UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Recommender System for Navigating Incentive Programs

by

Ridhaa Benefeld

Submitted in partial fulfilment of the requirement for the degree

Master's Degree (Information Technology)

in the Facility of Engineering, Built Environment and Information Technology

University of Pretoria, Pretoria

June 2019

Recommender System for Navigating Incentive Programs

by

Ridhaa Benefeld

Email: Ridhaa2503@gmail.com

Abstract

More and more companies and insurance brokers are offering wellness and lifestyle incentive programs. These incentives aim to improve the lifestyle of the individuals within these programs. Traditionally, these programs offer financial rewards based on behaviour within these programs. However, over the long term, these programs suffer from low engagement rates and insignificant long-term effects. This poor performance is due to the differences between individuals because users react differently to incentives. This work aims at improving behaviour within these programs by offering a personal behaviour recommender system. The developed recommender was designed to consider the behaviour and level of the respective user within the program and provided recommendations which would result in an ascension within the incentive program. This work consisted of a behaviour classification and behaviour recommendation phase. This classification was done using a random forest classifier to identify desirable and undesirable behaviour within the program. This behaviour is then inputted into the recommender system. A collaborative filter recommender type is used for the recommendation within this work. It should be noted that the recommender algorithm was designed to recommend a minimum action to the user which would ensure the ascension within the program. This limitation on the recommendation was imposed as to ensure that recommended actions are not so significant that users lose interest in the program or injure themselves.

Keywords: Behaviour recommender, collaborative filtering, incentive programs, behaviour analysis, personalised recommendation.

Supervisors : Prof. J. Grobler

Prof. A. P. Engelbrecht

Department : Department of Computer Science

Degree : Master of Information Technology

Acknowledgements

The following people and organisation have been instrumental in ensuring the successful completion of this work.

- Prof. Jacomine Grobler for all the advice and direction during my research.
- Prof. Andries P. Engelbrecht for all the support and valuable suggestions.
- Transnet Engineering for allowing me the time to complete this work.
- Tasneem for all the support and sacrifices over the last two years. Another big thank you for all the formatting help. It has been a long road.
- My parents, for their support. Without them, I would not be here today.

Contents

List of figures

List of tables

Chapter 1
Introduction

Multiply is a wellness program which aims to improve the financial, mental, and physical health of its members. Such improvement is made by creating a deep understanding of the behaviour of its members and proposing ways in which to improve their behaviour. Within the Multiply incentive program (MIP), members are given a score based on the healthy ways in which they live their lives. This score is an accumulation of financial, physical, mental, and safety levels of the member. Based on this score, each member is placed in a different category. These categories are defined as Bronze, Silver, Gold, Platinum and Private Club. Members are incentivised to ascend the respective categories, as more rewards are offered the higher the individual ascends. Currently, users are required to check a series of dashboards to identify areas where points can be obtained. These dashboards provide transparency within the respective categories. However, the current system does not provide users with a list of actions which is specifically tailored to their user profile. This dissertation aims to construct a personalised recommender system which would aid individuals in successfully ascending the program. The recommender is to account for a user's current actions and make recommendations based on the actions which would result in the rise of the user within the MIP.

1.1 Objectives

The objectives of this work are to:

- Create a personalised recommendation system within the MIP in order to aid users to ascend the respective categories.
- Provide users with a personalised list of actions which would result in the rise of users within the MIP.
- Create a personalised recommendation system which recommends the minimum action which would result in moving up one category.

1.2 Contributions

The contribution of this work, to the best of the author's knowledge, is that it is the first personalised behaviour recommender which is implemented within a lifestyle incentive program.

1.3 Limitations

This work is limited to individuals within the MIP and by the data supplied. Therefore, missing data is not accounted for in this work. An example of missing data is financial data within the MIP. Financial data was not supplied in the datasets. However, section 0 shows that financial behaviour does contribute to a user's category.

1.4 Dissertation Outline

Chapter 2 explores the use of recommenders in various fields. The different types of recommenders, as well as the respective methods, are explored. Chapter 3 focusses on providing an overview of the different algorithms used in this dissertation. Different pre-processing algorithms are explored as well as the different classification and recommendation techniques. These algorithms are evaluated critically with the aim of identifying the best performing algorithm for this study. Chapter 4 explores the pre-processing and classification development methodologies in detail. In addition, the classification results are shown and evaluated. Chapter 5 focusses on the development of the recommender system. This chapter investigates the phases, methodologies, and results of the various experimental techniques used within this dissertation. The performance of these algorithms is then critically discussed to evaluate performance within the MIP context. Chapter 6 provides conclusions and recommendations and opportunities for future work.

Chapter 2
Literature review

More and more companies and insurance brokers are offering wellness and lifestyle programs. These programs aim to promote healthy living, increase morale and reduce health costs (Gibson *et al.*, 2017). These programs are enabled by the rise of wearable technologies and the various database stores and tools. Lifestyle and wellness programs enable companies to obtain insight into the behaviour and living patterns of their customers. One of the main challenges with regards to wellness programs is that of motivation to change behaviour. In these programs, incentives are offered for desirable behaviours and penalties for undesirable behaviours. These incentives are highly effective in inducing short-term behaviour changes (Crespin *et al.*, 2016; Gibson *et al.*, 2017). However, over the long term, these incentive schemes suffer from low engagement rates and the long-term effect on behaviour becomes insignificant (Gneezy *et al.*, 2011). Furthermore, not all incentives are effective for all individuals. McComb *et al*. (2016) and Gibson *et al*. (2017) found that the effectiveness of incentives varies largely with the characteristics of the individual being incentivised and therefore, found personalised incentives to be more effective (McComb *et al.*, 2016; Gibson *et al.*, 2017). Between individuals, the behaviour is also not always consistent. Instead, behaviour varies depending on a number of factors. These could be time, location, context and intention (Bentley *et al.*, 2014). In order to change the behaviour of individuals, recommendations need to be personalised to account for this variation in behaviour between these individuals. Section 2 explores the different recommender system applications. Thereafter, the different types, techniques and challenges within the recommendation system space are analysed in sections 2.3 to 2.5. Finally, recommender systems focussed on lifestyle behaviour is explored in section 2.6.

2.1 Recommender system applications

Recommender systems are essentially information retrieval systems. Their primary objective is to find information which is most relevant to the user through various algorithms (Hors-Fraile *et al.*, 2018). Recommender systems have been a necessary development due to the explosion of information systems seen today. Companies are generating and storing vast amounts of data.

Additionally, the rise of e-commerce companies has resulted in more information being stored on users as well as providing more choices than ever before. However, a drawback to this

increase in choice is the risk of information overload. Users would need to navigate large amounts of data in order to find items which interest them. Thus recommenders help navigate these large databases to obtain items which would be most valuable to users (Jiang *et al.*, 2016). Examples of where recommenders are used include Amazon and Netflix. Amazon offers over 400 million products on its website. Therefore, it is imperative for recommender systems to be incorporated within their website architecture. Netflix aims to recommend content which the user prefers in order to keep users engaged. Recommender systems are used extensively in the e-commerce industry. However, they are also used in other industries such as tourism, education and health.

2.1.1 Tourism

Maroulis *et al.* (2016) investigated using recommender systems to determine tourist points of interests for users. These recommendations are built on location-based social networks which connected people through geotagged content such as pictures, texts and video (Maroulis *et al.*, 2016). Jiang *et al.* (2016) investigated a similar recommendation system for the tourism and travel industry. However, in addition to providing only points of interest for users, the sequence in which users should visit these points are recommended. This recommendation is based on social travelogues and community contributed photos (Jiang *et al.*, 2016).

2.1.2 Education

With the adoption of e-learning platforms, more and more users are participating in online courses. These sites produce vast amounts of data based on student and teacher behaviour. The primary objective of educational data mining is to obtain patterns of usage for both the teacher and student as well as to gain insight into student behaviour patterns (Dwivedi *et al.*, 2017). Dwivedi *et al.* (2017) investigated recommendation techniques used in the educational sector, with the aim of recommending additional elective courses based on the student's grades for certain subjects.

2.1.3 Health

Hors-Fraile *et al.* (2018) conducted a survey on the implementation of recommender systems within the medical field. Recommendation systems within healthcare are still in its infancy. One suggestion on why recommender systems have not been widely adopted is that the technology is not clearly defined and widely known by medical personnel. One of the

significant challenges with the use of recommendation systems within the health field is that of legal liability and regulatory compliance (Hors-Fraile *et al*., 2018).

2.2 Datasets for recommender systems

The datasets used within recommender systems greatly depend on the application or sector within which the recommender system is applied. The rise of social media and wearable technologies from various data sources can be aggregated to provide additional information and improve the performance of recommender systems. Balicki *et al.* (2015) investigated using internet of things (IoT) data to model the behaviour of individuals for a smart city application. In this research, IoT data was defined as data from mobile phones, tablets, cameras, smart cars, and wearables. These data sources were aggregated to obtain information on not just the behaviour of individuals within cities, but also the city infrastructure (Balicki *et al.*, 2015).

It should be noted that a recommendation system is not implemented in the research, but a behaviour modelling system. Behaviour modelling or profiling can, however, be seen to be a subcomponent of implementing a behaviour recommendation system and therefore, is relevant to this study. Another behavioural modelling system considered only using wearables to develop a robust activity recognition method which accurately identified physical activity and activities of daily living (Sztyler, 2017). Lim *et al.* (2017) developed a mobile wellness management system for healthy living which used both wearables and location data from mobile phones. This system used these datasets and aggregated them into a system which provided behavioural recommendations based on location context data and behavioural data (Lim *et al.*, 2017). Most of the literature for behavioural recommender systems use wearables as a data source as this source gives the highest resolution into human activity. However, some systems used pictures and social media data to obtain location and context-based information for recommender systems. These systems take into account the photos of individuals and their friend groups in order to offer recommendations (Jiang *et al.*, 2016; Kesorn *et al.*, 2017). An advantage of using social media is that it provides a wealth of upfront information and therefore, reduces the scope of the data gathering phase.

With the current growth in the information space and as the applications for recommendation systems grows, datasets which recommenders are to act on will become enormous. An additional issue which is to be addressed is that of computation. Recommendation systems can be computationally expensive (Sahu *et al.*, 2015). In addition, traditional databases do not provide in-house support for different recommendation algorithms. Therefore, overhead may

be required for existing databases, because data would need to be extracted, input into the algorithm, and then pushed back to the database (Sarwat *et al.*, 2017). Both PostgreSQL and Hadoop solutions are proposed to remedy this in Sahu *et al.* (2015), Sarwat *et al.* (2017), and Simović (2018). Although necessary, infrastructure considerations will not be the focus of this research and are not explored further in this dissertation.

2.3 Types of recommender systems

There is no one size fits all recommendation algorithm as different applications require different algorithms. Recommender systems can either be personalised or non-personalized. Details of these recommender types are discussed in detail in this section.

2.3.1 Non-personal recommender systems

Not all recommenders are personalised to specific users. In non-personalized recommender systems, recommendations are made based on the entire user group. These recommenders could be in the form of identifying the most popular items across all users (Khatwani *et al.*, 2017). This system is essentially a counting system, where most popular items are shown. These counting systems can be made more specific with regards to location, time of the year or other characteristics of the user or item. Non-personal recommender systems are useful for cold start situations (Tikk, 2016). A cold start is when a new user or item is added to the recommender system, and the algorithm has no ratings or history of making a recommendation. The cold start problem is elaborated upon in section 2.5.1.

Non-personal recommenders are not without its drawbacks. The most obvious being that items on the most popular list may not be of interest to the user. There is also an issue of visibility, where items which are not on the most popular list would remain hidden to the user. In order to provide more tailored recommendations for both users and items, a personal recommendation system is to be employed.

2.3.2 Personal recommender systems

As opposed to non-personal recommender systems, personal recommender systems use information on the user, user group, items and item groups in order to find similar users and items. Similar items or users are found, because it is assumed that similar users are more likely to prefer the same items, and similar items, are most likely to be preferred by specific users. Assumptions regarding personal recommenders are defined by Ren *et al.* (2015):

Assumption 1:

If two users rated a set of items similarly, then their future rating or consumer behaviour will tend to be similar too.

This assumption forms the basis of collaborative filtering techniques. Recommendations are made based on historical ratings of respective users.

Assumption 2:

Users prefer items similar to those previously liked.

This assumption simplifies the recommendation algorithm by refining the scope to a small number of items. On this assumption, content-based recommendations are made.

Assumption 3:

A user's item preference or an item's desirableness does not change over time.

This assumption along with assumption 2 is used to reduce the size of the sample space. However, assumption two and three will result in no new items being recommended and therefore, may result in the user losing interest. Temporal recommendation systems have been developed to address this problem (Ren *et al.*, 2015). These systems are discussed in section 2.4.5.

2.4 Recommendation techniques

This section explores the various recommendation techniques in greater detail. The aim is to explore the various recommender system techniques along with the advantages and disadvantages of each technique.

2.4.1 Problem formulation

Recommender systems comprise of a set of users and items. These can be seen below where $\boldsymbol{U} = \{u_1, u_2, \dots u_n\}$ represents a set of users $\boldsymbol{U}$ and $\boldsymbol{I} = \{i_1, i_2, \dots i_m\}$ represents a set of items. These user and item sets are combined in order to form a user-item rating matrix R_o

$$\boldsymbol{U} \times \boldsymbol{I} = R_o \tag{1}$$

where R_o is the user-item rating matrix, characterised as

$$\begin{bmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & \ddots & \vdots \\ r_{m1} & \cdots & r_{mn} \end{bmatrix}$$

and r_{nm} denotes the rating of item *m* by user *n* (Ren *et al.*, 2015).

2.4.2 Collaborative filtering

Collaborative filtering is one of the most commonly used recommender algorithms as it is found in almost all sectors. Collaborative filtering matches similar users based on their ratings. Recommendations are then made based on these ratings (Ren *et al.*, 2015). Ren *et al.* (2015) present two different types of collaborative filtering recommendation systems, namely memory-based collaborative filtering and model-based collaborative filtering. These are discussed below.

Memory-based collaborative filtering

This method comprises a two-step process: a neighbourhood selection step and a rating aggregation step. In the neighbourhood selection step, the neighbourhood ($n_k(u_x)$) of the active user U_x is defined by calculating user similarity. Thereafter, the rating step calculates the ratings r_{xi} and r_{yi} of an item. Two popular similarity algorithms used for finding the neighbourhood is the Pearson correlation coefficient (PCC),

$$sim(u_x, u_y) = \frac{\sum_{I_i \in I_{xy}}(r_{xi} - \overline{r_x})(r_{yi} - \overline{r_y})}{\sqrt{\sum_{I_i \in I_{xy}}(r_{xi} - \overline{r_x})^2 \ \sum_{I_i \in I_{xy}}(r_{yi} - \overline{r_y})^2}} \tag{2}$$

and the Cosine Correlation (COS),

$$sim(u_x, u_y) = \frac{\sum_{I_i \in I_{xy}} r_{xi} r_{yi}}{\sqrt{\sum_{I_i \in I_{xy}} {r_{xi}}^2 \ \sum_{I_i \in I_{xy}} {r_{yi}}^2}} \tag{3}$$

where $I_{xy} = \{I_i \in I \mid r_{xi} \neq \emptyset, r_{yi} \neq \emptyset\}$ denotes a set of items where item I_i is an item in set *I* that the user u_x and u_y have both rated. Therefore $\overline{r_x}$ and $\overline{r_y}$ are the average rating of u_x and u_y, respectively (Resnick,1994; Ren *et al.*, 2015). Once the neighbourhood is selected, the rating scores are aggregated for specific items. This is done using the following equations:

$$\hat{r}_{ai} = \frac{1}{K} \sum_{u_x \in n_k(u_x)} r_{xi} \tag{4}$$

$$\hat{r}_{ai} = \rho \sum_{u_x \in n_k(u_a)} sim(u_a, u_x) r_{xi} \tag{5}$$

$$\hat{r}_{ai} = \bar{r_a} + \rho \sum_{u_x \in n_k(u_a)} sim(u_a, u_x)(r_{xi} - \bar{r_x}) \quad (6)$$

where $\bar{r_a}$ is the average rating and K is the total number of rated items of the user u_a.

