Data mining and scalable unsupervised clustering of customer data

Stephen Dooley

April 8, 2016

Student ID: 12502947
Class: 4ECE
Supervisor: Dr. Michael Madden

Abstract

This project focuses on the implementation of a clustering model to customer data collected on an e-commerce website. Customer data has been retrieved from Altocloud's database and processed using data mining and machine learning tools. It consisted of both demographic and behavioural information aggregated to form profiles for multiple customers who have visited and shopped on an e-commerce website.

The objective of clustering the data is to uncover some trends or patterns in the data that would otherwise be unrecognisable. There is a pressing need for personalisation and identification of similar groups of customers. Without personalisation, the large amounts of data are not used to their full potential. After discovering some natural groups or clusters in the data, the clusters can be used to influence decisive and effective business decisions that will allow a company, like Altocloud, to capitalise. The results presented in this thesis show that it is in fact possible to accurately group customers according to their natural and inherent characteristics, uncovered by the K-means clustering algorithm. Additionally, the clusters can be used to influence market segmentation which can provide commercial value to a business.

Acknowledgments

Firstly, I would like to thank my supervisor Dr. Michael Madden, for his invaluable feedback and support throughout this project.

I would also like to thank the team at Altocloud for providing me with the tools to complete this project. The decision to carry out this research was inspired by my experience as an intern at Altocloud. I would like to pay special thanks to Joe Smyth, Maciej Dabrowksi and Conor Fennell for taking time off work to meet with my supervisor, and for providing constructive feedback throughout the entirety of this project.

Contents

List of Figures

List of Tables

1 Introduction

The following thesis is concerned with the application of a *clustering*[1] algorithm on customer data using open source and scalable technologies. The customer data was collected by Altocloud from visitors on an electronic commerce (commonly referred to as *e-commerce*[2]) website. The ability of the clustering algorithm is demonstrated by carrying out experiments on the moderately high dimensional customer dataset. This section will introduce and briefly describe the main components of the work carried out.

1.1 Overview

There are two core aspects addressed in this project. Both are described briefly in this overview.

Firstly, and more importantly, we look at topics such as machine learning, data mining and clustering. The design and architecture of a clustering model is followed by analysis and interpretation of results produced by the clustering algorithm. Over 4000 instances of customer *journeys*[3] on an e-commerce website are used for training, testing and validating a machine learning model. Forming distinct, meaningful clusters from the data is the aim of the clustering. Using the machine learning model, predictions about "future" customers on the website can be produced.

There is an abundance of research being carried out in the area of data mining and machine learning, with the aim of maximising profit. Huang and Song [1] "analyze the transaction database with different data mining methods", so that the potentially huge commercial value can be dug out. Morwitz and Schmittlein [2] apply market segmentation to partition a heterogeneous group of individuals into homogeneous subgroups so that these subgroups can offer market insights and detect common behaviours. Both works have a problem domain is similar to that of this project. Ultimately, the goal is to decrease the rate of cart abandonment and increase sales through insightful interpretation of the cluster formations and making astute business decisions accordingly.

Secondly, we look at the scalability of a distributed system for data processing. The Apache Spark engine is the core engine used for data processing. Apache Spark is a fast and general engine for big data processing, with built-in modules for streaming, Structured Query Language (SQL), machine learning and graph processing. The two Spark modules used in this project are the SQL and machine learning modules. Both the SQL module and the machine learning module allow you to run jobs on large amounts of data and distribute the work across multiple machines. Parallelising the work across multiple machines improves processing time which is a desirable solution in any software system. Not only is the processing time greatly reduced,

[1] Cluster analysis (or clustering) is explained in Section 2.2.

[2] E-commerce is the facilitation of trading in products or services over computer networks such as the Internet.

[3] A journey may be interpreted as the entire customer lifecycle from sitting at a computer or picking up a mobile device to arriving at an online checkout.

but it allows models to be built in near real-time thus giving owners of e-commerce websites the ability to communicate with visitors on their website as they shop live.

The thesis concludes by providing some directions in which future work could be carried out, along with some concluding remarks about the benefits of using machine learning and data mining in business.

1.2 Motivations

The motivation for this project originated in experiences gained whilst carrying out an internship at Altocloud. There is a growing need for companies like Altocloud to harness the data collected from customers to benefit both the company and the end user of the product. Processing large amounts of data real-time can be done using large distributed systems. Another area of interest for some of the leading companies in the software industry is machine learning. Exploring areas such as *data mining, machine learning and distributed computing*[4] is a personal interest of mine.

According to the Baymard Institute (updated May 8, 2015), 68.53% of online shopping carts are abandoned [3]. Abandoned carts can be a result of many factors such as slow Internet connection, loss of interest and unexpected costs like "shipping costs". A comprehensive understanding of customer behaviour as they make a *journey* can contribute to reducing the rate of abandonment by adopting new customer support techniques.

1.3 Research Questions and Objectives

As stated in Section 1.1, the goal of this project is to build a clustering model from historical customer data, and to reduce abandoned online carts and increase sales. The following research questions and research objectives have been identified based on these objectives.

1.3.1 Research Questions

R.1 *Can clustering techniques be used to effectively uncover important groups of customers, given an unlabelled dataset? If so, is it possible to predict the behaviour of future visitors on your website?*

R.2 *Under what circumstances is it necessary to adopt an architecture that provides scalability?*

R.3 *Can a scalable machine learning system, such as the one implemented in this project, be of commercial value to a company like Altocloud?*

[4]Data mining, machine learning and distributed computing are explained in Section 2.1 and 2.6 respectively.

1.3.2 Research Objectives

The following core research objectives were drawn up based on the preceding research questions.

- Design, develop and implement an architecture to retrieve customer data from Altocloud's database, build a machine learning model using a clustering algorithm from the data, and use the model to predict the behaviour of future customers. Carry out experimentation using a series of validation checks in order to analyse validity and further understand the clusters. This will allow us to determine if clustering is effective in this domain and enable us to identify the advantages and/or difficulties associated with such an approach. The completion of this research objective addresses research question **R.1** and is the main focus of this project.

- Design, develop and implement an architecture which can process small datasets or scale adequately to run in production, thus processing much larger datasets. Carry out experimentation using a collection of software applications that have been designed specifically to focus on scalability. This will allow us to investigate if such an architecture is suited to facilitate much larger, real-world datasets. The completion of this research objective addresses research question **R.2**.

- Develop and provide adequate test results to discover the potential benefits of applying machine learning techniques to unlabelled customer data in a real world situation. Carry out a series of tests to determine if clustering customer data can identify important groups of customers, and moreover, insignificant groups of customers. This will allow us to assess the impact of such a system if deployed in production. The completion of this research objective addresses research question **R.3**.

1.4 Thesis Structure

This thesis consists of six sections, the first of which provides an introduction to this final year project. The following is a brief outline of the remaining sections.

Section 2 provides a comprehensive review of the specified research areas. It is introduced with an explanation of the concepts of data mining and machine learning, followed by a description of a variety of clustering techniques. The area of unsupervised clustering is the primary focus of this literature review, specifically focusing on the K-means clustering algorithm. This is followed by providing details about existing software for implementing data mining and machine learning tasks that is both freely and commercially available. Finally, a brief overview of some scalable computing architectures applicable to this task are introduced.

Section 3 begins by providing an overview of the problem domain and the project requirements. The specific software requirements are a combination of feature requests from Altocloud and some necessary architecture requirements. This section concludes with a proposed design

solution which is described in detail. This includes initial and improved control flow diagrams with explanations for each.

Section 4 is concerned with an implementation of the improved design. This section is introduced by providing some insight to the project management tools used in this project. A description of the data retrieval process is followed by a data preprocessing implementation. The section concludes with an implementation of the clustering model.

Finally, Section 5 and Section 6 provide the results and conclusions drawn up from the tests and experiments carried out in this project. Section 5 focuses primarily on the presentation of the results, and provides some analysis on such results. Section 6 includes some concluding remarks and provides some directions in which future work could be carried out.

2 Literature Review

This section of the thesis introduces the core concept of the project; machine learning and data mining. A brief introduction of each will be followed by a breakdown of cluster analysis, focusing specifically on unsupervised clustering. A further in depth analysis of the K-means clustering algorithm is followed by some measures of *goodness of fit*[5] of the algorithm and its suitability to this particular task. Following the explanation of the cluster analysis and the K-means algorithm, is a section on various applications to which cluster analysis was applied. The final two sections cover the topics of open-source software suitable for experimenting and implementing the ideas described, and typical architectural designs for distributed and scalable systems.

2.1 Machine Learning and Data Mining

Machine Learning is the field of scientific study that concentrates on induction algorithms and on other algorithms that can be said to learn [4]. Taking the same approach as Witten and Frank, 2005, Chapter 1.1, it is important to first define what it means to learn. The dictionary defines "to learn" as follows:

> To get knowledge of by study, experience, or being taught;
> To become aware by information or from observation;
> To commit to memory;
> To receive instruction

The first two definitions offer an interpretation of what it means to learn as humans, as we know that computers cannot "study", "observe" or learn from "experience". As early as 1959, Arthur Samuel defined machine learning as a “field of study that gives computers the ability to learn without being explicitly programmed” [5]. The last two definitions are also concerned with human behaviour explicitly, as these functions are trivial tasks for a computer. This reinforces the idea that computers cannot learn without being programmed to do so.

The idea of machine learning now becomes a task of allowing computers to learn by feeding it historical data as an input, and achieving an *output*[6]. In classification learning, the model learns as it is presented with classified data and aims to predict the class of an unseen piece of data. Similarly with regression, the model aims to predict the numeric value of an unseen piece of data through the same process. When clustering, groups of similar characteristics are the desired output. The output is achieved through the processing of data through a specific algorithm, suited to the specific task. Regardless of the type of learning involved, we call the

[5] Measures of goodness of fit typically summarize the discrepancy between observed values and the values expected under the model in question.

[6] The desired output from a machine learning model varies per algorithm. For K-means clustering, clusters are the desired output. See Section 2.3.

thing to be learned the *concept* and the output produced by a learning scheme the *concept description* [6].

The desired output of machine learning is analogous to that of the data mining process; to understand and analyse data more comprehensively. Data mining has become an important process for modern businesses. Data mining is the extraction of implicit, previously unknown, and potentially useful information from data.

> "*Data mining is about solving problems by analyzing data already present in databases ... Data mining is defined as the process of discovering patterns in data... The patterns discovered must be meaningful in that they lead to some advantage, usually an economic advantage. The data is invariably present in substantial quantities.*" - (Witten and Frank [6], Chapter 1.1)

The overall goal of the data mining process is to extract information from a dataset and transform it into an understandable structure for further use [7]. Applying the findings to new data allows you to take advantage of potential opportunities that arise from the new data [6]. Failing to extract meaningful and valuable information from customer data can lead to loss of sales opportunities. Large scale data collection has become the keystone for software companies like Altocloud. Altocloud aims to improve conversions and enhance customer's experiences online and data mining is an important step in this process. Historical customer data now becomes vital information and can help direct agents in taking appropriate action when dealing with new visitors on a website.

2.2 Cluster Analysis

Put simply, when clustering, groups of examples that belong together are sought [6]. There are 3 main types of clustering; unsupervised (traditional), semi-supervised and supervised.

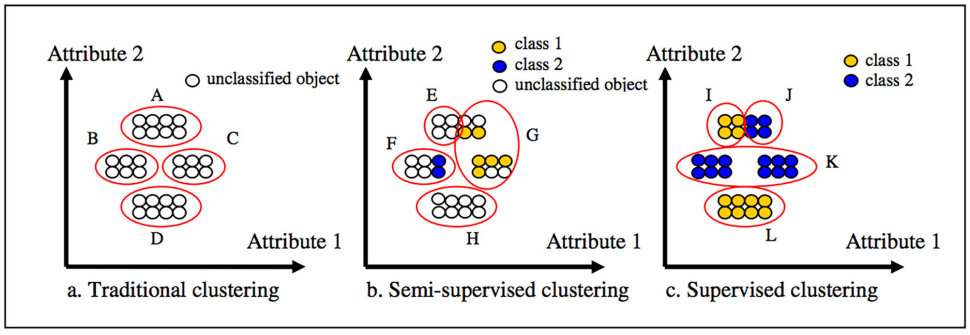

Figure 1: Unsupervised, semi-supervised and supervised clustering.
(Taken from Figure 1 in Zeidat *et al.* [8])

Clustering (or cluster analysis) aims to organize a collection of data items into clusters, such that items within a cluster are more "similar" to each other than they are to items in the other clusters [9]. These datasets can be of two types:

1. Unlabelled - Unlabelled data consists of samples of artifacts that you can obtain relatively easily from the world. Some examples of unlabelled data might include photos, audio recordings, videos, news articles, tweets or x-rays. There is no explanation for each piece of unlabelled data, it just contains the data, and nothing else.

2. Labelled - Labelled data typically takes a set of unlabelled data and augments each piece of that unlabelled data with some meaningful "tag," "label," or "class". For example, labels for the above types of unlabelled data might be whether this photo contains a horse or a cow, which words were used in this audio recording, what type of action is being performed in this video, what the topic of this news article is, what the overall sentiment of this tweet is or whether the dot in this x-ray is a tumor.

After obtaining a labeled dataset, machine learning models can be applied to the data so that new unlabelled data can be presented to the model and a likely label can be guessed or predicted for that piece of unlabelled data. With an unlabelled dataset, the problem involves uncovering natural, inherent groupings within the dataset (see Section 2.2.3).

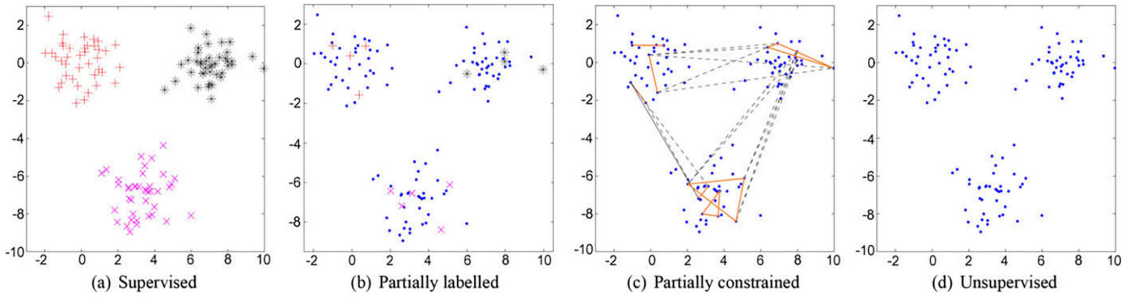

Figure 2: Supervised, partially labelled, partially constrained and unsupervised labelling. (Taken from Figure 1 in Jain [10])

2.2.1 Supervised Clustering

In supervised systems, the data presented to a machine learning algorithm is fully labelled. Therefore, all examples are presented with a classification that the machine is meant to reproduce. According to Eick *et al.* [11], supervised clustering is applied on classified examples with the objective of identifying clusters that have high probability density with respect to a single class. A probability density function[7] is a function of a continuous random variable, whose

[7]Probability density function demonstration available at: `http://stattrek.com/statistics/dictionary.aspx?`.

integral across an interval gives the probability that the value of the variable lies within the same interval.

2.2.2 Semi-Supervised Clustering

In semi-supervised systems, the machine is allowed to additionally take unlabelled data into account. Given two different and independent perspectives on the classification task ... learn a different model for each perspective ... then use each one separately to label the unlabelled data [6].

> "*The idea of semi-supervised clustering is to enhance a clustering algorithm by using side information in the clustering process that usually consists of a "small set" of classified examples; the objective of the clustering process, then, is to optimize class purity (examples with different class labels should belong to different clusters) in addition to the traditional objectives of a clustering algorithm.*" - (Eick *et al.* [11], Section 2)

Semi-supervised systems often outperform their supervised counterparts using the same labelled examples. The reason for this improvement is that more unlabelled data enables the system to model the inherent structure of the data more accurately.

This approach aims to incorporate the classification model with a clustering model by classifying unlabelled data. For example, take a set of unlabelled data and pick out two instances which best represent two classes, eg. positive and negative. An instance can be referred to as a single object of the world from which a model will be learned, or on which a model will be used (e.g., for prediction) [4]. Take an unlabelled instance and place it in a labelled pool that it is most similar to; positive for example. Repeat the process until the pool of unlabelled data has been assigned to either the positive or negative pool. There are many active areas of research in machine learning that are aimed at integrating unlabelled and labeled data to build better and more accurate models. However, the main focus of this thesis lies in the traditional clustering method - unsupervised clustering.

2.2.3 Unsupervised Clustering (Traditional)

Unsupervised systems are not provided any training examples at all. The results of clustering algorithms are data driven, hence more natural and better suited to the underlying structure of the data. This advantage is also its major drawback - without the possibility to tell the machine what to look for (like in classification), it is difficult to judge the quality of the clusters in a conclusive way[8]. But the absence of training example preparation makes the unsupervised paradigm very appealing. Due to the unlabelled nature of the dataset provided by Altocloud, all experimentation and testing is done using unsupervised clustering techniques, namely the K-means clustering algorithm.

[8] Cluster quality is explored in Section 2.3.2.

2.3 K-Means Clustering

K-means is one of the most commonly used clustering algorithms that clusters the data points into a predefined number of clusters. It requires a numeric input set. An example of a clustering problem can be seen below.

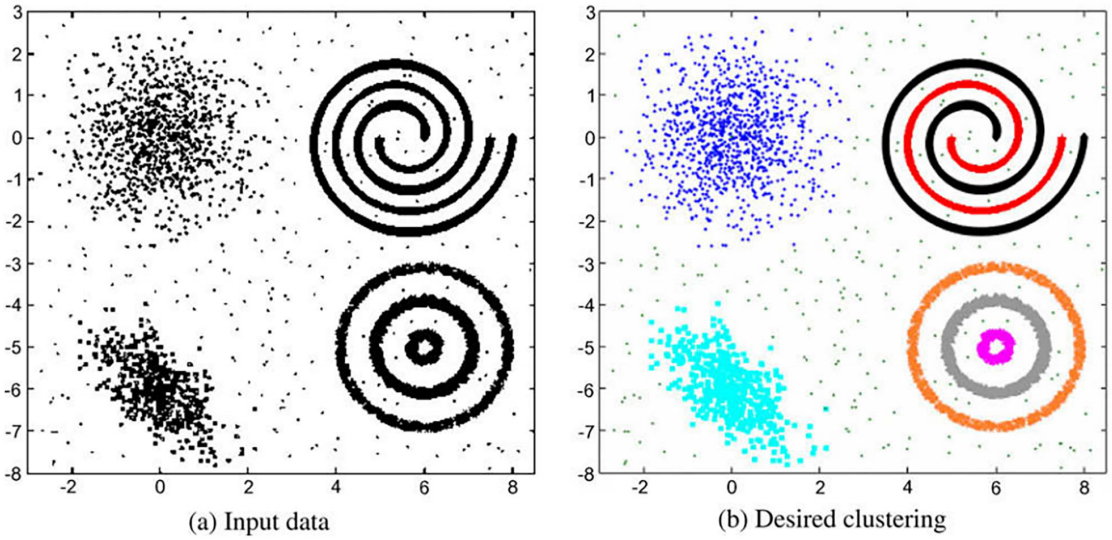

Figure 3: Diversity of clusters.
(Taken from Figure 2 in Jain [10])

The seven clusters in 3(a) (denoted by seven different colors in 3(b)) differ in shape, size, and density. Although these clusters are apparent to a data analyst, none of the available clustering algorithms can detect all these clusters [10].

An obvious drawback of the K-means clustering algorithm is its inability to differentiate between clusters such as those on the right-hand side of Figure 3(a) and 3(b). Despite the underlying assumption that the algorithm will not produce accurate models for this type of data, a machine learning model is built using K-means. The first of the three research questions stated in Section 1.3.1 is concerned with the suitability of clustering to this domain. It is important to note at this point that one of the disadvantages of the K-means clustering has now been identified, but Section 5 highlights the effectiveness of this algorithm and its suitability to such a task. The following derivation of the K-means algorithm is from a paper by Anil K. Jain [10] on data clustering, K-means and related topics.

