it covers most aspects that are interesting for a high number of different business areas (17).

The core of the Spark execution model is the communication between the driver and the executors in the cluster. A Spark application is mapped to one driver, referred to as the Master Node, that is connected to YARN which is available throughout the entire process and handles task scheduling among the executors. Each executor handles the assigned tasks and as previously mentioned stores intermediate results in the cache. Once all tasks are finished, the AM exits and the executors are released (21).

2.3.4 *Machine learning in interplay with big data*

Machine learning is a huge field, covering different areas where the aim is to reveal hidden patterns, predict or classify data. Even though machine learning has been giving a lot of attention the last couple of years in terms sensations such as IBM:s Watson and recommendations on Netflix, it has been around since the field of artificial intelligence was funded by Arthur Samuel and his work with applying machine learning to board games in 1967 (22). It is also a field that can be applied in many different areas, such as optimization problems, clustering, regression analysis, motor control and prediction. Even though the different fields differ in terms of what one wishes to accomplish, they all share the characteristic that the machine learning system learns the pattern of the data by training a model that is used to predict the outcome (23).

The characteristics of the data has an impact on the machine learning prediction. If we have both input and output, it is called supervised learning and the applied algorithm maps input and output to learn the outcome. In the opposite scenario, we only have input data and no output, it is called unsupervised learning (24). The applied algorithm then models the underlying structure of the data or the distribution of data and learns a distinct behaviour in the data points. In this project, the data lacks identifiers or class labels and thus algorithms that fall in the category of unsupervised learning are of interest.

Most machine learning algorithms become stronger and more accurate with increasing amounts of data, making the concept of big data a good sidekick. The combination of big data and machine learning is relevant for many areas that utilize large quantities of complicated data, since the aid of machine learning can help to reveal patterns and give deeper insights in the structure of the data. Spark offers a machine learning package that contains some of the more common and popular machine learning algorithms called *MLlib* (25), presented in Python, Java and Scala.

3 OBJECTIVES

The main objective of this project has been to investigate how approachable the big data technologies are for a retail company as ICA and what new types of analysis that can be done together with such technologies. The project seeks to understand the concept of big data and aims to give a thorough insight in what is possible and problems that might arise. This can be done as a proof of concept approach where the following statement summarises the main objective of this thesis:

1. A big data tech-stack enable new areas to be explored and new analyses to be carried out at ICA

To successfully investigate this, certain aspects needs to be covered. These aspects are connected to how ICA as a company should view a migration to the big data technologies and important things to consider. These are:

- How are the different big data technologies integrated for fullest potential?
- Design of applications. How is a use case designed and implemented?
- What is available and suitable for analysis? Appropriate algorithms and packages.
- What are suitable programming languages?

4 METHODS AND IMPLEMENTATION

To investigate the role of big data technologies and the impact on a large company as ICA, a combined approach with two different aims was conducted based on the bullet-points in section 5. The first part involved an investigation to see how ICA as a company successfully can approach the big data technologies. This was done by practically exploring the big data technologies in order to get an insight in code development and how easy it is to configure and set up a working platform. The other part involved an investigation of relevant algorithms and use cases were identified that either can bring new insights for ICA or replace/enhance current parts of existing systems.

4.1 Preparing the migration

When deciding on whether a migration to a big data technology should be done, it's important to have an idea of what the purpose is. What does one hope to achieve with a migration, and by this question one will be able to identify which technologies that are suitable. For this project, the goal was to set up a big data system that enables advanced analytics with fast response and high-level algorithms. Therefore, the following requirements were defined:

1. Easy and accessible storage solution
2. Fast analysis
3. Possibility of advanced analytics
4. Convenient repository for storing the results

ICA had in advance chosen a platform built on Hadoop for the big data platform. Therefore, the available technologies included in Hadoop were evaluated.

- For storage of data in tables, Apache Hive is the most promising technology. Apache Hive comes with a set of specific queries that later was seen to be valuable for data processing in later stages. It is also suitable for further usage due to its characteristics of storage in both SQL and NoSQL.
- Apache Spark 2 is the latest version of the parallelization framework and was therefore chosen. Apache Spark enables repartitioning of large workloads and reducing the overall computing time, making it a powerful tool when dealing with massive data.
- HDFS was deemed to be the best solution for storing results in CSV or TXT files. The files in HDFS can later be used for further analysis, since the filesystem itself is designed to be split up over machines for fast analysis. Apache Hive can also be used for storing the results in a table-like fashion.

4.2 Setting up the system architecture

As mentioned in section 5.1, ICA had in advance of this project decided to use a commercial platform built on Hadoop for the big data exploration phase. When the techniques provided by this platform had been evaluated and chosen, the next step was to investigate how the following aspects can be answered:

- How the different techniques integrate
- How a connection between the techniques are established

Once these questions were covered, a schematic illustration was designed to give a summarized understanding of the identified dependencies and integrations. This illustration can be seen in Figure 1, which illustrates the system architecture of a big data system built on Hadoop for advanced analytics and showcases the dataflow for this project.

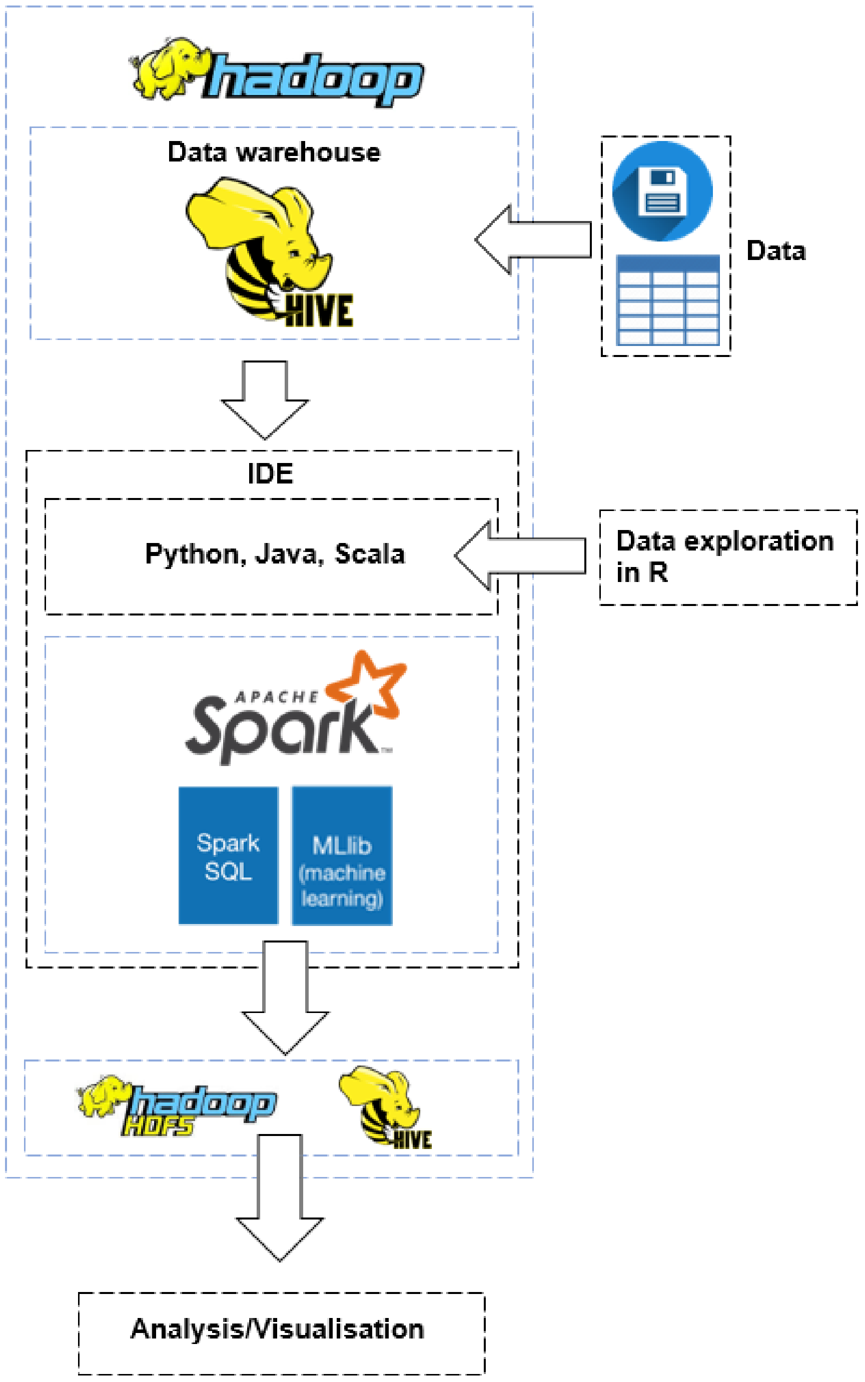

Figure 1: The figure describes the architecture and data flow from fetching the data to analysis used in this project. The data is extracted from the current RDBMS and uploaded to the data warehouse managed by Apache Hive. The data is then fetched with SparkSQL to be processed using machine learning algorithms in Apache Spark's MLlib package launched in Python, Java or Scala. Data exploration is first done in R and then migrated to one of the other languages. The result is stored in HDFS or Apache Hive and can then be fetched for further analysis or visualisation.
Areas marked with a blue border are subprojects included in Apache and areas marked with a black border are stand-alone software integrated in the Hadoop architecture.

4.3 Cluster setup

This section describes the Hadoop cluster setup that was used for this project.

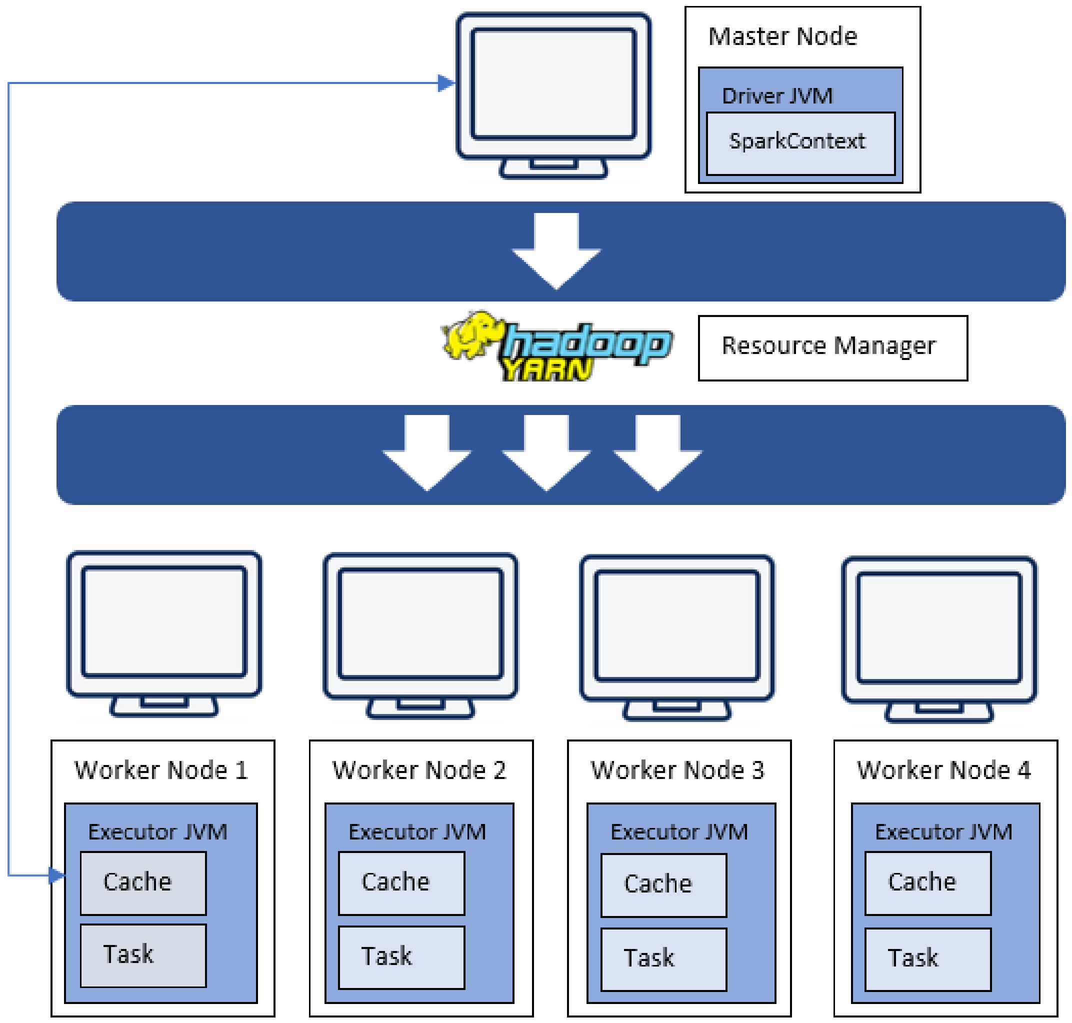

Figure 2: The figure illustrates the Hadoop cluster setup that was used in this project. In total, the cluster consisted of five nodes with one master node and four worker nodes. Each node had a CPU of 8 cores, memory (RAM) of 56 GB, disc memory space of 4 TB and CentOS as OS.

Figure 2 illustrates the Hadoop cluster used during this project, where Spark on YARN was used. The Master Node in the cluster handles the user-submitted applications by initiating a *SparkContext* object that is distributed across the Worker Nodes. The application request is sent to YARN that checks for data locality on the four Worker Nodes and schedules tasks accordingly. The executor JVMs on each Worker Node handles the assigned tasks and saves the result in cache which is sent back to the driver JVM that combines them into one final result.

4.4 Data collection

The available data at ICA was manually uploaded to the data warehouse in Apache Hive. It consisted of various tables containing different kinds of customer data such as article information, customer attributes and transaction data. It should be noted that all data was masked, meaning that there could be no traceback to specific individuals.

Later during the project, a new dataset was collected that was a combination of a set of tables from the current RDBMS. This dataset contained 6 million transactions for a total of 10.217 masked loyalty-card customers.

4.5 Setting up suitable IDE:s for code development

To successfully carry out this project, suitable programming languages needed to be identified. With respect to architectural design of the tech-stack in Hadoop, some limitations exist. The programming language needs to be fast and easy to deploy, and preferably compatible with extensions of the various parts in the Apache software hierarchy. By visiting Apache's web page, it was clear that packages and extensions in Scala, Java and Python were available. During an early meeting with the company that provided the platform used in this project, a discussion regarding these languages was carried out and they suggested Scala to be the fastest and most scalable one, followed by Java and Python. Unfortunately, neither Scala or Java offers good tools for visualisation and Python is quite limited. R is a powerful tool for data visualisation since it does not only offer tools for visualisation, but also handling and processing of data. R also includes packages for machine learning algorithms, but was during this project hard to integrate with the Hadoop architecture seen in Figure 1.