$\rho = \frac{1}{\sum_{u_x \in N_k(u_a)} sim(u_a, u_x)}$ serves as the normalisation factor. Recommendations can either be an average rating across all ratings for that item as given in equation (4) or a weighted majority prediction as given in equation (5). Equation (6) expands upon these equations by replacing the absolute value with the difference between the average rating and corresponding user ratings (Ren *et al.*, 2015).

Model-based collaborative filtering

Model-based collaborative filtering constructs a model of the rating matrix and thereafter uses these models to make predictions for items. These models span a wide range of types from supervised learning, unsupervised learning and matrix decomposition. These methods are further elaborated on in Ren *et al.* (2015).

Although the model-based collaborative filtering algorithm is used in many different sectors, it is not without drawbacks. One problem is that of scalability. As the user and item sets grow, so does the resultant rating matrix. Therefore, this method is computationally costly (Ren *et al.*, 2015).

2.4.3 Content-based recommendation

Content-based recommendation systems work differently to that of collaborative filtering in that it does not take the scores of other users into consideration. Instead, content-based recommendation algorithms construct a classifier for each user using various types of features. A comparison is made between the contents of the items and the user preference model, thereafter, items which have the highest degree of commonalities are recommended. For text-based items, a content vector ($\vec{v}_I$) could be constructed along with a preference vector ($\vec{p}_a$) for users. Thereafter, the similarity between the two vectors can be computed using the cosine distance formula,

$$sim(I_i, u_a) = \cos(\vec{v}_I, \vec{p}_a) = \frac{\sum_{K=1}^{N} v_{ik} p_{ak}}{\sqrt{\sum_{k=1}^{N} v_{ik}^2 \sum_{k=1}^{N} p_{ak}^2}} \quad (7)$$

where N is the dimensionality of the respective item. It is seen that v_{ik} and p_{ak} are the k^{th} element of the vectors $\vec{v}_I$ and $\vec{p}_a$, respectively. For text-based items, a vector will suffice for the algorithm. However, pre-processing and data mining will have to be done in the cases where items were video, pictures or songs. Another issue with this content-based recommendation algorithm is that of diversity, where only items in the preference vector will be recommended. This creates the chance that the user may lose interest (Baeza-Yates, 1999; Ren *et al.*, 2015).

2.4.4 Context-based recommendation

Context-based recommendation inputs additional information into the recommendation model which will affect the model. From work done in Cui *et al.* (2016), humans can be seen to behave differently based on a number of factors. These factors include time, location, and whether the user is alone or in a group. Additional information is incorporated into the traditional recommendation system to improve the recommendation algorithm. Either the input data will be pre-processed, where data matching the context will be chosen, or post-processed, where recommendations not matching the context will be removed (Ren *et al.*, 2015). An example of context pre-processing can be seen in Maroulis *et al.* (2016) where tensor factorisation is used to create a context-aware point of interest recommendation system.

2.4.5 Temporal recommendation

Assumption 3 stated that user-item preferences and item popularity do not change over time. This assumption is introduced to simplify the recommendation systems. In practice, individuals' transition between phases of interest. Items also transition through phases, where an item can be trendy over a specified period, and thereafter, fall out of popularity. Due to this phenomenon, temporal recommendation systems have been developed. Temporal recommendation systems operate under the following assumption:

Assumption 4:

User preferences change over time, and the temporal patterns of their preferences are similar to similar items (Ren *et al.*, 2015).

Within temporal recommender systems, there are two types of recommender information. Those aimed at users and the other aimed at items. For users, the following information is needed:

- User age – How long a user has been in the recommender system;

- User purchasing time – The time the user rated the item; and
- User preference pattern – The user's rating pattern over time.

For items, the following is needed:

- Item age – How long an item has been rated by users;
- Item launching time - Production time of the item; and
- Item popularity - How popular an item is during a specific period.

Temporal recommendation systems have not received much research interest until the TimeSVD++ algorithm, which was developed for the NETFLIX progress award. $\hat{r}_{ai}$ is calculated as:

$$\hat{r}_{ai} = \mu + b_i(t) + b_x(t) + q_i^T \left(p_x(t) + |R(u_x)|^{-0.5} \sum_{j \in R(u_x)} y_j \right) \quad (8)$$

where μ is the overall average rating, $b_i(t)$ denotes the time changing item bias, $b_x(t)$ denotes the time changing user bias. Both $\vec{q}_i$ and $\vec{y}_i$ are item vectors in the joint latent factor space and $|R(u_x)|$ is the item set rated by the user u_x and p_x, where p_x is used to capture changes over time.

2.4.6 Graph-based recommendation

For graph-based recommendation systems, a bipartite graph is constructed between users and items. Users are linked to items and items are thereafter linked to users. Figure 1 illustrates a graph-based recommendation system.

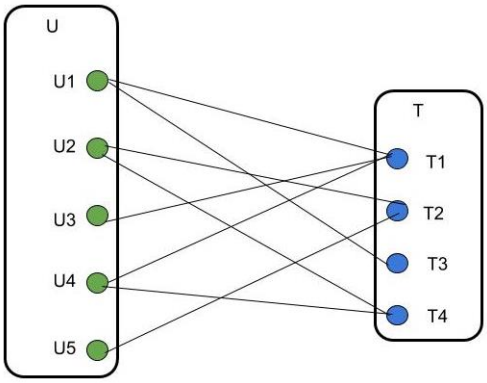


Figure 1: A bipartite graph showing the links between the user and item sets.

Figure 1 shows that both similar users and items can be identified by identifying the links between the respective sets. An underlying assumption with graph-based recommendation systems are elaborated upon below:

Assumption 5:

The fact that an individual rated a respective item indicates that the item is favourable to the user and this favourability can be distributed through the links of a bipartite graph (Ren *et al.*, 2015).

A drawback with this graph-based recommendation systems is that popular items within the environment will be amplified amongst users because popular items will have more connections to respective users (Zhou *et al.*, 2007). Therefore, graph-based item recommendations will depend mainly on the popularity of an item within a specific group.

2.4.7 Trust-based recommendation

Collaborative filtering methods compare input users to that of similar users in order to provide a recommendation. However, trust-based recommendation systems expand upon this concept, whereby recommendations are made based on users who are known or familiar to the input user. With the explosion of social media networks, trust-based recommendation methods have become more viable. With a trust-based recommendation, the cold start problem can be addressed for both new users and items (Massa *et al.*, 2007; Guy *et al.*, 2009).

For users within a social network, two recommendation methods exist, namely global and local. For global recommendations, the trustworthiness of a user is compared to that of the entire community. This trust-based recommendation type is similar to that of popular recommendation systems where the entire group is used to obtain a recommendation on the most popular item. For a more personalised recommendation, local recommendations should be employed. A representation of a trust-based recommender is illustrated in Figure 2.

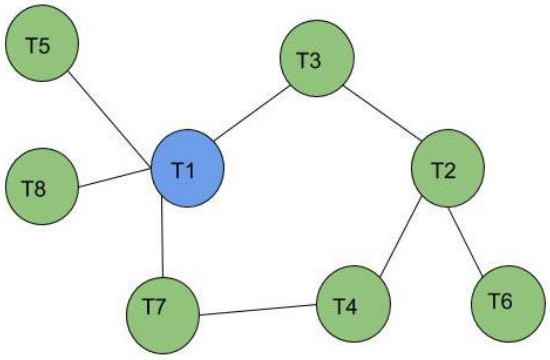


Figure 2: Trust-based graph-based network showing the link between active users and familiar friends.

Trust-based recommendations consider individuals within social networks to obtain trust information on other users within the chain. Trust is determined based on the social chain length and weighted trust average between individuals within the network. An issue with the trust-based recommendation is that recommendations cannot be made in cases where users have no local network of other users to draw from.

2.4.8 Sequential recommendation

Sequential recommendation systems consider not just the recommendation of individual items, but a recommendation into a sequence of items is explored. Sequential recommenders are investigated in He and McAuley (2017), where Markov chains and matrix factorisation are used.

2.4.9 Hybrid methods

Each recommendation system presented above has its advantages and disadvantages. In order to overcome the limitations of the various recommendation systems, hybrid methods have been developed. Hybrid methods consist of an ensemble of recommendation algorithms and are implemented and aggregated using a voting system (Ren *et al.*, 2015).

2.5 Challenges within recommender systems

The previous section presented several recommender algorithms. However, these recommendation systems are not without their drawbacks. Two common problems are the cold start problem and sparsity in the resultant user-item dataset.

2.5.1 Cold start problem

A common problem with recommender systems is the cold start problem. The cold start problem is defined by there being insufficient data present to make a recommendation. This problem can either be from the user or item perspective. The cold start user problem also called the new user problem, occurs when the user is new to the system and does not have any rating history.

Conversely, the new item problem is when a new item is added to the system where the item does not possess any ratings. Because of the lack of information, recommender systems are unable to make recommendations (Kesorn *et al.*, 2017). The most primitive method to overcome this problem is to wait for data to be collected on the individual. This problem can also be addressed by requesting users to provide an initial preference on respective items. These preferences are updated over time as the user adds more ratings to the system (Dharia *et al*., 2018). Another method is to initially adopt a trust-based system where social networks are used to find the initial preferences of individuals (Ren *et al.*, 2015).

2.5.2 Sparsity in datasets

Another issue within the recommender system space is that of sparsity within datasets. It is impractical to expect a single user to rate every item within an item set. Therefore, the larger the user and item sets become; the more sparsity exists within the user-item datasets. In order to address this problem, hybrid approaches between content-based and collaborative filtering have been developed to remedy the problem. Another method to remedy the data set sparsity problem is Markov chains and matrix factorisation (Tang *et al.*, 2013; He *et al.*, 2017).

2.6 Lifestyle behaviour recommender systems

Recommender systems are used to motivate individuals to behave in a certain way. These recommendations could be to motivate them to buy a specific product, visit a certain location or complete a particular educational course. This section investigates the recommendation systems aimed at improving lifestyle behaviour. These recommendations could either be focused on improving physical activity or improved calorie intake. Behaviour recommendation systems are different from traditional recommender systems in that the observed behaviour is highly variant depending on the individual observed.

Behaviour can be categorised to be temporal in that it is subject to change over time and context of the individual (Trang Tran *et al.*, 2018). However, behaviour patterns can be seen to repeat

over time. Therefore, recommendation systems implemented within this space can take temporal effects into account. This section explores the different methods of behavioural recommender systems, and the advantages and disadvantages of each method are considered and evaluated.

2.6.1 Healthy behaviour recommender system

Yürüten (2017) proposes a recommender system to recommend behaviours to aid individuals to achieve their health goals. Two constraints are imposed on the system, namely that the recommendations were to motivate steady improvement and that the recommended actions should not be too significant as to discourage or injure individuals. Yürüten (2017) proposes a method called FactorHabiTS which input wearable sensor data from individuals and provides recommendations thereafter. User data are measured through wearable sensors to monitor the effects of stimuli. The steps shown in Figure 3 are proposed to achieve behaviour recommendation.

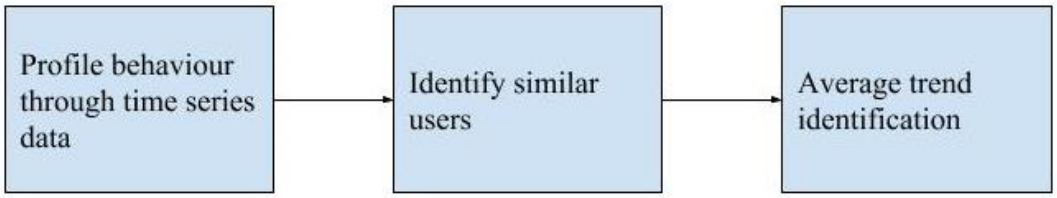


Figure 3: Flow chart showing the steps within the FactorHabiTS method for healthy behaviour recommendation.

Behaviour profiling

Raw time series data is used to obtain a temporal pattern of the user. This work is developed and explored in Yürüten *et al.* (2014). The behavioural profiling process is shown in Figure 4.

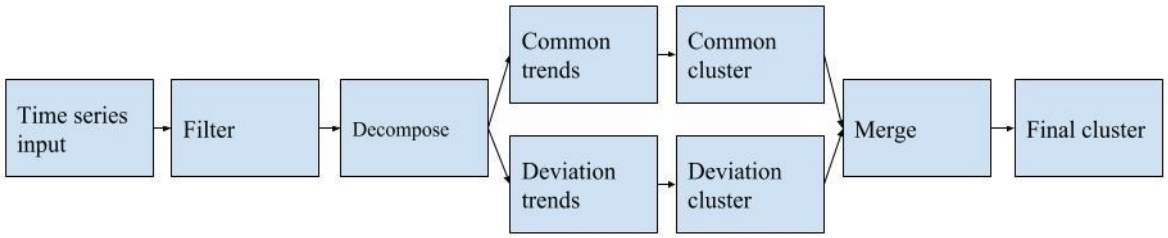


Figure 4: Behavioural profiling algorithm presented in *Yürüten, (2017).*

Figure 4 shows that the time series data is first filtered to remove noise within the system. In this behaviour profiling step, a Hodrick-Prescott filter is implemented (Ravn *et al.*, 2002). After the filter has been implemented, the data is decomposed into two matrices, namely the common

behavioural trends matrix and behavioural deviations matrix. Once split, these matrices are clustered individually to group the common behaviours and deviations. The matrix split and clustering algorithm aim to obtain the common behaviours within the sample space. These clusters are then collected and aggregated to obtain the final clusters. These clusters form the required temporal clusters (Yürüten, 2017).

Intervention profiles

As mentioned, the source of the data used in the system is obtained from wearable technology. Therefore, it allowed for the development of intervention profiles amongst the users. In the intervention profile phase, the behaviour of a user is measured before and after a recommendation is made. This behaviour is analysed immediately after the recommendation is made and over time. The before and after recommendation behaviour analysis are shown in Figure 5.

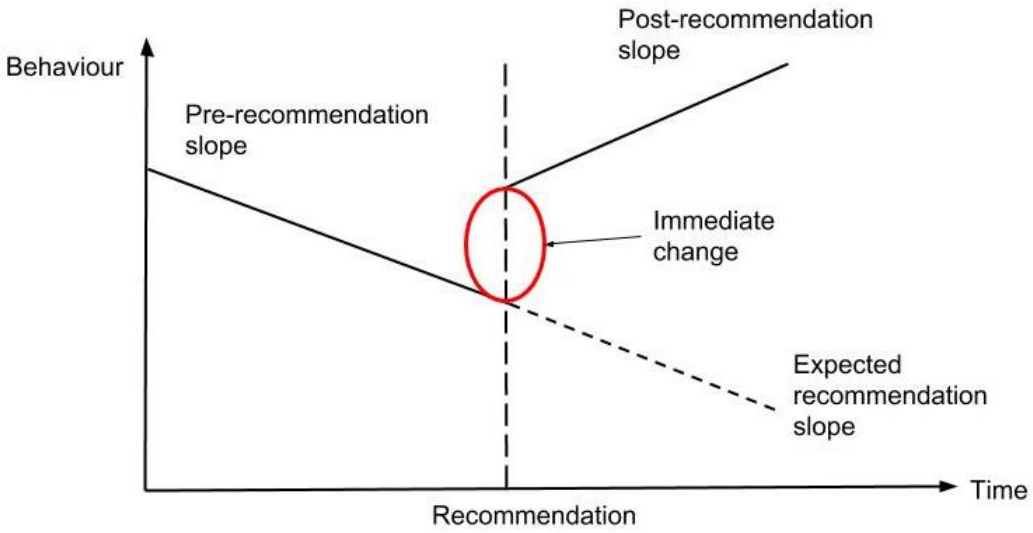


Figure 5: Graph showing the behavioural change before and after the intervention.

Figure 5 shows the different behaviour slopes before and after a recommendation is made. The immediate change is noted, as well as the change in behaviour over time. The definition of the behaviour slopes varies with the dataset. Therefore, a positive slope makes sense when physical exercise is being measured; however, in the event of calorie intake, a negative slope is used (Yürüten, 2017). Once the relative slopes are computed, the respective users are categorised into three groups. These groups are:

1. Responders – users who adopt the behaviour and show a steady increase in activity after the recommendation is made.

2. Temporary responders – users who respond immediately to the recommendation, but do not maintain the recommended behaviour over time.
3. Non-responders – users who do not change their behaviours after the recommendation.