Let $X = \{x_i\}$, $i = 1, ..., n$ be the set of n d-dimensional points to be clustered into a set of K clusters, $C = \{c_k, k = 1, ..., k\}$. K-means algorithm finds a partition such that the squared error between the empirical mean of a cluster and the points in the cluster is minimized. Let u_k be

the mean of cluster c_k. The squared error between u_k and the points in cluster c_k is defined as

$$J(c_k) = \sum_{x_i \in c_k} ||x_i - u_k||^2$$

The goal of K-means is to minimize the sum of the squared error over all K clusters,

$$J(C) = \sum_{k=1}^{K} \sum_{x_i \in c_k} ||x_i - u_k||^2 \tag{1}$$

where $||x_i - u_k||^2$ is the Euclidean[9] distance between x_i and u_k.

The following figure demonstrates the typical process of a K-means algorithm for an output of 3 clusters. The steps taken by a K-means algorithm are as follows [12]:

1. Select an initial partition with K clusters; repeat steps 2 and 3 until cluster membership stabilizes.
2. Generate a new partition by assigning each pattern to its closest cluster center.
3. Compute new cluster centers.

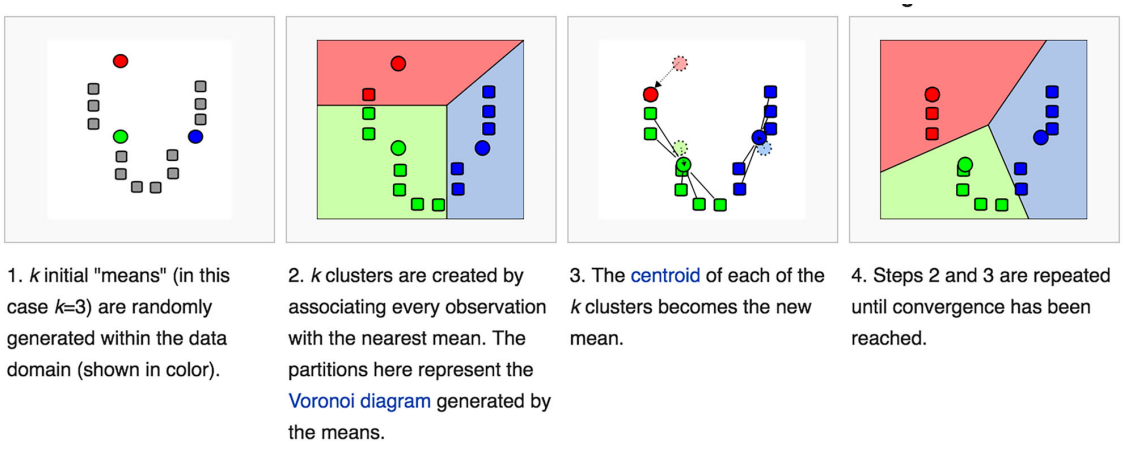

Figure 4: Demonstration of the standard algorithm.
(Taken from a K-means clustering example at:
`https://en.wikipedia.org/wiki/K-means_clustering`

[9] Euclidean Distance metric is described in more detail in Section 2.3.1.

2.3.1 K-Means Algorithm Parameters

The K-means algorithm requires three user-specified parameters: number of clusters K, cluster initialization, and distance metric [10]. K-means initialises with a partition of K clusters and assigns each instance from the dataset to a cluster so as to reduce the squared error. Since the squared error will always decrease as the number of clusters K increases (with $J(C) = 0$ when $K = n$), it can be minimized only for a fixed number of clusters. In other words, each instance of the dataset will lie within its own unique cluster if an initial number of clusters is not defined.

K-means is used with the Euclidean Distance metric for computing the distance between points and cluster centers. As a result, K-means finds spherical or ball-shaped clusters in data and may suffer when attempting to find structures such as those seen in Figure 3. An alternative version of K-means using the Itakura Saito distance has been used for vector quantization in speech processing [13]. Another distance metric used which is suitable for binary values is the Hamming Distance metric, which is *not* commonly applied to the K-means algorithm. A Hamming Distance metric is applicable to datasets where the difference between two binary strings needs to be calculated. The dataset used in this project contained primarily binary values, but the Hamming Distance metric will not calculate distances between continuous variables, which also exist in the dataset. Thus, the Euclidean Distance metric was used to encompass both discrete binary values and continuous values.

The Euclidean Distance between points p and q is the length of the line segment connecting them ($\overline{pq}$). If $p = (p_1, p_2, ..., p_n)$ and $q = (q_1, q_2, ..., q_n)$ are two points in Euclidean n-space, the Euclidean Distance is given by the equation:

$$d(p,q) = d(q,p) = \sqrt{(\sum_{i=1}^{n}(q_i - p_i)^2)} \tag{2}$$

The Euclidean Distance metric formula was taken from a paper called Euclidean Distance Mapping by Per-Erik Danielsson [14].

Figure 5 below depicts another illustration of the K-means algorithm. 5(a) shows a two-dimensional input data with three clusters; 5(b) three seed points selected as cluster centers and initial assignment of the data points to clusters; 5(c) and 5(d) intermediate iterations updating cluster labels and their centers; 5(e) final clustering obtained by K-means algorithm at convergence [10].

2.3.2 Cluster Validity Analysis

As briefly mentioned in Section 2.2, cluster validity analysis is an important step when dealing with an unlabelled dataset. Unlabelled datasets do not offer any *ground truth*[10] and hence it is often difficult to accurately interpret the clusters. Despite this, there are some measures which can help validate and visualise the cluster assignment. Some of the techniques discussed in

[10]Ground truth refers to the accuracy of the training set's classification for supervised learning techniques.

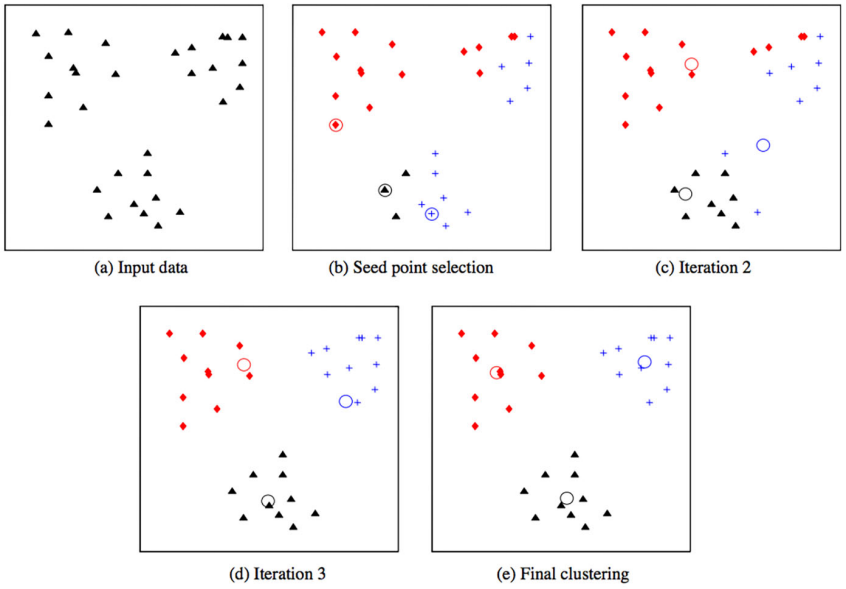

Figure 5: Demonstration of the k-means algorithm with 3 clusters on a 2-dimensional data.
(Taken from Figure 4 in Jain [10])

this section include *Variance*, *Silhouette Scores* and *Principal Component Analysis*. Clustering algorithms tend to find clusters in the data irrespective of whether or not any clusters are present [10]. Therefore, it is necessary to validate clustering.

According to [15] there are three types of validation procedures:

1. *External* validation can only be performed when prior knowledge of the problem is available. The prior knowledge may concern general characteristics of the clusters (e.g. expected compactness[11]) or relations between specific items (e.g. items A and B should belong to the same cluster and item C to a different one).
2. *Internal* validation is based on an evaluation of the "agreement" between the data and the partition.
3. *Relative* comparisons are usually the main application of the indices defined for the internal validation. Such comparisons are often employed for selecting good values for important parameters, such as the number of clusters.

[11] Since the K-means method aims to minimize the sum of squared distances from all points to their cluster centres, this should result in compact clusters [16].

With these validation processes in mind, some standard measures of spread in a dataset are looked at first: standard deviation and variance. These two measures of spread are introduced with the example below taken from *A tutorial on Principal Components Analysis* [17]. Take a set of numbers $X = \{1, 2, 4, 6, 12, 15, 25, 45, 68, 67, 65, 98\}$. The mean, or average, of the set is given by,

$$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n}$$

The standard deviation of the dataset measures how spread out the data is from the mean and is given by,

$$s = \sqrt{\frac{\sum_{i=1}^{n} (X_i - \overline{X})^2}{n-1}}$$

The variance is another measure of the spread of the data. It is given by,

$$s^2 = \frac{\sum_{i=1}^{n} (X_i - \overline{X})^2}{n-1}$$

The variance is simply the standard deviation squared in both the symbol (s^2) and the formula. Variance can be used as a measure of cluster compactness. Applying the variance formula to a set of clusters is covered in further detail below (See Formula 5).

Different initialisation settings can often lead to different final clusters because K-means converges to a local minima. One way to overcome the local minima is to run the K-means algorithm, for a given K, with multiple different initial partitions and choose the partition with the smallest squared error [10] or lowest silhouette score.

To determine whether the clusters are compact we use the *intra-cluster* distance measure, which is simply the distance between a point and its cluster centre and we take the average of all of these distances [16], defined as,

$$Intra = \frac{1}{N} \sum_{i=1}^{K} \sum_{x \in C_i} ||x - z_i||^2, \tag{3}$$

where N is the number of instances in the dataset, K is the number of clusters, and z_i is the cluster centre of cluster Ci.

Another measure that can be taken is the *inter-cluster* distance, or the distance between clusters, which we want to be as big as possible. Maximising the distance between clusters develops distinct, definite clusters. We calculate this as the distance between cluster centres, and take the minimum of this value [16], defined as

$$Inter = \min(||x - z_i||^2), i = \{1, 2, ..., K-1\}, j = \{i+1, ..., K\} \tag{4}$$

Taking a ratio of the intra-cluster and inter-cluster values, the validity then becomes,

$$Validity = \frac{Intra}{Inter}$$

Since the K-means algorithm aims to minimize the average intra-cluster distance, it is most likely that the cluster having maximum variance will be separated by the K-means procedure when the number of clusters is increased [16]. The variance for cluster C_i then becomes,

$$\sigma_{ij}^2 = \frac{1}{N}\sum_{x \in C_i}(x - z_i), i = \{1, 2, ..., K\}, j = \{1, 2, ..., A\}, \tag{5}$$

where N_i is the number of instances in cluster C_i, A is the number of attributes in each instance, and x is the vector representing attributes as $\{x_1, x_2, ..., x_a\}$ for each instance. Once convergence has been achieved, calculating the overall variance for the clusters can give a good indication of the differentiation between clusters.

The silhouette score measures how well an instance i matches the clustering at hand [18]. The following derivation is from Peter J. Rousseeuw's paper on Silhouettes [18].

Firstly, take any instance i in the dataset, and denote by A the cluster to which it has been assigned. When cluster A contains other objects apart from i, then we can compute,

$$a(i) = \text{average dissimilarity of } i \text{ to all other objects of A}$$

$$d(i, B) = \text{average dissimilarity of } i \text{ to all objects of } B \text{ (alternative cluster)}$$

$$b(i) = \min d(i, B)$$

Hence, the silhouette score given by the following equation:

$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if} a(i) > b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases} \tag{6}$$

An alternative version of the formula is:

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$$

When cluster A contains only a single object it is unclear how a(i) should be defined, and then we simply set s(i) equal to zero. From the above definition we see that,

$$-1 =< s(i) =< 1,$$

for each object i. The Silhouette Score can offer a measure of how easily the cluster was formed. Although calculations such as variance and Silhouette Scores perform as good indicators of how "valid" a cluster may be, they do not offer any form of visualisation on the dataset. To build a better understanding of the dataset and to attempt to visualise some natural groupings in the data, Principal Component Analysis (PCA) can be used.

What is PCA?

> "(PCA) is a way of identifying patterns in data, and expressing the data in such a way as to highlight their similarities and differences. Since patterns in data can be hard to find in data of high dimension, where the luxury of graphical representation is not available, PCA is a powerful tool for analysing data". - (Smith [17], Chapter 3)

Essentially, PCA attempts to reduce the dimensionality of a dataset by maintaining high variance. It does so by splitting the dataset through its line of most variance repeatedly until the number of dimensions required has been determined. The following illustration shows PCA reducing a 3-dimensional dataset to a 2-dimensional dataset.

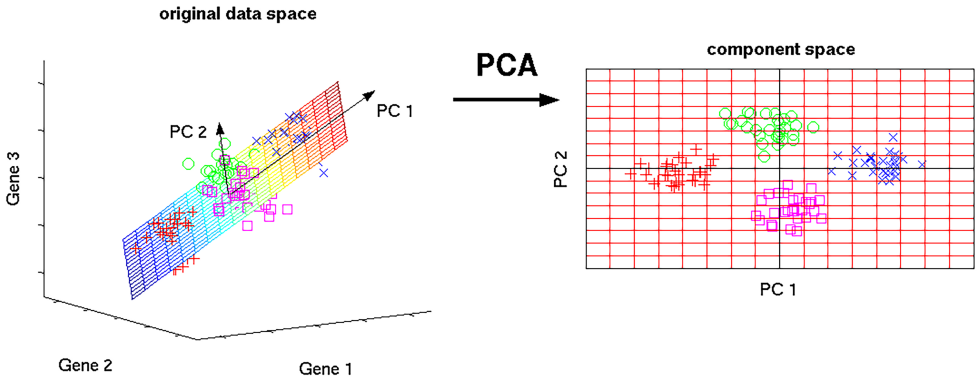

Figure 6: PCA on a 3-dimensional dataset.
(Taken from a PCA walk through at:
`http://www.nlpca.org/pca_principal_component_analysis.html`)

PC (Principal Component) 1 and PC 2 from Figure 6 become the new axes on which the 2-dimensional data (with the highest variance) lies. This technique is implemented in Section 4.4.1 to visualise the clusters within the dataset. It is important to note that PCA is highly affected by outliers and this was taken into consideration when preparing the dataset (see Section 4.3)

2.4 Applications of Clustering

In the following subsections, a variety of specific applications for clustering will be described. Clustering applications explored in this section are representative of some of the key constituents of the clustering model developed in this project. Section 3.1 introduces the problem domain in which applications, such as those described below, are used to define the domain. Cluster analysis has been applied to a broad range of areas but this section focuses primarily on applications relating to areas such as computing, human behaviour and e-commerce.

2.4.1 Unsupervised Clustering of Web Session Data

Stevanovic *et al.* [19] examine the use of two unsupervised neural network (NN) learning algorithms for the purpose web-log analysis. The aim of their research is to investigate the behaviour of malicious and non-malicious website users, and separate them accordingly. The detection of web crawlers was evaluated with the following two unsupervised neural network algorithms:

- SOM
- Modified ART2.

The SOM algorithm is very well known for its ability to produce natural clustering, i.e. clustering that is robust to statistical anomalies [19]. On the other hand, according to Stevanovic *et al.* [19], ART2 has a unique ability to identify statistically underrepresented but significant clusters, and is greatly suited for imbalanced datasets. The domain of their research is similar to that of this project whereby data consisting of link-traversal information is used to form clusters.

Their studies show that the neural networks managed to form separate clusters containing (a) malicious and (b) non-malicious web users based on their browsing behaviour. And, while human visitors tend to follow rather consistent browsing patterns, malicious web crawlers exhibit a range of browsing strategies. The results show that nearly 8% of web crawlers exhibit very much "human-like" browsing behaviour [19]. This study shows that clustering techniques can be used to identify groups of web users who share similar characteristics based on behavioural information collected while they navigate websites.

2.4.2 Classifying the User Intent of Web Queries

Web search engines are frequently used by people to locate information on the Internet. However, as Kathuria *et al.* [20] mention, not all queries have an informational goal. Instead of information, some people may be looking for specific web sites or web services. Their paper, *Classifying the user intent of web queries using K-means clustering*, aims to focus on automatically classifying the different user intents behind web queries.

Their research included 130,000 web search engine queries which were categorized as informational, navigational, or transactional using a K-means clustering approach based on a variety of query traits. The research findings show that more than 75 percent of web queries (clustered into eight classifications) are informational in nature, with about 12 percent each for navigational and transactional [20]. Interestingly, the results also show that web queries fall into eight primary clusters; six informational, and one each of primarily transactional and navigational.

This research shows that the K-means clustering algorithm can in fact detect the intentions of web users based on search terms. Like in the studies conducted by Stevanovic *et al.* [19], data collected online has been used to form significant groups of web users through clustering

techniques. Experiments, such as those carried out by Kathuria *et al.* [20], can be adapted to focus on data collected from a single website, rather than an entire search engine. The following application by Morwitz and Schmittlein [2] focuses on such experiments.

2.4.3 Segmentation to Improve Sales Forecasts

The following research was carried out to investigate whether the use of *market segmentation*[12] can improve the accuracy of sales forecasts based on stated purchase intent.

Morwitz and Schmittlein [2] objective of segmentation is to partition a heterogeneous group of individuals into homogeneous subgroups. Their ideology is equally applicable to the experiments carried out here where "homogeneous subgroups" are the desired output from the clusters. Their approach included the identification of "homogeneous segments based on demographic profiles and product usage variable". Morwitz and Schmittlein [2] show that more accurate intentions-based sales forecasts can be obtained by first segmenting consumers on the basis of other objective data such as demographic or product use data.

2.4.4 Cluster Analysis on E-commerce Transactions

Clustering transactional behaviour data from a website can give businesses further understanding of the behaviour and mentality of customers as they navigate through an e-commerce website. The final application discussed in this section is the application of cluster analysis to e-commerce transactions (based on K-means clustering). Huang and Song [1] "analyze the transaction database with different data mining methods", so that the potentially huge commercial value can be dug out. Their research is introduced by providing the following example:

> *We can analyze the consumers' buying habits through discovering relationships of different goods put into the shopping basket by consumers when shopping, which can help shopkeepers to know what goods are frequently purchased by consumers at the same time, so that they can develop better marketing strategy.*

This project focuses on the application of a K-means clustering algorithm, which is the main focus of both Kathuria *et al.* [20] *and* Huang and Song [1]. Huang and Song [1] state that the collected data is preprocessed before the experiment to achieve the transformation from non-numeric data to numeric data. This is due to K-means excellent clustering ability on numeric values. Similar techniques were carried out in this project (see Section 4.3).

[12] Market segmentation is the process of defining and subdividing a large homogenous market into clearly identifiable segments having similar needs, wants, or demand characteristics. For more see: `http://www.businessdictionary.com/definition/market-segmentation.html`.

Their data consisted of the following three main attributes:

- The product attributes of mobile phones
- The attributes of the seller
- The sales status

Their experimentation produced the following results:

- In comparison with single-pass clustering algorithm, K-means clustering algorithm has a very high inter-class dissimilarity and intra-class similarity when performing clustering analysis of e-commerce transaction
- The K-means clustering algorithm has an obvious superiority in dealing with high-dimensional data
- the K-means clustering algorithm puts up a high efficiency and strong elasticity.

The results from their experiments can be used to guide in the design process of this project (see Section 3.3)

2.5 Machine Learning and Data Mining Software

There are many software packages available for carrying out clustering tasks. Some of these are open source and freely available while others are commercially available. Some examples of software used, or considered for use, in this project are detailed below. All testing programs were built on Jupyter's iPython notebooks[13] (see Section 2.5.4 for more on Jupyter notebooks.

2.5.1 Waikato Environment for Knowledge Analysis (Weka)

We first look at the Weka open-source software. Weka is a collection of machine learning algorithms for carrying out data mining tasks. The software, developed at the University of Waikato, contains tools for classification, regression, clustering, data pre-processing, association rules and visualisation. The main user interface contains several different panels to give access to the main components of the workbench.

The preprocess panel allows you to import data and apply preprocessing filters to the dataset if necessary. Weka is compatible with file formats such as Comma Separated Value (CSV) and Attribute-Relation File Format (ARFF). The *classify* panel allows the user to select either a classification or regression algorithm to be applied to the imported data. The *cluster* panel gives access to the clustering techniques such as the K-means algorithm. The next panel is called *associate*. This panel includes facilities to create association rule learners that attempt to

[13] IPython provides a rich architecture for interactive computing. See more at `https://ipython.org/`.