In conclusion, Scala, Java and Python were chosen for programming languages. To enable code development, an investigation of suitable IDE:s needed to be conducted. For development in Python, Jupyter Notebook was deemed to be the most suitable one. Jupyter Notebook is a web based interactive JSON document that contains cells for coding, mathematical expression, text editor and creation of plots. It's provided in the package manager Anaconda developed by Continuum Analytics. A strong characteristic of Jupyter Notebook is that the architecture is designed to enable parallel and distributed computing.

For development in Scala and Java, Eclipse Neon was chosen. Eclipse uses a large set of plug-ins to provide functionality for programming. Coding in Eclipse is built up by creating projects and use Maven for managing dependencies. Maven uses a specific file where dependencies and build instructions are specified. A script is then packaged and converted to a compressed Jar file that is used to run the script on the cluster through WinSCP or PuTTy (if a Windows machine is used, as in this project).

4.5.1 *Enabling Spark for high speed computations*

To be able to handle the large amounts of transactional data at ICA, Apache Spark was used. Because of this, a script to be run in Spark needs to be initiated by a *SparkContext* object. The SparkContext works as the entry point for the Spark application and represent the connection to the cluster. Through this connection, RDDs can be created and data can be retrieved from other projects such as Apache Hive and HDFS. The SparkContext is created by building a *SparkConf* object, holding parameters that define the application. These parameters include the URL to the master node, the name of the app to be created, the location of the Spark installation on the nodes and environment variables to be initiated.

Setting up a connection in Scala:

```
val conf = new SparkConf().setAppName(appName).setMaster(master)

new SparkContext(conf)
```

Due to security issues, the name of the master node shouldn't be hardcoded in the script. This can be worked around by running the script in bash-mode using the *spark2-submit* command.

4.5.2 *Handling and creating objects in a Spark application*

Data can be fetched from databases and data warehouses by *SparkSQL*. SparkSQL is a module for structured data processing and can be used to set up a connection to the data warehouse Apache Hive. A *SparkSession* is instantiated with Hive support, enabling connectivity with Hive.

Creating a connection to Hive in Scala and running example:

```
import org.apache.spark.sql.Row
import org.apache.spark.sql.SparkSession

val spark_sess = SparkSession
    .builder()
    .appname("Application Name")
    .config("spark.sql.warehouse.dir", wareHouseLocation)
    .enableHiveSupprt()
    .getOrCreate()

import spark.implicits._
import spark.sql

val example_RDD = spark_sess.sql("SELECT * FROM TABLE.DATABASE")
```

The Scala object *example_RDD* is a RDD object, which can be seen as a data frame and can be operated on in parallel. Once a connection has been established to Spark and Hive, data can be fetched as seen in the example and used for its cause.

4.6 Development and implementation of use cases

Once the environment was set up, relevant use cases were identified and implemented. These were:

1. **Market basket analysis**
   Based on customer transactions, what item can be recommended to a larger set of items?

2. **Customer segmentation using clustering algorithms**
   Can customers be clustered based on their transaction history to give more insight of their personas?

4.7 Market basket analysis

Association rules and affinity analysis is a concept that is both interesting and relevant for most business areas, for example within sales, commercial or content on a media site. The aim is to find associations and connections between specific objects and by this infer correlations.

The most famous and most interesting example for ICA is the Market Basket Analysis. The goal of Market Basket Analysis is to investigate correlations between sets of articles and how they co-exist on transaction level or in-store physical display. There are a lot to gain from such an analysis, to name a few there are:

1. Recommendation engine: customers buying these items also tends to buy this/these items
2. Targeted marketing: based on these transactions, this is a suitable offer
3. Placement of products: both in-store and on website

4.7.1 *Theory of Market Basket Analysis*

As mentioned, the goal of market basket analysis is to infer associations between different set of items and how they correlate. For a retailer as ICA, we define a terminology that shows how each part of the analysis will be carried out. These definitions are based on the articles (26) and (27).

The **items** are the products in the store that we infer association between. We can define a set of items as:

$$I = \{i_1, i_2, \dots, i_m\}$$

Items are contained in a **transaction**, where they co-occur with a set of items. Of course, there might be transactions with only one item, but they will fail to be part in the algorithm due to parameter settings. With the definition of an item set, we can define a transaction database as a set of transactions:

$$TDB = \{T_1, T_2, \dots, T_n\}$$

where T_i $(i \in [1, .. n])$ is defined as a transaction containing items defined in I.

For an item or set of items, the **support** defines the fraction of transactions that contain that item or set of items contained in TDB. A high support is desirable, but not in all situations. This since it depends on what question that one wants an answer to. For example, the items with highest support will probably be items that a lot of customers buy, which is everyday items such as milk, onions and even plastic bags. It is therefore important to have some sort of threshold or business rule that apply a filter on what type of correlations we look for.

With an appropriate algorithm, **rules** to describe correlation of products will be inferred. A rule can be defined as:

$$\{i_1, i_2, \dots\} => \{i_m\}$$

A rule can be read as for a set of items on the LHS (antecedent), the item on the RHS (consequent) will be a suitable combinatory to that set. For a given rule, its **confidence** can then be calculated. The confidence of a defined rule is a measurement of the conditional probability that a transaction selected at random from the TDB will include all items in the consequent given that they are included in the antecedent.

The confidence can be defined as:

$$confidence(X => Y) = \frac{support(X \cup Y)}{support(X)}$$

Where the confidence of the rule $X => Y$ for a set of transaction *T* is equal to the proportion of transactions *T* that contains *X* and *Y*.

Let's apply the defined terminology on an example:

Table 1: Example table with 5 transactions with 3 unique products

Transaction ID	**Coffee**	**Milk**	**Sugar**
1	1	1	1
2	1	1	0
3	1	1	1
4	1	1	0
5	1	1	1

$supp(\{coffee, milk, sugar\}) = \frac{3}{5} = 0.6$

Since the item set {coffee, milk, sugar} occurs in 60% of all transactions.

$supp(\{coffee, milk\}) = \frac{5}{5} = 1$

Since the item set {coffee, milk} occurs in 100% of all transactions.

$conf(\{coffee, sugar\} => \{milk\}) = \frac{3}{3} = 1.0$

A confidence equal to 1.0 indicates that for each transaction where coffee and sugar is bought, milk is bought as well.

4.7.2 *Implementation of Market Basket Analysis*

The FPG algorithm in Spark's MLlib was identified to be suitable for the implementation of the Market Basket Analysis. The Apriori algorithm has previously been used in examples based on the Market Basket Analysis, but faces time complexity issues for processing larger data sets. The FPG algorithm solves for this by the design of the algorithm itself and can be divided into smaller subtasks by parallelisation, making it highly appropriate to be run in the Hadoop cluster through Apache Spark (27).

The FPG algorithm is built up by two steps, as outlined in the paper by Han *et. al* (27). In summary, the first step includes the construction of the FP-tree which is a compressed representation of the input and the core of the algorithm. Each transaction in the input is ordered based on a given support and then mapped to the FP tree, where the nodes in the tree represent the items and each item have a counter. Since transactions can include the same item, there is possibilities for overlap, but this allows for compression and thus the FP-tree won't grow to immense sizes throughout the process (27). This is one of the characteristics that makes the FPG algorithm highly scalable.

Once the FP-tree has been constructed, the frequent item sets can be extracted. This is done by the function *FP-growth* outlined in the paper (27), which takes the FP-tree as input and recursively computes the frequent item sets with a bottom-up strategy by looking at the minimum pattern base for each distinct item set. Depending on the path for each item through this bottom-up strategy, a large compression of the FP-tree occurs which also contributes to the high scalability of the FPG algorithm. These pattern bases are built up throughout the algorithm, until the minimum set is acquired. Once the tree is defined as the empty set, the algorithm quits.

The FPG algorithm was combined with the algorithm for generating association rules with a single item as the consequent. The algorithm for generating association rules was provided in the Spark documentation. A simplified code snippet of the implemented algorithms can be seen below.

```
import org.apache.spark.mllib.fpm.FPGrowth
import org.apache.spark.rdd.RDD

val all_transactions = spark.sql("SELECT * FROM
TRANSACTIONS.DATABASE_NAME")

val fpg_algorithm = new FPGrowth().setMinSupport(SUPP_VAL)
.setNumPartitions(NPARTITION_VAL)

val fpg_model = fpg_algorithm.run(all_transactions)

val min_confidence = CONF_VAL

fpg_model.generateAssociationRules(min_confidence).collect().foreach
{ rules =>
  println(
    rules.antecedent.mkString("[", ",", "]")
      + " => " + rules.consequent .mkString("[", ",", "]")
      + ", confidence: " + rules.min_confidence)
}
```

As seen in the code snippet above, the support (*setMinSupport)* is first specified which will affect the number of item sets we are able to withdraw with respect to the number of transactions. Depending on the amount of data, the support needs to be tuned so it doesn't exclude to little or include too many item sets. With a lower value for support, a larger FPG model will be generated. The confidence (*minConfidence*) is then set which affects the accuracy of the rules generated. For example, a confidence of 0.8 results in rules that are true in 80% of the cases.

4.8 Customer segmentation using clustering algorithms

Customer segmentation analysis is used to examine the characteristics of a company's entire customer base and group customers with similar characteristics into segments, or clusters. Objects with similar characteristics are grouped together into clusters, where objects belonging to different clusters diverge in some way. A deeper understanding of the customer base guides the company's efforts in for example marketing, pricing and product development. In terms of cluster analysis of customers, it is also possible to find groups of customers that share needs or interests or even behaviour. Based on the findings, certain groups might be more profitable for targeted offers or exposed to certain products.

The customer segmentation at ICA today is mainly done by defining business rules and then apply these on subsets of data. Another part of today's segmentation is also based on surveys where customers themselves has defined their profiles, resulting in a risk for biased data since this segmentation don't really reflect the true behaviour of the customer and doesn't change over time. Updating the segmentation also requires a new survey, which in turn is a time-consuming task. Therefore, a segmentation based on cluster analysis of customer data might give new insights and enables the possibility to follow a customer's persona on a new level. An investigation of suitable machine learning algorithms was carried out in order to see if it's possible to perform a segmentation using suitable clustering algorithms without any prejudices of what one should find.

For the success of a cluster analysis, there are several pre-processing steps that needs to be carried out.

4.8.1 *Data collection and filtering*

As a first step, the data needed for clustering needs to be identified and collected. The data available was transaction data combined with a smaller set of demographic data, such as age and sex. Due to this, the goal of the clustering was defined to investigate customer behaviour based on their transactions together with the available demographic data to see if there are some common patterns among different groups of customers. Each real customer ID was replaced with a randomised ID to apply a level of anonymity. ICA has defined an excellent *Product Description Hierarchy* (PDH) with 6 levels and a set of labels that defines each item.

The available data consisted of 6 million transactions, covering 10.217 loyalty-card customers spanning 6 months. Each row in the data set describes one purchased item being part of a transaction. The data set was provided as a combination of several tables from the existing database and stored as a table in the data warehouse Apache Hive in the cluster. An example illustrating the data is shown below.

Table 2: Example table showing transaction data logic. The table contains two unique customers, their age and transaction history. Each row in the table refers to one item contained in a certain transaction.

Customer ID	**Age**	**Transaction ID**	**…**	**Item description**	**N.o. units**	**Marked Organic**	**…**
2511747360	44	1	…	"Milk"	2	1	…
2511747360	44	1	…	"Bread"	1	0	…
2511747360	44	2	…	"Pork"	1	0	…
2311758925	75	1	…	"Shrimps"	2	0	…
2311758925	75	1	…	"Yoghurt"	1	1	…

As seen in Table 2, each row contains some demographic information about the customer as well as what the customer bought for a certain transaction. Each item is also identified with a set of labels. The labels in this data set was organic, healthy, ICA's own products (marked as EMV) and if it was on promotion. Each of these labels are identified as a Boolean expression, 1 if true and 0 if false as seen in example column "Marked Organic" in Table 2. These labels were identified as valuable parameters in the process to describe customers based on their transactions since one should be able to see if a person is for example more oriented towards organic products or more price concerned since there is a high number of items bought on promotion. These will be referred to as "markups".

Each item also has the number of units purchased associated to it, making it possible to get an idea of how many items were organic, healthy and on promotion. This was achieved by simply multiplying the number of units by the Boolean expressions for the different mark-ups. Since different customers will buy different amounts of units depending on factors, such as loyalty or type of purchase (small, medium, large), it will be difficult to compare customers to each other. Therefore, the number of mark-ups needed to be normalized as fractions of the total purchase resulting in a value between 0 and 1, where a low value indicates a small fraction and vice versa for a high value.

Before the data was transformed and relevant variables was identified, a filtering step was vital to maintain high data quality. The total number of different articles at ICA is very high, and only certain categories and areas will be interesting for the cluster analysis. Therefore, some business areas are less interesting and was removed from the data set by consulting the analytics team at ICA.

4.8.2 *Variable extraction and transformation*

Once irrelevant data had been removed, the data transformation step could be started. For implementation purposes, the data transformation procedure was first designed in R on a subset of the data for ease of use and experimental data mining. Once the desired result was achieved, the code was migrated to Scala to be able to run in the big data environment. Therefore, different schemas will be presented that shows available and suitable packages and libraries for data processing and aggregations in R and Scala for Spark. Packages and/or libraries are marked in **bold**.

Schema 1 Transaction Filtering and Transformation in R

1. Read CSV data
2. Extract relevant columns
3. Filter irrelevant categories
4. Compute organic, healthy, private, promotion by multiplication
5. Aggregate data with **dplyr** or **plyr**
6. Compute fraction of purchase of organic, healthy, private, promotion by division
7. Compute number of baskets purchased and average basket size
8. Reshape data to one row per customer with **reshape2**

Schema 2 Transaction Filtering and Transformation in Scala

1. Read data with **Spark SQL**
2. Filter irrelevant categories with **Spark SQL**
3. Select categories for computation
4. Compute organic, healthy, private, promotion by multiplication
5. Aggregate data with **RelationalGroupedDataset**
6. Compute fraction of purchase of organic, healthy, private, promotion by division
7. Compute number of baskets purchased and average basket size
8. Reshape data to one row per customer

Once relevant variables had been mined and transformed, a customer was defined by 9 variables:

- **Spend**, the total amount of money spent (SEK) during time-period.
- **Units**, the total number of units purchased. Distinguishes a small shopper from a large shopper.
- **ECO**, the fraction of a total purchase being marked as organic.
- **EMV**, the fraction of a total purchase being marked as ICA’s own products.
- **Healthy**, the fraction of a total purchase being marked as healthy.

- **Promotion**, the fraction of a total purchase being marked as on promotion. It should be mentioned that this does not include offers, but only items on promotion in a certain store or for all stores.
- **Age**, the age of the customer.
- **Number of Baskets**, the number of baskets the customer has bought during time-period. Distinguishes a small shopper from a large shopper or an infrequent from a frequent shopper.
- **Average Basket Size**, the average number of units per basket. Distinguishes a small shopper from a large shopper by other means than **Units**.