Therefore users who have a pattern similar to that of the input user and responds positively to the recommendation are used for the recommendation (Yürüten, 2017).

Output average trends of closest successful users

In this step the temporal behaviours of the users which best match that of the input user are averaged and then used as the output of the recommendation system.

The advantages of the FactorHabiTS method are that recommendations are made based on users who responded positively to recommendations. Therefore, users, even if they have been characterised as non-responders would be offered recommendations which will encourage positive change. A drawback of the method is that in cases where no feedback from users is accessible, intervention profiles may not be possible to obtain. The lack of intervention profiles will, therefore, affect the quality of the recommendation.

2.6.2 Introspective retrospective behaviour recommendation

Farrell *et al. (*2012) investigate the use of an introspective retrospective recommendation (IRR) method to facilitate healthy decision making. The recommender was mainly focused on calorie intake and physical activity. This recommendation method functioned by making use of an individual's health behavioural history to recommend future actions. IRR methods assume that an individual's lifestyle patterns are stable and change slowly over time. This method uses behavioural patterns of the past and compares past behaviour to current behaviour. Recommendations are made on the difference between these observed differences (Farrell *et al.*, 2012). For example, if an individual wanted to achieve a certain weight, the algorithm refers to a period or *prior self* where the individual was at the desired weight and compares the behaviour of the prior self to the current behaviour. Recommendations are then made to either adopt past habits or to reduce newly formed habits. As mentioned, a common recommender system is collaborative filtering; however, an IRR method may yield poor results within behavioural recommenders because of:

- Sparsity in the space – behaviours may be specific to different users and therefore introduce sparsity in the dataset;

- Diverse user characteristics and goals - all users may not wish to obtain the same result. For example, individuals may have different weight targets;
- Varied contexts – behaviours may be relative to an individual's context, location or availability of items; and
- Distinctiveness – individuals may be attracted to items which are entirely different from that of other individuals.

The advantages of an IRR system are that each user is considered in isolation. This consideration reduces the sparsity of the dataset, and only considers items in the particular user's history. Goals, context and distinctiveness, can be analysed and the history of changes are recorded over time. However, a disadvantage of this method is that it is highly susceptible to the cold start problem as no recommendation can be made if there is no history. Another drawback is that the basis of the algorithm is based solely on an individual user's history. Therefore, no novel actions will be recommended to the user which may cause the user to lose interest (Farrell *et al.*, 2012; Yürüten, 2017). A problem may also arise if the user never met their goals in the past. In order to remedy these drawbacks, a hybrid system could be implemented (Farrell *et al.*, 2012). The IRR recommendation method works well where there is a high fluctuation of desirable and undesirable behaviour by the user.

2.7 Social recommendations for personalised fitness assistance

Dharia *et al.* (2018) develops a personalised recommender system that considers multiple sources of information such as wearable sensors, calendar, interests and social networks. The system then uses a classifier algorithm to classify an individual's behaviour into inactive or active. This classification is input into a hybrid recommender system which recommends both new activities and similar users (gym buddies in this case) to a user of interest (Dharia *et al.*, 2018). The algorithm of this method is shown below.

Classifier algorithm

The method takes in accelerometer data from a user's wearable device; this data is then grouped according to time. This accelerometer data is then input into a gradient boosted decision tree in order to classify the behaviour of the user. The result of the classifier is inputted into the recommendation engine thereafter (Dharia *et al.*, 2018).

Recommendation engine

This method aims to recommend activities that the user may enjoy as well as a similar gym. Recommendations are made using a collaborative filtering technique, using the Pearson correlation coefficient equation 2. This personalised recommender system finds activities and users that are similar and would benefit from working together. Like all collaborative filtering techniques, the system suffers from the cold start problem. The personalised recommender system asks for user input to remedy this problem. Users are asked a series of questions in order to produce an activity preference score. This score was used to find fundamental similarities. This score, however, is updated each time the personalised recommender system is used using

$$pr_{ui} = \max\left(pr_{ui}, pr_{ui} + \frac{|a_{ui}|}{\sum_{j=1}^{n}|a_{ui}|}\right) \tag{9}$$

where pr_{ui} represents the preference score, and $|a_{ui}|$ represents the number of times a user performed an activity. The preference score is not updated until the user participates within the system. This personalised system was found to successfully recommend activities to individuals. However, the evaluation group is found to be small with only 24 people participating. The system was also found to be in its prototype phase and therefore, the social network element was not tested at scale.

2.8 Summary

This chapter explores the basis of behaviour and motivators in individuals. It was found that the behaviour of an individual is subject to not just personality but depends mainly on context. These contexts could be time, location, background, and motivation. Once understood, the various recommender systems are investigated. This investigation is done to identify the various sectors in which recommender systems have been employed and the reason thereof. Several recommender systems are explored to gain an understanding of their operations. The typical drawbacks of recommender systems, such as the cold start and sparsity problem are explored along with possible ways in these problems can be overcome. Behavioural recommender types are also analysed. Almost all recommender systems implemented in the behavioural analysis space took temporal behavioural patterns into account. Temporal systems in the behavioural analysis space profiles users in order to identify normal patterns to that of deviations. The literature outlined in this chapter provides a firm foundation to build on in this dissertation. This dissertation differentiates itself from the literature in that it aims to develop

a behaviour recommender system within an incentive scheme which does not require feedback from the user.

Chapter 3
Algorithm background

This chapter explores the various algorithms which are to be used at various stages in the development of the behaviour recommendation system. The various balancing and feature selection algorithms are explored in order to gain an understanding of their operation and requirements. After that, the various classifiers are investigated to gain an understanding of their advantages and disadvantages. This evaluation is imperative in order to ensure that the most optimum performing algorithms are used within this dissertation. Section 3 focusses on the analysis of the various balancing algorithms. Thereafter, section 3.2 explores the various feature selection algorithms. Finally, the various supervised and unsupervised methodologies, are explored and evaluated in section 3.3.

3.1 Balancing algorithms

Dataset balancing is done in the event where there is a disproportionate representation of classes. Balancing of datasets is essential to reduce bias within the machine learning model. Dataset balancing is done by either under-sampling or over-sampling the dataset or a combination of these methods. This section explores and evaluates the various dataset balancing algorithms.

3.1.1 Under-sampling

Balancing a dataset by undersampling can be achieved by randomly sampling the majority class to eliminate additional information. The major drawback of this method is that potentially vital information is removed from the model (Kotsiantis *et al*., 2006). Another method for under-sampling a respective dataset is the TOMEK Link Removal (TLR) method. A visual representation of this algorithm can be seen in Figure 6.

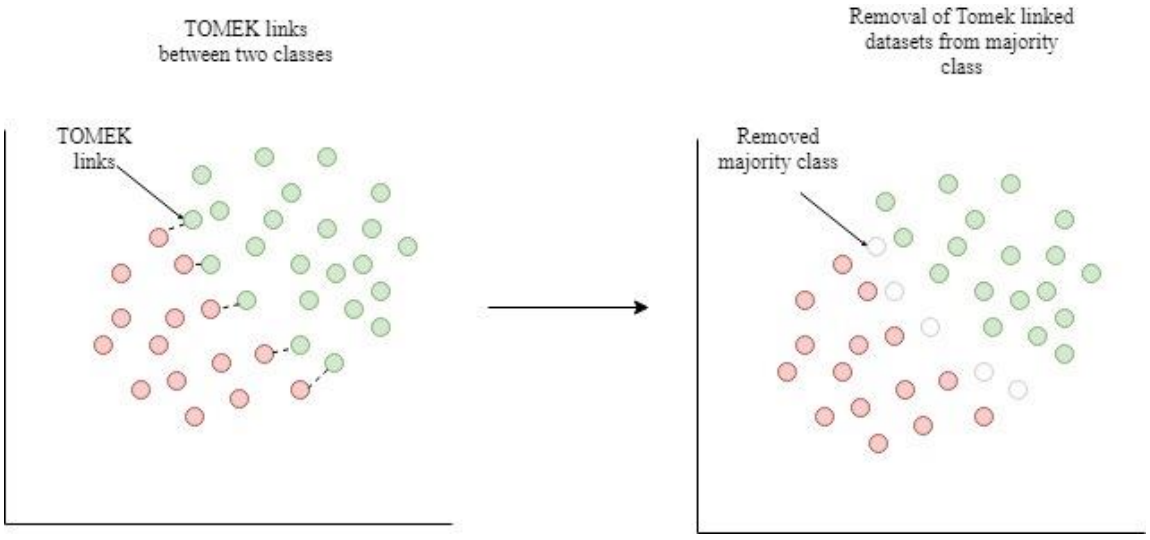


Figure 6: TOMEK link removal representation

A TOMEK link is defined as a pair of data points which are each other's nearest neighbours but which belong to different classes. This method removes these links from the dataset, thereby reducing the borderline data points and creating a more significant separation between clusters. The TLR can be expanded upon in order to only remove links which belong to the majority class (More, 2016).

3.1.2 Over-sampling

In contrast to under-sampling, over-sampling can be achieved by randomly replicating the data points within the minority class. However, random resampling and replication of the minority class increase the risk of overfitting. Another drawback of over-sampling is that additional computation is required to process the target dataset (Kotsiantis *et al*., 2006). Another way of oversampling is the synthetic minority oversampling technique (SMOTE) method (More, 2016). As the name suggests, this algorithm functions by creating new synthetic data points between two observed data points. This process is illustrated in Figure 7.

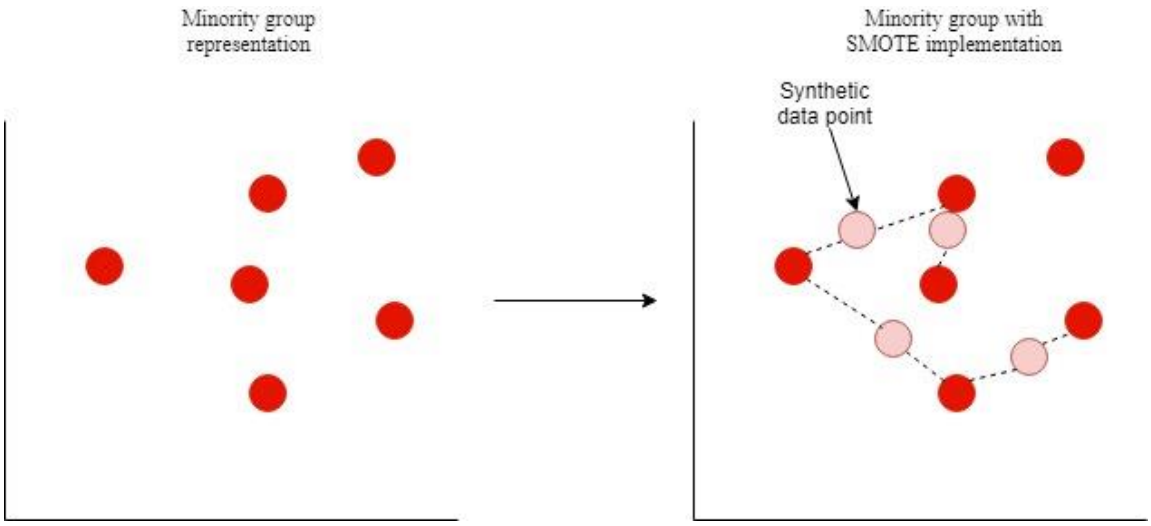


Figure 7: SMOTE implementation representation

Figure 7 shows the implementation of the SMOTE algorithm on the minority set within a dataset. This algorithm operates by implementing a K-Nearest Neighbour (KNN) algorithm for the entire feature space. The distance between the nearest neighbours are then identified, and the distance is multiplied by a random number between zero and one. A new data point is then generated along with the line between the two data points. This process is then repeated across all points within the dataset (More, 2016).

3.1.3 Combinational methods

An additional approach to either under-sampling or over-sampling is to implement a combination of both under-sampling and over-sampling, as this can yield better results (More, 2016). A hybrid approach is the SMOTE+TOMEK link removal algorithm (More, 2016; Lemaıtre *et al.*, 2017). This algorithm removes the TOMEK links of the majority classes and adds synthetic points to the minority classes.

3.2 Feature selection

Feature selection is essential to reduce the dimensionality of high dimensional datasets (Goswami *et al.*, 2014; El Aboudi *et al.*, 2016). Different approaches to feature selection include either a filtering method, wrapper method or hybrid method. These feature selection approaches aim to reduce the dimensionality of the dataset without compromising the information content (Goswami *et al.*, 2014). The details of these algorithms are described in the sections below.

3.2.1 Filter methods

Filtering methods implement statistical measures to rank features within the respective dataset. A threshold is then employed, and all features below this threshold are removed in order to reduce the dimensionality of the dataset (Goswami *et al.*, 2014; El Aboudi *et al.*, 2016). Filtering feature selection statistical measures include the Fisher score, Pearson correlation, Kendall correlation, Spearman correlation, Mutual information, Chi-Squared, and Count-based (Astala *et al.*, 2018).

3.2.2 Wrapper methods

Wrapper feature selection methods are different to filter based algorithms in that the method makes use of a learning algorithm to evaluate the feature selection subset (El Aboudi *et al.*, 2016). This feature selection approach is illustrated in Figure 8.

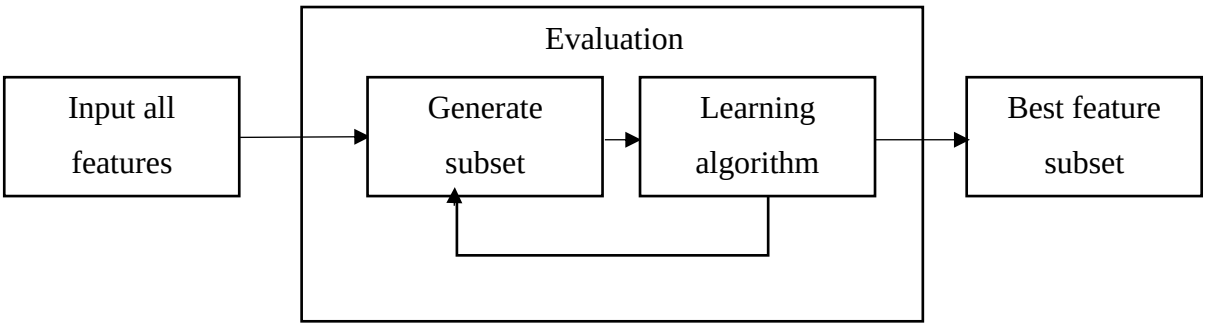


Figure 8: Feature selection wrapper method. This figure is adapted from El Aboudi *et al.* (2016).

Figure 8 illustrates the steps of wrapper methods. Wrapper methods input all features into the algorithm and thereafter generates a subset of the original dataset which is provided as an input to a learning algorithm. The performance of the algorithm is then evaluated, and used as the quality of the feature subset. The feature subset which produced the best performance for the learning algorithm is selected as the optimal set of features. This feature subset is output from the learning algorithm.

3.2.3 Hybrid methods

A hybrid approach can also be taken, where both filtering and a wrapper method is implemented. The filtering method is used first to reduce the feature dataset using statistical measures. Thereafter, the feature subset is input into the wrapper feature selection method. The

most optimum feature subset is output from the learning algorithm. (Goswami *et al.*, 2014; El Aboudi *et al.*, 2016)

3.3 Classification algorithms

One of the components of a behaviour recommender system is a classifier, used to classify existing behaviours. The behaviour classifier aims to observe desired and undesired behaviours from a dataset. This section explores the different classes of classification which is employed in behaviour classification systems.

3.3.1 Unsupervised classification

Unsupervised classification approaches aim to find connections or patterns within an input dataset without input from a teacher (Engelbrecht, 2007). This section explores the self-organising map (SOM).