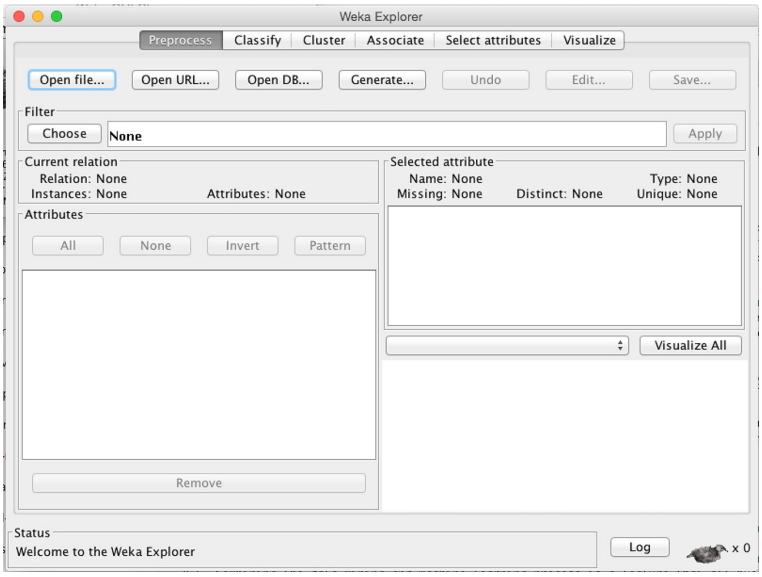

Figure 7: Weka graphical user interface

find relationships between attributes in a given dataset. This panel is followed by the select *attributes* panel used to identify the most predictable attributes in a dataset. Finally, the *visualise* panel has a display of several different scatter plot visualisations. The only workbench used in this project was the *Explorer* interface (See Figure 7). The Weka workbench also contains three other interfaces, *Experimenter*, *KnowlegeFlow* and *SimpleCLI*, which were not used throughout this project. More information about the Weka software package is available in "Data Mining: Practical Machine Learning Tools and Techniques" [6].

2.5.2 Scikit Learn

Scikit Learn is another collection of simple and efficient tools for data mining and data analysis. It is built on Python[14] modules NumPy, SciPy, and matplotlib. Scikit Learn is open source, but also has a commercial version available for those who require additional features and support. Scikit Learn contains facilities to build, test, and analyse machine learning models.

2.5.3 Apache Spark

Introducing Apache Spark

Apache Spark is an open source big data processing framework built with the aim of increased speed, ease of use, and sophisticated analytics. It was originally developed in 2009 in UC Berkeley's AMPLab, and open sourced in 2010 as an Apache project. Spark is continuously

[14]Python is a high-level, general-purpose, interpreted, dynamic programming language.

compared to other big data and *MapReduce* technologies like Hadoop and Storm. MapReduce is a programming model and an associated implementation for processing and generating large datasets with a parallel, distributed algorithm on a cluster [21].

A brief introduction on the history of Apache Spark is seen below, which was taken from a course by *edX*[15] called "BerkeleyX: CS100.1x Introduction to Big Data with Apache Spark",

- 1956-1979: Stanford, MIT, CMU, and other universities develop set/list operations in LISP, Prolog, and other languages for parallel processing
  (see `http://www-formal.stanford.edu/jmc/history/lisp/lisp.html`).
- Circa 2004: Google: MapReduce: Simplified Data Processing on Large Clusters by Jeffrey Dean and Sanjay Ghemawat.
  (see `http://research.google.com/archive/mapreduce.html`).
- Circa 2006: Apache Hadoop, originating from the Yahoo!'s Nutch Project Doug Cutting
  (see `https://yahooresearch.tumblr.com/post/139436148571/yahoos-new-research-model`).
- Circa 2008: Yahoo! web scale search indexing - Hadoop Summit, Hadoop User Group
  (see `https://developer.yahoo.com/hadoop/`).
- Circa 2009: Cloud computing with Amazon Elastic MapReduce (EMR), a Hadoop version modified for Amazon Elastic Cloud Computing (EC2) and Amazon Simple Storage System (S3), including support for Apache Hive and Pig.
  (see `http://aws.amazon.com/elasticmapreduce/` for more on AWS EMR).

Some of the reasons why Spark is preferred over its counterparts, such as Hadoop, are due to its ability to store data *in-memory*. Figure 8 below compares two different iterative machine learning algorithms, K-means clustering and logistic regression. On the x-axis, we have the run time of these algorithms for the different environments in seconds. For K-means clustering, Hadoop Map Reduce takes 121 seconds to complete that machine learning algorithm, whereas Spark takes 4.1 seconds. For logistic regression, it takes Map Reduce 80 seconds to complete the execution, whereas Spark completes execution in less than 1 second.

Both Spark and Hadoop were given the task of running the same algorithm on the same dataset, but clearly Spark outperforms the Hadoop MapReduce implementation substantially. In MapReduce, your only option for storage is the disk. Having access to in-memory storage increases processing speeds considerably and Spark supports both in-memory storage and disk storage. Spark will attempt to use as much in-memory storage as possible before spilling into the disk. In terms of operations, MapReduce only supports the map and reduce operations, whereas Spark supports Map Reduce, join, sample, and many other operations. The execution model for Map Reduce is limited to batch, whereas Spark supports execution models including batch, interactive, and streaming.

[15] edX hosts online university-level courses in a wide range of disciplines to a worldwide student body.

Figure 8: Two iterative machine learning algorithms.
(Taken from graphs generated at:
`https://courses.edx.org/courses/BerkeleyX/CS100.1x/1T2015/`)

Figure 9 displays results from the Daytona Gray 100 terabyte (TB) sort benchmark. Spark set the record for the 100TB sort benchmark in 2014 by reducing the time it took to perform that benchmark (set by Hadoop) from 72 minutes to 23 minutes. 1,000TB worth of data (1 trillion records) were sorted in only 234 minutes, running on 190 Amazon EC2 instances.

	Hadoop MR Record	**Spark Record**	**Spark 1 PB**
Data Size	102.5 TB	100 TB	1000 TB
Elapsed Time	72 mins	23 mins	234 mins
# Nodes	2100	206	190
# Cores	50400 physical	6592 virtualized	6080 virtualized
Cluster disk throughput	3150 GB/s (est.)	618 GB/s	570 GB/s
Sort Benchmark Daytona Rules	Yes	Yes	No
Network	dedicated data center, 10Gbps	virtualized (EC2) 10Gbps network	virtualized (EC2) 10Gbps network
Sort rate	**1.42 TB/min**	**4.27 TB/min**	**4.27 TB/min**
Sort rate/node	**0.67 GB/min**	**20.7 GB/min**	**22.5 GB/min**

Figure 9: First public cloud petabyte sort. See `https://databricks.com/blog/2014/11/05/` for more)

Spark Ecosystem

Spark also supports *lazy evaluation*[16] of big data queries, which helps with optimization of the steps in data processing workflows. It provides a higher level API which is easy to use and improves developer productivity through its simplicity. Spark lets you quickly write applications in Java, Scala, Python, Clojure or R. In addition to the Spark Core API, there are libraries that provide additional capabilities in Big Data analytics and Machine Learning areas. These libraries include:

- Spark Streaming - process real-time streaming data
- Spark SQL - run SQL like queries on Spark data
- Spark MLlib - scalable machine learning library
- Spark GraphX - for graphs and graph-parallel computation.

Resilient Distributed Datasets (RDD) are the core concept in the Spark framework. An RDD can be considered similar to a table in a database and can contain any type of data. Spark stores data in RDD on different partitions allowing for quicker computation and processing optimisation. RDD supports two types of operations:

- Transformation - return a new RDD without evaluating any results
- Action - evaluates and returns a new value for the RDD

An example of RDDs being stored 100% in-memory (and cached) can be seen below in Figure 10. This screenshot was taken from the Spark Web Console which is described in more detail in the paragraph following.

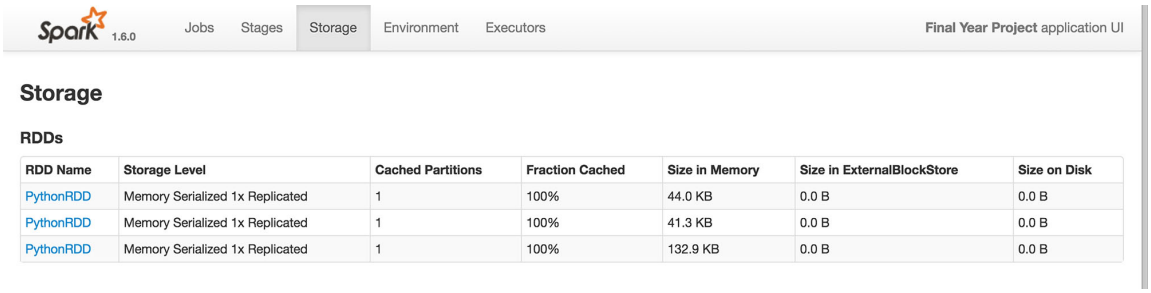

Figure 10: Spark using in-memory storage for RDD caching.

[16]Lazy evaluation is a strategy which delays the evaluation of an expression until its value is needed.

Graphical User Interface
While Spark is running, you can view the Spark Jobs and other interesting statistics through the Spark Web Console via the following URL:

```
http://localhost:4040
```

A typical Spark interface looks like Figure 11. The Spark Web Console contains important graphical statistics such as your job timeline, job execution times, available thread executors and general environment information.

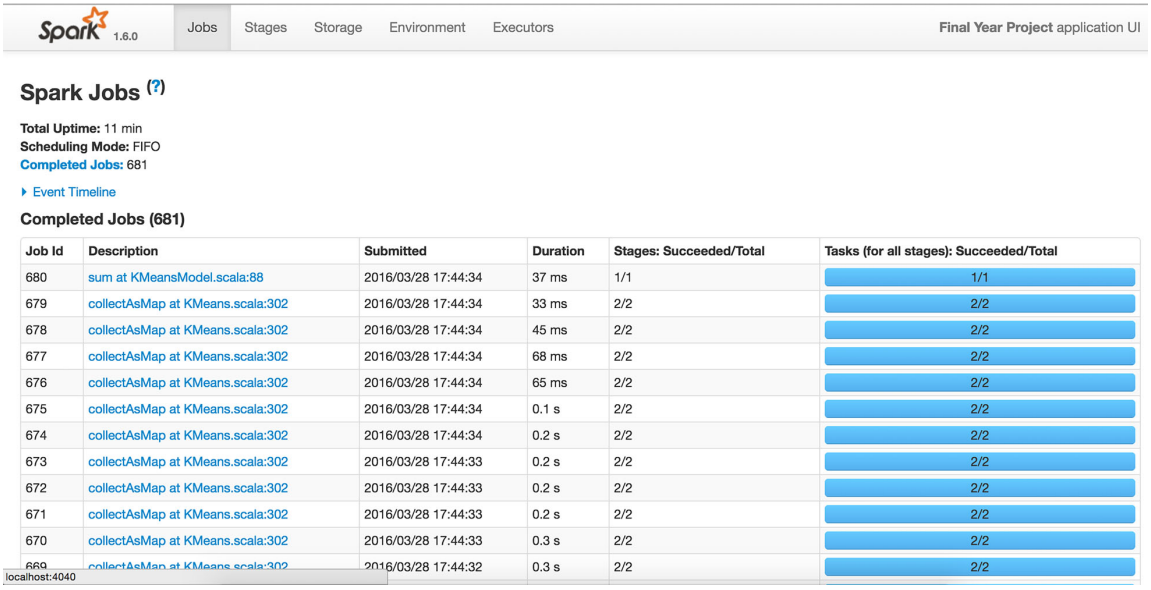

Figure 11: Spark Web Console.

2.5.4 Jupyter Notebooks

The Jupyter Notebook is a web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. The notebook is stateful, which means that variables and their values are retained until the notebook is detached (in Databrick's Cloud) or the kernel is restarted. Databricks, founded by the creators of Apache Spark, makes big data analytics simple through an integrated workspace hosted as a service in the cloud. Uses of the notebook include:

- support for over 40 programming languages
- data cleaning and transformation
- numerical simulation

- statistical modelling
- machine learning.

and much more.

A notebook is comprised of a linear sequence of cells. These cells can contain either *markdown*[17] or code, but we don't mix both in one cell. When a markdown cell is executed it renders formatted text, images, and links just like HTML in a normal web page. Python code cells allow you to execute arbitrary Python commands just like in any Python shell. The Python (version 3.0) programming language was the primary language used for all testing. Each chunk of code can be tested and inspected independently of preceding code. This makes debugging the code much more efficient as you only have to compile sections of the code that need to be adjusted or tweaked. The notebook allows you to build a sequential workflow and creating presentations becomes easy. Code and commentary, coding results, text, graphs and tables can all lie within a single notebook. Figure 12 below contains a typical Python Notebook running Spark.

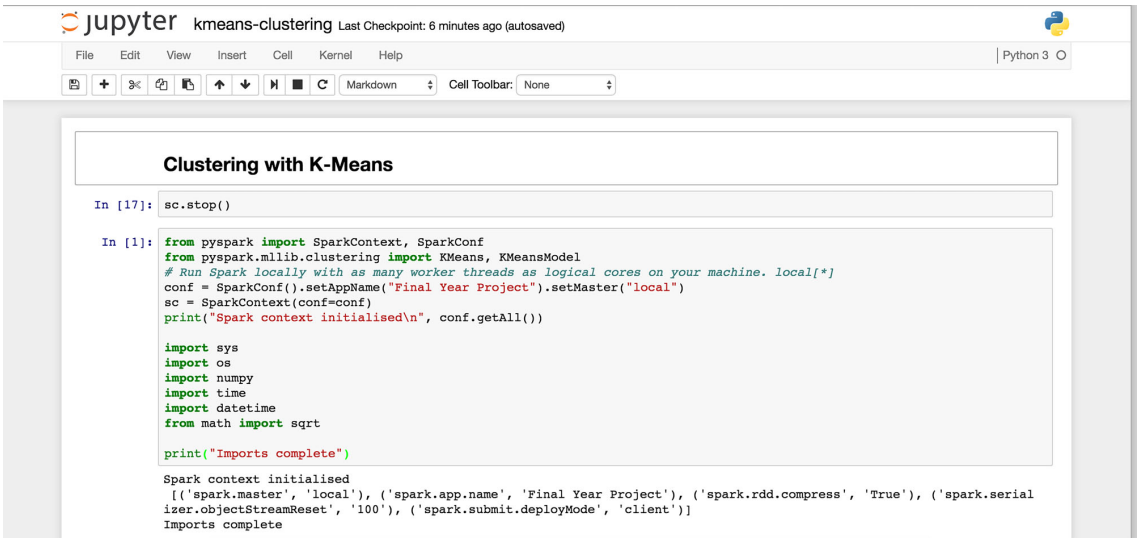

Figure 12: iPython notebook running Spark.

2.6 Distributed Computing and Scalability

An important feature of any data processing application is its ability to scale. As discussed in Section 2.1, large scale data collection and processing has become a common goal of many businesses. For example, Datalogix is a US-based data mining company that collects and analyses information about shoppers. Companies like Facebook use Datalogix services to track

[17]Markdown is a markup language with plain text formatting syntax designed so that it can be converted to HTML and many other formats.

consumers' retail purchases and gain valuable insights into the behaviour of visitors on their websites. This section focuses specifically on how a Spark based system can be built to process much larger datasets, in parallel.

2.6.1 Spark Context

Some further in depth analysis of the Spark architecture will be introduced in this section.

In Spark, communication occurs between a driver and executors. The driver has Spark Jobs[18] that it needs to run and these jobs are split into tasks that are submitted to the executors for completion. The results from these tasks are delivered back to the driver. As mentioned in Section 2.5.4, python code, without any Spark functions, can be executed via cells. When using Databricks Cloud this code gets executed in the Spark driver's Java Virtual Machine (JVM) and not in an executor's JVM, and when using an iPython notebook it is executed within the kernel associated with the notebook. Since no Spark functionality is actually being used, no tasks are launched on the executors.

When running Spark, you start a new Spark application by creating a SparkContext. When the SparkContext is created, it asks the master for the use of some worker cores. The master sets these cores aside for your application; they won't be used for other applications. Figure 13 below shows an example cluster, where the cores allocated for a Spark application are outlined in purple.

At a high level, every Spark application consists of a driver program that launches various parallel operations on executor JVMs running in a cluster, or even locally on the same machine. In the Databricks Cloud, *Databricks Shell* is the driver program. When running locally, *PySparkShell* is the driver program. Driver programs access Spark through a SparkContext object, which represents a connection to a computing cluster.

2.6.2 Scaling the System

In Spark, datasets (RDDs) are represented as a list of entries, where the list is broken up into many different partitions that are each stored on a different machine (or worker). Each partition holds a unique subset of the entries in the list. The following example demonstrates how an RDD can be processed (shuffled, searched, indexed, mapped, etc.) across multiple machines using the Spark Context. Take a list of 30 numbers $X = \{1, 2, ..., 30\}$ and partition the list into four partitions. Note in Figure 14 below, all subsets of the RDD are stored in-memory as discussed in Section 2.5.3. Figure 14 demonstrates the functionality and benefits of processing data across multiple workers instead of a single core within a single worker. Clearly processing times decrease significantly and data can be processed in near real-time.

[18] A Spark Job is a parallel computation consisting of multiple tasks that gets spawned in response to a Spark action (e.g. save, collect).

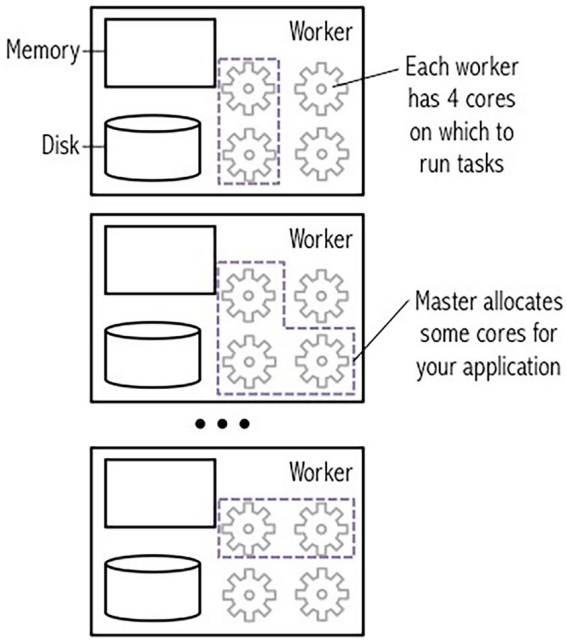

Figure 13: Core allocation for a Spark application.
(Taken from core allocation example at:
`http://spark-mooc.github.io/web-assets/images/executors.jpg`)

2.6.3 Cluster Management

Spark runs on top of existing Hadoop Distributed File System (HDFS) infrastructure and provides support for deploying Spark applications in an existing Hadoop v1 cluster or Hadoop v2 YARN cluster or even Apache Mesos. Apache Mesos is a cluster manager that provides efficient resource isolation and sharing across distributed applications or frameworks [22]. Mesos is an open source software originally developed at the University of California at Berkeley. Mesos makes it easier to deploy and manage applications in large-scale clustered environments and can run many applications on a dynamically shared pool of nodes [22]. Combining Apache Spark and Apache Mesos allows you to create a scalable, distributed, fault-tolerant system, capable of running applications in production.