4.9 The clustering analysis step

Once the data has been read, filtered and transformed, the data can be passed to the cluster analysis step. The hypothesis of the clustering is, as mentioned in section 6.6:

> *Can customers be clustered based on their transaction history to give more insight of their personas?*

Before this hypothesis can be tested, certain aspects needed to be considered for the success of the cluster analysis.

4.9.1 *Distribution of customers*

When conducing a cluster analysis, an important initial step is to investigate the distribution of the data to be part of the analysis. Most data sets follow a normal distribution, also referred to as a Gaussian distribution. In a normal distribution, the probability to find an object far from the center of the distribution decreases as the distance increases. This means that most objects in the population are close to an average point, and most objects are similar to each other in terms of describing variables. The objects seen far away from this center are referred to as anomalies or outliers, and can in many cases be interesting objects to capture (28). However, for this project it was decided that the focus of the analysis would not be to address these outliers directly, but instead a discussion regarding how to deal with them are presented.

4.9.2 *Principal component analysis*

The goal of a principal component analysis (PCA) is to reduce the number of initial dimensions without losing any information of the data. PCA extracts principal components, representing a transformation of the initial variables that describes the variance rotation of the prior vector space. This means that the first principal component will have the largest variance, the second component the second highest and so on. By this one hopes to find a smaller set of variables (principal components) than the initial number of dimensions that still accounts for most of the variance (29).

4.9.3 *Hierarchical clustering*

Hierarchical clustering (and other cluster algorithms) is based on computing distances between points in the feature space where the objects live. The method to calculate this distance can be seen as a measurement of similarity, or even dissimilarity, and the choice of distance measure is an important step (30). However, to choose the correct distance measure is not that trivial and there exist no well-defined guidelines to what is appropriate, especially for data of high dimension due to the curse of dimensionality (31). In the study described in Aggarwal *et. al.* (31), it was shown that the Manhattan distance metric and Euclidean distance metric can be applied to high dimensional data. In addition, PCA was performed on the data set used in this project where the Euclidian space by definition is the reference space (32). Therefore, the Euclidian distance was selected as distance metric to compute dissimilarities between the objects.

The Euclidian distance *d* between two points *x* and *y* are given by equation 1:

$$d(x,y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2} \qquad (1)$$

where *n* is the number of dimensions of the data set, x_k and $y_k$the k^{th} variables of *x* and *y* (28).

Hierarchical clustering can be divided into divisive and agglomerative, and agglomerative was chosen for this project. Agglomerative hierarchical clustering creates a tree structure referred to as a dendrogram, which contains k number of block partitions where k ranges from 1 to n (number of observations to be clustered). The algorithm allows the user to decide on different levels of granularity, since the cut in the dendrogram can be changed simply by visual inspection. When using agglomerative hierarchical clustering, the algorithm starts by assigning each point to its own cluster and merges clusters based on an aggregation criterion to reduce the number of clusters until k = 1 (33). The aggregation method in this project was Ward’s method. The method was selected since it is based on the squared Euclidian distance between clusters, and as mentioned the Euclidian distance was selected to infer dissimilarities between objects. Different implementations of Ward’s method are based on different merging criterions, and the one used in this project is based on Huygen’s theorem, which refers to the decomposition of the total variance, or intertia, in between and within-group variance of the clusters. Since PCA, which as mentioned is based on multivariate variance, was used as a pre-processing stage it makes Ward’s method based on Huygen’s theorem highly suitable. The total inertia can be defined as:

$$\sum_{k=1}^{K}\sum_{1=1}^{Q}\sum_{i=1}^{I_q}(x_{iqk}-\bar{x}_k)^2=\sum_{k=1}^{K}\sum_{q=1}^{Q}I_q(\bar{x}_{qk}-\bar{x}_k)^2+\sum_{k=1}^{K}\sum_{q=1}^{Q}\sum_{i=1}^{I_q}(x_{iqk}-\bar{x}_{qk})^2 \quad (2)$$

Total inertia Between-inertia Within-inertia

where x_{iqk} is the value of the variable k for object i of cluster q, $\bar{x}_{\mathrm{qk}}$ the mean (center) of the variable k for cluster q, $\bar{x}_k$ the overall mean of variable k and I_q the number of individuals in cluster q.

Ward's method then seeks to fuse pairs of clusters that results in the smallest value of within-inertia. With reference to equation 2, Ward's method minimizes the change in between-cluster inertia, in other words maximizing it and by this we are minimizing the within-cluster variance (inertia). Ward's method can then be defined as function 3:

$$\delta(c_1, c_2)=\frac{|c_1||c_1|}{|c_1|+|c_2|}||c_1-c_2||^2 \quad (3)$$

where c_1 and c_2 are vectors of original data or mean and |.| is both cluster cardinality and mass and ||.|| the Euclidian norm and δ is the function sought to be minimised (32).

The within-inertia is in turn a measurement of how homogenous a cluster is, in our case meaning customers with similar behaviour. The resulting dendrogram will then illustrate an hierarchy indexed by the gain of within-inertia, where we seek a number of clusters resulting in a low value for this measurement (32) (34).

5 RESULTS

The aim of this project was to examine and understand the different systems of the big data technologies and how it can be combined with machine learning for fast analysis and high-throughput data management. The aim was also to get insight in what a big company as ICA needs to take into consideration to successfully approach the different technologies of big data.

5.1 Investigating the IDE:s for code development

A part of the thesis was to explore what programming languages that can be deployed on the Hadoop platform. There are 4 APIs available for Spark integration, namely Python, Scala, Java and R. Python, Scala and Java was found to be best suited for developing Spark applications, since packages are available in the Spark documentation. The R API was evaluated to be not as mature as the other APIs to wrap around Spark.

5.1.1 *Eclipse Neon*

Eclipse Neon with Maven was set up for Scala and Java. Maven makes it possible to install dependencies, such as Apache packages, containing all the extensions for designing Spark applications such as connectivity with Hive and HDFS and access to the machine learning library *MLlib*. Eclipse Neon offers a user-friendly interface with intuitive toolbars and a smart design.

5.1.2 *Jupyter Notebook*

Jupyter Notebook was set up for Python on each node in the Hadoop cluster. Once the IDE was set up, it was soon realised that some packages in the Spark documentation wasn't included for Python. For example, the FPG algorithm was available but not the package for association rules that works as the last phase for creating the recommendations. As an attempt, the apriori algorithm (a similar algorithm to FPG but slower) that also includes the construction of association rules was implemented. It was however quickly seen that it was troublesome to convert the Spark data frame or RDD that was created from an SQL query into the correct format for the algorithm. Of course, this could have been worked around but due to lack of time this wasn't investigated any further. It was also realised that the Apriori algorithm is much slower than the FPG algorithm, which eventually will be a big problem in terms of growing data quantities to process (35).

The advantages with Python through Jupyter Notebook is however the ability to launch the IDE in the web browser from a Worker Node in the cluster. It enables fast development by trial and error and debugging. Python is also in some cases more suitable for developing your own implementations and has a lot of packages that can help in the process, such as the machine learning library *scikit-learn*, and offers tools for visualisation. The Python API is also easier to grasp and understand from an early stage.

5.1.3 *Rstudio*

The purpose of setting up Rstudio was to enable a good API for data aggregations and transformation together with powerful visualisation options and statistical analysis. To get the data stored on the cluster into Rstudio locally, the R package Rimpala was used. The connection was set up successfully, but a bottleneck was soon identified. The limitation was that it's only possible to query a certain number of rows, resulting in data being chopped of and thus losing parts of the data.

Another solution to integrate R with Spark is to install and run R and Rstudio within the cluster, from one of the nodes. The heavy computations are then made in the backend of Spark and the results are collected in Rstudio for visualisation and interpretation. This is enabled by the R package "sparklyr" (36).

5.2 Market Basket Analysis

The Market Basket Analysis was performed by extracting data from two tables, one holding a total of 25.874.658 transactions and 138.080 customers and one table with the article descriptions. The data was collected by creating views with the function *registerTempTable* provided in SparkSQL to store intermediate results, making filtering possible. An example is shown below.

```
val details = spark.sql("SELECT art.articlename, pur.transactionid
FROM
DATABASE.ARTICLE_TABLE AS art,
DATABASE.PURCHASETRANSACTION_TABLE as pur
WHERE
art.ean=pur.ean")
details.registerTempTable("transactions")
```

There cannot be multiples of the same article in a transaction, since the FPG algorithm computes the frequency for each unique item in a set of items.

```
val unique_transactions = spark_sess.sql("SELECT DISTINCT
articlename,transactionid
FROM transactions
ORDER BY transactionid")
unique_transactions.registerTempTable("unique_transaction")
```

As a last step, each set of articles are concatenated into a list and saved as a CSV file in HDFS. This is achieved by calling a Hive specific query and then write the result to HDFS.

```
val trans = spark_sess.sql("SELECT COLLECT_SET(CONCAT(article AS
STRING)) ORDER BY transaction_id")

val filepath = new Path("/PATH-TO-HDFS/FPG_data.csv")

trans.write.csv("/PATH-TO-HDFS/FPG_data.csv")
```

The file is then read and converted to an RDD object and passed as input to the FPG algorithm.

```
val data = sc.textFile("/PATH-TO-HDFS/FPG_data.csv")

val transactions: RDD[Array[String]] = data.map(s =>
s.trim.split(','))

val fpg_algorithm = new FPGrowth().setMinSupport(0.0001)
     .setNumPartitions(1000)
val fpg_model = fpg_algorithm.run(transactions)
```

The value for the minimum support was set to 0.0001, meaning that the model will include item sets that occurs more than 0.01%, resulting in a very large model. When the FPG model is created with its input, the association rules are computed. The association rules are saved to a list and stored throughout the iteration. The list is then written to a file and stored in HDFS.

```
val min_confidence = 0.5
var all_rules = new ArrayBuffer[String]()

model.generateAssociationRules(min_confidence).collect().foreach {
rules =>
    rules.antecedent.mkString("[", ",", "]")
      + " => " + rules.consequent .mkString("[", ",", "]")
      + "", confidence: " + rules.min_confidence

      var rule = rules.antecedent.mkString("[", ",", "]") + " => " +
         rules.consequent .mkString("[", ",", "]") + ", confidence:"
             + rules.min_confidence

      all_rules += rule
      all_rules += System.lineSeparator()

}

val config = new Configuration()
config.addResource(new Path("/etc/hadoop/conf.platformX.hdfs/core-
site.xml"))

val fs = FileSystem.get(config)
val filenamePath = new Path("/PATH-TO-HDFS/market_b_res.txt")
```

Each row in the resulting text file from the FPG and association rules mining contains the item set and the recommended item as well as the confidence for that rule. In the above code snippet, a confidence of 0.5 results in rules that are true in 50% of the cases, resulting in a larger set of rules. A part of the result is seen in Table 3.

Table 3: A small set of the result from the Market Basket Analysis. Note that the resulting rules from the test run worked solely as a proof of concept, meaning that no deeper or further analysis was done to find relevant rules or new insights for ICA.

Items		**Recommendation**	**Confidence**
[Baguette vitlök, Paprika Röd]	=>	[Banan Eko 1 kilogram]	0.8387
[ICA Eko Gurka, Hasselnötskärnor 200 gram, ICA Lök gul]	=>	[Sötmandel 200 gram]	0.5106
[Äpple Sverige, Apelsin 1 kilogram,KG BANAN kg, Banan Eko 1 kilogram]	=>	[Fast potatis lv 1 kilogram]	0.5402

5.3 Data mining and cluster analysis

This section describes the process of the cluster analysis. The first two sections will describe the data mining phase where the characteristics of the original data consisting of 10.217 loyalty-card customers were examined. The last section will illustrate the result of applying a cluster algorithm on the examined data.

5.3.1 *Data extraction and transformation*

The extracted data was transformed to create the 9 variables describing a customer, listed in section 6.8.2. The design of the script and how to extract these features was first implemented in R on a smaller data set as seen in Schema 1, section 5.8.2. The process was then implemented in Scala for the larger data set as seen in Schema 2, section 5.8.2 The Scala script fetches 6 million rows of transaction data from Apache Hive, transforms it and saves the result to Apache HDFS. This process took around 220 seconds. The data was then loaded in R for further analysis. An extract of the data is seen in Table 4 below.

Table 4: The 9 discerning variables for 10 out of the 10.217 customers available. The variables describe the purchase behaviour for each individual customer during a 6 months' period.

Customer ID	**Spend**	**Units**	**ECO**	**EMV**	**Health**	**Promo**	**Age**	**N.o. Baskets**	**Avg. Basket.Size**
2513401690	36042	1714	0.013	0.311	0.139	0.221	46	187	9.2
2528666255	6392	312	0.006	0.131	0.138	0.039	80	23	13.6
2539769570	10369	622	0.042	0.198	0.269	0.297	72	35	17.8
2545968210	16430	857	0.057	0.389	0.380	0.327	59	38	22.6
2549578930	18838	1012	0.090	0.279	0.218	0.426	64	40	25.3
2579603405	15083	694	0.043	0.244	0.264	0.200	66	42	16.5
2583209025	14512	552	0.009	0.234	0.150	0.339	46	88	6.3
2592060410	14008	516	0.050	0.357	0.128	0.159	79	37	13.9
2596379085	24102	1068	0.002	0.042	0.048	0.117	65	81	13.2
2744599945	16416	593	0.688	0.401	0.371	0.119	38	62	9.6

5.3.2 *Identifying patterns: variable correlations and distributions*

A correlation plot of the data set was created in R with the function *pairs* from the "graphics" package. The correlation plot provides insight in how the variables relate to each other and showcases the complexity of the data. The result is seen in Figure 3.