Self-organising maps

A self-organising map (SOM) is an unsupervised artificial neural network (ANN) (van Heerden, 2017). A SOM can be defined as a scaling method used to project *I*-dimensions into a discrete two-dimensional output space. This output space consists of a map structure of either orthogonal or hexagonal connected neurons (Engelbrecht, 2007; van Heerden, 2017). During the training of the SOM, the map learns and represents both the distribution and topology of the input space (Westerlund, 2005). This dimensional scaling is done in order to reduce the complexity of the set of inputs, as well as to represent the input with a lower dimensionality (van Heerden, 2017). It should be noted that there are many variations of this algorithm, such as neural gas, batch training and growing maps. However, only the operation of the stochastic SOM is explored in this section.

The first step within the stochastic SOM is to initialise the weights of the neurons. This weight initialisation can either be assigned randomly, using the principal component of the input vectors or by constructing a hypercube to uniformly cover the majority of the data (Engelbrecht, 2007; van Heerden, 2017). The next step is to adjust the weights of the neurons iteratively within the map. A SOM is trained using a competitive learning methodology, where a single winning neuron or best matching unit (BMU) is determined per iteration. The BMU is the neuron whose weight vector best matches the input training vector. This BMU is identified most commonly using the Euclidean distance between the respective neurons and input vectors. Upon identifying the BMU, the weights of the neurons of the entire map are adjusted. However,

the BMU responds more strongly than the other neurons which are further away within the SOM (Westerlund, 2005; van Heerden, 2017). The training process can be seen in Figure 9.

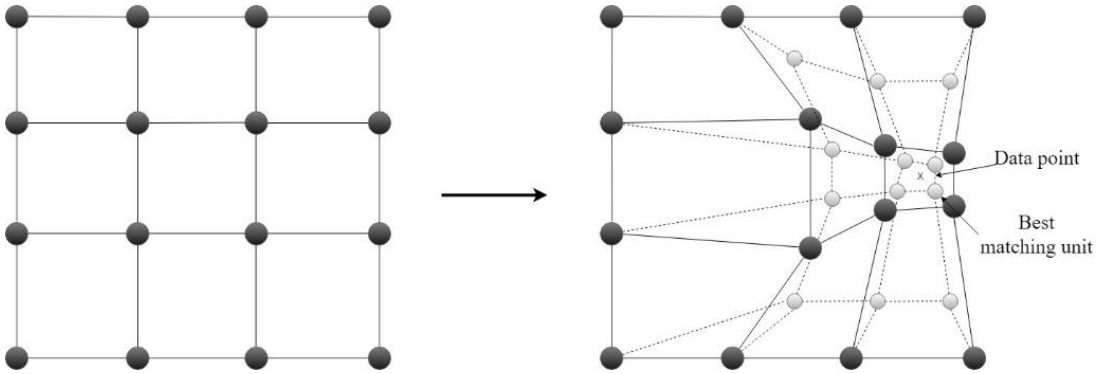


Figure 9: SOM map before and after the training process. This image is adapted from Westerlund (2005).

The SOM training process is iterated until a stopping criterion is met. The stopping criterion could either be the convergence of the map or after a number of training iterations (van Heerden, 2017). One design consideration when initialising a SOM is to decide on the size of the map. A map with too many neurons is computationally expensive as well as increases the chance of overfitting. Conversely, too few neurons create high variance within clusters (Engelbrecht, 2007). One approach to remedy this problem is to implement a growing map. Another issue with SOMs is that the training process is slow. Therefore, to speed up this process, a decaying neighbourhood function and learning rate can be implemented to reduce the weight change during the training of the SOM (van Heerden, 2017). As mentioned, a trained SOM produces a set of clusters which best represents the input vectors. However, no target information is input during the training process. Therefore, in order to use the SOM as a classifier, the map should be labelled either by a supervised or unsupervised labelling approach (Engelbrecht, 2007; van Heerden, 2017).

3.3.2 Supervised classification

This section explores the various supervised classification algorithms. This exploration is done in order to evaluate the advantages and disadvantages associated with each algorithm, in order to ensure that the classifier used within the recommender system is best suited for the application. The supervised classifier algorithms to be discussed are artificial neural networks, support vector machines, decision trees, and random forests. These algorithms are discussed throughout the rest of this section.

Artificial neural networks

An artificial neural network (ANN) is modelled on the biological neural system of the brain. Within the brain, neurons are connected between each other in a mesh. Signals are passed into this mesh of neurons where the individual neurons either excite or prevent signals. ANNs contains artificial neurons which are represented by a weight which either activate or prevent signals from propagating through the ANN architecture (Engelbrecht, 2007). The architecture of an ANN model consists of three parts: an input layer, hidden layers and an output layer (Almási *et al.*, 2016). These layers are either fully or partially connected between layers (Engelbrecht, 2007). A single neuron (SN) is first considered to gain an understanding of the operation of the entire ANN. An SN is illustrated in Figure 10.

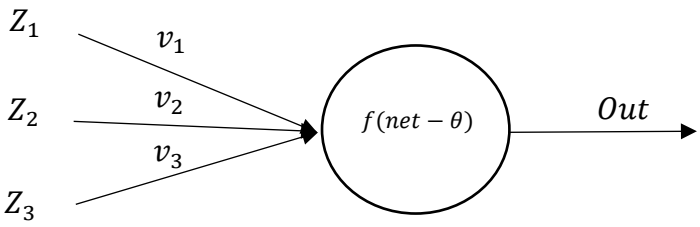


Figure 10: Single neuron adapted from Engelbrecht (2007).

An SN allows for the input of multiple inputs. These inputs are all multiplied by a respective weight before they are passed into an activation function, which either enhances or diminishes the input signal. The output is obtained usually by computing either the weighted sum or product of all the input signals. The output of the SN is further controlled through a threshold, also known as the bias. The activation function used within the SN is most commonly either a linear, step, ramp, sigmoid, hyperbolic tangent or gaussian function. Besides the linear function, all these functions can be seen to be monotonic. Therefore, an SN can be used to split linearly separable data to cases which are above or below the defined threshold. It should be noted that an SN only works in cases where data is linearly separable. In the case where the data is not linearly separable, then additional neurons should be introduced (Engelbrecht, 2007).

Various multilayer ANN methodologies have been developed for supervised learning. These methodologies include Feedforward Neural Networks (FFNN), Recurrent ANNs and Time-delay ANNs. It should be noted that these methodologies by no means cover all the ANN types, but provide a broad overview of the supervised ANN space. The FFNN architecture is shown in Figure 11.

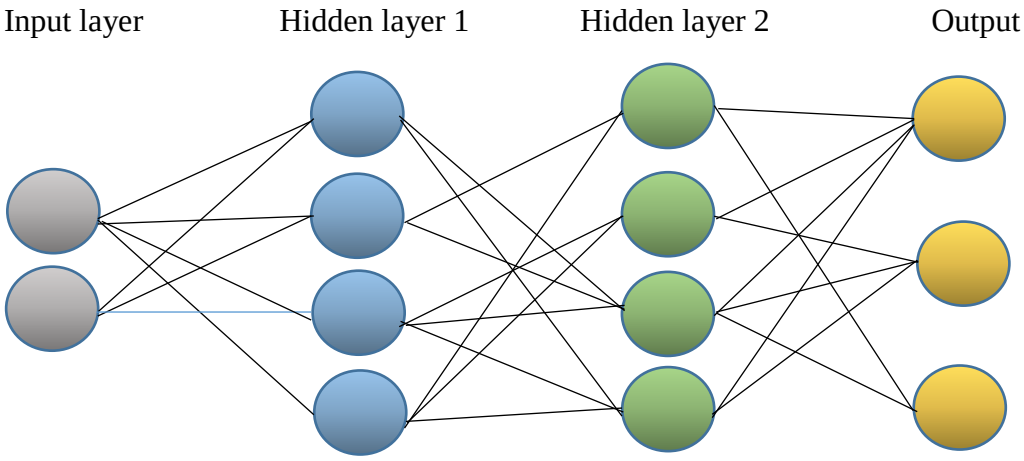


Figure 11: Feedforward neural network architecture illustration.

The architecture of an FFNN consists of an input layer, hidden layer(s) and an output layer. It should be noted that there can be multiple hidden layers. FFNNs receive single external signals which are then propagated through the hidden layer(s) to obtain the result. This process does not consist of any feedback connections to previous layers. Recurrent ANNs does, however, have this feedback link to previous layers. These feedback links allow for the learning of temporal characteristics of the input dataset. Time-delay ANNs considers a window delayed in time. Time-delay ANNs, therefore, allow for the temporal characteristics to drive the shaping of the learning function (Engelbrecht, 2007).

An advantage of ANNs is that they can continue operating with incomplete data. Another advantage is that once the architecture has been found, the cost of forward calculations is significantly reduced (Engelbrecht, 2007). An issue with ANNs is that of dimensionality. A neural network with a large input layer will result in many weights to be created within the model. Therefore, processing high dimensional datasets would be computationally unfeasible (Wójcik *et al.*, 2018). Another issue with ANNs is that they do not generalise well and therefore produce poor accuracy on test sets. A solution to poor generalisation performance is to increase the number of data points within the dataset. Concept drift also impacts the accuracy of the ANN. Concept drift occurs when the input target variable changes over time to values not present in the training data (Almási *et al.*, 2016).

Support vector machine

A support vector machine (SVM) performs a supervised classification by finding hyperplanes to separate linear separable classes, by maximising the distance between the hyperplane and the two classes (Ng, 2000). This separation is done by finding a minimal set of support vectors. Support vectors are the points which lie closet to the hyperplane. These points are the most difficult to classify and therefore has a direct bearing on the location of the separating hyperplane. Therefore, moving a support vector would result in the movement of the decision boundary or hyperplane (Berwick, 2003). An illustration of the SVM technique is shown in Figure 12.

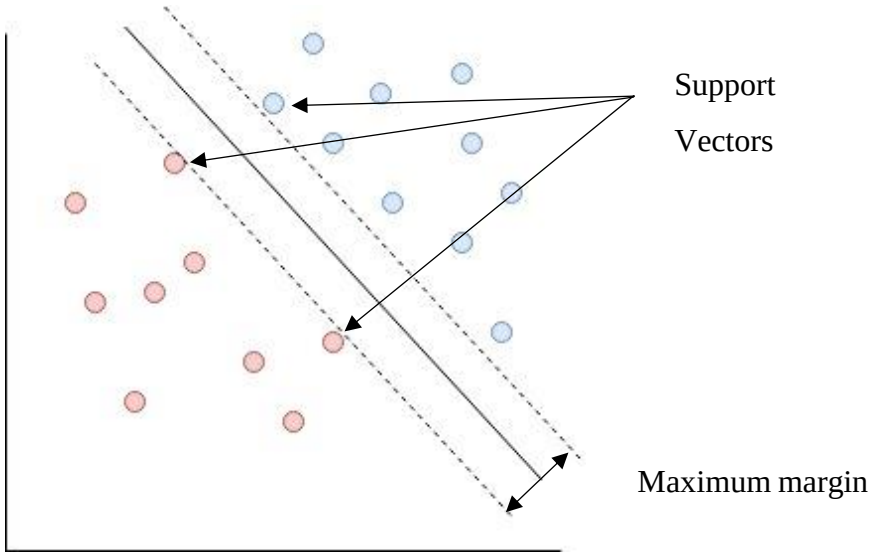


Figure 12: SVM technique showing the support vectors and the maximum margin between data classes.

The SVM illustration shown in Figure 12 is specific to linearly separable data. However data is not always linearly separable, in these cases, a soft margin is imposed, where some data points are allowed to cross the margin or by implementing a kernel function which maps data to a higher dimensional space to achieve separation (Berwick, 2003; Noble, 2006). An advantage of SVMs is that they perform well for a high dimensional number of features. SVMs contains a good out of sample generalisation and is robust to bias within the training samples as only data points which lie near the hyperplane are focused on (Auria *et al*., 2009). By only focussing on data near the hyperplane, computation is reduced by not attempting to construct a complete distribution of all the data, thereby making SVMs scalable with large datasets (Noble, 2006). Another advantage of SVMs is that they solve a convex optimisation problem and, therefore, always converge to a unique solution (Noble, 2006). A disadvantage of SVMs is that they lack transparency into the resultant ruleset. Although it is possible to obtain

confidence scores, it is not possible to view the ruleset explaining how the classification was done (Noble, 2006).

Decision trees

Decision trees are sequential models which enable multistage decision making. The basic idea behind decision trees is that complex decisions are broken down into a series of simpler decisions, where the final decision would resemble the desired solution (Safavian *et al.*, 1991; Kotsiantis, 2013). Decision trees have been studied extensively with a number of different decision trees being developed. These decision trees range from classification, regression trees, and model trees amongst others (Kotsiantis, 2013). For the remainder of this section, classification decision trees will be the focus.

There are generally two phases to inducing decision trees. The first being the growth phase and then the pruning phase. The growth phase aims at recursively partitioning the training data to obtain two or more leaf nodes. These leaf nodes results either in pure classes or classes with a purity of a predefined threshold. The pruning phase aims to generalise the decision tree that was induced in the growth phase by creating a subset of the growth phase decision tree. By pruning the tree overfitting of the decision tree is avoided (Kotsiantis, 2013). A representation of a decision tree is shown in Figure 13.

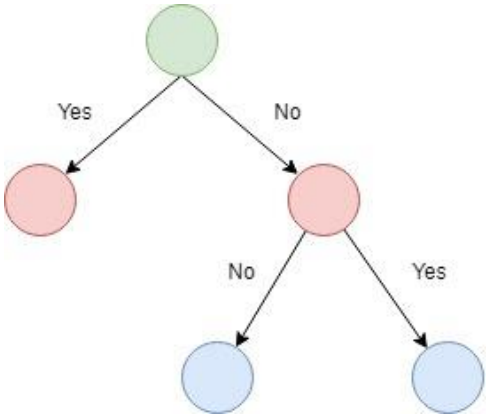


Figure 13: Decision tree representation.

As mentioned, a decision tree makes use of a recursive function in that the method splits data until all data within a subset is from a single class. In the case where a pure class is not produced, the process is repeated. However, with each split, purity or uncertainty is measured (Utgoff, 1989; Kotsiantis, 2013). This measurement is done by making use of an entropy equation

$$Entropy(S) = -p_{(+)} \log_2 p_{(+)} - p_{(-)} \log_2 p_{(-)} \quad (10)$$

where S is the subset of training examples and $p_{(+)}$ and $p_{(-)}$ are the positive and negative examples within S. The decision tree splits the data in order to minimise the entropy of each resulting subset. In addition, the information gain for each split is computed to find the most optimum feature to split on. The information gain measures the amount of information a feature split provides on a specific class. The information gain is given by

$$Gain(S, A) = \underset{v \in A}{Entropy}(S) - \sum \left(\frac{|S_v|}{|S|} \right) Entropy(S_v) \tag{11}$$

where v is the possible values of attribute A, and S_v is a subset containing values which is equal to v. (Kotsiantis, 2013). The decision tree splits on features which maximises the information gain.

An advantage of decision tree classifiers is that the rulesets can be extracted from the decision tree, therefore, making classifications within the decision tree more interpretable (Kotsiantis, 2013). Another advantage of a decision tree is that a comprehensive analysis can be extracted where consequences from possible decisions can be obtained. A disadvantage with this algorithm, however, is that decision trees are susceptible to overfitting and therefore, trees need to be pruned to ensure that they generalise to unseen examples. Another disadvantage is that a decision tree is susceptible to noise within the dataset because the decision tree splits on noise within the data (Safavian *et al.*,1991).

Random forests

A random forest consists of an ensemble of trees, and the result is voted on by these respective trees in the algorithm (Breiman, 2001). Each decision tree in the random forest operates by randomly splitting training sets into multiple subsets. An ensemble of decision trees are implemented on these subsets. The output of specific input vectors is voted on by these individual decision tree classifiers within the forest. Classes with the most votes in the forest win the classification (Boulesteix *et al.*, 2012). A representation of this algorithm is shown in Figure 14.