2.6.4 Spark Environment Used

Question **R.2** from Section 1.3.1 inquires the necessity of a scalable system. When dealing with relatively small datasets (approximately 4000 instances for example), is a massively scalable architecture required? As mentioned in Section 2.6 above, scalable architectures can be built from technologies such as Apache Spark and Apache Mesos ***if*** required. For the purpose of this

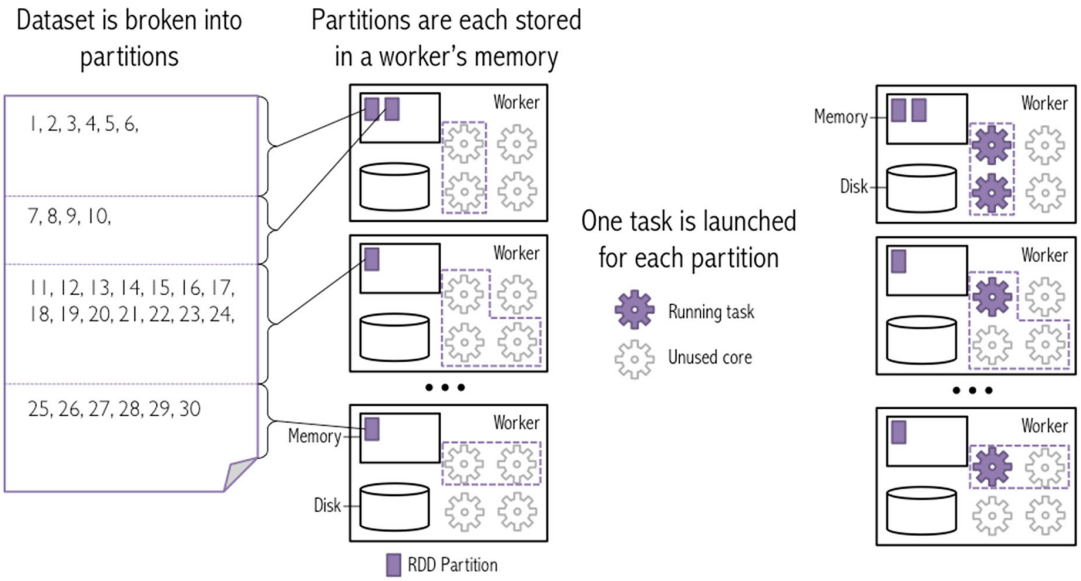

Figure 14: Partition dataset across multiple machines.
(Taken from core allocation example at: `http://spark-mooc.github.io/web-assets/images/executors.png(&)partitions.png`

project, all Spark Jobs were run locally on a MacBook Pro. The laptop contains 4 cores, memory and disk, and can simulate a single worker in a cluster. This is sufficient for all experimentation and the machine described by Figure 15 can replicate accurately the behaviour of a segment of a larger distributed system.

Figure 15: MacBook Specs.

3 Project Requirements and Design Architecture

The following section introduces the problem domain. A brief description of the project requirements is followed by a proposed design solution. The research questions are considered when such design solution is developed. This section concludes with an implementation of the proposed design solution.

3.1 Problem Domain

This subsection will define the problem domain by aggregating the domains of related applications. These domains are introduced through various applications of clustering (see Section 2.4) and in research areas associated with the research questions that define this project (see Section 1.3.1). The mains areas that define the problem domain include:

- Finding valuable insight in unlabelled data i.e., the customer dataset
- Cluster analysis
- Applying machine learning and data mining techniques to acquire commercial value and maximise profit
- Scalable and distributed system architectures

This project focuses on the combination of these areas. Altocloud, like many other *SaaS*[19] companies, contain an abundance of customer data. Due to the nature of SaaS companies, vast quantities of information is gathered through subscriptions, support tickets, customer success meetings, etc. Making sense of all the data, not to mention analysing that data in meaningful ways to drive decision-making, is not a simple task. Many businesses neglect this process and the extremely valuable customer data goes to waste. The more you understand about your customers, the better you are able to cater to their unique needs. Breaking the problem domain down into smaller segments appears to be the most logical approach and the first problem to tackle is the one aforementioned; the customer data collected by companies needs to be analysed and used to their advantage.

Cluster analysis is one particular method of machine learning that has been used frequently in the area of market insight and market segmentation. Section 2.4 introduces some relevant applications. The next challenge now involves applying clustering techniques to the processed and prepared data. Data collected by companies such as Altocloud may contain information on billions of individuals each month. Naturally, volumes of data this large cannot be processed and interpreted by humans. Many software applications have been designed to cater specifically to big-data processing problems like this one. Incorporating the use of these facilities is necessary in this project. The K-means clustering algorithm, as previously mentioned by Huang

[19] Software as a service (SaaS) is a software licensing and delivery model in which software is licensed on a subscription basis and is centrally hosted.

and Song [1], is effective for processing high dimensional data and is very well suited to this clustering task. But what are the benefits of this system?

If the correct measures are taken, the combination of data mining and machine learning can become a very valuable process for businesses. For instance, take large amounts of noisy customer data, process it and analyse it, and apply clustering algorithms to it to provide some market segmentation. Research such as that carried out by Morwitz and Schmittlein [2] has proven that market segmentation can produce a commercial advantage[20]. Dividing the infinite pool of potential clients into smaller, better defined and more manageable groups can allow business owners to make more astute business decisions, with less risk involved. Now that it has been seen that data mining and machine learning techniques can be used to provide commercial value, the final challenge encountered is scalability. Is it possible to scale this system indefinitely?

The answer to this question is provided by a proposed design solution to a scalable system which is discussed in more detail in Section 3.3.2. Furthermore, the following subsections include some project requirements, a design solution to solve the problems mentioned above, and a physical implementation of the proposed solution.

3.2 Project Requirements

Before introducing the project requirements, some marketing terminologies used frequently by Altocloud, and throughout the remainder of this thesis, are defined:

- **Persona** - A persona is a representation of a customer based on market research and real data about observed behaviour on a website. An example of a persona may include "*Viewed Checkout*" or "*High Value Cart*". A customer who has reached an online checkout page, but has not confirmed the purchase yet, may match such persona.

- **Outcome** - An outcome can be defined as the determination and evaluation of the results of an activity or journey. An example of an outcome may include "*Checkout Confirmation*" or "*Abandoned Cart*". Outcomes are manually formulated and inserted into the Altocloud software, by business owners and marketing teams.

Each time a customer matches a persona or outcome, Altocloud's software appends this information to the customer's profile in real-time. Following the above definitions, the requirements of this project are put forward.

3.2.1 Obtain a Real-Life Customer Dataset

Finding the right excerpt
Due to the large quantities of customer data collected by Altocloud, only a sample of the entire

[20] An interesting article on *The Advantages of Market Segmentation* can be found at: `http://smallbusiness.chron.com/advantages-marketing-segmentation-67156.html`.

database is required to build an accurate model. Inversely, a sample large enough to represent the entire *population*[21] is also needed. Hence, a suitable excerpt of the data must be chosen for training, testing and validation of the model.

Model a single business
The excerpt must contain information about customers from a *single* business, to focus on the behaviour of customers observed on a *single* website. Altocloud contain information about customers from a multitude of e-commerce websites so this is an important feature.

Personas and Outcomes Matter
Additionally, the single business from which the data is taken must have defined an Outcome which can be used to validate clusters; an Outcome such as "*Checkout Confirmation*". Finally, we only want data containing customers who have matched at least one persona.

3.2.2 Data preprocessing and Data Encoding

Data Cleaning
It is normal for bad data to exist in the dataset. The data preprocessing phase includes measures to to alleviate errors, inconsistencies and missing data.

Important Features
A meeting was held with engineers and data scientists from Altocloud where it was recommended that "unnecessary features, such as *unique customer IDs*, *business IDs* or any other features with low information gain can be omitted from the dataset". Similarly, "important features such as *personas matched*, *time of visit* and *browser used* should be included in the dataset."[22]

Encoding the Data
Most of the values in the dataset take the form of categorical data. The K-means algorithm requires numeric values and so all categorical data must be numerically encoded.

Data Protection
Lastly, the data contained within Altocloud's database contains sensitive information about customers. Personally identifiable information must be protected under the SaaS Application Service Provider Agreement[23].

[21] The *population* in this case is regarded as the entire heterogeneous group of individual customers.

[22] The features mentioned were recommended by Maciej Dabrowski (Chief Data Scientist, Altocloud) and Conor Fennell (Software Engineer, Altocloud) in January, 2016.

[23] See more about the SaaS ASP agreement at: `http://www.bodlelaw.com/saas/confidential-information`.

3.2.3 Construct a Machine Learning Model

The machine learning model should be trained with historical data from the Altocloud database, using the an appropriately processed data excerpt (as described in Sections 3.2.2 and 3.2.1). The model should infer the behaviour of "future" customers by making predictions about which cluster customers belong to. The clusters should be checked for validity before drawing conclusions about the results observed.

3.2.4 Apply Findings to Persona Creation and Other Marketing Strategies

The objective of clustering is to create market segmentation that can influence persona creation and other marketing strategies. Instead of arbitrarily making a persona, use the clusters to guide and influence persona creation. For example, each cluster can define a persona, some more commercially valuable than others. The system should be as automated as possible; from the data retrieval, to the clustering, to the persona creation. This project does not attempt to build a completely autonomous system as it is out of the scope of work, but it aims to build the key components of the overall system.

3.3 Design Solution

In the initial design phase of the clustering model architecture, a high level flow control diagram is created and explained in detail. The flow control is based on the Cross Industry Standard Process for Data Mining (CRISP-DM) process model (see Figure 16). The CRSIP-DM model describes common approaches used to tackle problems and is often used by data mining experts. Each component of the following flow diagrams are broken down into sub diagrams to explain the individual operations. Over time software coding and analysis techniques improved, and these diagrams changed to reflect the updated architecture and overall improvements to the flow of control. The model was updated and improved on a continuous basis and diagrams such as these help to communicate such changes. The following section provides an insight into the early development of the system architecture and how it has changed over time.

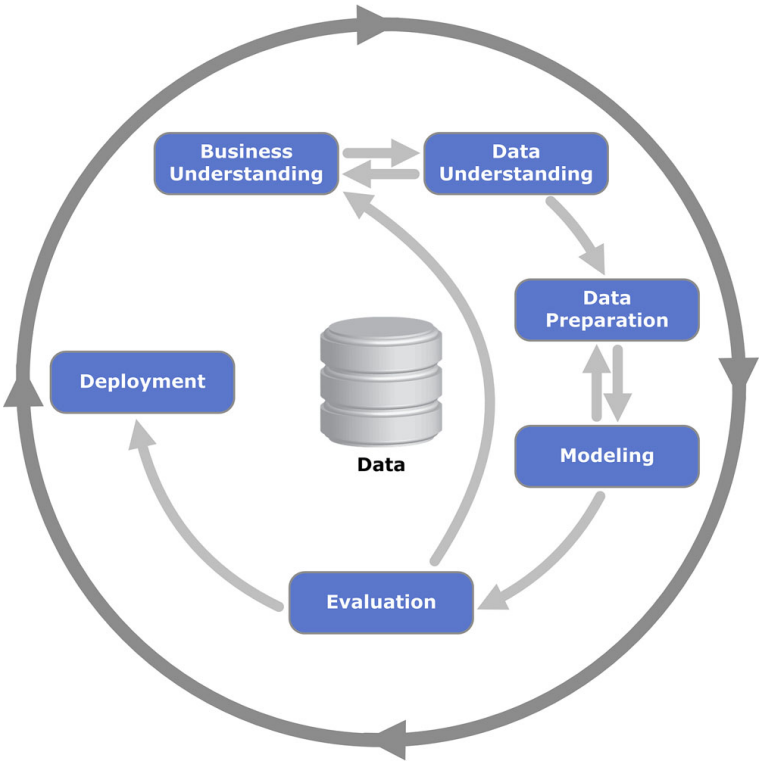

Figure 16: CRISP-DM process model.
(Taken from lectures slides in module CT475 at NUIG)

3.3.1 Initial Design Phase

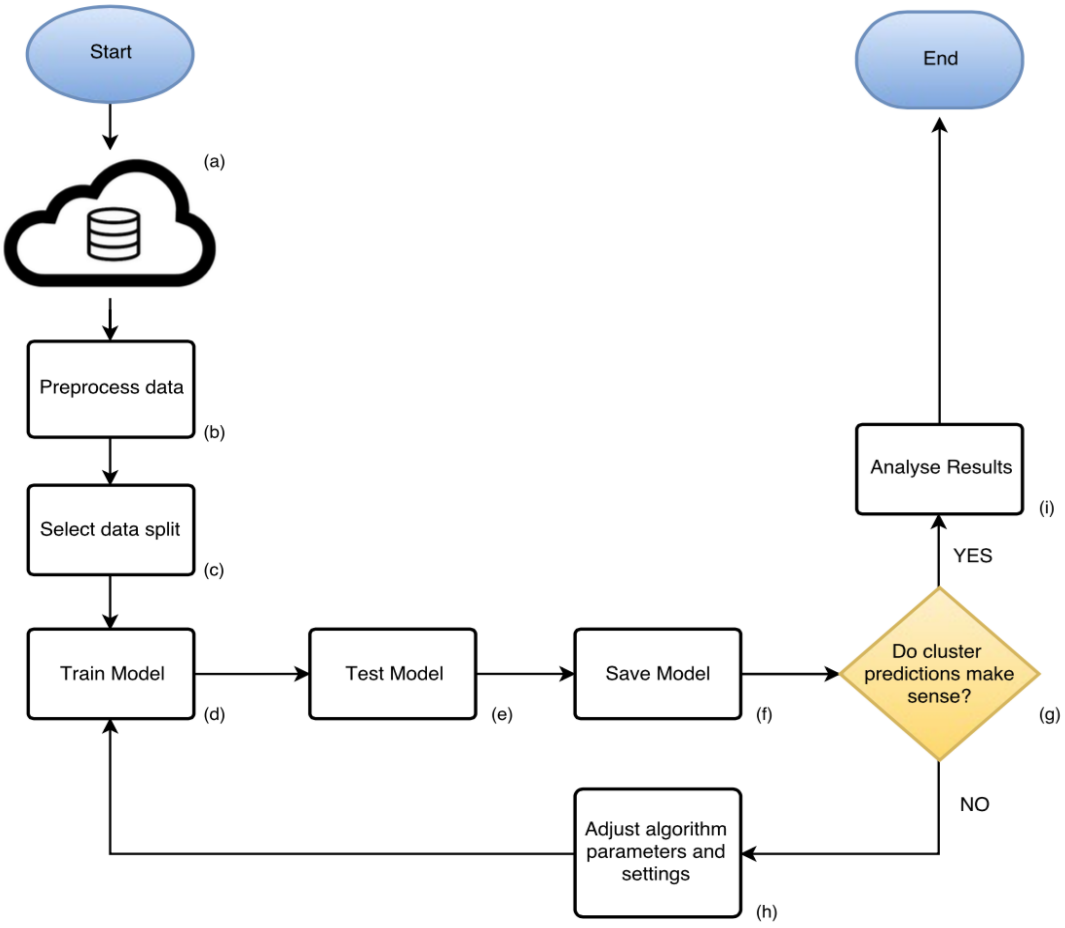

Figure 17: Initial flow of control design

(a) Altocloud's customer data is contained in cloud based data storage, hosted by AWS. The S3 storage facility can be accessed through the AWS Command Line Interface (CLI) or through the S3 user interface. Authentication credentials for Altocloud's S3 buckets are required to access customer data when the CLI is used. A data dump is taken from the database and stored locally on an encrypted Solid State Drive (SSD). The *data dump*[24] should meet the requirements described in Section 3.2.1.

[24]A data dump is a large amount of data transferred from one system or location to another.

(b) The data stored on the encrypted SSD is processed and cleaned. This process is done using a combination of programming scripts and manual processing using tools and software such as Python, CLI, Weka and Microsoft Excel. Once the dataset is processed and prepared for modelling, 5000 individual entries remain. The dataset included 4 features:

- Page Views - Number of unique pages viewed on website
- Activities - Number of activities carried out while browsing. Some example activities may include "item added to cart", "page viewed", "support chat requested"
- Duration - The length of time spent on a website in seconds
- Device - Device used by customers to access the website e.g., Mobile, Desktop, Tablet, etc.

(c) Split the dataset into the following proportions:

- Training (60%)
- Testing (20%)
- Validation (20%)

Each dataset is created so that the training phase is independent of the testing and validation phase. This ensures no overlap of data and reduces potential *over-fitting*[25].

(d) The clustering model is trained using the K-means algorithm and the training examples resulting from the dataset split.

(e) The model is tested using the testing portion of the dataset.

(f) The model is saved for further analysis after the testing phase is complete. The model is also saved locally on the encrypted SSD.

(g) The model is loaded from the SSD. Cluster formations and cluster predictions are analysed. Using the validity checks described in Section 2.3.2, clusters are validated and predictions are interpreted. If results do not meet the requirements or there is room for efficiency improvements, the parameters for the algorithm are adjusted and the model is trained again.

(h) The K-means algorithm is implemented initially with the default settings. If analysis at phase **(g)** finds the model does not meet the project requirements, alternative algorithm settings and parameters are implemented, and the model is trained and tested again.

(i) If the model appears adequate and meets the project requirements, further analysis is carried out. The clusters may then be used to influence persona creation and drive marketing decisions.

[25] Over-fitting occurs when a statistical model describes random error or noise instead of the underlying relationship. This can occur when the model is trained and tested with the same dataset.

3.3.2 Improving the Design

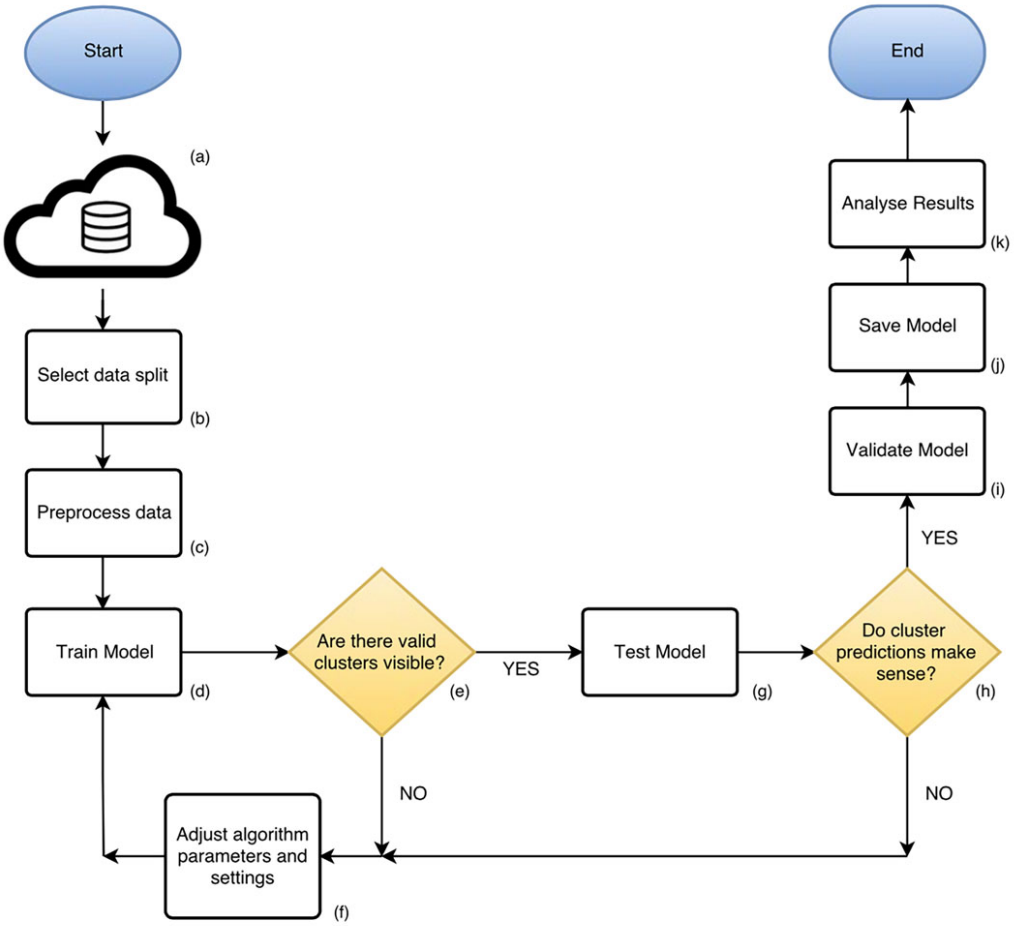

Figure 18: Improved flow of control design

(a) Phase **(a)** does not change in this design. The data dump was carried out once only, in the early stages of this project. If an entirely autonomous system is to be deployed in real-time (data arriving in continuous streams) the system should query the database at a constant time interval, rather than just once. For example, when data arrives in a stream, we may want to estimate clusters dynamically, updating them as new data arrives. Question **R.2** in Section 1.3.1 inquires what circumstances would require a scaled architecture, and the example described above may lead to such architecture.

(b) The improved flow control involves splitting the data prior to the data preprocessing. The

division between the training and test/validation sets is an attempt to replicate the situation where you have past information and are building a model which you will test with *unknown future* information: the training set takes the place of the past and the test set takes the place of the future.