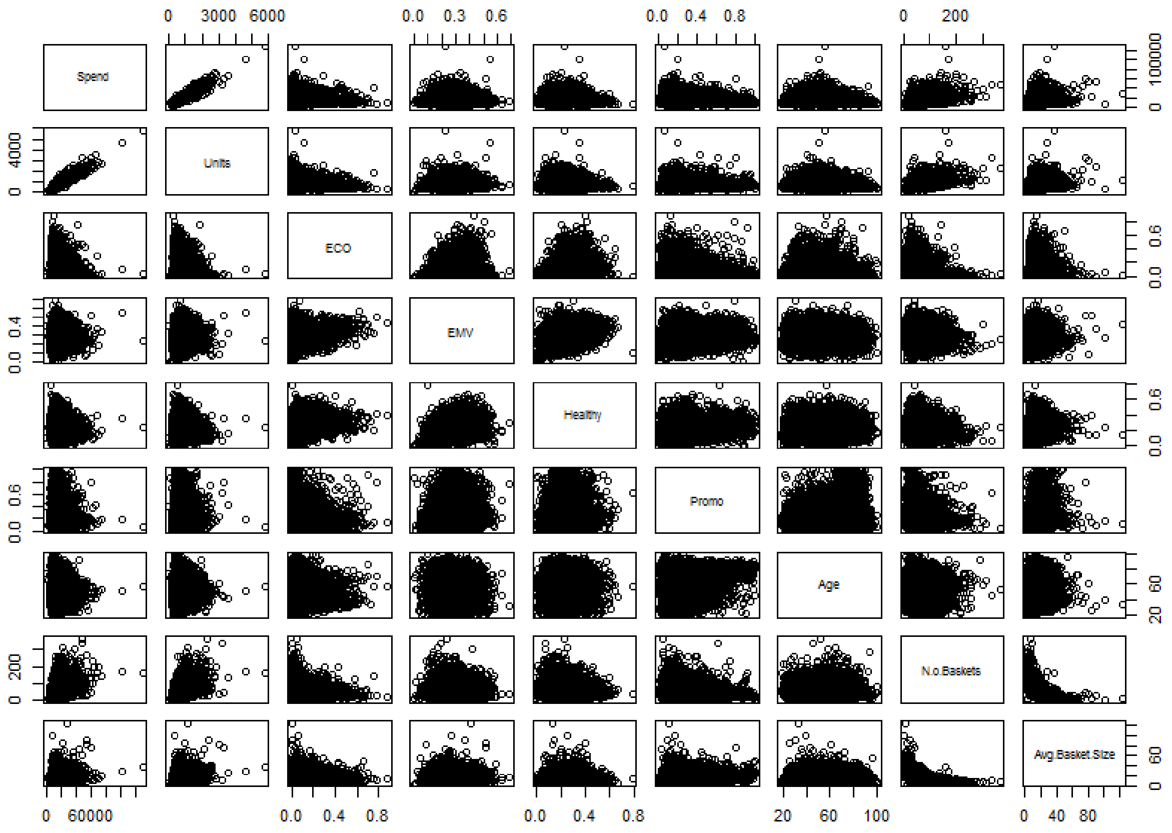

Figure 3: Correlation plot of each pairwise variable. The plots suggest that there exists a lot of correlations, anti-correlations and interdependencies between most variables.

Figure 3 indicates that the data is complex with a lot of correlations, anti-correlations and interdependencies. For example, the variables "Spend" and "Units" show a high correlation, which isn't so surprising since the more you buy, the more you also spend. We also see that there are some tendencies for anti-correlations between "Spend" and almost all other variables and the same goes for "Units" and all other variables. We also see that there appears to be complex interdependencies between the different "markups". The overall correlation of these variables can be addressed by computing the covariance matrix using the R function *cor* from the "stats" package. The covariance matrix can be seen in Table 4.

Table 4: Covariance matrix showing the covariance between the variables. A value equal to 1 indicate a perfect positive correlation, and a value equal to -1 indicate a perfect negative correlation.

	Spend	**Units**	**ECO**	**EMV**	**Health**	**Promo**	**Age**	**N.o. Baskets**	**Avg. Basket Size**
Spend	**1.000**	0.949	0.028	-0.032	-0.037	-0.058	-0.069	0.474	0.300
Units	-	**1.000**	-0.027	-0.003	-0.035	0.035	-0.036	0.528	0.291
Marked.ECO	-	-	**1.000**	0.143	0.327	-0.072	-0.104	-0.096	0.071
Marked.EMV	-	-	-	**1.000**	0.349	0.090	-0.074	-0.053	0.041
Marked.Healthy	-	-	-	-	**1.000**	0.094	0.107	-0.129	0.083
Marked.Promo	-	-	-	-	-	**1.000**	0.338	-0.095	0.143
Age	-	-	-	-	-	-	**1.000**	-0.027	0.005
N.o.Baskets	-	-	-	-	-	-	-	**1.000**	-0.486
Avg.Basket.Size	-	-	-	-	-	-	-	-	**1.000**

Table 4 summarizes the plots seen in Figure 4. We see for example that, as seen in the plot in Figure 4, “Units” and “Spend” has a high covariance of 0.949.

As a next step to investigate the pattern of the data is to look at the distribution of the data points for each variable. Figure 5 illustrates the distribution of customers with kernel density plots, generated with the R function *density* from the “stats” package.

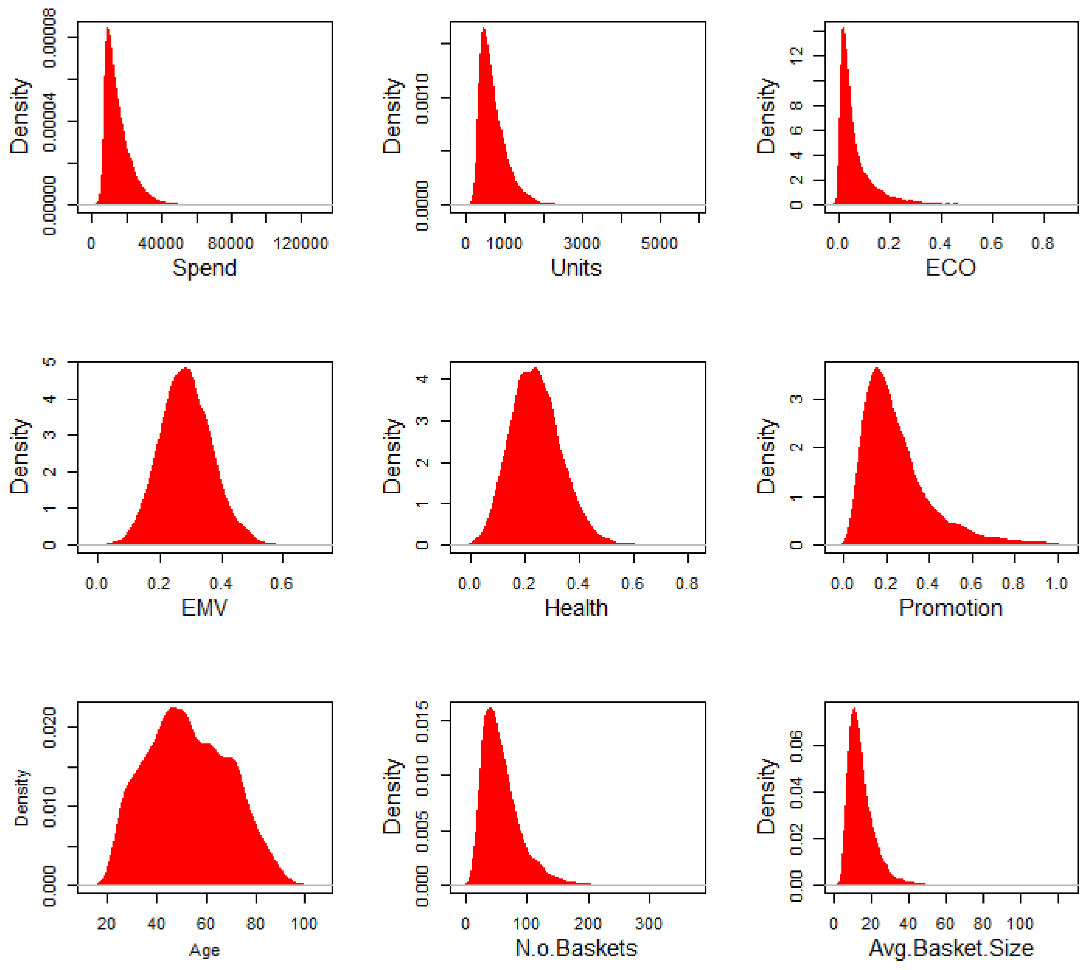

Figure 4: Kernel density plots for all 9 variables describing a customer. The y-axis in each plot shows the density and x-axis the value interval of each variable. Most plots illustrate a positive skew, meaning that extreme values are present in the data for that variable.

The plots in Figure 4 suggests that there exist extreme values for some variables, since the peak is seen far to the left indicating that some points exist higher up along the x-axis, also referred to as a positive skew. Some of the plots fails to show this since these extreme values make out a low fraction of the total distribution. This is for example clearly true for “Spend”, “Units” and “Avg.Basket.Size”, and therefore Figure 6 and Figure 6 shows a zoomed in version of these variables.

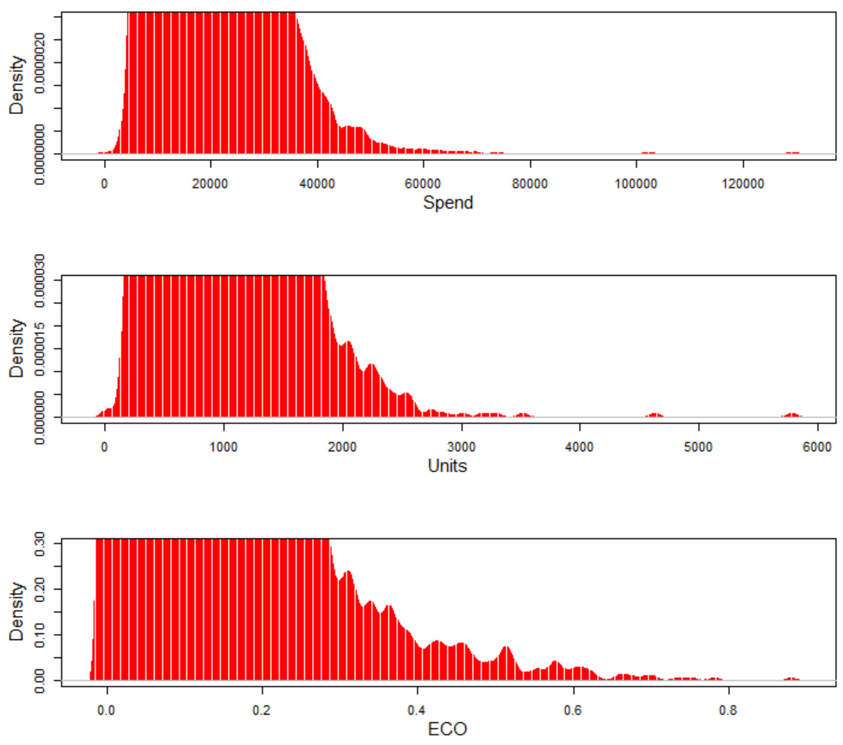

Figure 5: The three kernel density plots shows the extreme values for variables "Spend", "Units" and "ECO". The y-axis in each plot shows the density and x-axis the value interval of each variable.

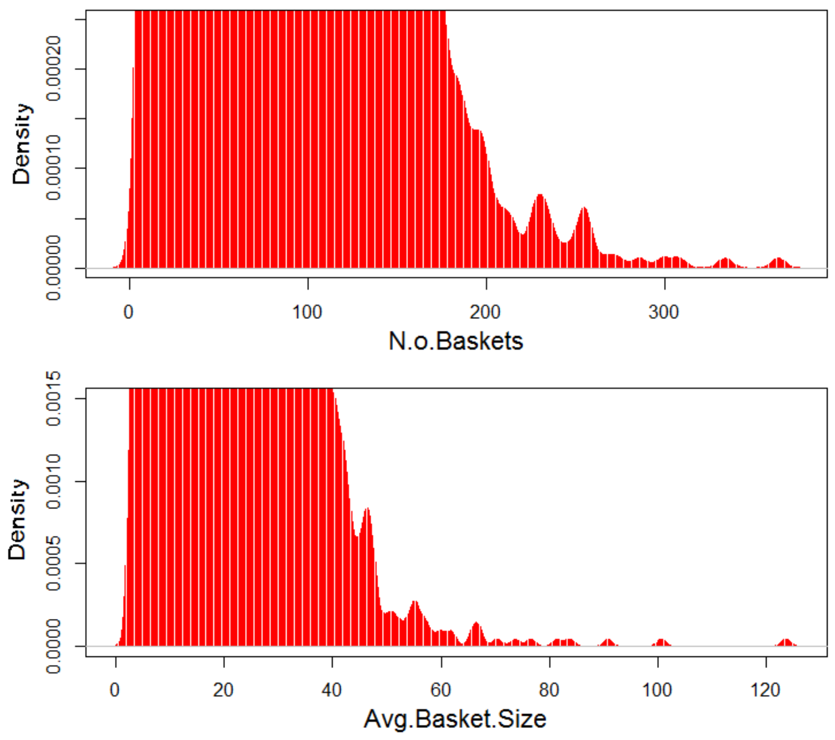

Figure 6: The two kernel density plots shows the extreme values for variables "N.o.Baskets" and "Avg.Basket.Size. The y-axis in each plot shows the density and x-axis the value interval of each variable.

Extreme values in this context can be defined as outliers, which in turn are defined as customers with a purchase behaviour that differs from the average customer. As seen in the kernel density plots on the previous pages, they can be defined as customers that spends more or buys more organic articles than the average customer. However, as mentioned in section 6.9.1, outliers defined in one variable is not the same as outliers defined in multiple variables. As mentioned earlier, the goal of the cluster analysis was not to address these outliers but to see how the proposed solution works in capturing customers based on the generated data. A discussion regarding treatment of multivariate outliers is presented instead.

5.3.3 *Hierarchical cluster analysis*

PCA was performed to investigate if the number of dimensions could be reduced and create new variables that reflect the data on a new level. The data was first normalized by calculating z-scores to give equal weights to all variables, which was achieved either by using the R function *scale* from the “base” package or use the inbuilt scaling property in the R function *PCA* from the “FactoMineR” package. A scree plot was generated to investigate the principal components. The result is seen in Figure 8.

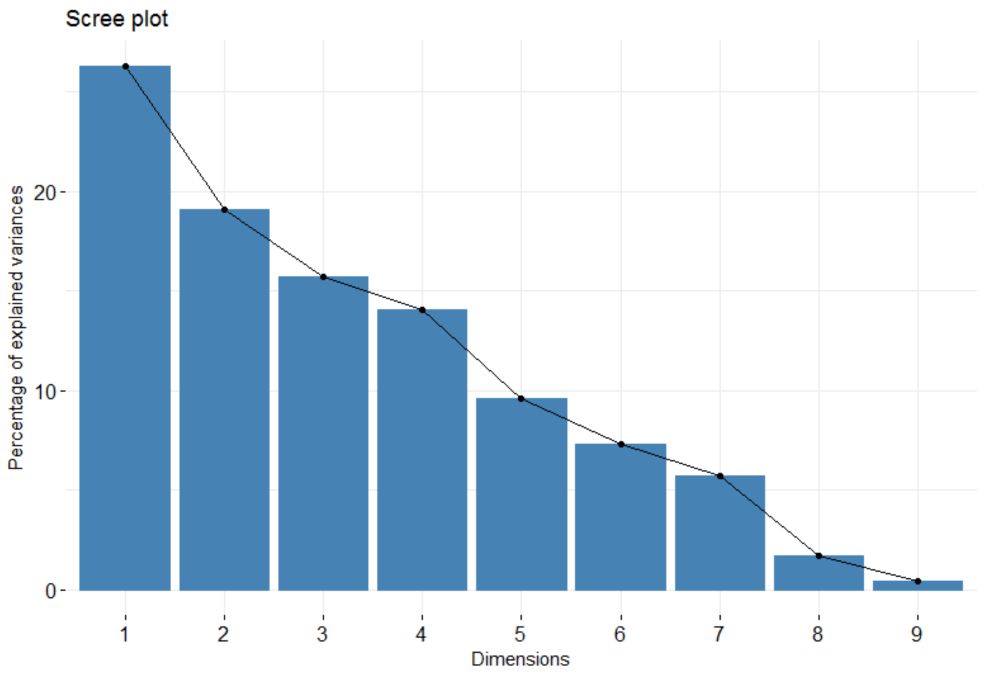

Figure 7: Visualization of the variance for each principal component. The principal components are seen on the x-axis and the amount of variance explained on the y-axis in percentage. The first component has the highest variance explained around 26%.