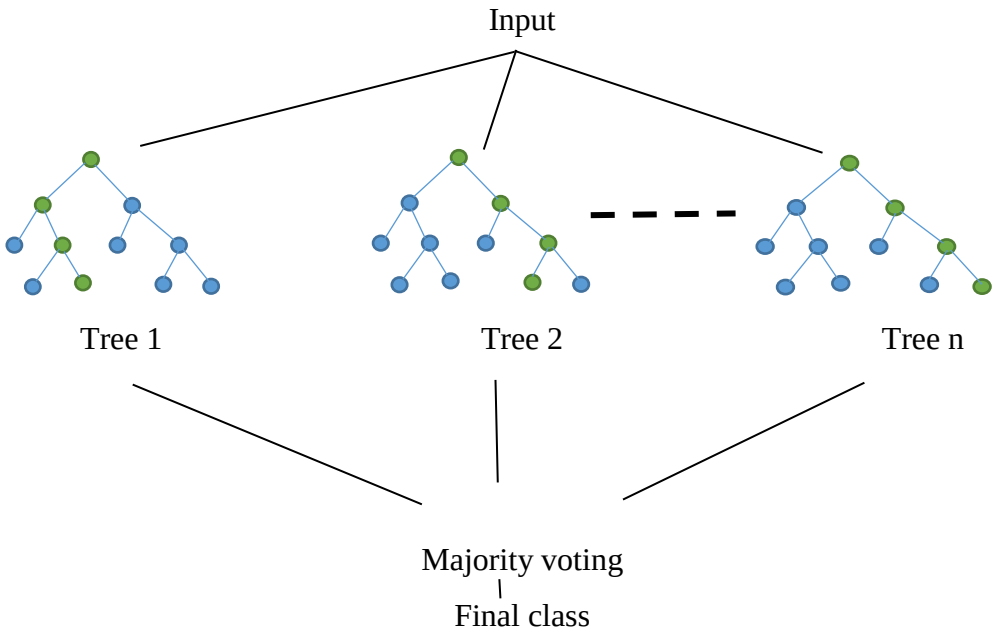


Figure 14: Decision forest representation adapted from Verma *et al.* (2018).

The advantages of a random forest classifier are that it performs well with high dimensional data. This performance is observed because not all features are considered when a split is searched for. Other advantages of this classifier is that the algorithm is robust to over-fitting and resilient against noise and missing values (Touw *et al.*, 2013). However, a disadvantage of the algorithm is that the internal ruleset is not accessible. The random forest classifier uses a voting methodology when classifications are made and aims to minimise the overall error rate and will, therefore, aim to optimise the prediction accuracy of the majority class. This optimisation would result in poor performance of the minority class. Care should also be taken to account for unbalanced datasets to reduce this bias (Liaw *et al.*, 2002; Chen *et al.*, 2004).

K-nearest neighbour classification

A K-Nearest Neighbour (KNN) classification algorithm is considered a semi-supervised learning algorithm as it requires the training data and a user-defined *K* value. KNN classification works by selecting the *K* number of elements and a defined distance metric which is closest to the point of interest. Once *K* elements have been identified, the point of interest will be classified based on the majority group of these *K* elements (Chomboon *et al.*, 2015). An example of the KNN algorithm is shown in Figure 15.

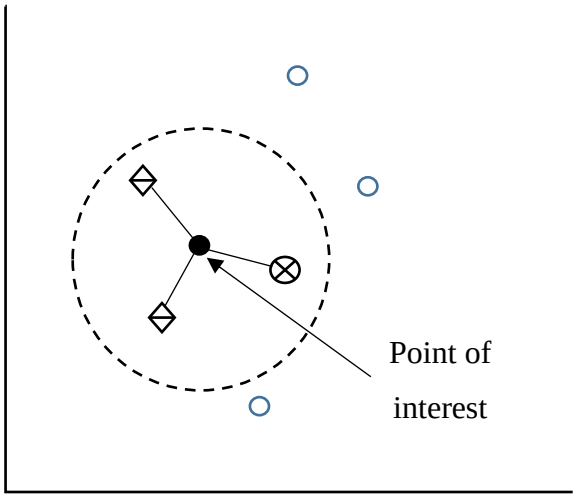


Figure 15: Example of the KNN algorithm where $K = 3$.

Several different distance metrics can be implemented in order to find the distance between points, the most common being the Euclidean distance. Other distance metrics include the standardised Euclidean distance, Mahalanobis distance, City Block distance, Minkowski distance, and Chebychev distance (Cunningham *et al.*, 2007; Chomboon *et al.*, 2015). Additional distance measurement metrics are shown and explored in Chomboon *et al. (*2015).

An advantage of KNN algorithms is that they are robust to noisy training data and capable of processing large datasets (Teknomo, 2017). Another advantage is that the classification is based directly on the training examples. Therefore, training can be performed on the entire dataset (Cunningham *et al.*, 2007). A drawback to this algorithm is that bias is introduced by a user-defined *K* selection. Points of interest specifically on the border of classifications can be affected by the number of *K* elements used within the KNN algorithm. The KNN algorithm has a high computation cost, as the distance between every point and the various K elements needs to be computed (Teknomo, 2017).

3.4 Summary

This chapter evaluated the various algorithms which can be used within the development of the behaviour recommendation system. The chapter considered the various dataset balancing techniques along with the advantages and disadvantages of each algorithm. The input dataset of this study is of high dimensionality, therefore, various feature selection algorithms were explored and evaluated. Attention was then paid to both supervised and unsupervised classification methods which can be used to classify the behaviour of individuals within the MIP program.

Chapter 4
Multiply incentive program

The developed behaviour recommender system is implemented within the MIP. It is assumed that all users within the program want to move to higher categories. The behaviour recommender system aims to aid users in successfully navigating the MIP to achieve a higher status. In order to achieve this aim, it is essential to understand how the current program operates and what the present required actions of users are. This chapter analyses the current MIP point structures and user interface to obtain an understanding of the architecture of the program.

4.1 Exploration of Multiply

Multiply works on a point basis. Desirable actions of the users produce points. These points are accumulated and calculated on a yearly basis. Based on these points, users are placed in different categories. The incentive categories are given in Table 1.

Table 1: Multiply Incentive Program Categories

Category	Single	Family
Bronze	0	0
Silver	250	500
Gold	500	1000
Platinum	650	1300
Private club	750	1500

Table 1 shows that users in a family would need to produce double the number of points when compared to single users. Users navigate these groups depending on their points. These points are awarded based on four factors, namely health, safety, financial and policies received from Momentum. Each of these factors is described in the subsections that follow.

4.2 Health

The points awarded for healthy behaviour are summarised in Table 2.

Table 2: Healthy points

Activity	When	Points	Single point limit	Family point limit
Complete the physical health and activity questionnaire	Once a year	20	20	40
Known healthy heart score	Once a year	Green – 100 Amber – 60 Red – 30	100	200
Have an active day	Once a day	1	80	160
Or				
Fitness assessment	Every six months	Level 5 – 40 Level 4 – 30 Level 3 – 20 Level 2 – 10 Level 1 – 5	80	160

Points awarded to individuals depend on the category and health of the individual. These categories are the healthy heart score and fitness test. The healthy heart score ranges from 100 points for a green status to 30 points for a red status. Users also have a choice to track their active days or to do a fitness assessment every six months. Behaviours which contribute to an active day are:

- 300 calories burned
- One gym visit
- 10000 steps taken
- One sporting event finished

One active day is given if any of the conditions above are met. Users can only earn one active day per day, irrespective of the number of conditions met for that day. If the user does not wish to track their fitness each day, they could opt for an assessment every six months. This assessment covers:

4. Blood pressure
5. Muscle flexibility
6. Lung function using a peak flow meter
7. Height, weight, and waist circumference
8. Cardiac efficiency

Based on the results of this fitness test, users are placed in categories and points are awarded depending on their category. These point allocations are shown in Table 2.

Fitness points range from 40 to 5 points depending on the fitness of the user. Users can only receive these points twice a year, and therefore a limit of 80 points is applicable.

4.3 Safety

The points awarded for safe behaviour is summarised in Table 3.

Table 3: Safety points

Activity	When	Points	Single	Family
Complete safety questionnaire	Once a year	20	20	40
Safety score	Once a year	>80% - 50 71% - 80% - 40 55% - 70% - 30 <55% - 0	50	100
Safe day	Once a day	1	80	160

Users receive a safety score based on the results of their safety questionnaire. These results depend on the following aspects:

- Does the user have a fire extinguisher at home?
- How frequently they inspect their electrical fence.
- How frequently their geyser is checked for faults.
- Is there an inventory list kept of their belongings.

- Are there chips or cracks on their windscreen.
- The presence of security features on their vehicle.

The user obtains a score which depends on the number of safety aspects the user has. Users also receive a point per day if the day is defined as a safe day. These safe days are defined in Table 4.

Table 4: Safe day

Safe day category	Points
Drive according to the rules	1
Take the train	1
Have a drive free day	1

It should be noted that driver behaviour is recorded through the MIP app. It is seen that one point is awarded per day if any of the conditions shown above are met. A maximum of 80 safe points is available per year.

4.4 Financial

This section discusses how points are allocated for financial behaviour. The point system is summarised in Table 5.

Table 5: Financial points contributors

Activity	When	Points	Single	Family
Complete questionnaire	Once a year	20	20	40
Yearly financial review	Once a year	100	100	200
Track money spent	Once a month	20	80	160

4.5 The cover received from Momentum

This section shows the points received when users take additional products with Momentum. These points are summarised in Table 6.

Table 6: Additional cover points

Activity	When	Cover	<1 year	One year	Two years	Three years	Four years	Five+ years
Cover with Momentum	Once a year	Risk	70	80	90	100	110	120
		Car and home cover	70	80	90	100	110	120
		Health	70	80	90	100	110	120
		Wills	70	80	90	100	110	120
		Retirement	70	80	90	100	110	120
		Savings	70	80	90	100	110	120

Additional points are obtained the longer a user holds the policy. From the table above, a user receives an additional 10 points for each year the cover is held.

4.6 Summary of point structure

If a user complies with all of the behaviours in the health, safety and financial sections, the points obtained will result in gold status. Therefore, it is impossible to move to private or platinum without taking out an additional policy with momentum. The more years that an individual has these policies, the less needs to be done in other categories. Points are not given for items bought or for the location at which the individual shopped.

In this dissertation, the structure of the incentive scheme presented is assumed to be static from year-to-year. If this was not the case the problem would be dynamic in nature.

4.7 User Interface

To provide a behaviour recommender system which best aids the user, an analysis of the required user actions and interfaces with dashboards is done. This analysis dictated where the behaviour recommender system would be best suited in order to provide maximum impact. With the current system, users track their progress using various dashboards. These dashboards show user statistics and point totals. The dashboards also show the number of points needed to go to the next level. These dashboards are shown in Figure 16.

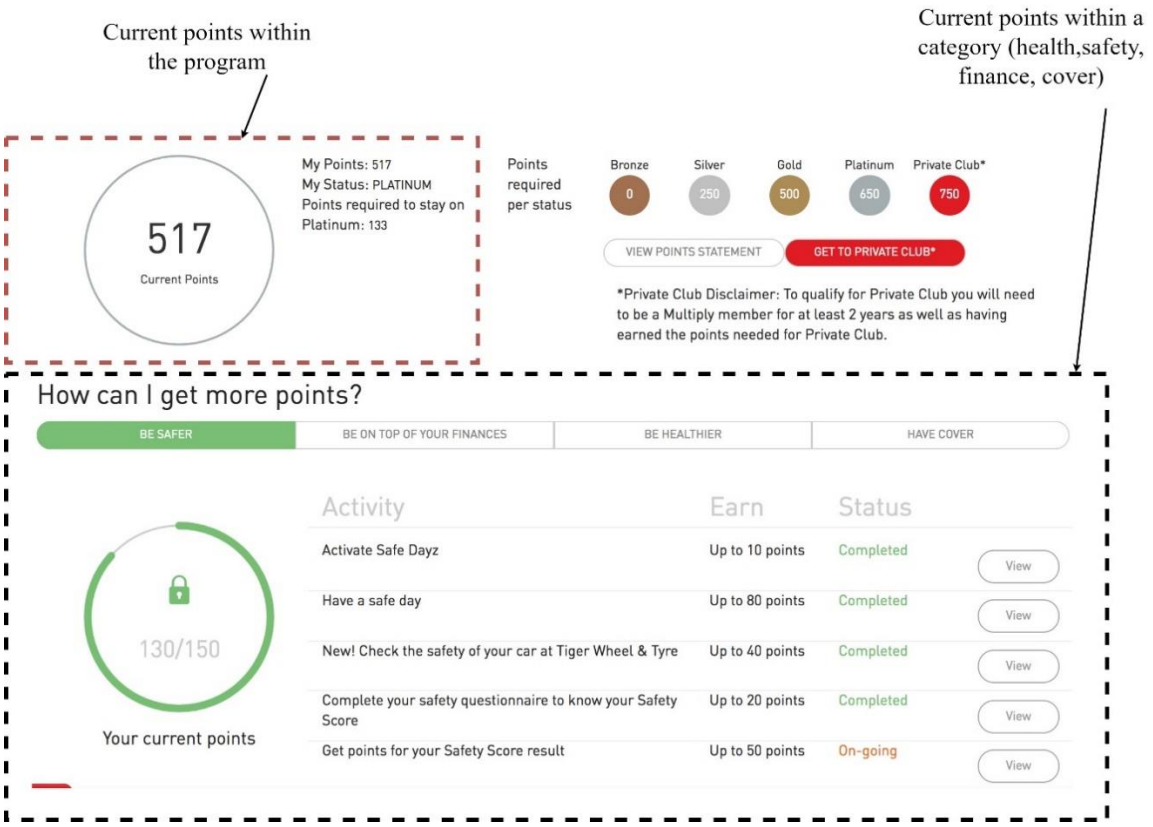


Figure 16: MIP user dashboards.

Each dashboard shows the points earned in that sector, along with points that can still be obtained per sector. This system gives the user transparency with regards to their points and the contributors thereof. However, a drawback of this system is that it does not provide recommendations on actions which need to be implemented to move to the next category. For this, a behaviour recommender system is best suited.

4.8 Summary

This chapter explores the current point system employed within the MIP. The point contributing actions are investigated for each section, along with the critical analysis of the program. The user interface is also explored to find the method users interact with the MIP.

Chapter 5
Solution methodology

This chapter outlines the methodology, experiments and investigations into the development of a behaviour classifier which is implemented in the behaviour recommender. The available dataset is explored and pre-processed. The developed behaviour recommender is split into two phases: the first being the behaviour classification phase and then the recommendation phase. The classification phase aims to train a model to successfully identify the behaviours which most contribute to users being placed into a specific group and to develop a classifier which can be used for validation of the recommendation algorithm. This chapter discusses the solution architecture and data exploration in sections 5, 5.2 and 5.3, respectively. Thereafter, the chapter elaborates on the pre-processing steps in section 5.4. Finally, the classifier development and results are shown in section 5.5 and 5.6, respectively.

5.1 Solution Architecture

This section provides a high-level architecture of the steps taken to develop a behaviour recommender system. The architecture is shown in Figure 17.