Keeping the past/future analogy in mind, preprocessing should be done on the training set alone. You can then apply the same processing techniques, as used on the training set, to your test set so that both sets are completely independent and replicate past and future customers.

As in earlier designs, the dataset is split into the following proportions:

- Training (60%)
- Testing (20%)
- Validation (20%)

(c) In addition to the tools and software used in the initial design, the *Spark SQL* module is used. The Spark SQL module is used to convert the large data dump into a SQL dataframe, allowing for more efficient data processing. This is due to Sparks parallelising capabilities as described in Section 2.5.3. Again, once the dataset is processed and prepared for modelling, 4,389 individual entries remain. Earlier versions of the dataset contained 5,000 instances which was reduced to 4,389 in the improved design due to additional preprocessing steps taken such as outlier removal and threshold settings (see Section 4.3 for more on data preprocessing techniques used). Additionally, the dataset now includes 34 features, as opposed to earlier versions of the dataset which only contained 4 features. This is due to project requirements and additional feature requests from members of the Altocloud team.

(d) The clustering model is trained using the K-means algorithm and the training set resulting from the dataset split.

(e) Before testing the model, some validation checks such as cluster variance and PCA are conducted (see Section 2.3.2 for more on validation checks). Earlier versions of the design involved validation checks at a later stage in the process. Validating clusters after training and before testing can prevent unnecessary testing.

(f) Again, the K-means algorithm is implemented initially with the default settings. The cluster parameters are refined continuously using the validation checks until a stable and accurate model is trained.

(g) The model is tested using the testing portion of the dataset (aka "future data").

(h) Run some additional checks to see if cluster predictions adhere to the project requirements and answer the research questions. If not, revert back to phase **(f)**.

(i) Once the model is trained and tested and the algorithm parameters have been configured correctly, produce the final results using the validation portion of the dataset.

(j) The model is saved on the encrypted SSD for further analysis after the validation phase is complete.

(k) The model is loaded for final analysis and interpretation. The clusters are used to influence persona creation and drive marketing decisions, as in earlier designs.

4 Implementation

The following implementation aims to apply the machine learning and data mining techniques, as described in Section 2, to produce the proposed design solution from Section 3. This section includes a brief overview of the project management tools used throughout this project, followed by the data processing implementation. This section concludes with the implementation of a clustering model.

4.1 Project Management

All project management is carried out using a project management tool called Redmine. Redmine is a free and open source, web-based project management and issue tracking tool. It allows users to manage multiple projects and assign multiple users to various tasks. Redmine user interface is easy to navigate, containing a calendar and Gantt charts to aid visual representation of projects and their deadlines.

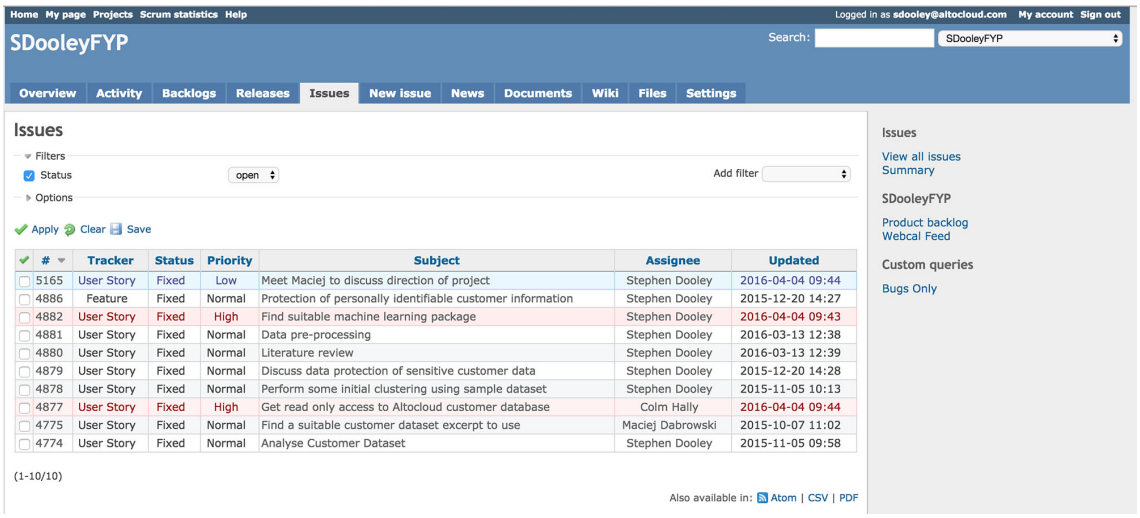

Figure 19: Redmine User Interface

The two main panels used for the management and planning of this project are the *Issue* panel and the *New Issue* panel. As seen in Figure 19, the *Issue* panel contains information about the all open issues (tasks). Each issue is tracked through its *Status*. The *Status* can be one of the following: New, In Progress, Fixed, Feedback, Closed or Rejected. As all issues in Figure 19 have been completed, a *Fixed* status applies to all. A custom *Priority* can be set per issue so that high priority tasks can be focused on first, and vice versa. Figure 19 displays a set of issues that range from *Low* priority to *High* priority. Other priority measures include *Urgent* and *Immediate*. The *Assignee* is used to assign certain members of the team to certain

tasks. Redmine is excellent for collaborative projects and allows multiple users to track a single project or even a specific *Issue*.

The *New Issue* panel is used to create new *Issues*. A brief description of the problem should be provided with each *Issue*. *Issues* are updated regularly with a brief overview of the solution. Screen-shots or images of solution may be uploaded to provide a more comprehensive overview.

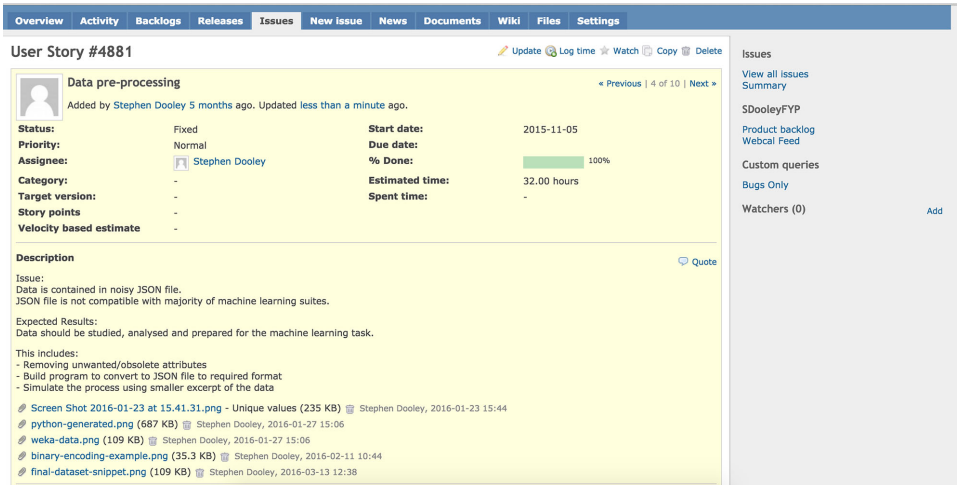

Figure 20: Overview of a *Fixed Issue* in Redmine

4.2 Retrieving the Dataset

Section 2.5.3 briefly introduces *Amazon Web Services* (AWS). This section focuses on AWS S3 which accommodates Altocloud's data, and hence the data used for this project.

A series of steps are taken to retrieve and transform the data from the raw *JSON*[26] files to the final dataset used for modelling. Firstly, a data dump is retrieved from the database and stored locally using the AWS CLI. To gain access to Altocloud's database, authentication credentials are required. The Altocloud authentication credentials are set in the machines *bash*

```
# aws s3 bucket authentication
export AWS_ACCESS_KEY_ID="XXXXXXXXXX"
export AWS_SECRET_ACCESS_KEY="XXXXX-XXXXX-XXXXX"
export AWS_REGION="us-east-1"
```

Figure 21: Example authentication credentials for AWS S3

profile. There is a hidden file in your Mac's user directory named *.bash_profile*. This file is

[26] JavaScript Object Notation (JSON) is a lightweight data-interchange format, easy for humans to read and write.

loaded before Terminal loads your shell environment and contains all the startup configuration and preferences for your command line interface. Within it you can change your terminal prompt, create aliases for command prompts and set environment variables. Figure 21 demonstrates how the credentials for the S3 bucket were assigned as environment variables, and thus granting access to the database. Once configured, the CLI is used to communicate with the database. The command below is used to copy data from the database to a location locally.

```
aws s3 cp s3://eu-west-<project-name>/data.json ./<destination>
```

The command below is used to get a list of data available in a directory.

```
aws s3 ls <path>
```

The data collected from the database contained one month of customer data. After the data is retrieved, data processing techniques were used to create the final dataset for modelling.

Note This architectural design requires you to download vast quantities of data locally, which is not feasible when dealing with larger datasets. However, newer versions of Spark allow you to query AWS S3 data buckets directly using the Spark SQL module. Querying the database directly avoids wasting time unnecessarily downloading large quantities of data. Such features can become a key part of a scalable architecture which provides answers to research question **R.2** (see Section 1.3.1).

4.3 Data Preprocessing

Data preprocessing is an important step in any machine learning task. Real world data are generally:

- Incomplete - lacking attribute values, lacking attributes of interest, or containing only aggregate data[27]
- Noisy - containing errors and outliers
- Inconsistent - containing discrepancies in codes or feature names.

Tasks in data preprocessing include:

- Data reduction - reducing the volume but producing the same or similar analytical results
- Data cleaning - fill in missing values, remove errors and outliers and resolve inconsistencies
- Data transformation - normalise and aggregate data where necessary
- Data encoding - replacing categorical values with nominal values
- Data protection - anonymise and encrypt data to protect personally identifiable information.

A collection of iPython notebooks are written to break the data prcoessing into a series of tasks. The notebooks consist of two primary tasks: data preprocessing and clustering. The scripts are stored in a Dropbox folder which backs up the scripts in real-time. Flow control services such as Github were not used due to the sensitivity of the data being processed. The iPython interface contains a sub-directory for each of these primary tasks.

[27] In statistics, aggregate data are data combined from several measurements.

4.3.1 Data Reduction

Customers from a Single Business

The initial data dump[28] contained information about customers from multiple business over the course of one month. The objective of the data reduction task is to isolate one business and study and analyse the behaviour of customers on a single e-commerce website. Using the Spark SQL module, a single business is extracted from the data. For demonstration purposes, a business called *my-business* is used throughout the following sections.

```
df = sqlContext.read.json(<path to data dump>)
# Infer the schema, and register the DataFrame as a table called "visits"
df.registerTempTable("visits")

# SQL can be run over DataFrames that have been registered as a table.
# ci937cyf2000701pje91u121f = unique business identification number
business = sqlContext.sql('SELECT * FROM visits WHERE business =
    "xxYYEcyf3333701phui1u001f"')

# count how many instances are in the new subset
print("Number of instances in subset:", business.count(), "\n")
# check the url of last page viewed by customers from the subset to
    verify
# there is only data from a single business
business.select('lastPage.uri').show()

# save file as backup
file_path = '/Volumes/encrypted-ssd/fyp/data/backup/json/'
business.select('*').write.save(file_path + 'business', format='json')
print("File saved to external drive")
```

OUTPUT:

```
Number of instances in subset: 3,000,001
+--------------------+
|http://www.my-business.com/<path>|
|http://www.my-business.com/<path>|
|http://www.my-business.com/<path>|
|http://www.my-business.com/<path>|
|http://www.my-business.com/<path>|
+--------------------+
only showing top 5 rows
File saved to external drive
```

[28] A data dump is a large amount of data transferred from one system or location to another.

The above Spark SQL demonstration is used to query the original data dump and return all instances of visits on an e-commerce website owned by *my-business* (ID = "xxYYEcyf3333701phui1u001f"). This implementation can be used to find similar results for any business, provided the business ID is known.

Customers Who Have Matched a Persona

The dataset contains 58 unique features. An important feature in this dataset is *Persona* (See **Important Features** in Section 3.2.1). As stated in Section 3.2, a *Persona* is a representation of a customer based on market research and real data about observed behaviour on a website. The next step taken to *reduce* the data is to find all customers who have matched at least one persona. Customers who have matched at least one persona are likely to be customers who have spent a reasonable amount of time on the website. Web-crawlers and accidental visitors may arrive at a website and leave less than a second later. In which case, they will not have matched a persona. The model should be trained with data representative of the ideal visitor i.e., someone who intended on visiting this website and thus is interested in purchasing a product. Using the Spark SQL module again, all customers who matched at least one persona were extracted and a new subset is created.

```
# register the business subset as a new table called "my_business"
business.registerTempTable("my_business")
# find all instances which have matched at least one persona
personas = sqlContext.sql("SELECT * FROM my_business WHERE personas[0]
    IS NOT NULL")
# count how many instances are in the new subset
print("Number of instances in subset:", personas.count())

# save file as backup
file_path = '/Volumes/encrypted-ssd/fyp/data/backup/json/'
personas.select('*').write.save(file_path + 'personas', format='json')
print("File saved to external drive")
```

OUTPUT:

```
Number of instances in subset: 195,892
File saved to external drive
```

Sample the Subset Further

The dataset now contains 195,892 (down from 3,000,001) instances. All instances contain customers from a single business, who have matched at least one persona. The data is randomly sampled even further to extract 5,000 instances as a dataset of this size is more manageable. The following code snippet demonstrates how the dataset is randomly sampled.

```
sample = personas.sample(False, 5000/195892, 12345)
print("Number of instances remaining:", sample.count())

# save file as backup
file_path = '/Volumes/encrypted-ssd/fyp/data/backup/json/'
sample.select('*').write.save(file_path + 'sample', format='json')
print("File saved to external drive")
```

OUTPUT:

```
Number of instances remaining: 5,044
File saved to external drive
```

Section 3.3.1 describes the initial design solution where only 4 features were included in the training and testing dataset. The final step in data reduction is to remove unwanted features. The final dataset contained 34 features, some of which created through the division of an aggregate feature (such as *device*) into multiple features. See Section 4.3.4 for encoding techniques used to form these additional features. A full list of features available and a list of features used can be seen in Table 12 in the Appendix of this thesis.

4.3.2 Data Cleaning

Once the data is reduced to a manageable size, errors, outliers and inconsistencies are removed from the dataset. The reduced dataset consists of 34 selected features. Of the 34 features there exists 9 primary features: *activities*, *duration*, *pageviews*, *time of day*, *outcomes*, *persona*, *referrer*, *device* and *browser*. Of these 9 primary features, 3 consist of positive, continuous, random values and the rest discrete. The continuous features include: *activities*, *duration*, *pageviews*. To remove outliers from these features some maximum and minimum thresholds were set. Table 1 below displays the thresholds set for the features mentioned.

Feature	**Minimum Threshold**	**Maximum Threshold**
activities	3	100
duration	10 (seconds)	3600 (seconds)
pageviews	2	100

Table 1: Continuous random value thresholds

If missing or inconsistent data is detected, the entire instance is removed from the dataset. Visual analysis of the data is conducted to detect common discrepancies in the data which may be undetectable by a computer. With the exception of the features mentioned in Table 1, all other features consisted of categorical data.

4.3.3 Data Transformation

The data transformation included normalisation of the continuous random values of the *activities*, *duration* and *pageviews* features. All values were positive so $0-1$ normalisation is used. The formula for $0-1$ normalisation is as follows,

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)}$$

where $x_i = (x_1, ..., x_n)$ and z_i is the i^{th} normalised data.

Normalisation is carried out using Microsoft Excel's *function* feature. See Figure 22 for example implementation of normalisation. The function displayed in the function tab (above the cells) is applied to the entire dataset.

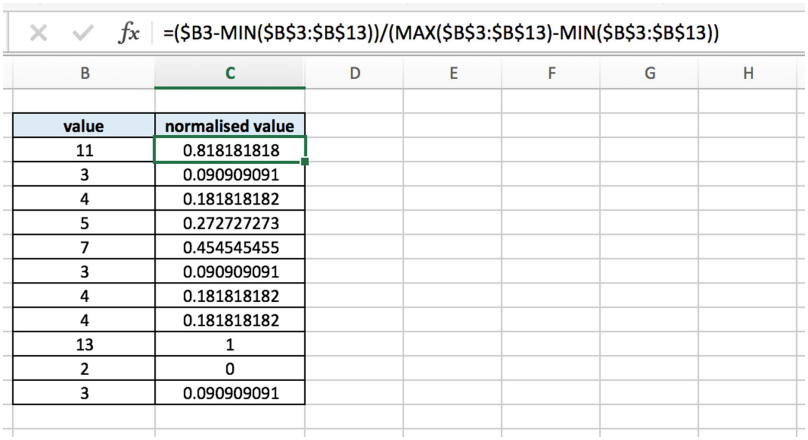

Figure 22: Normalising continuous random variables in Microsoft Excel

4.3.4 Data Encoding

The data encoding implementation is an important process as it transforms categorical data into K-means compatible numeric data. Figure 23 provides a demonstration of typical binary encoding which is implemented on all categorical data.

As mentioned in Section 2.3, the K-means algorithm requires a numerical input dataset. Other encoding techniques such as sequential numbering systems were considered. For example, instead of creating a new feature for each device, assign a value of *desktop* $= 1$, *tablet* $= 2$, *mobile* $= 3$ and *other* $= 4$. Let's say for instance, customer 1 used *desktop* and customer 2 used *mobile*, then the feature *device* would contain a value 1 for customer 1, and a value 3 for customer 2. Although at first this seemed like a suitable encoding, it may be interpreted by the algorithm as an order of preference, whereas the binary encoding simply states whether or not

device	device_desktop	device_mobile	device_tablet	device_other
desktop	1	0	0	0
desktop	1	0	0	0
desktop	1	0	0	0
mobile	0	1	0	0
other	0	0	0	1

Figure 23: Binary encoding of categorical data

the feature is "achieved" by the customer. The binary encoding method eliminates the sense of order and is the preferred encoding technique for this model.

Another feature which required special attention is *time of day*. The traditional binary encoding technique (used on all other discrete features) does not suffice. The feature contains the last time the customer was active **according to the server time**. Altocloud's servers, hosted by AWS, are located in the European Union. This includes two server locations; one in Dublin and the other in Frankfurt. As Dublin is the primary server used by Altocloud, the servers are running at Greenwich Mean Time (GMT). The *time of day* feature may look like the following before the encoding process,

```
2015-06-08 19:26:47.152 Z
=
year-month-day hour:minute:second.millisecond timezone
```

This unfortunately, did not reveal much about what time the customer was actually active, as the timezone will be different for each user. In order to infer the actual time of browsing, the location of the customer (or *client*) is taken into consideration. The latitude and longitude of each customer is also known, and this information is used to translate location to timezone.

The following steps were taken to convert the server time into the actual time of browsing:

1. Get timezone from the customer's latitude and longitude. This is done through *the AskGeo REST API*[29]. The API allows you to infer the timezone of any location through a RESTful request. The following code snippet demonstrates a typical REST request.

```
account_id = "XXX"; api_key = "XxxxXxxxX"
response_format = "json"; databases = "TimeZone"

path = os.path.join("https://api.askgeo.com/v1/",
       <account_id>, <api_key>,
       "query." + response_format)

def make_api_request(rest_request):
    # get REST response containing the timezones
    response = requests.get(rest_request).json()
    # check if request was successful
    response_code = response['code']
    response_message = response['message']
    # data is a list of JSON objects
    if(response_code == 0):
        response_data = response['data']
        # return the data containing the timezone
        return response_data
    else:
        print('API request was not successful: ' +
            response_message)

# get the latitude/longitude co-ords from the dataframe
lat_lng_coords = df['coordinates']
# create list to add timezones to
timezones = []
points = ""
# for each client, get the timezone by querying the API based on
# on the latitude/longitude co-ords
for i, points in enumerate(lat_lng_coords):
    # query the rest API with the following params
    query = "?databases=" + databases + "&points=" + points
    rest_request = path + query
    response_data = make_api_request(rest_request)
    timezones.append(hours)
```

[29]Representational State Transfer (REST) is the software architectural style of the World Wide Web. For more on AskGeo see: `https://askgeo.com/`.

The above script will create a list of timezones (in milliseconds) that correspond to the timezone of each customer. The list is then converted to hours using a basic Python script.