Figure 7 shows that the top principal components accounts for a rather small proportion of the variance explained, meaning that we will need to keep a number of components close to the initial number of dimensions. To get an accurate idea of how many components are needed, a statistics table was created using the R function *get_eig* from the “factoextra” package.

Table 5: Statistics for each principal component. The more dominant a component is, the higher its eigenvalue becomes. The cumulative sum of the percentage of variance explained shows that 6 components accounts for 92.09% of the total variance, meaning that 6 components are sufficient to keep for further analysis.

	Eigenvalue	**Percentage of variance**	**Cumulative percentage of variance**
PC 1	2.37	26.29	26.29
PC 2	1.72	19.10	45.39
PC 3	1.41	15.72	61.11
PC 4	1.27	14.09	75.21
PC 5	0.86	9.61	84.81
PC 6	0.66	7.28	92.09
PC 7	0.52	5.74	97.83
PC 8	0.16	1.73	99.55
PC 9	0.04	0.45	100.00

Table 5 shows that the first 6 components explain around 92% of the total variance, meaning that they are good candidates for the clustering. By this, the dimensionality is reduced from 9 to 6.

The principal components were passed as input to the R function *HCPC* from the "FactoMineR" package. The function performs an agglomerative hierarchical clustering on the result from a factor analysis such as PCA. Ward was used as agglomeration method and Euclidian as distance metric. The resulting dendrogram is seen in Figure 9.

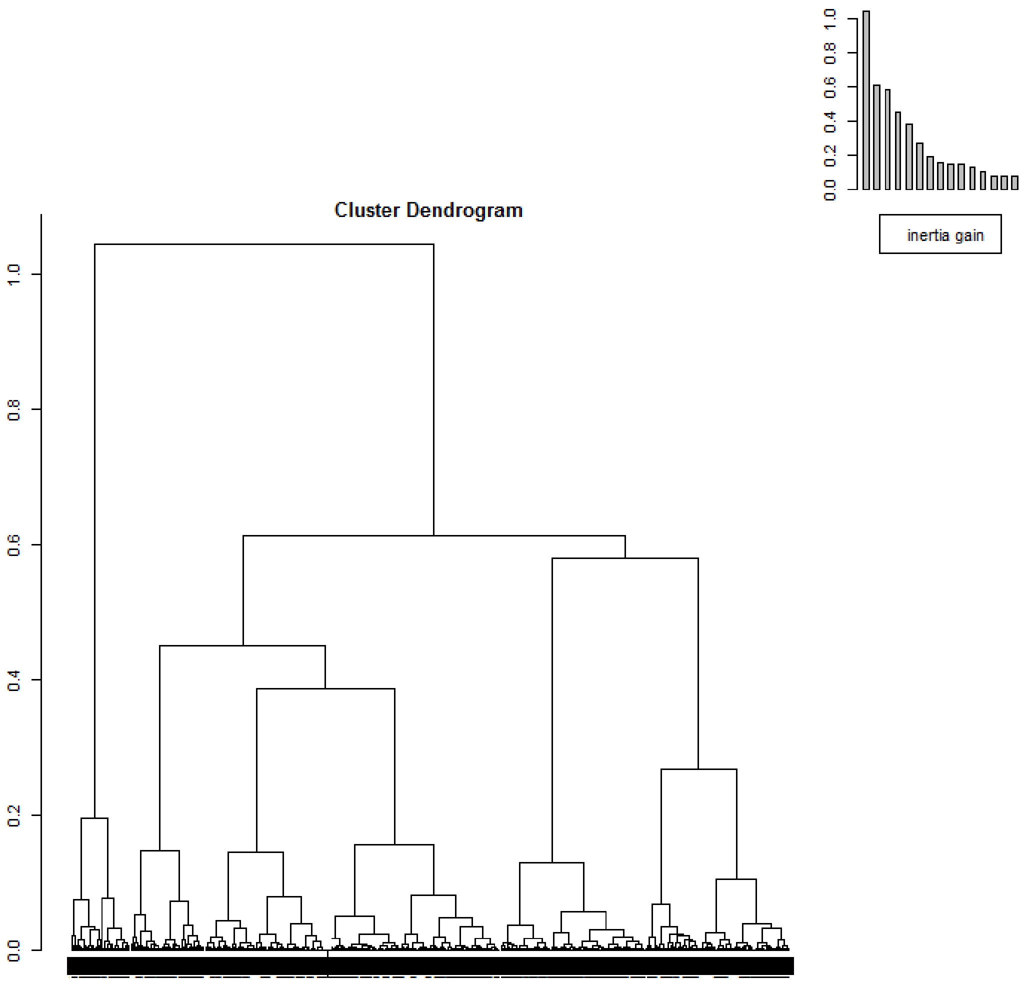

Figure 8: Cluster dendrogram for 10.217 customers, using the first six principal components. The dendrogram was computed using Ward as agglomerative method and Euclidian distance to compute dissimilarities between customers. The barplot in the upper right shows a measurement of how the inertia, or variance, decreases the deeper down in the dendrogram we go. This means that the within-cluster variance decreases and the clusters becomes more and more homologues.

The dendrogram seen in Figure 8 is based on the first six principal components. The next step is to perform a cut in the dendrogram, resulting in the actual clusters. The level of granularity can be selected by the user by applying a cut where appropriate based on the shape of the dendrogram, and based on the dendrogram in Figure 9 and the barplot showing the inertia gain, k=6 resulting in 6 clusters appears to be suitable for exploration purposes. A cut equal to 6 clusters results in an inertia gain just before the slope flattens out in the barplot, meaning that the within-inertia appears to be sufficiently low thus indicating quite homogenous clusters. The dendrogram with a cut k=6 is seen in Figure 9.

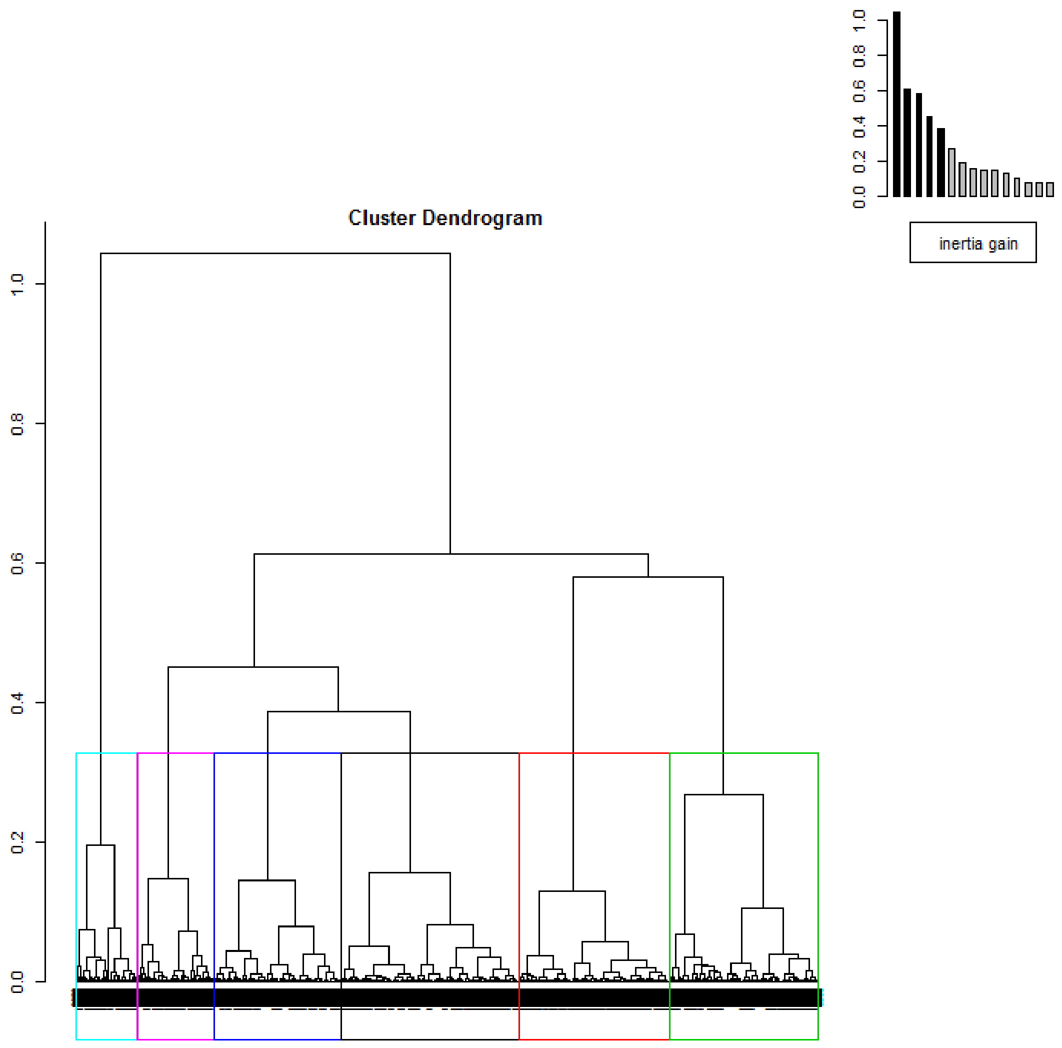

Figure 9: Dendrogram with a cut k equal to six, resulting in six distinct clusters for the first six principal components. The dendrogram was computed using Ward as agglomerative method and Euclidian distance to compute dissimilarities between customers.

The final step is to investigate the definition of each cluster based on the original 9 variables. This can be done by computing the mean and median for each cluster. By computing both, we also get an idea of the robustness of each cluster.

Table 6: The median value for each variable of the 6 clusters.

Cluster	Spend	Units	ECO	EMV	Health	Promo	Age	Num. Baskets	Avg. Basket.Size
1	10914	536	0.046	0.343	0.285	0.198	50	48	11.8
2	11228	602	0.032	0.280	0.258	0.481	73	44	13.6
3	12509	569	0.261	0.316	0.316	0.185	45	45	13.4
4	11157	530	0.025	0.227	0.171	0.166	48	53	10.4
5	20663	967	0.042	0.280	0.245	0.234	50	38	25.2
6	25671	1211	0.031	0.270	0.213	0.191	49	101	12.3

Table 7: The mean value for each variable of the 6 clusters.

Cluster	Spend	Units	ECO	EMV	Health	Promo	Age	Num. Baskets	Avg. Basket.Size
1	11615	559.7	0.058	0.347	0.289	0.210	49.3	50.5	12.5
2	12023	631.0	0.044	0.285	0.260	0.510	73.0	48.9	14.6
3	13799	619.5	0.295	0.321	0.324	0.216	47.4	48.6	14.5
4	11847	552.9	0.040	0.224	0.169	0.182	48.8	56.8	11.1
5	21007	977.5	0.066	0.281	0.247	0.257	51.1	38.4	27.7
6	27220	1287.9	0.053	0.273	0.215	0.214	50.2	110.7	12.8

The results in Table 6 and 7 differs in some variables, but they follow the same pattern. As a last step, the clusters can be defined in terms of the values of each variable in these tables.

Cluster 1 is defined by mid-aged customers that probably does their main-shopping elsewhere, since they spend a small amount of money and number of units purchased is low. They do however have a high value for number of baskets considering the small amount spent, indicating that they might be complementing their purchase at ICA. They also have the highest fraction of EMV products purchased, with a relatively large fraction in healthy products as well.

Cluster 2 is defined by upper-aged customers that spends a rather small amount of money at ICA. They appear to spend a rather large proportion on healthy products, and they are heavily promotion-oriented indicating that they are price concerned. This might reflect a senior cluster.

Cluster 3 is defined by mid-aged customers that spend a rather small amount of money at ICA. They have a large proportion of organic, healthy and EMV products purchased. This cluster can be defined by environmental-concerned customers.

Cluster 4 is defined as more frequent shoppers with a small average basket size but a higher number of baskets. This in combination with a rather small amount spent, they might be mostly buying cheaper products or some sort of complementary shopping.

Cluster 5 is defined by larger shoppers that spends a high amount of money. The average basket size is very large and in combination with a smaller value for number of baskets, they are probably a large-buyer that does the weekly shopping at ICA. They also have a rather large number of items bought on promotion.

Cluster 6 is defined by the largest buyers that spends the largest amount of money. They buy a large number of units and has a high value for number of baskets, but a rather small value for average basket size.

If we compare the describing variables in Table 6 and 7 with the density plots in Figure 4, we can see that we're not really able to reflect the extreme values in any of the 6 clusters. This doesn't come as a great surprise, since these extreme customers are seen in such small numbers compared to the bulk of customers, meaning that applying a larger cut would create more accurate clusters for these customers. We can however assume that cluster 6 for example contains the customers seen with a large value for "Spend", which can easily be controlled by looking at the top 5 customers from each cluster that appears to be capturing high values for the discerning variables. The result is seen in Table 8.

Table 8: Top 5 customers for cluster 2, 3 and 6. The most discerning variables for each cluster are marked in red, indicating the extreme values.

Cluster	**Spend**	**Units**	**ECO**	**EMV**	**Health**	**Promo**	**Age**	**Num. Baskets**	**Avg. Basket.Size**
	7901	290	0.083	0.262	0.383	1.000	45	6	48.3
	8582	640	0.039	0.208	0.269	1.000	82	28	22.9
2	11512	624	0.009	0.248	0.293	1.000	86	25	25.0
	12611	762	0.005	0.285	0.319	0.997	81	28	27.2
	13889	651	0.078	0.222	0.182	0.997	86	29	22.4
	9820	275	0.880	0.429	0.404	0.1273	57	20	13.8
	7858	320	0.781	0.503	0.409	0.7844	70	30	10.7
3	42052	1823	0.752	0.308	0.210	0.1355	56	143	12.7
	10661	368	0.734	0.337	0.326	0.0897	33	30	12.3
	8487	337	0.703	0.409	0.362	0.4065	70	25	13.5
	129326	5779	0.035	0.229	0.231	0.0557	56	159	36.3
	102210	4625	0.111	0.548	0.353	0.1933	50	170	27.2
6	73725	2726	0.039	0.329	0.201	0.1985	52	142	19.2
	69848	2863	0.018	0.177	0.135	0.1603	50	200	14.3
	67626	2601	0.294	0.367	0.395	0.1196	47	164	15.9

Table 8 shows the top 5 customers based on the most discerning variable(s) for cluster 2, 3 and 6. The discerning variables are marked in red. We see that the customers with extreme values are captured in the clusters where that variable is the most discerning for that cluster. This does not come as a great surprise, since we can expect that customers with a high value in these variables will be mapped to the cluster with that discerning variable.