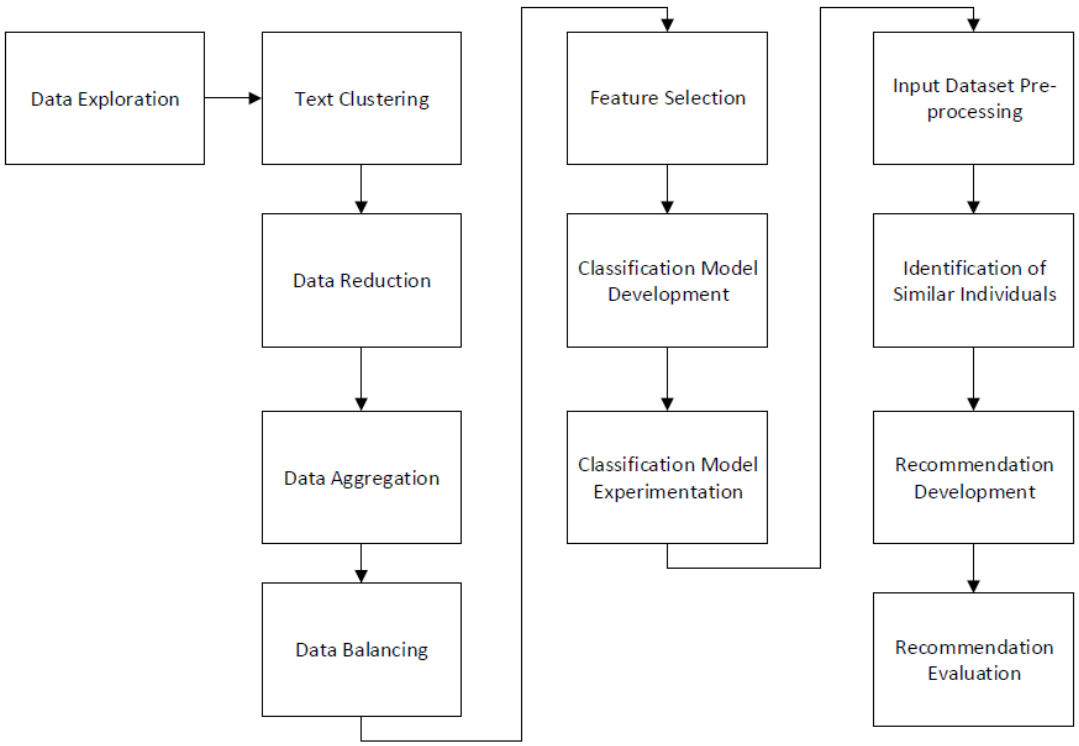


Figure 17: Flowchart showing the steps followed in order to develop a behaviour recommender system for the MIP.

Data exploration is done to identify the range, missing data and types of data within the dataset. Thereafter, the data is pre-processed. This step prepares the data for the classification model. Text fields are labelled, data is reduced and then aggregated. Once aggregated, the complete dataset is balanced and evaluated to ensure that the data is in the required form. The classifier is trained on this data and evaluated to ensure that a satisfactory performance is achieved. Following the evaluation of the classifier, the prominent features are selected and extracted from the dataset. The selected features form the new dataset which is provided as input to the behaviour recommender system. Once the features have been extracted, the data is input into an additional pre-processing phase. This phase processes the features into the required form for the recommender algorithm. After that, the recommender system is developed and evaluated. It should be noted that this architecture forms the basis of this chapter and chapter 6. Each step of the flowchart shown in Figure 17 is elaborated upon in the following subsections.

5.2 Data exploration

Multiply, in collaboration with Momentum, provided the data for this dissertation. The data was provided in nine different comma separated value type tables. These tables contained anonymised personal information, membership information, and health, transactional, product and driving data. The details of these datasets are provided in Table 7.

Table 7: Provided data characteristics

Table	Table focus	Size (MB)	Rows	Columns
1_Personal_Data	Personal	76.245	143743	7
2_Membership_Information	MIP	231.712	3259142	6
3_Healthy_Heart	Health	268.010	3872242	5
4_Activity_Information	Exercise	1797.358	23763295	6
5_Partner_Data	Cinema	33.173	361481	7
5_Partner_Data_CarRental	Car rental	7.360	53737	11
5_Partner_Data_Netsense	Shopping	0.874	9943	5
5_Partner_Data_onlineShopping	Shopping	0.789	10755	4
5_Partner_Data_Dischem	Shopping	2421.454	9864068	13
5_Partner_Data_P&P	Shopping	203.621	2477873	6
6_Product_Reward	Shopping	54.468	928466	4
7_DrivingData	Driving	139.184	614588	22
8_Engagement_Data	MIP	161.493	2476342	5

The datasets are shown to be of varying size and features. Each dataset describes the different behaviours of users within the MIP. An analysis of these data tables is given in Appendix A. It should be noted that the tables are linked using the clientNo and policyFull columns. The clientNo column contains the client number data of each respective client while the policyFull column contains the policy information of the respective policies. It was found that multiple clients could be on the same policy, as well as multiple policies could be held by a single client. In the exploration phase, the data was explored in order to gain an understanding of the types of data present, the size of the respective datasets and the absence of data within respective rows. A description of the data contained in the datasets given in Table 8.

Table 8: Dataset descriptions.

Table	Category	Description
1_Personal_Data	Personal	Member detail account, age, gender, address and policy description
2_Membership_Information	Policy	Details on the member status and points
3_Healthy_Heart	Health	Heart score of member and period
4_Activity_Information	Health	Details on active information, e.g. workout and gym visits
5_Partner_Data_Netsense	Transactional	Description of beauty treatment
5_Partner_Data_onlineShopping	Transactional	Description of the item bought online
5_Partner_Data_Dischem	Transactional	Basket data on individual
5_Partner_Data_P&P	Transactional	Points and amount spent at PnP and the location of the transaction
6_Product_Reward	Transactional	Details on total spend at momentum and partners
7_DrivingData	Safety	Details on driving trip and behaviour
8_Engagement_Data	Policy	Date and method of engagement
5_Partner_Data	Transactional	Details on cinema visits
5_Partner_Data_CarRental	Transactional	Details on location and trip of car rentals

The dataset was split into five categories, namely: personal, policy, health, transactional and safety information. Users are placed into the different incentive tiers based on their recorded behaviour with respect to these datasets.

5.3 Exploration of datasets

A Jupyter notebook in conjunction with the *pandas* and *numpy* libraries were used for the exploration of the dataset. The aim of the exploration phase was to explore the respective tables

for missing values, duplicates within the datasets and data types. An understanding of how the tables linked together was also investigated. It should be noted that each dataset was evaluated independently. The exploration of one dataset is provided in Figure 18.

```
PolicyFull description:
PolicyFull
False    2056258
True     1202884
Name: PolicyFull, dtype: int64
```

```
description of Nulls:
CryptoSet                      0
PolicyFull                     1
InceptionDate                  1
MultiplyMemberStatusDesc       0
FinancialDateKey          890907
TotPts                    890907
dtype: int64
```

Figure 18: Dataset exploration output.

The respective tables contained associative entity columns, clientNo and policyFull, which linked all tables together. These columns were investigated for duplications. Figure 18 shows the details of a duplicate's investigation in the policyFull description section, where false indicated unique values and true indicates duplicates. As can be seen in Figure 18 the dataset was found to have a large number of duplications. These duplicates indicated that the data within the tables were transactional. This transactional data was reduced to show all transactions of a unique client on a single row. The details of this are shown in section 5.4.2.

The description of nulls in Figure 18 investigated the missing values within the respective tables to get an insight into the sparsity of the dataset. Rows with missing values in either the policyFull or clientNo columns were removed because these rows were not linkable to the rest of the datasets.

During the exploration phase, a subset of the columns was found to be in a string or date format; these fields are elaborated on in Appendix A. It was found that several columns of string type contained datasets of a large variation. These columns were the ItemDD, MultiplyActiveDayEventDesc, PickUpPoint, DropPoint, ItemDDMovieTitleDD and the various date columns. Columns with a large variation of items posed a problem, in that they created highly sparse aggregated datasets. Therefore, these columns were processed to reduce the variation contained in the tables. This process is described in section 5.4.1.

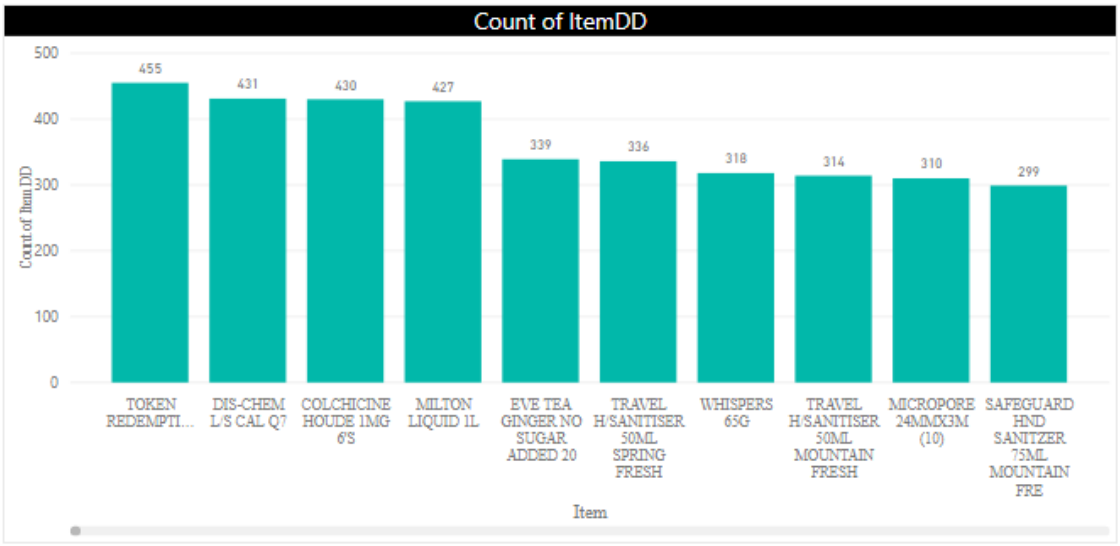
5.4 Pre-processing data

This section describes the steps taken in the pre-processing phase. The objective of this phase is to process the data and model the data into a form which enables the training of a decision forest classifier, which is discussed in more detail in section 5.5.4. The data model requirement of the decision forest classifier was for all tables to be aggregated into a single dataset. The pre-processing phase was implemented with the aim of reducing high variant text columns to produce richer data, reducing transactional type data tables to allow for the conservation of information, while ensuring efficient aggregation of the respective tables.

5.4.1 Text processing

Dis-Chem basket data

As mentioned in the exploration phase, the ItemDD column within the Dis-Chem dataset contained all product item names sold from the various Dis-Chem Stores. It was observed that a high variation of items was present in the ItemDD column. To ensure that a rich dataset was used for the analysis, the patterns in the ItemDD column were labelled into respective categories based on the item type. No reference dataset was provided for the categorisation of this data. Therefore, data within this column is categorised by manually labelling the data within the dataset. Labelling of the dataset is done using the visualisation tool in Figure 19.



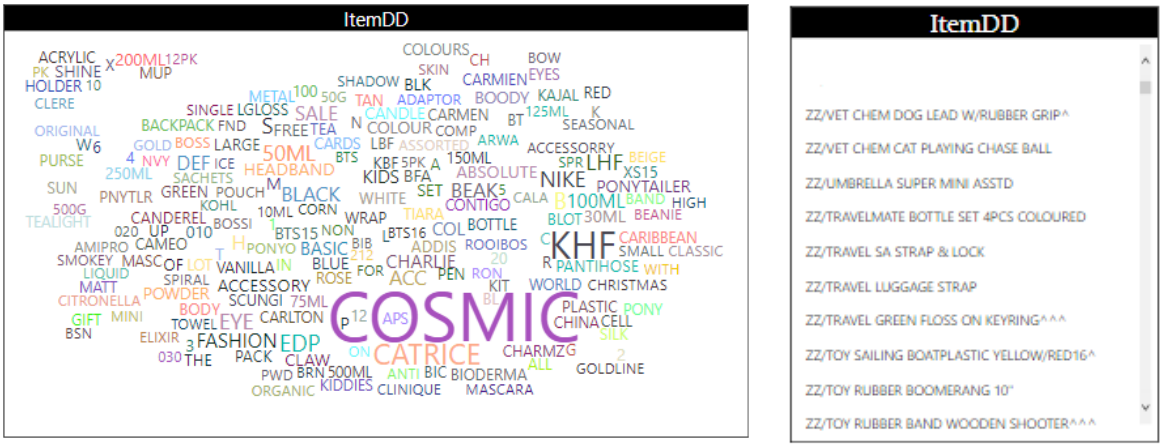


Figure 19: ItemDD manual labelling tool.

The visualisation tool, which was developed in Microsoft Power Bi consisted of a histogram, word cloud, table view and count indicators. The histogram showed the distribution and frequency of each item within the dataset. While the word cloud showed high-frequency words contained in all the items in the dataset. The word cloud, therefore, allowed for an intuitive view into all the individual words within the entire dataset in a single view. The table in Figure 19 was used to obtain the specific item name of the chosen words in the respective column. The ItemDD column was labelled by selecting keywords of high-frequency items within the dataset and placing these items into categories. For example, items containing the word ‘cosmetic’ was placed in the cosmetic category. This method is repeated until all of the items are categorized. The resultant categories are:

- Water
- Food
- Cosmetics
- Detergent
- Medical equipment
- Miscellaneous
- Medication

The resultant distribution of the defined categories is shown in Figure 20.

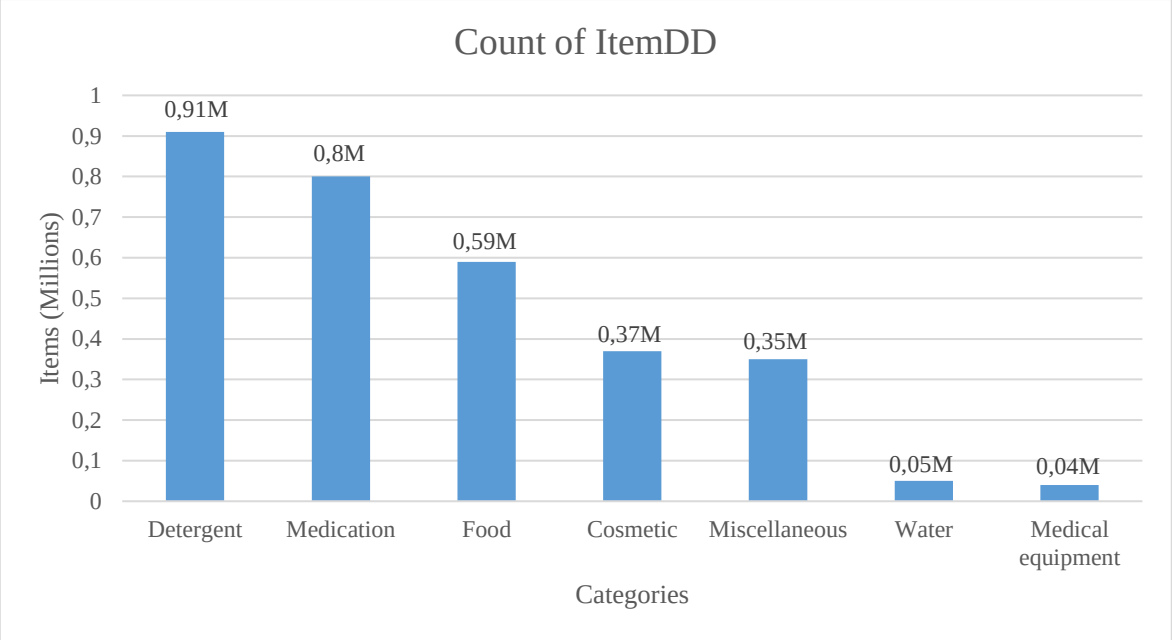


Figure 20: ItemDD labelled distribution.

Figure 20 shows that the most common item is detergent, followed by medication, food and cosmetics. The variation within the ItemDD column is, therefore, reduced to seven categories.

Rental Data

Another dataset which requires text processing is the rental dataset. This dataset contained information on locations of drop-off and pick-up of rental cars. The rental dataset contained a large variety of locations within South Africa. The same visualisation tool shown in Figure 19 is used for the rental data. This visualisation can be seen in Figure 21. It was found that most cars were hired from airports, however, several other rental branches across the respective provinces were used. All rental locations were categorised according to the respective province of the location in order to reduce data variation.