2. Add the timezone (in hours) to the recorded server time for each customer in the dataset.

3. Encode the customer's actual time of browsing using the following encoding:

```
12am -> 8am = 0
9am -> 5pm = 0.5
6pm -> 11pm = 1
```

4.3.5 Data Protection

Certain measures were taken to ensure the protection of personally identifiable customer information[30]. Firstly, an external hard drive is used to maintain all customer data. The drive is encrypted and password protected. Additionally, all personally identifiable information is either omitted from the dataset or binary encoded. Information such as name, address, date of birth or latitude/longitude co-ordinates must be omitted to protect customer privacy.

4.3.6 Final Dataset Snapshot

Figure 24 below displays first 10 rows of the final dataset after all preprocessing has been applied (note that only the 5 of 34 features are displayed).

activities_norm	duration_norm	pageviews_norm	time_of_day_night	time_of_day_work
0.677083333	0.45611591	0.659574468	0	0
0	0.02591251	0.010638298	0	1
0.03125	0.060183895	0.042553191	0	0
0.229166667	0.406241293	0.223404255	0	1
0.0625	0.059069379	0.063829787	1	0
0.270833333	0.163555308	0.255319149	1	0
0.1875	0.29980496	0.20212766	0	1
0	0.173585957	0.010638298	0	1
0.322916667	0.240735581	0.329787234	0	0
0.416666667	0.48202842	0.436170213	0	1

Figure 24: Snapshot of first 10 rows of final dataset

Note See Table 12 in the Appendix section of this thesis for list of features used in final dataset.

[30]To read more about what confidentiality provisions need to be included in a SaaS agreement see: `http://www.bodlelaw.com/saas/confidential-information`.

4.4 Clustering

The data preprocessing resulted in a dataset of 4,389 instances with 34 features. The data split (as described in Section 3.3.2) resulted in three subsets of the dataset:

Dataset Purpose	Percentage Split (%)	Number of Instances
Train Model	60	2661
Test Model	20	835
Validate Model	20	893

Table 2: Data split

The Spark API has a *randomSplit()* method in which the RDD containing the entire 4,389 instances is split into the proportions described in Table 2. The *seed* parameter is a random seed value to initialise the randomisation. The average Silhouette Score for 5 splits (for $seed = 0$ and $seed = 10$) is recorded to detect the most suitable seed value to split the data. Table 3 and Table 4 in Section 5.1.2 display the average Silhouette Scores, for two arbitrary seed values. The following code snippet demonstrates the used of the Spark *randomSplit()* method.

```
# split the dataset into tranining, validation and testing sets
trainingRDD, validationRDD, testRDD = visitsRDD.randomSplit([6, 2, 2],
    seed=10)
# cache datasets for quick access
trainingRDD.cache()
validationRDD.cache()
testRDD.cache()

print("Training: %s, validation: %s, test: %s\n" %
      (trainingRDD.count(), validationRDD.count(), testRDD.count())
     )
```

OUTPUT:

```
Training: 2661, validation: 893, test: 835
```

Once the datasets have been established, the cluster analysis is carried out.

4.4.1 Validate Clusters with Scikit Learn

The Scikit Learn Python module is used to implement some validation checks on the training data before the model is built using Spark's MLib module. As stated in Section 2.3.2, calculating the *variance* and the use of *PCA* can be used as cluster validation measures. Once convergence

has been achieved, calculating the overall variance for the clusters can give a good indication of the differentiation between clusters. PCA is used to reduce the dimensionality of a dataset, thus allowing for easier visualisation of clusters. The following code snippet is used to calculate the variance for a range of values of $K = (1, ..., 15)$.

```
# Determine your k range
k_range = range(1,15)
# Fit the kmeans model for each n_clusters = k
k_means_var = [KMeans(n_clusters=k).fit(training_data) for k in k_range]
# 'centroids' contain cluster centers in each model
centroids = [X.cluster_centers_ for X in k_means_var]
# Calculate the Euclidean distance from each point to each cluster center
k_euclid = [cdist(training_data, cent, 'euclidean') for cent in
    centroids]
dist = [np.min(ke,axis=1) for ke in k_euclid]

# Total within-cluster sum of squares
wcss = [sum(d**2) for d in dist]
# The total sum of squares
tss = sum(pdist(training_data)**2)/training_data.shape[0]
# The between-cluster sum of squares
bss = tss - wcss

# Plot the elbow curve
fig = plt.figure(figsize=(20, 10))
ax = fig.add_subplot(111)
ax.plot(k_range, bss/tss*100, 'b*-')
```

Scikit Learn is also used to implement PCA and determine whether natural clusters exist in the training data. This visualisation is a useful technique when dealing with reasonably high dimensional data (34 features for example). The model is trained for a range of values of $K = (1, ..., 15)$. Secondly, the initialisation algorithm is set to two different modes. This can be either *random* to choose random points as initial cluster centers, or *k-means++*. Scikit learn implements the standard k-means++ "to select initial cluster centers for K-means clustering in a smart way to speed up convergence[31]". However, Spark implements a parallel variant of k-means++ (Bahmani *et al.* [23]) called *k-means||*. Initial testing is done in Scikit Learn in order to determine which initialisation method should be used for final testing in Spark. Results from this test can be seen in Table 3 and Table 4.

[31] See more at: `http://scikit-learn.org/stable/modules/generated/sklearn.cluster.k_means.html`.

The following code snippet demonstrates the implementation of PCA on the training data to observe the existence of natural clusters. Both initialisation modes mentioned above were modelled and Silhouette Scores for each model recorded.

```
number_of_clusters = range(1, 15)
# create a range of models for a range of k
for k in number_of_clusters:
    # model for init='k-means++'
    k_means_plus_plus = KMeans(n_clusters=k, init='k-means++')
    k_means_plus_plus.fit(training_data)

    # model for init='random'
    k_means_random = KMeans(n_clusters=k, init='random')
    k_means_random.fit(training_data)

    # plot cluster assignments
    plot_cluster_centers(k_means_plus_plus, k_means_random,
                         training_data, test_data, str(k))
```

The optimum initialisation algorithm is determined through analysis of the variance plot, the PCA plot and the Silhouette Scores. Additionally the optimum value for K is inferred through similar analysis. These findings are used in the final testing which is carried out using a Spark implementation, rather than Scikit Learn.

Note Results from the above implementations can be seen in Section 5.

4.4.2 Train Model with Spark

To train a K-means clustering model using Spark the following API is available:

```
classmethod train(rdd, k, maxIterations=100, runs=1,
                  initializationMode='k-means||',
                  seed=None, initializationSteps=5,
                  epsilon=0.0001, initialModel=None)
```

The Spark K-means train method takes the following parameters:

- *data* - training points stored as RDD [<double>]
- *k* - number of clusters
- *maxIterations* - max number of iterations
- *runs* - number of parallel runs, defaults to 1. The best model is returned.
- *initializationMode* - initialization model, either "random" or "k-means||" (default).
- *seed* - random seed value for cluster initialization

Now that a suitable, intuitive value for k and $initializationMode$ have been determined, final testing using Spark can be carried out. The model is trained for a range of values of $K = (2, ..., 30)$. Unless stated otherwise, all other parameters for the training model are left as the default. The following code snippet demonstrates the training of a K-means clustering model using Spark:

```
# test for clusters of 2 to 30
min_clusters = 2; max_clusters = 30;
number_of_clusters = range(min_clusters, max_clusters)
cost_values = []

# create a list of cost values and a list of cluster for various cluster
    sizes
for k in number_of_clusters:
    clusters = KMeans.train(dataset, int(k), maxIterations, runs,
                            initializationMode, seed, initializationSteps,
                                epsilon)

    cost_values.append(clusters.computeCost(dataset))
```

The "cost_values" variable contains a list of *Within Set Sum of Squared Errors* for a range of models. The sum of errors is plotted against the range of K and another "elbow curve" is determined. This plot (like the variance plot) can be used to further determine a suitable value for K. See the plot in Figure 26 in Section 5.

4.4.3 Infer Predictions from Trained Model

The optimum value for the training method parameters are used to train the model and the model is saved locally for future analysis. The model is loaded form a file to make predictions. The following code snippet demonstrates typical methodology for making predictions using the *validation set* (the validation set is used only when all testing is complete and the optimum parameters for the algorithm have been found).

```
# LOAD CLUSTER MODEL
clusters = KMeansModel.load(sc, model_location)
print("There are", clusters.k, "clusters\n")

# PRINT CLUSTER CENTRES
for num, cluster in enumerate(clusters.clusterCenters):
    print(num, "- Cluster Centres:\n", cluster, "\n")

# PREDICT CLUSTERS
predictions = clusters.predict(validation_data)

# Create key/value pairs and count the number of cluster assignments
# map() - for each word, create the tuple (word, 1)
# reduceByKey() - go over the tuples "by key" (first element) and sum
    the second elements
count_cluster_assignments = predictions
      .map(lambda cluster: (cluster, 1)).reduceByKey(lambda a, b: a + b)
print("Cluster assignments <cluster, number of assigned customers>:\n",
      count_cluster_assignments.collect())

# CALCULATE COST
WSSSE = clusters.computeCost(validation_data)
print("\nWithin Set Sum of Squared Errors:\n", WSSSE)
```

The *predict* method is used to predict which cluster a customer belongs to. The above snippet demonstrates predictions being made for the validation dataset (893 customers). The *computeCost* function calculates the Within Set Sum of Squared Error for the model. Further description of the code functionality can be seen within the comments of the snippet.

Note Results seen in the output below are used for demonstration purposes only and may differ from actual results presented in Section 5.

OUTPUT:

```
There are 6 clusters

0 - Cluster Centres:
 [0.342406,0.407879,...,0.0,0.0]

1 - Cluster Centres:
 [0.259629,0.314902,...,0.0,0.0]

2 - Cluster Centres:
 [0.178880,0.233646,...,0.0,0.0]

3 - Cluster Centres:
 [0.214504,0.266328,...,0.0,0.0,0.001636]

4 - Cluster Centres:
 [0.211863,0.254076,...,0.001862,0.001862]

5 - Cluster Centres:
 [0.168243,0.246981,...,0.054054,0.027022,0.0]

Cluster assignments <cluster, number of assigned customers>:
 [(0, 109), (1, 87), (2, 183), (3, 209), (4, 184), (5, 133)]

Within Set Sum of Squared Errors:
 1781.3427057814708
```

5 Results

The following section presents the results produced through the implementation of the proposed design solution. The optimum algorithms parameters are inferred through a series of plots and results developed in the cluster validation process. Some cluster visualisation plots produced from the model training are followed by some test results. This section concludes with a set of predictions and related analysis.

Note *Phrases such as "outcome achieved" or "achieved an outcome" suggest that a customer(s) has successfully bought something on the website.*

5.1 Infer Optimum Algorithm Settings

This subsection presents the results that ultimately determine of the optimum K-means algorithm parameters.

5.1.1 Variance Curve

Figure 25 shows the first of two "elbow-curves" used to detect the natural number of clusters in the data. When plotting the percentage of variance explained against the number of clusters, the first clusters adds much information (explain a lot of variance), but at some point the marginal gain will drop, giving an angle in the graph. The number of clusters is chosen at this point, hence the "elbow criterion". This "elbow" cannot always be unambiguously identified [24] and so it can simply be observed by eye. From the curve it can be seen that the most favourable value for K lies in the range of $K = (4, 5, 6, 7)$.

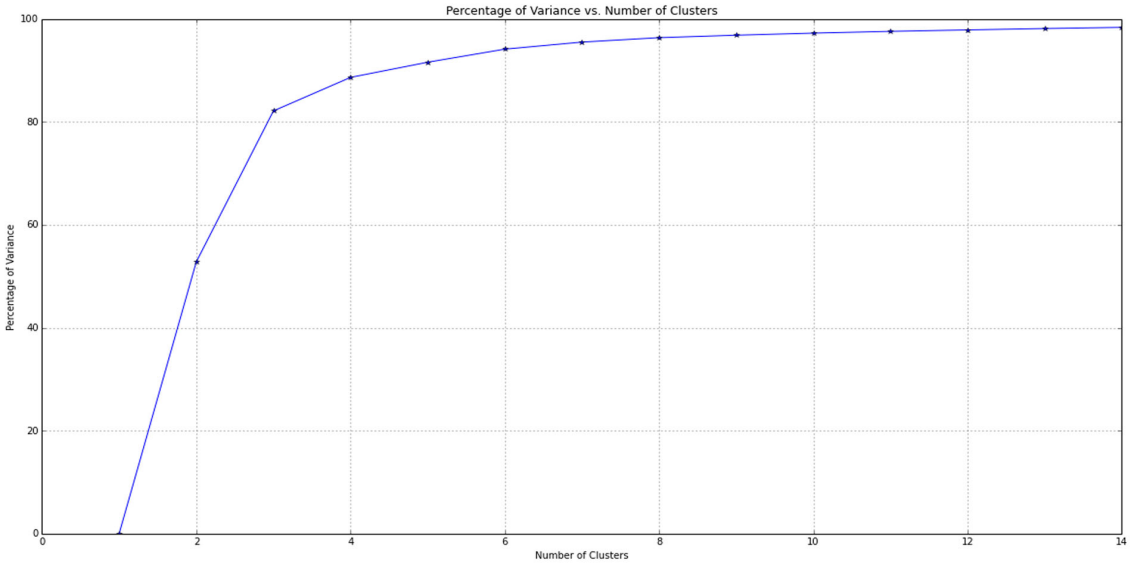

Figure 25: Percentage of Variance vs. Number of Clusters

5.1.2 Silhouette Scores

The average silhouette of the data is another useful criterion for assessing the natural number of clusters. The silhouette of a datum is a measure of how closely it is matched to data within its cluster and how loosely it is matched to data of the neighbouring cluster, i.e. the cluster whose average distance from the datum is lowest [18]. Table 3 and Table 4 display the Silhouette Scores for two data split seed values (as described in Section 4.4). The scores are an average value for 5 models.

Number of Clusters	Model Initialisation	Silhouette Score
4	K-means ++	0.591217492
4	random	0.591192328
5	K-means ++	0.562976408
5	random	0.562928604
6	K-means ++	0.600869591
6	random	0.600869591
7	K-means ++	0.605156248
7	random	0.605156248
8	K-means ++	0.578028483
8	random	0.578117369

Table 3: Silhouette score for multiple cluster sizes (Data Split Seed = 0)

Another test is carried out on a dataset with random split seed = 10

Number of Clusters	Model Initialisation	Silhouette Score
4	K-means ++	0.589737684
4	random	0.589562726
5	K-means ++	0.56741507
5	random	0.567418436
6	K-means ++	0.604476876
6	random	0.604565536
7	K-means ++	0.603144533
7	random	0.603418864
8	K-means ++	0.57972674
8	random	0.571564296

Table 4: Silhouette score for multiple cluster sizes (Data Split Seed = 10)

The average Silhouette Scores show almost no difference when comparing the data split seed's influence:

Table Reference Number	Average Silhouette Score
3	0.587651236
4	0.588103076

Table 5: Average Silhouette Scores

Table 5 shows that the random split seed does not have a significant effect on the Silhouette Score of the model. It can be concluded that the random split does not have an effect on the cluster formation of the dataset. Additionally, the important features of Table 3 and Table 4 are Number of Clusters (K) and the Model Initialisation (*initializationMode*). Despite the data split seed not having a significant influence on the Silhouette Score, an intuitive decision would be to train the model using the dataset used to deduce the results in Table 4 (the average Silhouette Score is marginally better for this dataset).

Following the decision to use the results from Table 4, a suitable value for K and *initializationMode* must be determined. Referring to Section 5.1.1, it is assumed that the most favourable value for K lies in the range of $K = (4, 5, 6, 7)$. Additionally, setting $K = 6$ and *initializationMode* = *'random'* provides the best Silhouette Score. Without judiciously concluding that the optimum values for K and $initializationMode$ have been found, some more tests are conducted.

5.1.3 Cost Curve

Another "elbow-curve" developed to infer the optimum number of clusters is the curve seen in Figure 26 below. The plot shows the Within Set Sum of Squared Errors for a greater range of clusters, $K = (2, 3, ..., 30)$. Again, from the curve it can be seen that the most favourable value for K lies (on the "elbow") in the range of $K = (4, 5, 6, 7)$. This check further verifies that setting $K = 6$ and *initializationMode = 'random'* is most favourable for this clustering task.

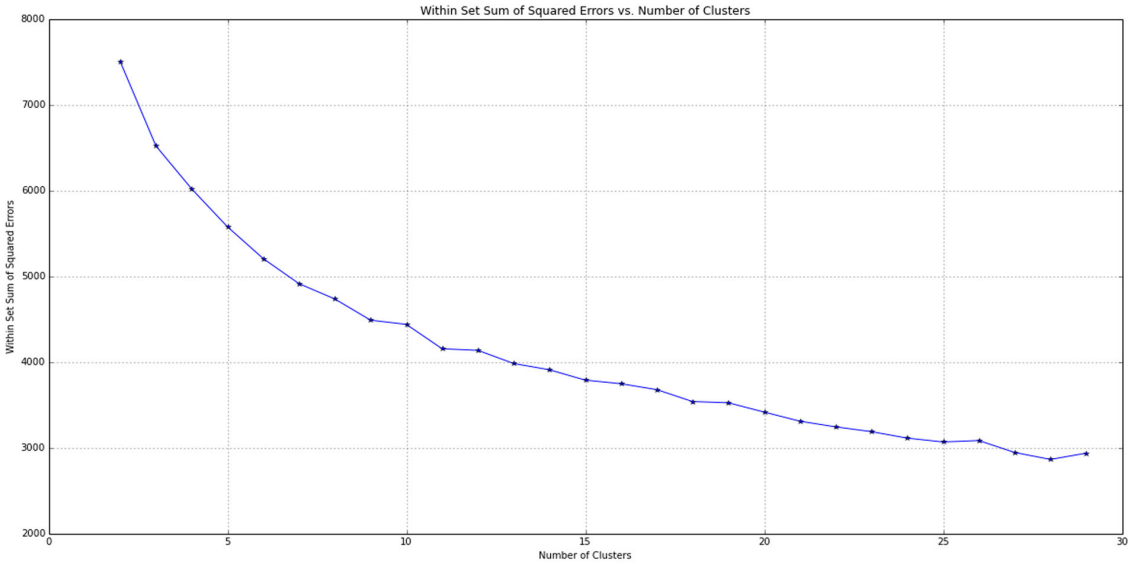

Figure 26: Within Set Sum of Squared Errors vs. Number of Clusters

5.1.4 PCA Visualisation

Principal Component Analysis is applied to the dataset to view the natural inherent clusters of the training set in 2 dimensions. A mesh grid is created to encapsulate every instance of the dataset in a single grid. Each pixel of the grid represents a potential customer in the 2 dimensional space. Clusters can be observed in Figures 27, 28, 29 in which the white dots denote the centre of the clusters. It is clear from Figure 27 that a value of $K = 4$ or $K = 5$ does not split the data into its natural clustering. Clusters assigned by the model contain more than one natural cluster within them, and should therefore be split into two or more clusters.