The result does however show that the hypothesis of performing cluster analysis on data derived from customers' transaction history can work to describe their personas, and should be employed over time as customers' needs changes.

6 DISCUSSION

6.1 Technical limitations

The Hadoop cluster that was used during this project was quite small in comparison to what will be used in the future. There was an ambition to set up a benchmarking where the current RDBMS in Oracle would be compared to Apache Hive for ad hoc analysis, but during a telephone meeting with the company providing the platform it was understood that 4 nodes is not sufficient for a successful benchmarking. The results wouldn't be reliable since 4 nodes isn't comparable to a RDBMS database such as Oracle. The number of nodes also affects the run-time of the scripts that was used during this project. If a larger cluster would have been available, the run-time would probably be much lower than stated in this report. However, the size of the data used during this project didn't quite reach big data-sizes, making 4 nodes acceptable.

6.2 Setting up the IDE:s and programming languages

One of the goals of this project was to investigate and highlight the possibilities in code development in various programming languages. To get the different IDE:s to work was a tedious process since there is a lot of dependencies that needs to be configured. Different version of software also plays a vital role, since it will affect other programs and packages available for the different languages. These are all problems that was hard to identify in advance and instead they were encountered along the way. The most troublesome side to this is that once a dependency is solved, it might create a new issue later on during the configuration. This was especially the case when deciding which R version to use. R comes with a lot of different packages and many of them are especially designed for one or previous versions. The opposite is also true when an R version is too new for a certain package. Therefore, deciding which packages that is most important for a certain application can be more complicated that one might think. A similar issue was seen when trying to set up a connection to Spark through R. Some documentation and packages stated that it was possible, but when investigated through implementation, a lot of limitations was seen. For example, setting up a connection to Impala on the remote cluster was done, but due to the nature of R and how it is designed it wasn't possible to fetch more than a couple of hundreds of thousands of rows. There are some workarounds to this, like setting up R and RStudio on one of the data nodes in the cluster. This was however not tested due to time constraints.

Another issue arose when configuring Jupyter Notebook to enable coding in Python. At a first glance of the Spark documentation, Python seems to have the same functionality as Scala and Java, but it was soon realised that some of the most important parts was not included for Python. This was true for the implementation of the Market Basket Analysis, where the functionality to generate association rules was absent for Python. Of course, it is possible to find a way around this by implementing your own version, but then you are most probable to lose the parallelisation functionality enabled through Apache Spark. Since the implementation process can be quite time consuming as well, and the Market Basket Analysis would work as a proof of concept, it was decided that a self-written implementation was out of question.

The steps needed to successfully configure Eclipse with Maven for code development has a pretty steep learning curve, but once learned it is easy. The first time is however hard and without any experience in similar setups it’s a process that requires a lot of debugging and trial-and-error. Once the setup was configured, it is however very powerful and enables a good way to run scripts on the remote cluster by using for example WinSCP or PuTTy.

Apache Spark is built in Scala, making Scala a suitable scripting language. Unfortunately, Scala was not so suitable for data processing and data transformation. If data is fetched through SparkSQL, it is stored as a data frame. Data frames in Scala are not indexable, making high dimensional data troublesome to handle. This since each column needs to be referred to by its name and by this it’s not possible to iterate over the data frame. However, if the data is fetched as an RDD from for example HDFS, the columns are indexable. A workaround is therefore to save a data frame created through SparkSQL and then save it to HDFS to read it as an RDD.

6.3 Data modelling

The success of data modelling is highly dependent on the actual data. This since data is the fundamental requirement for any type of following analysis, and the steps needed to successfully perform the modelling phase relies heavily on how the data is collected and selected, its quality and that its business focused.

The data that was available for this project had been selected in advance as a test set for exploration, and to acquire new data wasn’t possible due to time constraints. This poses a problem in terms of the two implemented use cases, the Market Basket Analysis and the Customer Segmentation, since the two analysis is highly dependent on the selected data. In both cases, the implementations worked as proof of concepts and was mainly for exploration, but it’s important to put emphasis on that the data selection phase plays a vital role. By having a clearer view of the selected data, it also becomes much easier to validate the result of an analysis since we would have a deeper understanding of the data and what we can expect in terms of the analysis.

6.4 Market Basket Analysis

The implementation of the Market Basket Analysis worked as a proof of concept and no new inferences or correlations was expected to be found, but its however interesting to discuss the scenario where we would expect to discover patterns. By using a larger set of relevant transaction records and use a data modelling phase to filter irrelevant articles and applying a business perspective on what correlations that are profitable, it would be possible to find unexpected correlations of items that will affect how marketing is done or for example how items are physically placed in the store to increase revenue.

The implementation of the Market Basket Analysis could also be used to build a recommendation engine, where the resulting rules could be integrated in an online shopping environment and make the shopping experience easier. It would however be important to exclude irrelevant recommendations by inferring some threshold. The inferred rules could also be used to provide ICA's customers with more accurate offers, since they will be directly based on the individuals shopping patterns.

6.5 The cluster analysis

The cluster analysis during this project faced a series of problems. One of the perhaps largest issues were to understand what type of result we would expect to find from the selected data, and what could be valuable to ICA. The cluster analysis became an exploration task where I wanted to see how well we can use transaction data to describe a customer's behaviour, and how well the variables produced during the modelling phase achieves this. The resulting clusters certainly shows some patterns, but the question is whether it provides with any value. The goal was to investigate if transaction data can be used to model customer behaviour, which appears to be possible to some extent. The modelling could however be improved by spending more time on the feature analysis as well as integrate the people with business insight. By doing so, the resulting clusters would probably be more revealing and easier to translate. The final step of a cluster analysis is to in some way validate the result, which was hard in this case since there was no prior idea of what one should find. Therefore, it would have been of immense value if a similar analysis had been carried out at some point, where the result in this project could have been compared to an existing segmentation based on similar data. By this it would be possible to analyse and evaluate the performance and interpret the clusters. Another aspect of validating the result of a cluster analysis is to compare with class labels, meaning that the data points in advance has been labelled as belonging to a certain class or not, also referred to as supervised learning. However, the result in this project could work as the starting point for the construction of such labels. By continuing the data extraction phase and improve the design of the data modelling I am confident that better insights could be achieved.

During the exploration and data transformation step, many different approaches were tested. For example, a customer's purchase behaviour was described by the fraction of "markups" for all business areas, resulting in a data set with 48 dimensions. This was later discarded since the fractions became very low and the result would be hard to put into context and value. This type of problem was the result of a difficulty in understanding what variables that becomes valuable to ICA, and how to understand the data. Since the data used in this project was collected by me personally, it was hard to dive into the meaning of the data and what I expected to find. This process of data exploration and data mining should be given a lot more care, and would probably have become a very large proportion of the project and thus it was hard to invest too much time and face the risk of corrupt or inadequate data.

The cut in the dendrogram defines various levels of granularity, and the deeper down in the dendrogram we cut, the more accurate or discerning the clusters become. The cut used in this project was set to six to illustrate an example, but a larger cut would result in more discerning clusters and a lower value of the within-inertia. However, there exists a trade-off in this since not only will we acquire more accurate or defined clusters, but we will also generate more clusters that might not be of high relevance. Depending on the problem and goal with the cluster analysis, this type of procedure might be interesting or not, it basically depends on the formulation of the cluster analysis as well as what the stakeholders hope to find. One can also apply a larger cut to find clusters of outliers, depending on what we classify as an outlier or group of outliers. This type of analysis needs further investigation and proper formulation, but would be an interesting case for marketing teams where the aim would be to evaluate how accurate their offers are with respect to these smaller groups of customers.

I also believe that there exists a problem in applying hierarchical clustering on such a large set of objects. The dendrogram will become more complex with increasing number of included objects, making it hard to find a suitable cut. However, the hierarchical clustering provides a nice visualisation of the complexity and works as a good initial step and as a proof of concept for ICA.

6.5.1 *Other algorithms for the cluster analysis*

Agglomerative hierarchical clustering was deemed suitable due to the bottom-up nature of the algorithm, making it easy to find various levels of clusters and appropriate for exploration purposes. The algorithm does however have some drawbacks. In general, hierarchical clustering has problem dealing with data of higher dimension. As mentioned, the data used in this project isn't of such high dimension that this is a big problem, but for future aspects a more robust algorithm should be tested. This since the number of variables might increase in the future, making a more robust algorithm more suitable. Another downside of hierarchical clustering is the complexity of the algorithm. The complexity of agglomerative clustering is $O(n^2 \log(n))$, which makes it slower with increasing data sizes. There are however some optimised agglomerative methods with a complexity of $O(n^2)$.

One suitable approach would be to use subspace clustering, where the aim is to identify the most valuable variables in the data set that still produces good results with respect to the original variables, similar to the purpose of PCA (37). Subspace clustering can be seen as a higher level of feature selection that finds groups of objects in different subspaces in the data, and is divided into two main categories: **Bottom-Up Subspace Search Methods** and **Iterative Top-Down Subspace Search Methods**. Methods in the first category is density-based and uses an Apriori-like approach. The algorithms in this category create histograms for each individual dimension and selects bins for a given threshold. The methods then utilize the downward closure-property of density that states that if dense units exist in *n* dimensions, they exist in all dimensional projections. Subspaces are then formed based on the dense units until no more are to be found. Examples of methods in this category are ENCLUS and CLIQUE (38). Although not shown, this approach could suffer complexity issues for larger sets of data since it's Apriori-based.

Methods in the second category are initiated by finding clusters for all dimensions, which are assigned equal weights. The weight of each dimension is then updated for each cluster, which are used to regenerate the clusters. The process of these methods is expensive, and some methods uses certain techniques to improve performance. The methods also require a pre-definition of the number of clusters and the size of the subspaces, which the user need to define in advance which often is a challenging task. The most popular methods in this category are PROCLUS, ORCLUS and COSA (38).

6.5.2 *Dealing with outliers*

In this project, the goal was not to address the existence of multivariate outliers by removing or capturing them. Instead, a discussion regarding the difficulties in dealing with multivariate outliers will be presented in this section.

To define a multivariate outlier isn't an easy task. First of all, one has to declare what qualifies as an outlier. In a uni- or bivariate Gaussian space, this is easily achieved by visual inspection where the quantiles of the distribution, corresponding to extremely low and high values compared to the bulk of the objects, can be regarded as outliers. This can also be easily computed mathematically. But when the number of dimensions increase, as in the case in this project, the situation becomes much more complicated. There can exist outliers that don't qualify as an outlier by only looking at the describing variables one at a time, but by the combination of these variables and the problem almost becomes philosophical. It is also important to have a business aspect of what defines an outlier, meaning that an outlier can be defined more than just based on the describing variables. In other words, this might be more of a business-related definition than a mathematical or statistical one that needs to be answered by the people with deeper insight in the business models. However, statistical models can be applied to some extent but to find relevant literature that aims to capture outliers in the context presented in this project has been hard. Most papers refer to outliers in retail as fraud or intrusion detection, not as a customer with a unique behaviour that differs from the everyday-customers. The discussion henceforth will be based on what I believe might be suitable ways to address multivariate outliers defined as customers with extreme purchase behaviour.

One way to assess outliers is with PCA, since PCA can be seen as a projection pursuit technique. The aim is to reduce the number of dimensions and in this subspace, classify which points that can be marked as outliers. The authors of Franklin *et. al.* (39) cites the book "Robust Statistics" by Huber (1981) which states that the leading principal components pick projections with good separations and they collect the systematic structure of the data. By this, the top components reflects the major linear trends and objects far away from these can be deemed as outliers (39). However, more recent studies suggest that the more insignificant principal components, meaning the ones with low variance explained, should be given attention as well. The authors of O'Neill *et. al.* (40) suggests that multivariate outliers can be identified using PCA, and present four outlier criteria based on the book by Joliffe (41). They show that the insignificant components reflect objects that have a defective behaviour (40). The authors of Shyu *et. al.* (42) also argues that the more insignificant components are sensitive to observations that are inconsistent with the overall correlation of the data but are not outliers with respect to the original variables, which is exactly the case in this project. In their study, they managed to show that outliers can be successfully classified by combining the top components that account for around 50% of the variance with the less significant components with an eigenvalue below 0.2. In their study, they compared their developed approach with 4 other outlier detection methods and the results indicated that their approach outperformed for all false alarm rates. This study was however performed for the sake of intrusion detection and hasn’t been tested on retail data as used in this project. I investigated how well it worked to perform an outlier detection based on the two most insignificant components, and it appeared to work to some extent. The hard part was to validate this, and one way I tried to look at it was to go back to the density plots for the original variables. The distributions for each variable became a lot more manageable and almost conformed to a normal distribution, which indicated that the customers with extreme values based on the individual variables had been removed. The result was however inconclusive and to theoretically validate this procedure probably needs a study on its own, and to be performed on data where we have some idea what defines an outlier so the result can be validated.

Another suggested way to identify outliers in multivariate data is to use an angle-based approach. The authors of Kriegel *et. al.* (43) argues that distance metrics become vaguer with increasing dimensionality, and thus suggests using the distance metric together with the directions of the vectors in the space. They propose that comparing angles between pairs of distance vectors with data points assists in discerning between similar points and outliers. They argue that the angles between pairs of difference vectors of points within a cluster differ a lot and the angles decrease for points approaching the border of a cluster, but the variance will still be higher than the variance of an outlier not belonging to any cluster. Thus, if the spectrum of surrounding angles for a point is large, it indicates that the point is surrounded by other points and belong to a cluster. If the opposite is true, the spectrum of surrounding angles for a point is small, it indicates that other points only are seen in some directions and thus the point is located outside a set of points. This means that small angles for a point that are similar to surrounding points imply that the point is an outlier. The authors showed that this approach works well to counter-act the curse of dimensionality and offers a parameter-free outlier detection approach for high-dimensional data (43).

Another appropriate approach would be to use density based clustering, which is able to identify arbitrary cluster shapes and handles noise effectively, thus very sensitive to the presence of outliers. The density based algorithms are designed to recognise clusters of points based on evaluating connected regions and regions with high density. The density is defined as the neighbourhood of points close to an object, which are defined by the user that specifies two parameters. The first parameter is an indication of the radius of the neighbourhood, and the second parameter defines the number of points included in this neighbourhood (44). This approach was tested, but to specify these parameters for the data set used in this project was hard, since the data was of quite high dimension making it hard to visually inspect dense regions. High dimensional data also becomes a problem for density based clustering, since the density of objects decreases and the measurement becomes more irrelevant.