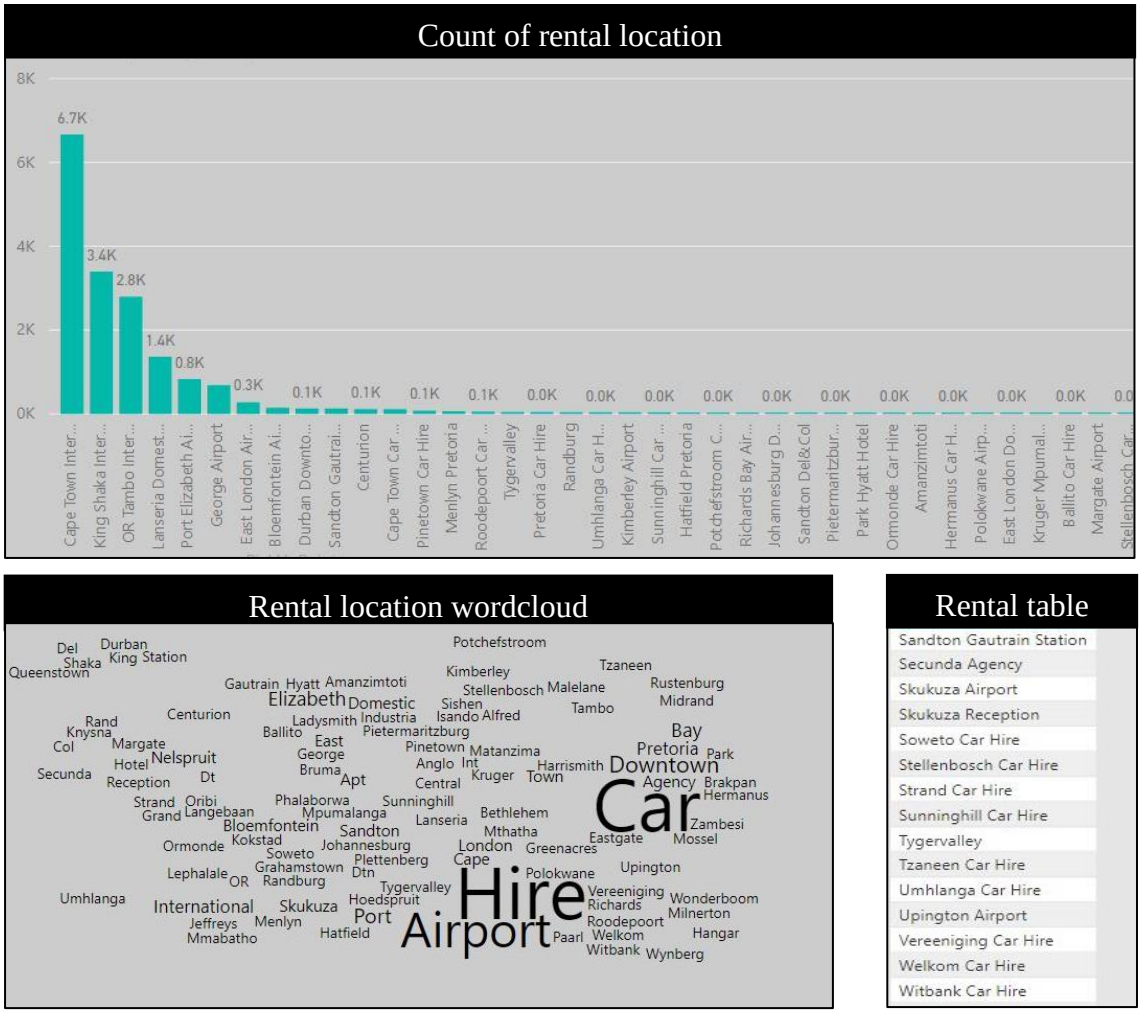


Figure 21. Rental manual labelling tool.

Active data

The active dataset contains information on active activities which were recorded by users. This dataset was more straightforward to categorise because it contained only 28 variations. The categorisation was done by manually assigning activities to the different groups. The groups are labelled as follows:

- Step
- Gym activity
- Calorie activity
- Sport
- Cycling

This categorisation aims to group activities. Therefore, walking and running is categorised as step activities, while all field and court sports activities are labelled as a sport.

Movie name

The movie name column within the NuMetro dataset is categorised differently to that of the others as the open-source reference dataset was available for use. Each movie title was removed and replaced with the genre of the movie to reduce the variation within the dataset. In the exploration phase, it was seen that the genre of the respective movies was not supplied. Therefore, the open-sourced movielens review dataset was used as the dataset contained the genre and title of most movies in the dataset (Maxwell *et al.*, 2015). Therefore, the genres of the movies defined in the movielens dataset were aggregated with the movies in the NuMetro dataset. A 'no information' label was given to movies not found in the movielens dataset.

Date columns

Twelve of the thirteen datasets provided contain dates of transactions of users. For this application, dates are used to split the data into transactions of users per year. A year resolution is chosen because categories are calculated on an annual basis. The exploration into this resolution is further elaborated upon in section 5.4.3.

5.4.2 Dataset reduction

The data provided was for three years, namely the years 2016 to 2018. Duplicates within the dataset cause a challenge for aggregation, because as the data size would grow exponentially in the event of a many-to-many relationship between the primary and foreign keys within the datasets. The data within the respective datasets are of a transactional type. Therefore, a large number of duplicates of the foreign and primary keys are present. The objective of this phase was to reduce the transactions of respective users to a single row without losing information. The dataset reduction was made by splitting the dataset into years of interest. Data were sorted based on the year of each respective transaction. The *pandas* library *groupby* function was used to join all the transactions of a client or policyholder into a single array in each row. The values within each row are converted to a dictionary showing the count of each transaction. This process is illustrated in Figure 22.

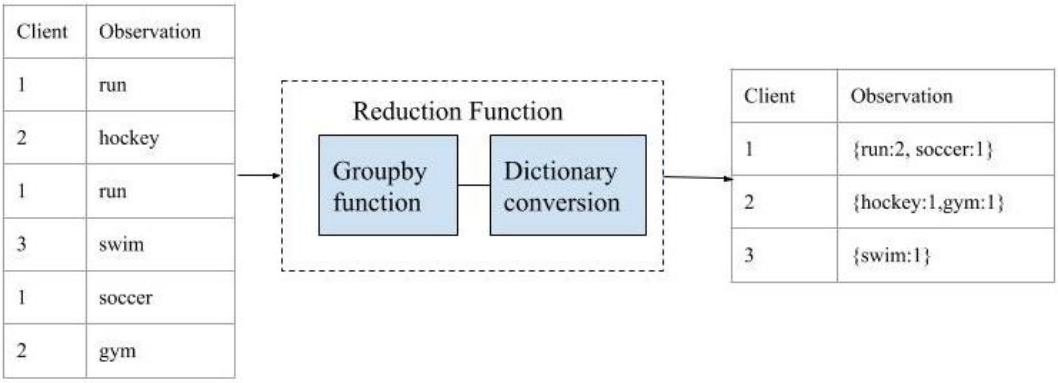


Figure 22: Visualisation of the reduction algorithm.

5.4.3 Dataset aggregation

The aggregation phase aims to aggregate all datasets into a single dataset. Each year's dataset can be seen in Table 9.

Table 9: Table showing the aggregated datasets for the respective years

Year	Client	Observation	Observation 1	observation 2
2016	1	{run:2, soccer:1}	{Green:3, Amber: 2}	{gold: 1}
	2	{hockey:1, gym:8}	{Green:5}	{silver: 1}
	3	{swim:1}	{Amber:3, red: 1}	{private: 1}
2017	1	{run:2, swim:1}	{Green:2, Amber: 2}	{gold: 1}
	2	{soccer:1, walk:4}	{Amber: 1}	{silver: 1}
	3	{swim:1, gym:3}	{Green:1, Red: 2}	{private: 1}
2018	1	{run:2, soccer:1}	{Red: 2}	{gold: 1}
	2	{hockey:1, walk:4}	{Green:4, Amber: 2}	{silver: 1}
	3	{hike:5}	{Amber: 2}	{private: 1}

Rows within the respective datasets were split into the different years ranging from 2016 to 2018 using the transactional date stamps. As mentioned, these datasets were aggregated to the personal_information table using a many-to-one relationship using the clientNo and policyFull associative entity columns present in the tables.

Once successfully joined, the datasets were appended to create one dataset. Within the Membership_information dataset, the MultiplyMemberStatusDesc indicates the category of the user. All members within the program were placed into a category on entering the MIP. In the event of a user not being assigned a tier for the respective year, the implication is that the

user was not a member in that year. Therefore, rows with missing values within the MultiplyMemberStatusDesc column was removed. The resultant dataset consisted of 65 columns and 136535 rows.

The next step was to expand the reduced dictionaries into respective columns. The expansion into the respective columns was to ensure that all transactions made by users are represented. The expansion process is represented in Figure 23.

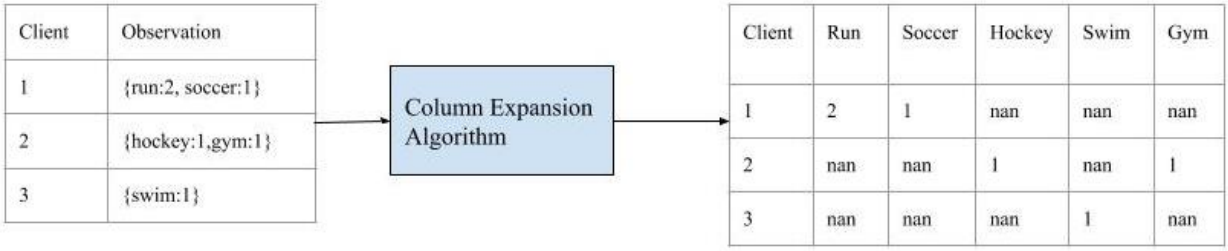


Figure 23: Column expansion process.

Each dictionary observation in the respective rows was expanded, and each unique key in the respective dictionaries was converted to a column. The value of this key was input in the row for each client. A *nan* value was given to clients who had no observations in respective columns. For example, in Figure 23 client three had an observation of one for the swim key, therefore, on aggregation, the swim key is converted to a column and a value of one is input in client three's row. It follows that a nan value is an input in all other columns as these activities were not initially observed.

The advantages of this process were that it preserves the transactional nature of the data within the dataset, converts the entire dataset into a single numeric datatype, and reduces the size of the dataset file. This conversion also creates a common criterion for comparison between clients as all clients and actions are present. However, expansion of the columns is not without its drawbacks. Splitting of the columns creates a highly sparse dataset. Therefore, the input 65 columns are expanded to 483 columns.

5.4.4 Dataset balancing

On completion of the aggregation phase, the MultiplyMemberStatusDesc (target variable) column was evaluated to identify the distribution of the client categories. This exploration was done to ensure that classes used in the classification are equally represented as to reduce bias within the behaviour classifier. Figure 24 illustrates the distribution of the target variable after the aggregation phase.

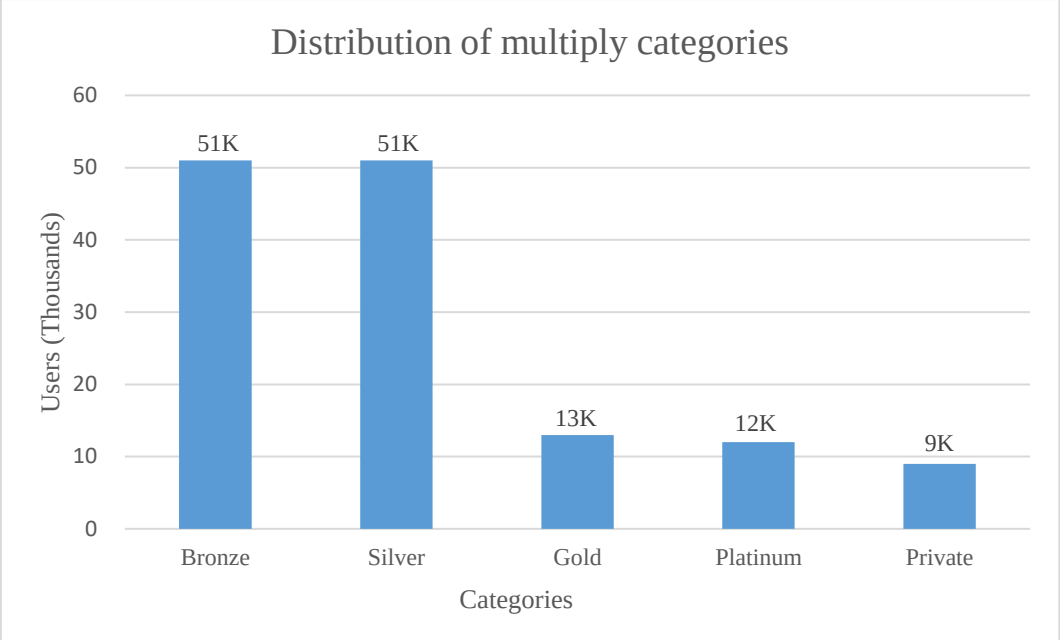


Figure 24: Initial distribution of the aggregated target variable, i.e. MIP categories.

Figure 24 shows that there was a substantial imbalance in the resultant dataset. This imbalance shows approximately 51000 members currently in the Bronze and Silver category, while gold, platinum and private only contained 13000, 12000, and 9000, respectively. This distribution was plausible as members who join the program are by default placed into the Bronze category. Moving up through the categories requires more effort and engagement from users. Therefore, it is understandable that there are substantially fewer individuals in higher categories.

In order to remove bias from the classification model, the dataset was balanced. Balancing of the dataset was implemented using the SMOTE+TOMEK algorithm (Lemaıtre *et al.*, 2017). This algorithm makes use of both under and oversampling techniques as discussed in section 3.1.3. The algorithm was implemented using the python *imbalance-learn* library in conjunction with the *encoding* library. The balancing algorithm requires all features to be represented numerically as the SMOTE+TOMEK method computes the distance between the respective data points as specified in section 3.1.3. Hence the Python *Encoding* library was used to encode the identification columns (policyFull or ClientNo) and all non-numeric columns (Lemaıtre *et al.*, 2017). Figure 25 illustrates the distribution of the target variable after the balancing algorithm. It is observed that there is variation within the balance of the classes. However, all classes are within 20% of each other. The resultant dataset contains 227 738 rows and 483 columns.

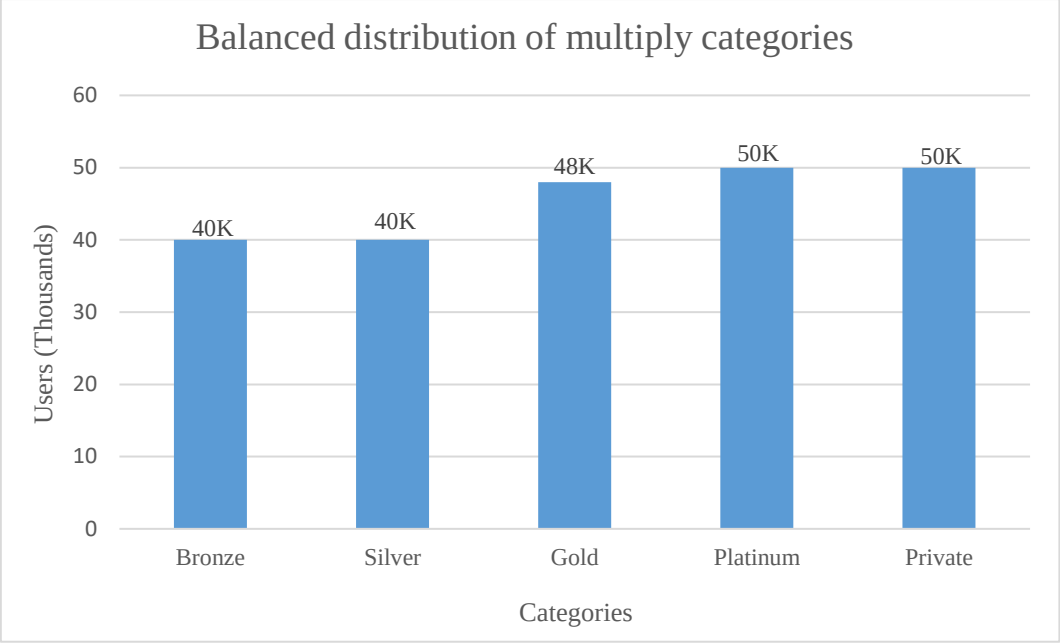


Figure 25: Results of the SMOTE+TOMEK algorithm used to balance the dataset successfully.

5.5 Classification

On completion of the pre-processing, the dataset was provided as input into the classification phase. In this phase, a classification model was constructed with the aim of successfully classifying individuals into a respective incentive category. The status of each user was provided as the target feature in the dataset. Therefore, a supervised learning classification model is developed. This section elaborates on the steps taken to train and evaluate the model.

5.5.1 Development environment

The training phase is implemented using the Azure machine learning studio (AMLS) (Astala *et al.*, 2018). This environment was chosen as it allowed for flexible experimentation, which incorporated a drag and drop interface. The environment also allowed for quick tuning of model parameters. Figure 26 shows the AMLS interface.