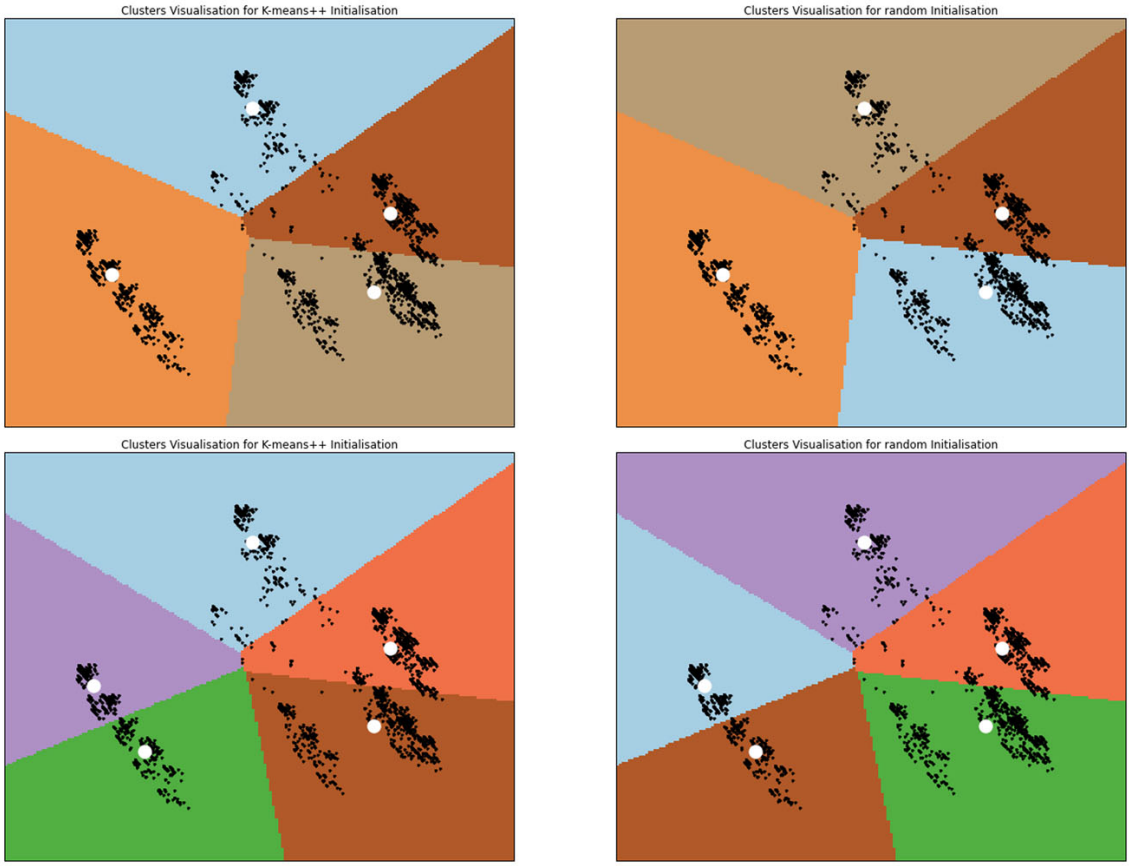

Figure 27: Cluster visualisation for clusters $K = 4$ (top left and top right) and clusters $K = 5$ (bottom left and bottom right)

Inversely, Figure 28 shows that a value of $K = 7$ or $K = 8$ is too high, and the model has split natural clusters into multiple clusters.

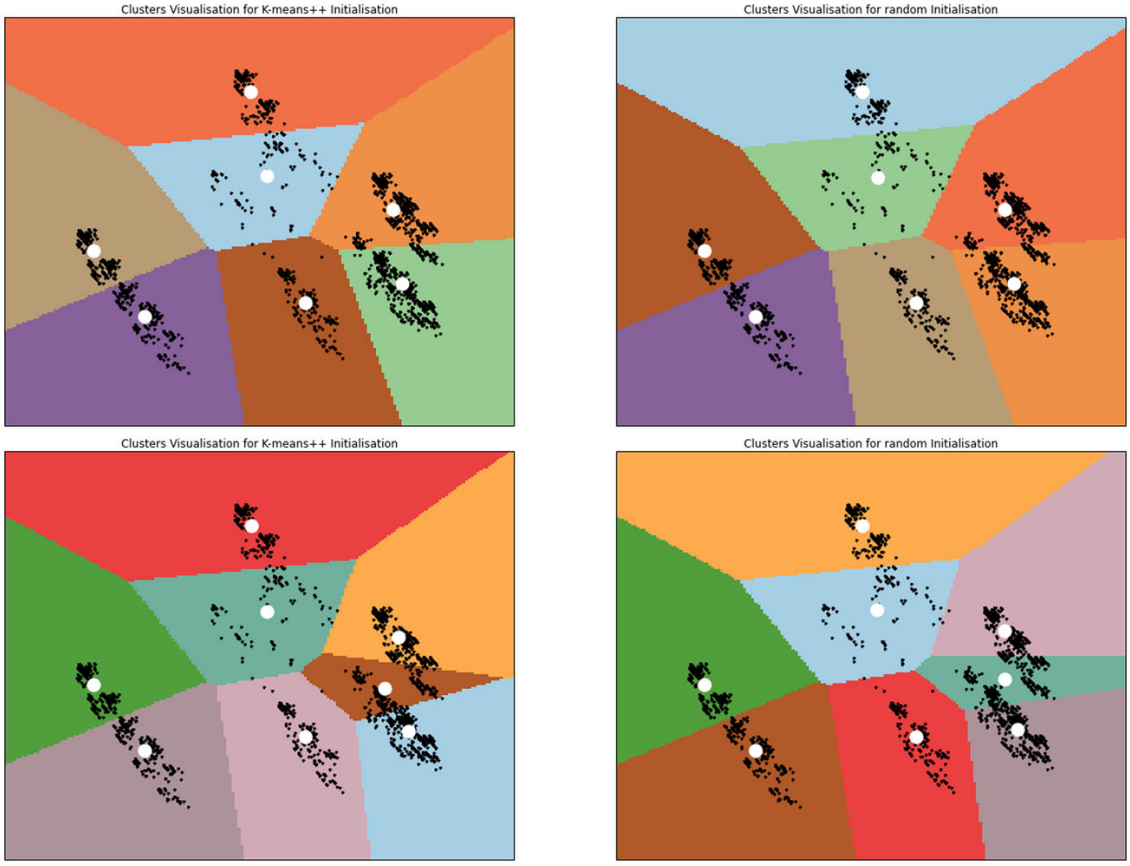

Figure 28: Cluster visualisation for clusters $K = 7$ (top left and top right) and clusters $K = 8$ (bottom left and bottom right)

Figure 29 shows that a value of $K = 6$ allows the model to form a single cluster for each natural cluster, concluding that $K = 6$ is in fact the most favourable value for this task. The results presented in the following subsections are based on studies carried out for a value of $K = 6$ and $initializationMode = random$ due to the validation checks carried out above.

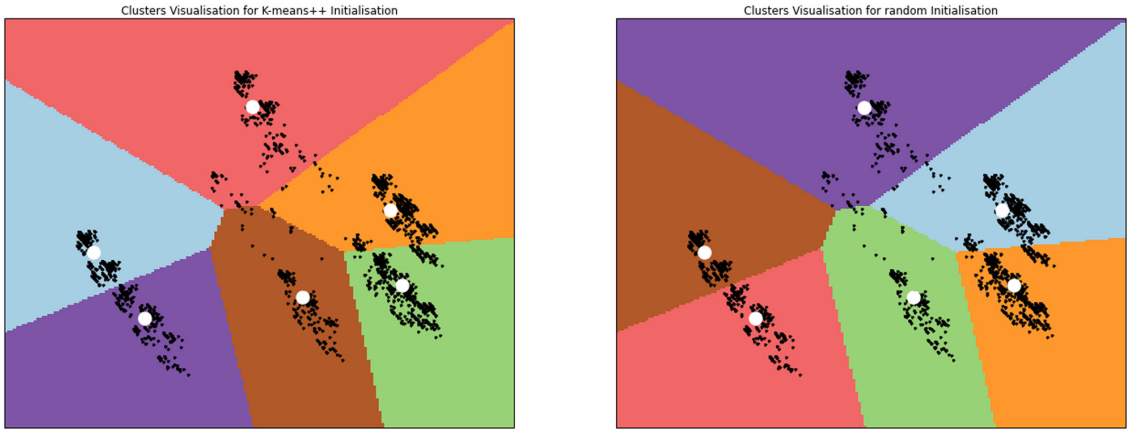

Figure 29: Cluster visualisation for clusters $K = 6$

5.2 Clustering Model

A model is trained, tested and validated using the datasets designated for such tasks. It has been discovered that a value of $K = 6$ is ideal for this task and so the following results are presented for such value (along with other parameters that have been deemed best fit for this task). Further tests are carried to verify if the clustering model can predict, with reasonable accuracy, whether or not a customer is likely to make a purchase.

5.2.1 Training the Model

The model is trained for $K = 6$ and the following cluster centres are found:

		Cluster Centres					
Feature		**Cluster #1**	**Cluster #2**	**Cluster #3**	**Cluster #4**	**Cluster #5**	**Cluster #6**
1	time_of_day_night	0.19838	0	0.28336	0.26478	0.45955	0.1048
2	time_of_day_work	0.4453	1	0.44789	0.44473	0	0.58467
3	time_of_day_evening	0.35627	0	0.26873	0.29048	0.54044	0.31048
4	device_desktop	0	0.99075	0	0	0.93975	0.93548
5	device_mobile	0	0	1	0.97429	0.00688	0.04838
5	device_tablet	1	0	0	0.02570	0	0.01209
7	device_other	0	0.00924	0	0	0.05335	0.00403
8	browser_chrome	0	0.34822	0	0.91516	0.34423	0.56451
9	browser_safari	1	0.10015	1	0	0.15146	0.22580
10	browser_ie	0	0.42372	0	0	0.31497	0.01612
11	browser_chrome_mobile	0	0	0	0.85347	0	0.01209
12	browser_mobile_safari	1	0	1	0	0	0
13	browser_firefox	0	0.10015	0	0.00257	0.09294	0.17741
14	browser_gomeza	0	0.02465	0	0	0.05335	0
15	browser_chromium	0	0	0	0	0.03958	0
16	browser_chrome_mobile_ios	0	0	0	0.03856	0	0.01209
17	browser_android	0	0	0	0.05912	0	0.01209
18	browser_amazon_silk	0	0	0	0.02056	0.00344	0.00403
19	browser_opera	0	0.00308	0	0	0	0
20	referrer_yahoo	0.04453	0.04160	0.01462	0.00771	0.01549	0.04838
21	referrer_bing	0.00809	0.0677	0.01096	0.00514	0.04991	0.01209
22	referrer_other	0.07692	0.09244	0.04021	0.02827	0.08089	0.08467
23	referrer_google	0.17004	0.22496	0.17184	0.15167	0.21170	0.25806
24	referrer_facebook	0	0	0.00365	0.00257	0.00344	0
25	persona_a	0.74898	0.78428	0.90310	0.87917	0.77452	0.94758
26	persona_b	0.34412	0.29275	0	0.0128	0.26506	0.24193
27	persona_c	0.25910	0.04776	0.22851	0.1696	0.06024	0.98387
28	persona_d	0.09716	0.10477	0.07129	0.08226	0.12392	0
29	persona_e	0.13360	0.05084	0.10054	0.0694	0.06024	0.64919
30	persona_f	0	0	0	0	0	0
31	outcome_achieved	0.20647	0.10631	0.15722	0.09768	0.09294	1
32	pageviews_norm	0.23555	0.2343	0.17007	0.17051	0.22541	0.32824
33	activities_norm	0.24046	0.23711	0.16613	0.16693	0.22606	0.35706
34	duration_norm	0.29436	0.29061	0.22250	0.25366	0.27390	0.39148

Table 6: Cluster Centres

The results from Table 6 are produced by the model using the training dataset. Each feature $F = (1, 2, ..., 34)$ has one corresponding cluster centre, for each cluster $C = (1, 2, 3, 4, 5, 6)$.

The results from Table 6 show that Cluster 6 appears to be the most likely cluster to contain customers who have achieved an Outcome.

		Cluster Centres					
Feature		Cluster #1	Cluster #2	Cluster #3	Cluster #4	Cluster #5	Cluster #6
31	outcome_achieved	0.20647	0.10631	0.15722	0.09768	0.09294	1

Table 7: Cluster Centres for the Outcome feature

Table 7 above highlights the Outcome feature. Note that Cluster 6 has a cluster centre of 1, whereas all other clusters have cluster centres less than 0.21. The Outcome feature is quantified by a binary 1 or 0 and so values other than 1 or 0 for clusters centres can be interpreted as an average value. It is assumed that cluster centre values less than 0.5 correspond to a majority 0 value in the cluster, and cluster centre values more than 0.5 correspond to a majority 1 value in the cluster. Therefore it can be interpreted that Cluster 6 contains the customers who have (by majority) achieved an Outcome.

Features have been numbered from 1 to 36 in Table 6 to replicate precedence in a customer *journey* (see Section 1.1 for more on customer journeys). The following example introduces a typical journey:
A person who is browsing at work[32] uses their desktop[33] to open a Google Chrome browser[34]. Within the Google Chrome browser, the customer searches for e-commerce website using a Yahoo! search engine[35]. Once on an e-commerce website, they browse through pages and items while information about *page views*, *duration on website*, *matched personas*, etc. are recorded[36]. Each step of the journey (1 through 5) is defined by a specific set of features.

[32] Step 1 includes features 1 - 3
[33] Step 2 includes features 4 - 7
[34] Step 3 includes features 8 - 19
[35] Step 4 includes features 20 - 24
[36] Step 5 includes features 25 - 34

Figure 30 demonstrates 6 typical journeys illustrated by the 6 clusters and their centres.

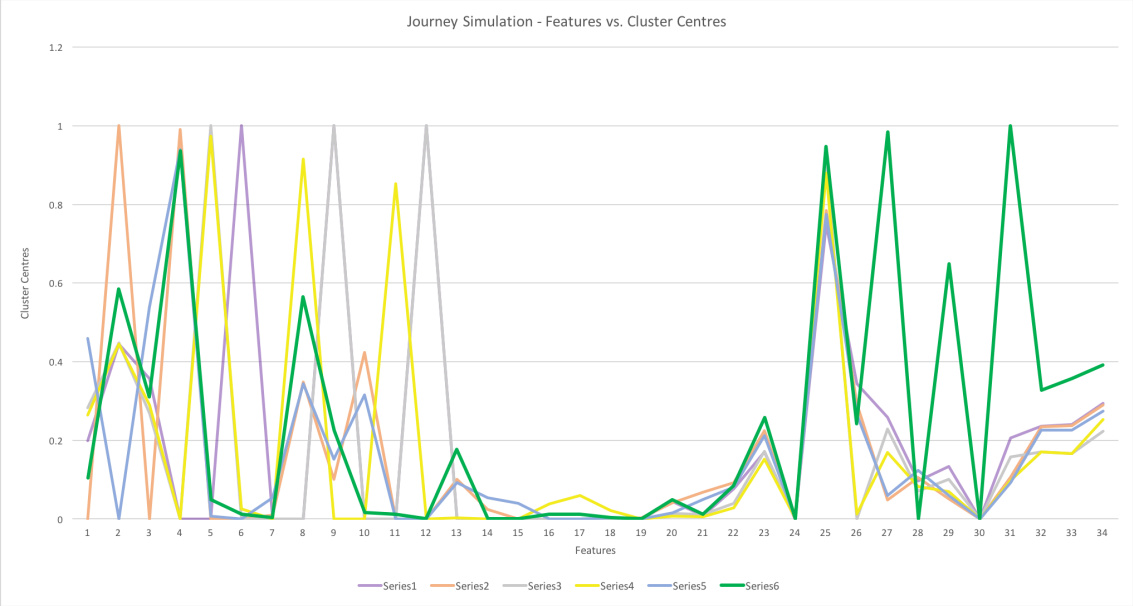

Figure 30: Journey simulation

The legend below the graph denotes each cluster as a "series". Cluster 6 (series 6) is highlighted in green and is the main focus of the graph because as mentioned previously, Cluster 6 contains the majority customers who have achieved an Outcome. It is important at this point to focus on the reasons why members of Cluster 6 may achieve an outcome and why members of other clusters may not. The peaks and troughs in the graph are what define a cluster. The following analysis contains references to features by numbers rather than name to coincide with Figure 30. Refer to Table 6 for translations.

Why customers in Cluster 6 are likely to purchase

It can be seen that Cluster 6 contains some notable peaks for Features 27, 29, and 31 (peaks are much less significant in all other clusters). Feature 31 corresponds to the Outcome feature and it is expected that Cluster 6 shows the highest peak for this Feature. Features 27 and 29 correspond to Personas. Again, these features are not be entirely unique to customers who have achieved an outcome but are certainly more common amongst these types of customers. Some of the less notable peaks that define Cluster 6 are those seen for Features 13, 20 and 23. Despite cluster centre values for these features being closer to 0 than to 1, it is still noted that they are highest in Cluster 6.

Equally as important as the peaks are the troughs. Cluster 6 contains a notable trough for Feature 28. Feature 28 is again a Persona feature which may suggest that customers who have matched this Persona are not likely to purchase. Each of the features (8, 13, 20, 23, 27, and 29) mentioned in this analysis are the influential features that define Cluster 6. Thus, it can be assumed that these features influence the behaviour of customers who are likely to achieve and outcome. Figure 30 provides another visual comparison between Cluster 1, 4, and 6 (Clusters 1 and 4 were arbitrarily chosen to demonstrate a method of comparative analysis).

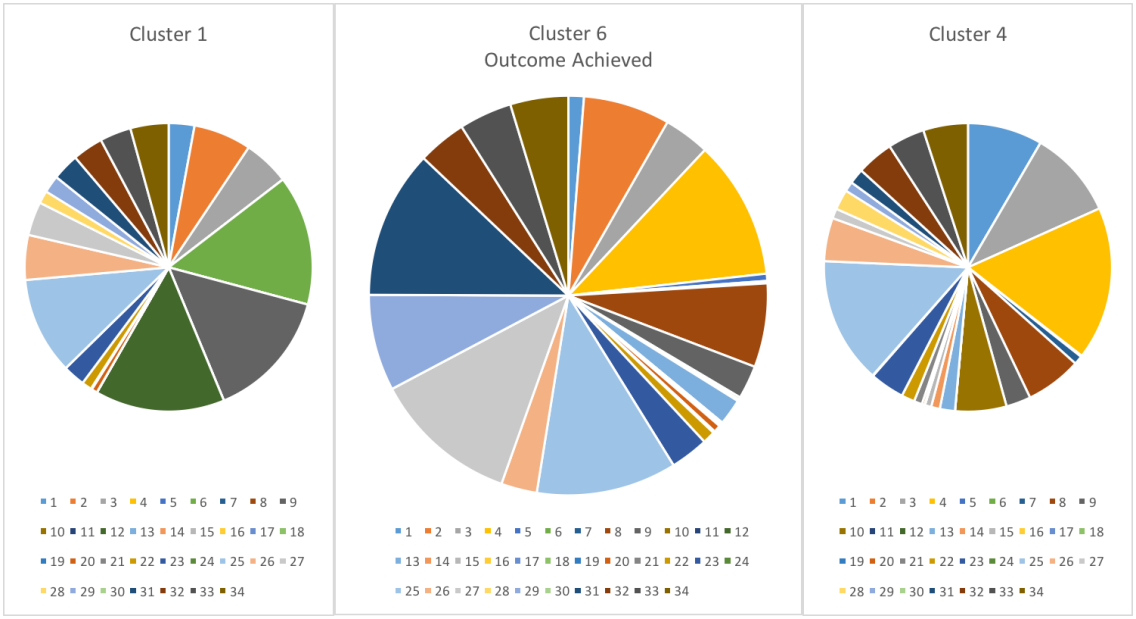

Figure 31: Pie-charts showing feature significance for Clusters 1, 4 and 6

Why customers in Cluster 4 are not likely to purchase
Some other interesting analysis can be done on other clusters such as Cluster 4 (series 4). Cluster 4 shows dominating peaks for Features 5, 8 and 11. These features correspond to *device tablet*, *browser chrome* and *browser chrome mobile* respectively. The data presented infers that customers who use tablets or mobile version browsers are less likely to purchase. This can be due to a number of reasons, for example:

- Website feature restrictions on tablets and mobile devices
- Limited visibility of items due to smaller screens sizes

Similar analysis to that carried out above can be carried out by businesses who have now obtained this information through cluster analysis. Results such as those presented in this section can be used to drive marketing decisions and gain a market advantage. The results from a range of tests to predict the behaviour of customers are presented in the following subsections. Additional analysis is carried out to determine whether the clustering model, such as that implemented above, can be of commercial value to a company (refer to question **R.2** in Section 1.3.1).

5.2.2 Testing and Validating the Model

Table 8 below shows the predictions made for the validation dataset (893 instances). The table contains the number of customers assigned to each cluster. At the bottom of the table lies the *Within Set Sum of Squared Error* (WSSSE) associated with the predictions[37].

Cluster	**# Customers**
1	76
2	232
3	170
4	130
5	187
6	98
WSSSE	1774.29

Table 8: Predictions of cluster assignments for customers in validation dataset

Figure 32 displays the proportion of the validation data that is assigned to each cluster. Using Cluster 6 as the benchmark example, it can been see that 11% of the customers are likely to achieve an outcome due to the dominating outcome feature that defines Cluster 6. **Note that other clusters may contain customers who have achieved an outcome (see Table 7).**

[37] Section 2.3 introduces the Sum of Squared Error measure.