So far, the discussion regarding outliers has been to find a way to remove them. However, the opposite might be interesting for ICA, where the aim is to find these outliers to offer an even more personalised experience. This could lead to better customer relations since these outliers will feel that ICA is able to see them and capture their personas. How this will be achieved is a matter of how an outlier should be viewed, which has been a philosophic aspect throughout this project. It isn’t an easy task to succeed with this definition, and with an appropriate scope of an analysis this definition might be easier to define. For example, it might be hard to identify the scope for a multivariate data set similar to the one used in this project, since this project has highlighted the issues in defining such customers. Instead, I believe it would be easier to look at a smaller set of variables or even one variable for a marketing campaign with the goal of identifying unusual behaviour. As mentioned, hierarchical clustering could be used to investigate the existence of clusters of outliers and perhaps gain more insight in what appears to be defining them.

In summary, there are several approaches to address multivariate outliers. However, without any knowledge how the data should be defined in the context of an outlier, it’s hard to select an appropriate algorithm. I believe that the first step in dealing with outliers in the context of retail data needs to start with a proper definition. The authors of Banek *et. al.* (45) analysed outliers in promotion data for a retail company, where the analysis was started by properly stating what the goal of the analysis were and by this easily identify what defines an outlier. A similar approach should be carried out for the data used in this project, where a closer collaboration with people with insight in the business models would bring high value.

6.6 Big data vs. traditional systems

This project has mostly put emphasis and argued for the possibilities in using the big data technologies, especially in terms of data analysis. I do however feel that it's important to provide a discussion that compares the performance of big data systems with more traditional ones since one can be preferable over the other in certain aspects.

As with all modern technologies, there exist bottlenecks or hidden limitations in certain aspects. First of all, there is a steep learning curve in understanding and eventually mastering the new technique to harvest the full potential. I also believe that one must keep in mind that all problems or areas aren't directly resolved by switching to the newest and most hyped technology on the market. For example, MongoDB, a NoSQL DBMS is in some aspects better than a SQL DBMS, such as Oracle, since MongoDB is designed for storing unstructured data. MongoDB is also able to store larger data, which unstructured data usually infers in terms of clickstream or web logs. MongoDB has also been shown to be faster in inserting, deleting and updating existing tables, which can be suitable depending on the task at hand. Oracle on the other hand enables more complex and robust database structures since MongoDB doesn't provide relations between tables. These aspects conclude that it's important to have a clear view of what one hopes to accomplish in the long run, and what characteristics that are important. If the system for example needs fast response time, a NoSQL DBMS such as MongoDB is preferable, but if a more complex and robust DBMS is needed, then a SQL DBMS such as Oracle might be better (46). The same analogy can be made when dealing with the big data technologies, it all boils down to the task at hand and what characteristics of the system that is desirable. If the possibility to analyse large unified data sets or unstructured data in parallel where high response time is the highest priority, then Apache Spark is a very good candidate. If the possibility to analyse nested data in a complex DBMS is desirable where the response time isn't a limiting factor, then a DBMS such as Oracle should probably be used.

I also believe that there is an aspect of ease of usability for the different systems. The people using the system is probably more accustomed to a traditional SQL system than a NoSQL system. This disables the quite time-consuming task of learning a new system and increases productivity, as well as better knowledge in common system-related issues and problems that might arise. However, with newer systems comes also new application areas and possibilities, but there might be a limitation in the number of tools that are possible to integrate, especially for data analysis. Compared to a SQL system, where third-party software and tools have been developed around SQL for centuries, most tools would be needed to be developed in-house and current systems might be hard to integrate with the new one. However, in the age of digitalisation it is extremely important for companies to keep up with new emerging technologies to maintain their position on the market and at least evaluate the impact and if it would bring any value.

6.7 Scalability for future aspects

The datasets used in this project doesn’t really reach “big data sizes”, therefore it hasn’t been possible to pinpoint certain bottlenecks or limitations with the Hadoop cluster used in this project. As mentioned, the cluster was quite small in comparison to what should and probably will be used in production later, where probably a two- or three-digit number of nodes would be needed for a highly scalable system. However, it’s important to speculate whether one’s data actually will reach sizes that can be characterized as “big data” and what data that actually is worth storing in the Hadoop cluster. Here we would need to exceed terabytes of data that will be used for analysis, otherwise the big data technologies won’t play a vital part and thus data analysis or processing might be slower than if a traditional system would be employed. It also boils down to the actual data, for example if more transaction data would be stored or data in unstructured format such as clickstream data. We have seen that transaction data can be further investigated to describe a customer’s persona, and by this more data would mean that we could capture more customers in the same analysis. Then again, it’s important to have an idea of the goal of the analysis and if more data is needed to achieve the desired result. For ICA’s own sake, I believe that there are certain trade-offs with a Hadoop cluster compared to a platform for analysis built on top of an RDBMS and it’s important to define why one of the systems are more valuable than the other. In terms of data analysis, I do believe that a Hadoop cluster is superior in the sense of scalability and parallelisation through Apache Spark, and for storage of unstructured data such as clickstream data. But if analysis of such large quantities of data isn’t needed, the technologies provided in Apache Hadoop shouldn’t probably be employed.

7 FUTURE WORK

7.1 Deeper insights in the data

As mentioned in the discussion section, the were no prior knowledge or idea of what the cluster analysis would yield. This was of course an exciting thing, but led to troubles when evaluating the modelled data and the result of the cluster analysis. The work regarding the data modelling in this project can be seen as a first step which shows that it might be possible to use transaction data to describe customer behaviour, but for future work better insights would probably be achieved with more knowledge or an idea of what can be found and by this perform a more successful data modelling. To do this, the business team at ICA should be consulted and the technical possibilities and limitations should be presented so a mutual understanding can be set.

7.2 More data

Another part of data analysis is the actual data at hand. By using more data or a broader spectrum of data, new insights might show and new patterns might be revealed. By increasing the amount of data in this project, I think there might be a possibility to find even better groups of customers since new groupings might form that before were too small to see. Since the aim in this project was to investigate the role of big data technologies, more data would also have improved the context of handling large quantities of data. The clustering step could however suffer time complexity issues since hierarchical clustering, as mentioned, becomes slower with increasing data quantity. Therefore, more scalable algorithms should be studied and employed.

7.3 More variables for the cluster analysis

This project has revealed the possibility to use transaction data to describe customers on a new level, and the result indicates that a hierarchical cluster analysis might be suitable. For future aspects, it would be interesting to add more describing variables which would reveal other characteristics about ICA's customers, or replace some of them used in this project. In this section, a couple of interesting features are proposed and discussed.

7.3.1 *A measurement for brand loyalty*

A feature that would bring high value is a measurement of customers' brand loyalty. This can either be performed on a specific category such as dairy products or an article for a promotion. By doing this, ICA would get insight in which customers or households that would be suitable targets for promotions or discounts. This attribute was successfully computed in R on a smaller set of transactions (about 700.000 transactions of 1.000 customers), but was not implemented in the Scala script. This since the Scala

API lacked the convenient tools used in R and time constraints played a large role. The logic behind the computation in R is presented below.

The variable **Mean Brand Loyalty** was calculated by looking at the combinations of *CustomerID*, *Category* and *Brand*. The hypothesis was as follows:

"For a given category (e.g. yoghurt), how many different brands were purchased?"

The first step was to use an aggregation in R which computes each unique combination of *CustomerID*, *Category* and *Brand*, resulting in a measurement of the frequency of the combinations seen as *count_brand* in Table 14 below.

Table 9: Table illustrating the result from calculating how many times a customer purchased each brand within each category, seen as variable "count_brand".

Customer ID	**Category**	**Brand**	**count_brand**
2511747360	Yoghurt	Brand 1	2
2511747360	Yoghurt	Brand1	2
2511747360	Yoghurt	Brand2	1

We see in Table 9 that the combination [2511747360; Yoghurt; Brand1] is seen 2 times, indicating that *Brand1* was purchased with a frequency of 2 from this *Category*. We also see that these rows are duplicates, which is taken care of in a later step.

The next step is to calculate the combinations of *CustomerID* and *Category*, resulting in an idea of how many times a customer purchased items from a certain category, resulting in variable *count_category* seen in table 10 below.

Table 10: Table illustrating result from calculating how many times a customer purchased an item from a certain category, seen as variable "count_category".

Customer ID	**Category**	**Brand**	**count_brand**	**count_category**
2511747360	Yoghurt	Brand 1	2	3
2511747360	Yoghurt	Brand1	2	3
2511747360	Yoghurt	Brand2	1	3

The next step is to filter for duplicate rows and computing the measurement for brand loyalty by dividing *count_brand* with *count_category*. This results in a number indicating the level of loyalty towards a brand in a certain category, seen as variable *brand_loyalty* in Table 16 below.

Table 11: Table illustrating the result from calculating the brand loyalty as the ratio of "count_brand" and "count_category", seen as variable "brand_loyalty".

Customer ID	**Category**	**Brand**	**count_brand**	**count_category**	**brand_loyalty**
2511747360	Yoghurt	Brand 1	2	3	0.667
2511747360	Yoghurt	Brand2	1	3	0.333

The calculation for variable *brand_loyalty* above might suffer some complexity issues since duplicate rows are created. Therefore, if a future implementation would be of interest, some optimisation would be needed to avoid this.

7.3.2 *A measurement for exclusive customers*

A measurement for customers' spending level has been used during this project, but this measurement doesn't reveal if the individual customer spends a high fraction on more costly or cheaper articles. Therefore, it would be interesting to compute a variable describing how much customers spends on more exclusive brands or categories. The implementation of this variable would be quite straightforward, the harder part would be to infer the logic of which articles or categories that should be classified as exclusive. The last step would then be to infer a threshold that defines if a customer spends more on exclusive articles/brands than other customers in general.

7.3.3 *A measurement for redeeming of offers*

A big part of the retail business is to provide customers with offers on popular or suitable articles. However, it can be hard to know whether or not the offers are appreciated by the customers or if the customers are totally uninterested in them. Therefore, a valuable variable would be a measurement of the fraction of offers redeemed by a customer. By then inferring a threshold for this variable, it would be possible to discern customers with high interest in offers and those with low.

7.3.4 *New article/demographic labels*

The article labels used in this project, referred to as "markups", was seen to be very useful in creating variables that describe the persona of a customer. For future aspects, new labels could be introduced. It would for example be very interesting to infer labels such as "vegetarian" or "vegan" that could bring valuable insights. One could even go beyond that and introduce labels for over-sensitivity and allergies as demographic attributes. However, some of these labels would be dependent on whether or not the customer is a single person or a household.

By inferring more labels or high-level machine learning, I believe it would be easier to follow a customer’s journey through life. People change behaviour over time and a person can go from being a meat-lover to vegetarian in weeks, making offers of meatier products highly irrelevant and the customer might feel offended. I believe there exists value in evaluating this type of thinking in order to take the retail business to a new level. One can argue that the data becomes too complex or large by doing this, but with the concept of big data and new technologies, I don’t think that this will pose such a large problem.

8 CONCLUSION

The aim of this project was to investigate the role of big data and machine learning in retail from both a technical and practical point of view. During the project, problems arose in different phases that showcases both the complexity of a big data system as well as how to formulate data related analytics. Even though the field of big data promises a lot in different contexts, it's important to realise that it requires some fundamental understanding of the different technologies.

Even though some technical problems were encountered, this project shows how powerful the big data technologies can be for data analysis. The most promising aspect is that the different steps of a traditional analysis can be made in one single script. The data is loaded, parsed, transformed, evaluated and saved in separate parts of the system but merged as one during the analysis. This leads to ease of use and gives a clear view of the entire analytical pipeline, but might not be optimal during early development stages.

The presented workflow in this project validates the hypothesis stated in section 5. The big data technologies provided by Apache Hadoop contained a good set of tools to set up an environment for advanced analytics of transaction data. This in turn provides ICA with good insights in advanced technologies and proves that more advanced analytics can be done inhouse. The findings also highlight the suitable programming languages, certain drawbacks that was identified and hopefully a sense that nothing is impossible. It was also shown that the item hierarchy ICA uses today is excellent for extracting variables that describes a behaviour, but also puts emphasis on that deeper insights can be achieved with more accurate and improved data. The result of the cluster analysis also shows the complexity of dealing with data of higher dimensions and highlights the difficulties in assessing outliers. More work is needed to investigate how data should be defined and what one hopes to accomplish with the analysis, as well as how a multivariate outlier should be defined and whether or not they are of interested in an analysis.

As a concluding mark, this project shows that with minor prior knowledge of a big data system, it is possible to successfully implement and run interesting analytics. It also shows the value of implementing these systems in-house, which for future aspects will be both economically effective as well as technically important for future competition. It is however important to evaluate the exact purpose of a migration to these technologies, since current traditional systems are more appropriate in some aspects.

9 Acknowledgements

First of all, I would like to thank my supervisor at ICA, Lotta Silverberg. Thank you for having me in your project group and giving me the opportunity to conduct my thesis project at ICA. I also want to thank you for the huge support through challenging times and believing in me. I would also like to thank the consultants from TCS that was part of the GCV 360 project, especially Sooraj Koroth whom I worked closely with during the exploration phase of the big data technologies. I would also like to thank all other persons that I met during my time at ICA, it was a very nice working environment and I learned a lot about a big company as ICA. I would show my appreciation to my supervisor Jan Andersson for all support. Last but not least, I would like to thank my girlfriend and family for all the support.

10 References

1. ICA Sverige. Available from: http://www.icagruppen.se/om-ica-gruppen/var-verksamhet/ica-sverige/. Cited 2017 Jan 18.

2. Apache Trademark FAQs. Available from: https://www.apache.org/foundation/marks/faq/#insidebook. Cited 2017 May 28.

3. Marz N, Warren J. Big Data: Principles and best practices of scalable real-time data systems. Manning; 2015.

4. Exabyte. In: Wikipedia. 2017. Available from: https://en.wikipedia.org/w/index.php?title=Exabyte&oldid=767915608. Cited 2017 Apr 4.

5. IBM - What is big data? 2017. Available from: https://www.ibm.com/software/data/bigdata/what-is-big-data.html. Cited 2017 Apr 4.

6. The Four V:s of Big Data. IBM Big Data & Analytics Hub. Available from: http://www.ibmbigdatahub.com/infographic/four-vs-big-data. Cited 2017 Jan 25.

7. Taleb I, Kassabi HTE, Serhani MA, Dssouli R, Bouhaddioui C. Big Data Quality: A Quality Dimensions Evaluation. In: 2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld). 2016. p. 759–65.

8. Ahmed FD, Jaber AN, Majid MBA, Ahmad MS. Agent-based Big Data Analytics in retailing: A case study. In: 2015 4th International Conference on Software Engineering and Computer Systems (ICSECS). 2015. p. 67–72.

9. Manikandan SG, Ravi S. Big Data Analysis Using Apache Hadoop. In: 2014 International Conference on IT Convergence and Security (ICITCS). 2014. p. 1–4.

10. Welcome to ApacheTM Hadoop®!. Available from: http://hadoop.apache.org/. Cited 2017 Jan 18.

11. Thusoo A, Sarma JS, Jain N, Shao Z, Chakka P, Zhang N, et al. Hive - a petabyte scale data warehouse using Hadoop. In: 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010). 2010. p. 996–1005.

12. Apache Hadoop HDFS. Hortonworks. 2016. Available from: http://hortonworks.com/apache/hdfs/. Cited 2017 Feb 17.

13. HDFS Architecture Guide. Available from: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html. Cited 2017 Jan 18.

14. Apache Hadoop 2.7.2 – Apache Hadoop YARN. Available from: https://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/YARN.html. Cited 2017 Feb 17.

15. White T. Hadoop: The Definitive Guide. Available from: http://shop.oreilly.com/product/0636920033448.do. Cited 2017 Apr 23.

16. Chiang R, Dawson D. Untangling Apache Hadoop YARN, Part 1: Cluster and YARN Basics. Cloudera Engineering Blog. 2015. Available from: http://blog.cloudera.com/blog/2015/09/untangling-apache-hadoop-yarn-part-1/. Cited 2017 Apr 23.

17. Maheshwar RC, Haritha D. Survey on high performance analytics of bigdata with apache spark. In: 2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT). 2016. p. 721–5.

18. IBM - What is MapReduce? - United States. 2016. Available from: https://www.ibm.com/software/data/infosphere/hadoop/mapreduce/. Cited 2017 Feb 8.

19. Owen S. How-to: Translate from MapReduce to Apache Spark. Cloudera Engineering Blog. 2014. Available from: http://blog.cloudera.com/blog/2014/09/how-to-translate-from-mapreduce-to-apache-spark/. Cited 2017 Feb 8.

20. tutorialspoint.com. Apache Spark RDD. www.tutorialspoint.com. Available from: https://www.tutorialspoint.com/apache_spark/apache_spark_rdd.htm. Cited 2017 Feb 8.

21. Cloudera © 2017, Hadoop IA rights reserved A, trademarks associated open source project names are trademarks of the ASFF a complete list of, Here C. Spark Application Overview . Available from: https://www.cloudera.com/documentation/enterprise/5-6-x/topics/cdh_ig_spark_apps.html. Cited 2017 Apr 23.

22. Bowling M, Fürnkranz J, Graepel T, Musick R. Machine learning and games. Mach Learn. 2006 Jun 1, 63:211–5.

23. Molter TW, Nugent MA. Machine Learning with Memristors via Thermodynamic RAM. In: CNNA 2016; 15th International Workshop on Cellular Nanoscale Networks and their Applications. 2016. p. 1–2.

24. Brownlee J. Supervised and Unsupervised Machine Learning Algorithms. Machine Learning Mastery. 2016. Available from: http://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/. Cited 2017 Jan 26.

25. MLlib - Spark 2.0.0 Documentation. Available from: https://spark.apache.org/docs/2.0.0-preview/mllib-guide.html. Cited 2017 Feb 8.

26. Hahsler M, Grün B, Hornik K. A computational environment for mining association rules and frequent item sets. 2005. Available from: http://epub.wu.ac.at/132/. Cited 2017 May 17.

27. Han J, Pei J, Yin Y. Mining Frequent Patterns Without Candidate Generation. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. New York, NY, USA: ACM; 2000. p. 1–12. (SIGMOD ’00). Available from: http://doi.acm.org/10.1145/342009.335372. Cited 2017 Apr 10.

28. Tan P-N, Steinbach M, Kumar V. Introduction to data mining. 1. Harlow: Pearson; 2014.

29. Mukhopadhyay U, Sinha S, Ghosh P, Ghosh R, Kole DK, Chakroborty A. Enhancing the Security of Digital Video Watermarking Using Watermark Encryption. In: Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology. New York, NY, USA: ACM; 2012. p. 145–150. (CCSEIT ’12). Available from: http://doi.acm.org/10.1145/2393216.2393241. Cited 2017 Apr 11.

30. Jain AK, Murty MN, Flynn PJ. Data Clustering: A Review. ACM Comput Surv. 1999 Sep, 31:264–323.

31. Aggarwal CC, Hinneburg A, Keim DA. On the Surprising Behavior of Distance Metrics in High Dimensional Spaces. In: Proceedings of the 8th International Conference on Database Theory. London, UK, UK: Springer-Verlag; 2001. p. 420–434. (ICDT ’01). Available from: http://dl.acm.org/citation.cfm?id=645504.656414. Cited 2017 Apr 13.

32. Murtagh F, Legendre P. Ward’s Hierarchical Agglomerative Clustering Method: Which Algorithms Implement Ward’s Criterion? J Classif. 2014 Oct 1, 31:274–95.

33. Davidson I, Ravi SS. Agglomerative Hierarchical Clustering with Constraints: Theoretical and Empirical Results. In: Knowledge Discovery in Databases: PKDD 2005. Springer, Berlin, Heidelberg; 2005. p. 59–70.

34. FactoMineR: Exploratory Multivariate Data Analysis with R; Hierarchical Classification on Principal Components (HCPC). Available from: http://factominer.free.fr/classical-methods/hierarchical-clustering-on-principal-components.html. Cited 2017 Jun 3.

35. Dharmaraajan K, Dorairangaswamy MA. Analysis of FP-growth and Apriori algorithms on pattern discovery from weblog data. In: 2016 IEEE International Conference on Advances in Computer Applications (ICACA). 2016. p. 170–4.

36. Kestelyn J. Introducing sparklyr, an R Interface for Apache Spark. Cloudera Engineering Blog. 2016. Available from: https://blog.cloudera.com/blog/2016/09/introducing-sparklyr-an-r-interface-for-apache-spark/. Cited 2017 Apr 11.

37. Song Q, Ni J, Wang G. A Fast Clustering-Based Feature Subset Selection Algorithm for High-Dimensional Data. IEEE Trans Knowl Data Eng. 2013 Jan, 25:1–14.

38. Parsons L, Haque E, Liu H. Subspace Clustering for High Dimensional Data: A Review. SIGKDD Explor Newsl. 2004 Jun, 6:90–105.

39. Franklin S, Thomas S, Brodeur M. Robust Multivariate Outlier Detection Using Mahalanobis' Distance and Modified Stahel-donoho Estimators - Semantic Scholar.

40. O'Neill PM. Production Multivariate Outlier Detection Using Principal Components. In: 2008 IEEE International Test Conference. 2008. p. 1–10.

41. Principal Component Analysis | I.T. Jolliffe | Springer.

42. Shyu M-L, Chen S-C, Sarinnapakorn K, Chang L. A Novel Anomaly Detection Scheme Based on Principal Component Classifier. 2003.

43. Kriegel H-P, S hubert M, Zimek A. Angle-based Outlier Detection in High-dimensional Data. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, NY, USA: ACM; 2008. p. 444–452. (KDD '08).

44. Amini A, Wah TY, Saboohi H. On Density-Based Data Streams Clustering Algorithms: A Survey. J Comput Sci Technol. 2014 Jan 1, 29:116–41.

45. Banek M, Osrecki D, Vranic M, Pintar D. Outlier detection as the primary step for promotion planning in retail. In: 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). 2015. p. 1405–10.

46. Boicea A, Radulescu F, Agapin LI. MongoDB vs Oracle – Database Comparison. In: 2012 Third International Conference on Emerging Intelligent Data and Web Technologies. 2012. p. 330–5.

47. Andreu-Perez J, Poon CCY, Merrifield RD, Wong STC, Yang GZ. Big Data for Health. IEEE J Biomed Health Inform. 2015 Jul, 19:1193–208.

48. Kashyap H, Ahmed HA, Hoque N, Roy S, Bhattacharyya DK. Big data analytics in bioinformatics: architectures, techniques, tools and issues. Netw Model Anal Health Inform Bioinforma. 2016 Dec 1, 5:28.

49. Ye K, Schulz MH, Long Q, Apweiler R, Ning Z. Pindel: a pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads. Bioinformatics. 2009 Nov 1, 25:2865–71.

50. Chen S, Lau TK, Zhang C, Xu C, Xu Z, Hu P, et al. A method for noninvasive detection of fetal large deletions/duplications by low coverage massively parallel sequencing. Prenat Diagn. 2013 Jun 1, 33:584–90.

51. Creighton C, Hanash S. Mining gene expression databases for association rules. Bioinformatics. 2003 Jan 1, 19:79–86.

52. Jiang D, Tang C, Zhang A. Cluster analysis for gene expression data: a survey. IEEE Trans Knowl Data Eng. 2004 Nov, 16:1370–86.

Appendix A - software and versions used

Software/Program	Version
Scala	2.11.5
R	3.3.2
Python	2.6.6
Java	1.7.0_95
Jupyter Notebook	4.2.1
RStudio	0.99.892/1.0.143
Eclipse	Jee Neon
Maven	1.7
Junit	5.8.1
Apache Spark	2.0.0

Appendix B – copyrights and trademarks

In this project, logos and products provided by Apache are referred to. As of this, trademark attributions will be listed in this section as outlined on their website (2).

Apache®, Apache Hadoop, Hadoop, and the elephant logo are either registered trademarks or trademarks of the Apache Software Foundation in the United States and/or other countries. No endorsement by The Apache Software Foundation is implied by the use of these marks.

Apache®, Apache Spark, Spark, and the star logo are either registered trademarks or trademarks of the Apache Software Foundation in the United States and/or other countries. No endorsement by The Apache Software Foundation is implied by the use of these marks.

Apache®, Apache Hadoop HDFS, Hadoop HDFS, and the elephant logo are either registered trademarks or trademarks of the Apache Software Foundation in the United States and/or other countries. No endorsement by The Apache Software Foundation is implied by the use of these marks.

Appendix C - Big Data and machine learning in bioinformatics

As mentioned in the introductory parts of this report, a lot of different areas are today conforming to the big data paradigm. A field that is working with large data sets is bioinformatics. This section will therefore mention the areas of which big data technologies can improve some of the problems that exist today as well as mention what the used algorithms in this project can be used for when dealing with biological data.

Handling large datasets

One of the main areas of bioinformatics is to analyse sequence data generated from NGS technologies, which creates extremely large results. This is especially true for whole genome sequencing where the aim is to study complex diseases in for example the human body. Depending on the depth of the sequencing, a whole human genome sequence can reach as much as 200 GB which can be a problem to effectively store and handle, making big data technologies a suitable solution (47). One can also imagine the need to run an analysis on multiple sequences, resulting in even larger data sets used for analysis. The technology that would be suitable for this is Apache Impala, which is the fastest data warehouse of the big data technologies.

Another field within bioinformatics that has emerged is image analysis where the morphology of an organ is used to define functionality. One of the most studied organs for image analysis is the brain, since different brain structures and tissues are responsible for processes that explains cognitive processes. This is referred to as Neuroimaging and the analysis is highly dependent on RAM. In the paper by Andreu-Perez *et. al.* (47), they showed that the forecasted amounts of RAM needed for an analysis by 2020 could reach 260 GB, which might pose a big problem when running on local machines. Big data offers the possibility to scale horizontally instead of vertically, which means that more nodes or virtual machines are added when the workload grows instead of adding new costly hardware. This is an important aspect from an economical point of view, since we only need to add more virtual machines for a much smaller amount of money than if we would need to add more hardware. It is also a lot faster and more efficient to scale horizontally than to order and install new RAM.

Even though there has been an immense increase in data generation within the field of bioinformatics, the big data challenges in this field diverges from the ones seen in other fields. This is mainly due to the fact that the data is heterogenous, meaning that multiple heterogenous and independent databases are needed for validation and inference of analysis. Another issue is that most data sources are spread out geographically, leading to troubles in transferring large data sets efficiently. Due to this, a big data problem that has become unique for bioinformatics data is the global distribution of the data, in combination with the 4 V:s volume, velocity, veracity and variety mentioned in the introductory part of this report. This problem is often resolved by setting up a cloud service, making the sharing of large data sets possible and has been shown to work efficiently (48).

Frequent Pattern Growth and association rules mining

The output from NGS technologies can also be used to investigate genetic variants, such as insertion and deletions. NGS technologies are also able to detect SNPs, and the paper (49) mentions a study where Illumina was used to detect around 4 million SNPs and 400.000 indels of sizes 1-16 bp in the human body. The paper mentions that small indels are easy to detect and common to study, but longer variants requires an assembly such as *de novo* assembly or using unmapped reads. However, this approach is mainly successful for shorter genomes such as bacteria, and higher organisms above the eukaryotic hierarchy are problematic due to repetitive regions. This results in low quality *de novo* assemblies and the authors suggests a pattern growth algorithm that detects large deletions of sizes between 1 bp and 10 kb. Their algorithm is based on *PrefixSpan* which is a sequential pattern mining algorithm (the *PrefixSpan* algorithm is available in the Spark MLlib package). Their developed software successfully identified deletions of sizes of 10 kb using this algorithm. This paper was published in 2009, and new methods to detect even larger deletions has been designed. In the paper by Shengpei *et. al.* (50), they designed a sequencing approach that was able to detect deletions/duplications over 10 Mb. However, the paper Kai *et. al.* (49) shows that frequent pattern growth algorithms can be used for this type of problems and improved algorithms might already have been developed. Their implementation suffered some problems in run-time and memory usage, which in theory could be solved in a big data system. It would therefore be very interesting to see how a similar implementation in a big data environment would compete with the results in by Shengpei *et. al.* (50).

Association rules mining are, as seen in this project, often applied in Market Basket Analysis and was successfully combined with the FPG algorithm. Association rules mining can also be applied on its own, and Chad *et. al.* (51) implemented an algorithm that is used to mining association rules from gene expression data. The paper compares the gene expression mining to the Market Basket Analysis, where a gene expression profile can be seen as a transaction and each gene as an article, i.e. one row of the table used in this project. The difference is that an article is always purchased in a transaction, but a gene is either marked as up, down or neither up or down regulated. The authors implemented an Apriori-like algorithm, similar to the FPG algorithm used in this project, that first finds frequent items from which association rules are computed. The result from such an algorithm shows which genes that are highly expressed with one single gene. This can reveal new gene interactions and quite fast give insight in an organism's genome. The paper also argues that association rules can be used to relate gene expression to environmental factors and give insights in gene networks. They however state that identified association requires some prior knowledge of the biological background or requires some further investigation of the including genes.

Cluster analysis

Cluster analysis can be applied in many different areas within bioinformatics. For example, microarray techniques enable the possibility to study gene expression levels on thousands of genes in parallel. By applying hierarchical cluster algorithms on this data, one can get new insights in gene functionality and regulation and gain understanding in complex cellular processes. This is achieved since gene expression profiles with similar functionality are grouped together, which reveals new insights and points towards coregulation of these genes (52).