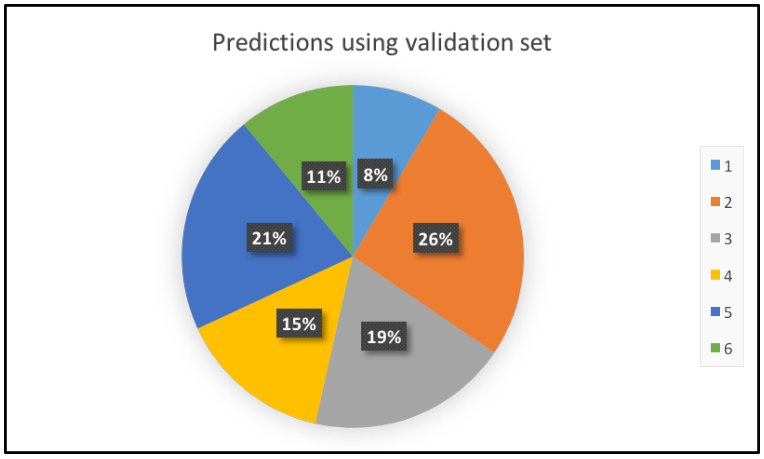

Figure 32: Proportion of the validation dataset that is assigned to each cluster

5.2.3 Making Predictions about Important Customers

Additional testing datasets are created to infer predictions about customers who specifically **had** or **had not** achieved an outcome. These datasets were created by combining instances from the testing and validation datasets, which contained a mixture of both customers who achieved an outcome and customer who didn't. Table 9 below shows the predictions made for 3 such test sets: (a), (b) and (c). Test sets (a), (b) and (c) contain 355 instances each.

Table 9 (a) - This dataset comprises of 355 customers (taken from the test and validation set) who **have not** achieved an outcome. Therefore, the Outcome feature comprises of *0's* for all customers.

Table 9 (b) - This dataset comprises of 355 customers (taken from the test and validation set) who **have** achieved an outcome. Therefore, the Outcome feature comprises of *1's* for all customers.

Table 9 (c) - This dataset also comprises of 355 customers (taken from the test and validation set) who **have** achieved an outcome. However, the Outcome feature is "hidden" for all instances in this dataset. This is to replicate a real life situation where the outcome of a particular journey is not known yet.

Analysis of the results from Table 9 is carried out on the following page. As in previous analyses, Cluster 6 is used as a benchmark demonstration.

	(a)	(b)	(c)
Cluster	**# Customers**	**# Customers**	**# Customers**
1	26	31	31
2	103	45	81
3	76	46	46
4	60	20	26
5	90	31	70
6	0	182	101
WSSSE	659.26	938.56	938.02

Table 9: Predictions of cluster assignments for customers who have/have not achieved an Outcome

Table 9 (a) - Of the 355 customers in dataset (a), the model predicts that none of the customers are assigned to Cluster 6, and therefore not likely to achieve an outcome. This is an expected result from dataset (a) which contains all the customers who have not achieved an outcome.

This may due to the existence of high clusters centres for features such as Feature 27, 29 and 31 in Cluster 6, which is common only to customers who **have** achieved an outcome. This may be also due to the existence of low clusters centres for features such as Feature 8 in Cluster 6, which again is only common to customers **have** not achieved an outcome.

Table 9 (b) - Of the 355 customers in dataset (b), the model predicts that 182 (51.27% of the dataset) customers are assigned to Cluster 6, and therefore are likely to achieve an outcome.

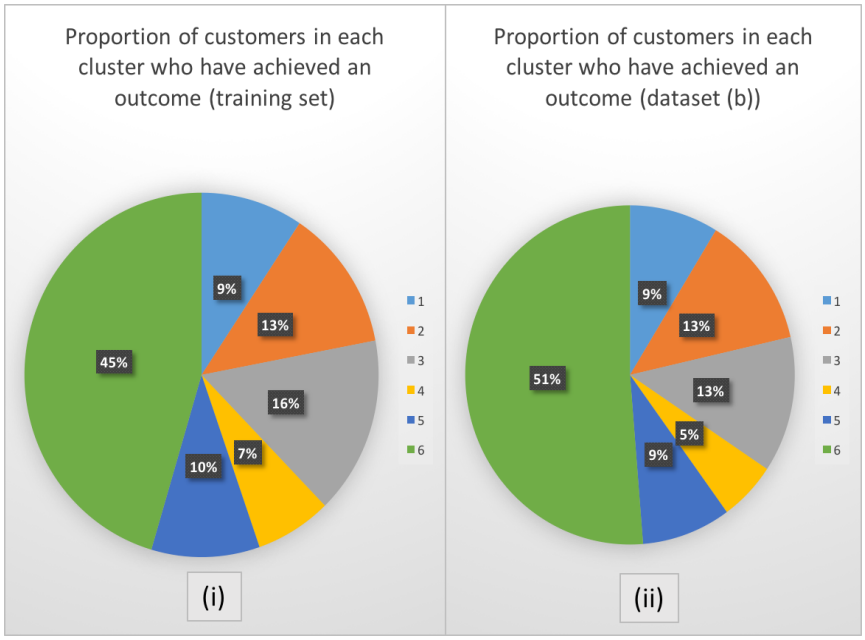

Figure 33: (i) Proportion of customers who have achieved an outcome (trained data) v.s.
(ii) Predicted proportion of customers who have achieved an outcome (dataset (b))

Referring to Table 10, it can be seen that Cluster 6 is **not** the only cluster that contains customer who have achieved an outcome. Figure 33(i) displays the proportion of the customers (who have achieved an outcome) within each cluster for the trained data. Interestingly, similar ratios can be seen Figure 33(ii) which contains the predicted proportion of customers in each cluster for dataset (b) i.e., the dataset containing the customers who have achieved an outcome. Note that the number customers who have achieved outcome Figure 33(i) can be calculated by getting the product of the cluster centroid and the number of customers in the cluster. This is because the centroid value is a binary value.

Table 10 contains the results used to produce the pie charts in Figure 33.

	Cluster					
	C1	**C2**	**C3**	**C4**	**C5**	**C6**
cluster centroids for Outcome feature	0.2064	0.1063	0.1572	0.0974	0.0929	1
# cutomers in trained clusters	247	649	547	389	581	248
# customers who have achieved outcome (i)	51	69	86	38	54	248
# customers who have achieved outcome (ii)	31	45	46	20	31	182

Table 10: Cluster centres vs. Number of customers per cluster

The model demonstrates the system's ability to use significant features to influence predictions. The accuracy of the model can be evaluated by calculating the absolute difference, or error, per cluster by comparing results in Figure 33(i) and Figure 33(ii). The error for dataset (b) is 12%; therefore, the model shows 88% accuracy when clustering customers from dataset (b) i.e., customers who have achieved an outcome. Although the Outcome feature is used as a deciding factor when evaluating the whether the cluster assignment is appropriate, the clustering process is not based solely on this feature; the additional 33 features are taking into consideration when clustering the customers.

The same analysis could be carried out for any feature. For example, Cluster 3 has a cluster centroid of 1 for the *device mobile* feature. Instead of tyring to detect customers who have achieved an outcome, you could run tests to find the impact of using mobile when shopping online.

Table 9 (c) - Of the 355 customers in dataset (c), the model predicts that 101 (28.45% of the dataset) customers are assigned to Cluster 6, and therefore are likely to achieve an outcome.

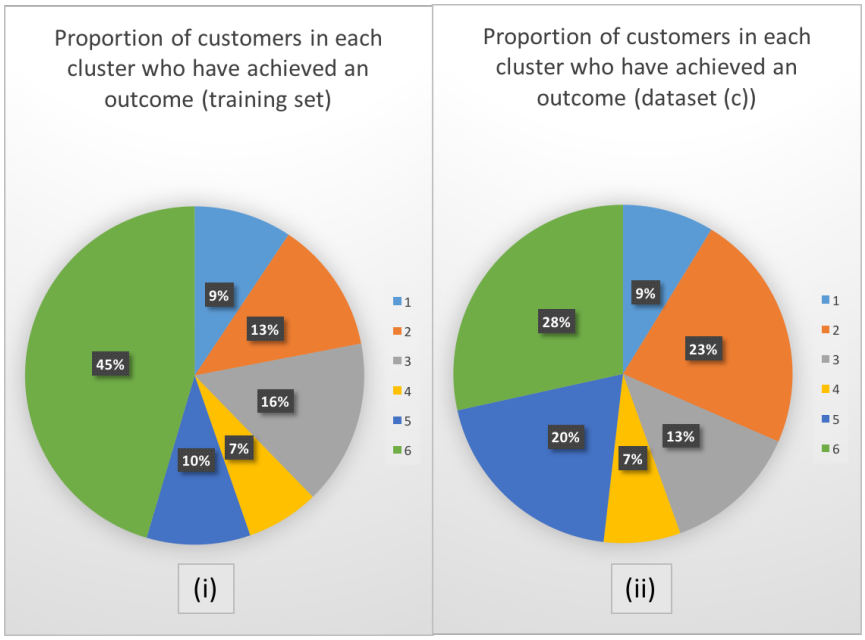

Figure 34: (i) Proportion of customers who have achieved an outcome (trained data)
v.s.
(ii) Predicted proportion of customers who *may* have achieved an outcome (dataset (c))

The same accuracy analysis can be carried out here as in the previous test. The error calculated using dataset (c) is 40%; therefore, the model shows 60% accuracy when clustering customers from dataset (c) i.e., customers who have achieved an outcome but the Outcome feature has been "hidden". This appears at first like a rather poor accuracy, but note that this is not a classification task, and we are not trying to classify customers as having achieved an outcome or not. There are more appropriate measures of accuracy when dealing with hidden data in clustering. One such measure of accuracy would be to calculate the cluster persistence.

Table 11 contains the results used to produce the pie charts in Figure 34.

	Cluster					
	C1	**C2**	**C3**	**C4**	**C5**	**C6**
cluster centroids for Outcome feature	0.2064	0.1063	0.1572	0.0974	0.0929	1
# cutomers in trained clusters	247	649	547	389	581	248
# customers who have achieved outcome (i)	51	69	86	38	54	248
# customers who have achieved outcome (ii)	31	81	46	26	70	101

Table 11: Cluster centres vs. Number of customers per cluster

The cluster persistence is a measure of how many customers were re-assigned to their original clusters after a feature (such as the Outcome feature) is hidden. It was discovered that of the 355 customers, 274 customers were re-assigned to the same cluster when the Outcome feature was "hidden". The only customers who are assigned to new clusters are the 81 customers who are taken out of Cluster 6 (see last row of Table 11). This proves a 77% persistence accuracy. Accurate reproducibility is an important feature of the model when inferring about the future behaviour of customers.

This model can be used to detect, as early as possible, customers who are likely to purchase. If customers are likely to purchase but may abandon their carts in the final stages of the checkout (as do 68.53% of online shoppers [3]), it is important to deter them from abandoning their cart. This can be done through the timely use of communications software, such as that provided by Altocloud, to offer support and ultimately encourage customers to complete the purchase.

Results to focus on:

- 88% accuracy when assigning customers to *appropriate* clusters when features **are not** hidden.
- 60% accuracy when assigning customers to *appropriate* clusters when features **are** hidden.
- 77% accuracy when re-assigning customers to original clusters when features **are** hidden.

6 Conclusions

This section will discuss the conclusions drawn from this final year project. All of the findings will be first summarised, followed by the directions for future work. This section, and the thesis as a whole, will then be brought to a close with some concluding remarks.

6.1 Work Summary

An extensive literature review was carried out at an early stage of this research. The concepts of data mining and machine learning were introduced, followed by a comprehensive introduction to clustering and the K-means algorithm. Some drawbacks and pitfalls of the K-means algorithm were identified, but so too were the benefits of this well developed clustering algorithm. Next, a selection of validation techniques used to validate clusters were described in detail, providing the algorithms to implement such techniques. This was followed by a review of some applications of clustering to a variety of tasks in the field of computing and marketing. This work summarised the diverse range of practical problems in which clustering had shown promising results and it was noted that the applications reviewed were restricted to relevant domains. A selection of existing software packages suitable for data mining and machine learning were then identified and described. This was an important step to discover what software was currently available as implementing a scalable architecture was a key feature of this project. The literature review was concluded with a brief introduction to scalable, distributed architectures using Spark.

Once a thorough literature review had been completed, the design phase of the clustering model began. Early drafts for the design were followed by an implementation which ultimately lead to the improved and final design.

6.2 Directions for Future Work

The are a considerable number of directions in which this project can take. These include:

- **Improving the Model:** The model could be improved by implementing a more suitable clustering algorithm such as the K-median method as described by Desirel *et al.* [25]. A variation of this algorithm would improve distance calculations greatly due to the majority binary values in the dataset. Another upgrade could involve the incorporation of additional features in the training and testing data. This would help build a more comprehensive and complex model. A final upgrade could include the development of an entirely autonomous system, rather than the semi-automatic system that has been develop here.

- **Implementing a Scalable Architecture:** The scalable Spark architecture described in Section 2.6.2 could be fully implemented to build an elastic system. Cluster managers, such as Apache Mesos, could be used to manage clusters.

- **Deploy Model as Feature for Altocloud:** If deemed suitable and of commercial value to Altocloud, the clustering model implemented in this project could be designed to integrate with Altocloud's architecture. This is the ultimate goal of the project and if achieved future work could involve deployment and maintenance of the system.

6.3 Concluding Remarks

Data mining and machine learning are certainly interesting areas of research. With companies such as Google, Microsoft and Facebook spending millions on research into advanced neural networks and deep machine learning, applying machine learning tasks to solve real-world problems will soon be the norm. The results reported in this thesis show that a clustering model can be of commercial value to a business. Additionally, the clustering model, if suitably modified, can be deployed to production by any business to gain a market advantage.

Appendix

Full List	Features used in Initial Design	Features used in Improved Design
account	activities	activities
active	duration	duration
activities	page views	page views
away at date	device	time of day night
business		time of day work
created at date		time of day evening
customer		outcome achieved
customer ID		persona a
duration		persona b
finished at date		persona c
idle at		persona d
idle at date		persona e
ip		persona f
language		referrer yahoo
last active at date		referrer bing
last page		referrer other
last page published date		referrer google
last page title		referrer facebook
last page uri		device desktop
location 2		device mobile
location 3		device tablet
location country		device other
location display name		browser chrome
location latitude		browser safari
location locality		browser ie
location longitude		browser chrome mobile
location organization		browser mobile safari
location postal code		browser firefox
location region		browser gomeza
location source		browser chromium
next best action		browser chrome mobile ios
outcomes 0		browser android
outcomes 1		browser amazon silk
pageviews		browser opera
personas 0		
personas 1		
personas 2		
referrer		
referrer hostname		
referrer known		
referrer medium		
referrer search term		
referrer source		
referrer url		
state		
device browser family		
device browser major		
device browser minor		
device browser patch		
device capabilities webrtc		
device category		
device device		
device os family		
device os major		
device os minor		
device os patch		
device raw		
updated at date		

Table 12: List of features

References

[1] Xuan Huang and Zhijun Song. “Clustering analysis on E-commerce transaction based on K-means clustering”. In: *Journal of Networks* 9.2 (2014), pp. 443–450.

[2] Vicki G Morwitz and David Schmittlein. “Using segmentation to improve sales forecasts based on purchase intent: Which "intenders" actually buy?” In: *Journal of marketing research* (1992), pp. 391–405.

[3] Baymard Institute. *Cart Abandonment Rate*. Accessed 29 October 2015. 2015. URL: http://baymard.com/lists/cart-abandonment-rate.

[4] Stanford University. *Glossary of Terms Journal of Machine Learning*. Accessed 25 March 2016. 1998. URL: http://ai.stanford.edu/~ronnyk/glossary.html.

[5] Phil Simon. *Too Big to Ignore: The Business Case for Big Data*. Wiley, Mar. 5, 2013. 258 pp. ISBN: 978-1-118-64210-8.

[6] Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.

[7] Soumen Chakrabarti et al. “Data mining curriculum: A proposal (Version 1.0)”. In: *Intensive Working Group of ACM SIGKDD Curriculum Committee* (2006).

[8] Nidal Zeidat, Christoph F Eick, and Zhenghong Zhao. “Supervised clustering: algorithms and applications”. In: *University of Houston, Houston, TX* (2005).

[9] Nizar Grira, Michel Crucianu, and Nozha Boujemaa. “Unsupervised and semi-supervised clustering: a brief survey”. In: *A review of machine learning techniques for processing multimedia content, Report of the MUSCLE European Network of Excellence (FP6)* (2004).

[10] Anil K Jain. “Data clustering: 50 years beyond K-means”. In: *Pattern recognition letters* 31.8 (2010), pp. 651–666.

[11] C. F. Eick, N. Zeidat, and Z. Zhao. “Supervised clustering - algorithms and benefits”. In: *16th IEEE International Conference on Tools with Artificial Intelligence, 2004. ICTAI 2004*. 16th IEEE International Conference on Tools with Artificial Intelligence, 2004. ICTAI 2004. Nov. 2004, pp. 774–776. DOI: 10.1109/ICTAI.2004.111.

[12] Richard Dubes and Anil K Jain. “Clustering techniques: the user’s dilemma”. In: *Pattern Recognition* 8.4 (1976), pp. 247–260.

[13] Yoseph Linde, Andres Buzo, and Robert M Gray. “An algorithm for vector quantizer design”. In: *Communications, IEEE Transactions on* 28.1 (1980), pp. 84–95.

[14] Per-Erik Danielsson. “Euclidean distance mapping”. In: *Computer Graphics and image processing* 14.3 (1980), pp. 227–248.

[15] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. “Data clustering: a review”. In: *ACM computing surveys (CSUR)* 31.3 (1999), pp. 264–323.

[16] Siddheswar Ray and Rose H Turi. “Determination of number of clusters in k-means clustering and application in colour image segmentation”. In: *Proceedings of the 4th international conference on advances in pattern recognition and digital techniques*. Calcutta, India. 1999, pp. 137–143.

[17] Lindsay I Smith. “A tutorial on principal components analysis”. In: *Cornell University, USA* 51.52 (2002), p. 65.

[18] Peter J. Rousseeuw. “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53 –65. ISSN: 0377-0427. DOI: http://dx.doi.org/10.1016/0377-0427(87)90125-7. URL: http://www.sciencedirect.com/science/article/pii/0377042787901257.

[19] Dusan Stevanovic, Natalija Vlajic, and Aijun An. “Unsupervised Clustering of Web Sessions to Detect Malicious and Non-malicious Website Users”. In: *Procedia Computer Science* 5 (2011). The 2nd International Conference on Ambient Systems, Networks and Technologies (ANT-2011) / The 8th International Conference on Mobile Web Information Systems (MobiWIS 2011), pp. 123 –131. ISSN: 1877-0509. DOI: http://dx.doi.org/10.1016/j.procs.2011.07.018. URL: http://www.sciencedirect.com/science/article/pii/S1877050911003437.

[20] Ashish Kathuria et al. “Classifying the user intent of web queries using k-means clustering”. In: *Internet Research* 20.5 (2010), pp. 563–581.

[21] CNET. *Google spotlights data center inner workings*. CNET. 2008. URL: http://www.cnet.com/news/google-spotlights-data-center-inner-workings/ (visited on 03/28/2016).

[22] Sachin P Bappalige. *Open source datacenter computing with Apache Mesos*. 2014. URL: https://opensource.com/business/14/9/open-source-datacenter-computing-apache-mesos (visited on 03/29/2016).

[23] Bahman Bahmani et al. “Scalable k-means++”. In: *Proceedings of the VLDB Endowment* 5.7 (2012), pp. 622–633.

[24] David J Ketchen and Christopher L Shook. “The application of cluster analysis in strategic management research: an analysis and critique”. In: *Strategic management journal* 17.6 (1996), pp. 441–458.

[25] Desirel. Massart, Frank Plastria, and Leonard Kaufman. “Non-hierarchical clustering with masloc”. In: *Pattern Recognition* 16.5 (1983), pp. 507 –516. ISSN: 0031-3203. DOI: http://dx.doi.org/10.1016/0031-3203(83)90055-9. URL: http://www.sciencedirect.com/science/article/pii/0031320383900559.