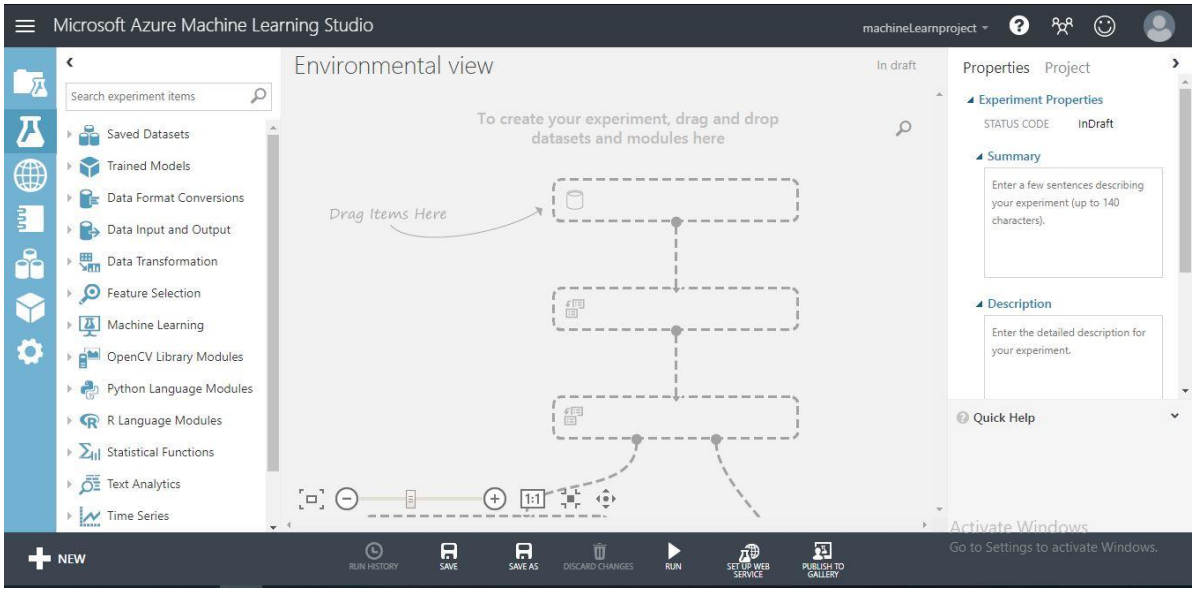


Figure 26: Azure machine learning studio which was used for the experimentation and development of the behaviour classifier model.

The environment represents each phase in the training and pre-processing process as a self-contained block. The environment enables the aggregation and the tuning of model parameters and objectives of the respective blocks.

5.5.2 Experimental setup

The objective of the classifier is to classify users into the various categories based on their observed behaviour within the MIP. Figure 27 provides an overview of the experimental setup.

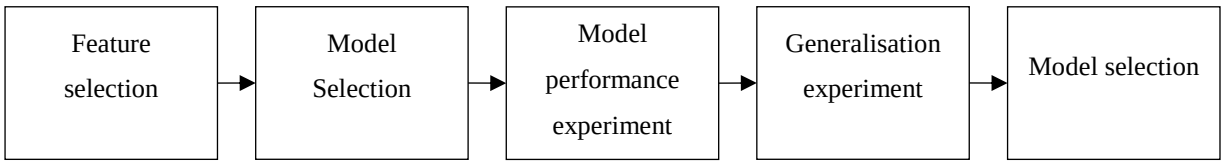


Figure 27: Classifier development process

First, the various feature selection algorithms are implemented on the dataset. In this phase, all features which are found to be significant to the model is selected. The classifier type is then chosen and tuned in order to obtain the optimum control parameters for the classification model. The classifier is then evaluated for accuracy, precision, recall and F1-score. The classifier is then provided as input to a ten-fold cross-validation experiment in order to assess the generalisation of the model.

5.5.3 Feature selection

The feature selection phase aims to identify the most prominent features within the dataset. The following feature selection methods were evaluated:

- Fisher score
- Pearson correlation
- Kendall correlation
- Spearman correlation
- Mutual information
- Chi-Squared

These feature selection methods were evaluated as they preserve the dataset and features. These methods also allowed for each feature selection method to be scored and therefore, enabling a comparison to be made between these feature selection methods. Each feature selection method was implemented on the dataset, and each feature selection method scores was plotted on a graph. The optimum number of features of each feature selection method was approximated by selecting the elbow of the respective feature selection graph where the score tended to zero. This section expands on the various feature selection methods used in the development of the personalised recommender.

Fisher score

The main idea behind the Fisher score is to identify a subset of features where the distance between data points within the same class are minimised, while the distance between data points of different classes are maximised (Gu *et al.*, 2012). The Fisher score is calculated as follows:

$$F(Z) = tr\{(\tilde{S}_b)(\tilde{S}_t + \gamma I)^{-1}\} \tag{12}$$

Where γ is the positive regularization parameter, I is the identity matrix and $\tilde{S}_b$ is the between class matrix, defined as:

$$\tilde{S}_b = \sum_{i=1}^{n} n_i \, (\tilde{\mu}_i - \tilde{\mu})(\tilde{\mu}_i - \tilde{\mu})^T \tag{13}$$

Where n_i is the size of the *ith* class and $\tilde{\mu}_i$ is the mean vector. $\tilde{\mu}$ is defined as the overall mean vector of the reduced dataset, defined as:

$$\tilde{\mu} = \sum_{i=1}^{n} (\tilde{\mu}_i)\mathrm{n_i} \quad (15)$$

$\tilde{S}_t$ is the total scatter matrix which is defined as follows:

$$\tilde{S}_t = \sum_{j=1}^{n} (z_j - \tilde{\mu})(z_j - \tilde{\mu})^T \quad (16)$$

The Fisher score equation is computed independently for each feature within the dataset and assigns each feature a Fisher score (Gu *et al.*, 2012). The features with the highest-ranking fisher scores were selected, while the features with low scores were dropped from the dataset. As mentioned, features are selected by selecting the elbow point of the graph, where the graph flattens to near zero as shown in Figure 28.

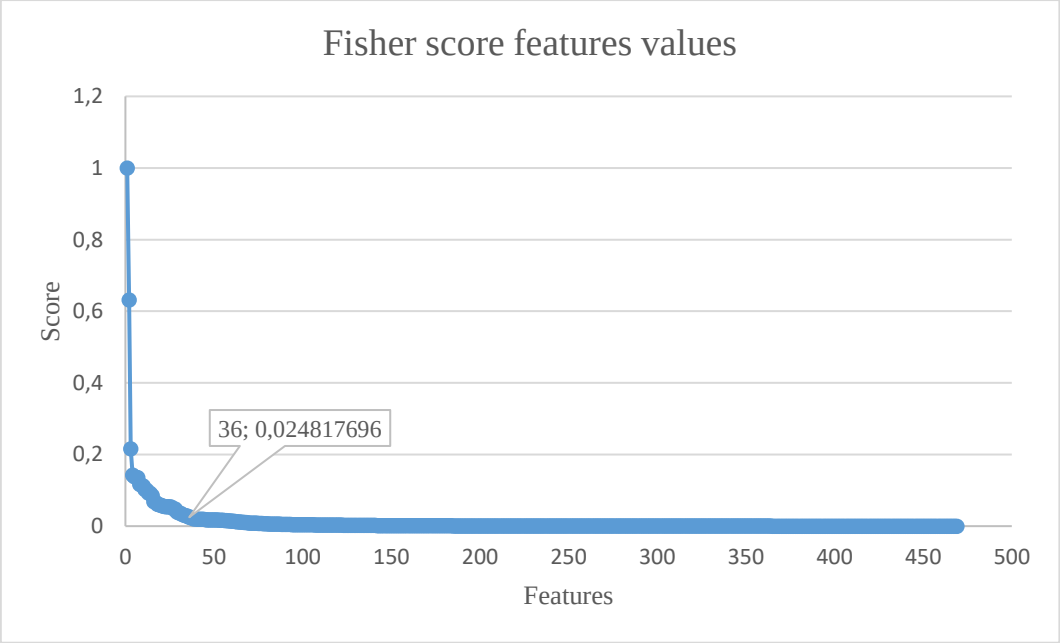


Figure 28: Fisher scores of the features in the aggregated dataset.

From the graph, the Fisher score indicates that the optimum number of features within the dataset was 36.

Pearson correlation

The Pearson correlation is a standard measure of association between continuous variables. The Pearson correlation was defined as the ratio of the covariance of two variables to the product of their respective standard deviations (Astala *et al.*, 2018; Lani, 2018). The Pearson correlation is given by:

$$r = \frac{\sum_{i=1}^{n}((x_i - \bar{x})(y_i - \bar{y}))}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad (17)$$

Where x_i and y_i are feature vectors in the dataset and $\bar{x}$ and $\bar{y}$ are the mean vectors, defined as:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \quad (18)$$

$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n} \quad (19)$$

Where n is the number of samples in the feature space. In the Pearson correlation feature selection method, the correlation for each feature is calculated (Shong, 2010). These scores are graphed and ordered in descending order and plotted in Figure 29.

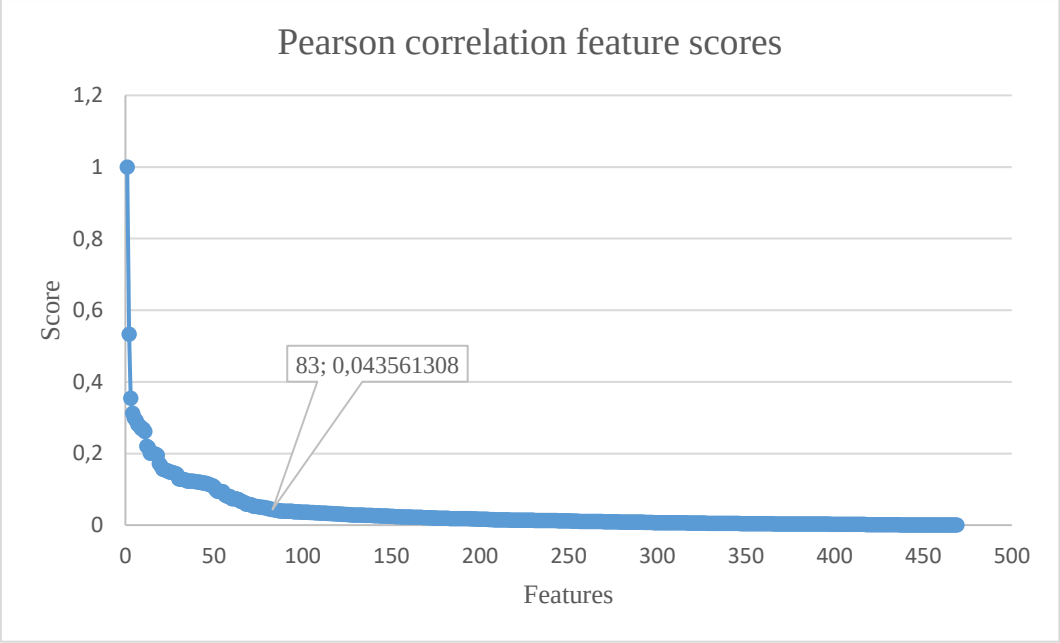


Figure 29: Results of Pearson correlation measure.

The elbow of the plot was manually selected to approximate the most optimum number of features. According to the graph in Figure 29, 83 features were found to be relevant using the Pearson correlation method.

Spearman correlation

Spearman correlation is a rank-based version of the Pearson correlation method (Lani, 2018). The Spearman correlation aims to measure the degree of association between two variables.

This measure is designed to operate with data of ordinal type (Lani, 2018). The Spearman correlation coefficient is calculated as follows:

$$r_s = \frac{\sum_{i=1}^{n}\left(\left(rank(x_i) - \overline{rank(x)}\right)\left(rank(y_i) - \overline{rank(y)}\right)\right)}{\sqrt{\sum_{i=1}^{n}\left(rank(x_i) - \overline{rank(x)}\right)^2 \sum_{i=1}^{n}\left(rank(y_i) - \overline{rank(y)}\right)^2}} \quad (20)$$

Where $rank(x_i)$ and $rank(y_i)$ are the ranks of the observations in the dataset (Shong, 2010). The absolute value of the Spearman correlation coefficient describes the strength of the monotonic relationship. The closer the coefficient is to 0, the weaker the monotonic relationship between the two variables. In addition, Spearman correlation coefficients can have a value of 1 for a linearly related variable and some type of monotonic relationship (Lani, 2018). The Spearman correlation feature selection method generates a correlation score for each feature within the dataset. These correlations scores were then graphed and using the same method as in the Pearson correlation section, the optimum number of features are selected. This graph is shown in Figure 30.

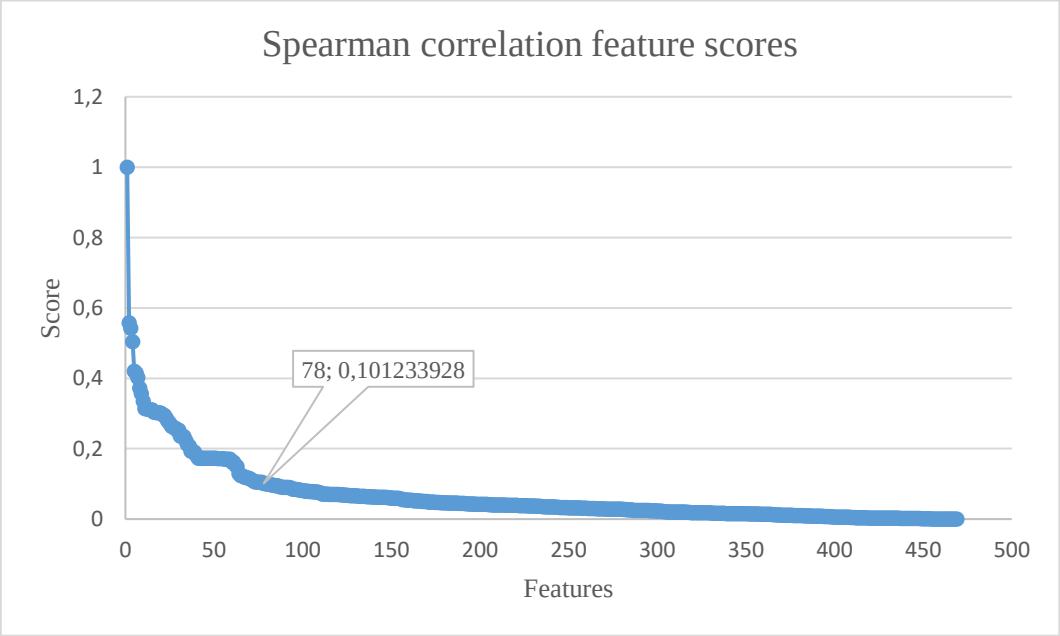


Figure 30: Results of Spearman correlation measure.

According to the graph, 78 features were found to be relevant within the dataset.

Kendall correlation

The Kendall correlation is similar to the Spearman correlation in that the objective of the Kendall correlation is to identify the association between two ordinal variables (Lani, 2018). The Kendall correlation can be expressed by:

$$\tau = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} sgn(x_i - x_j)sgn(y_i - y_j)}{n(n-1)} \quad (21)$$

$$sgn(x_i - x_j) = \begin{cases} 1\ if\ (x_i - x_j) > 0 \\ 0\ if\ (x_i - x_j) = 0 \\ -1\ if\ (x_i - x_j) < 0 \end{cases} \quad (22)$$

$$sgn(y_i - y_j) = \begin{cases} 1\ if\ (y_i - y_j) > 0 \\ 0\ if\ (y_i - y_j) = 0 \\ -1\ if\ (y_i - y_j) < 0 \end{cases} \quad (23)$$

Where the coefficient quantifies the discrepancy between the number of concordant and discordant pairs. τ represents the strength of the monotonic relationship of the two variables (Shong, 2010). As was done previously, all correlations were ordered and graphed. The optimum number of features were approximated when the correlation scores were seen to be near zero. The Kendall correlation for the dataset can be seen in Figure 31.

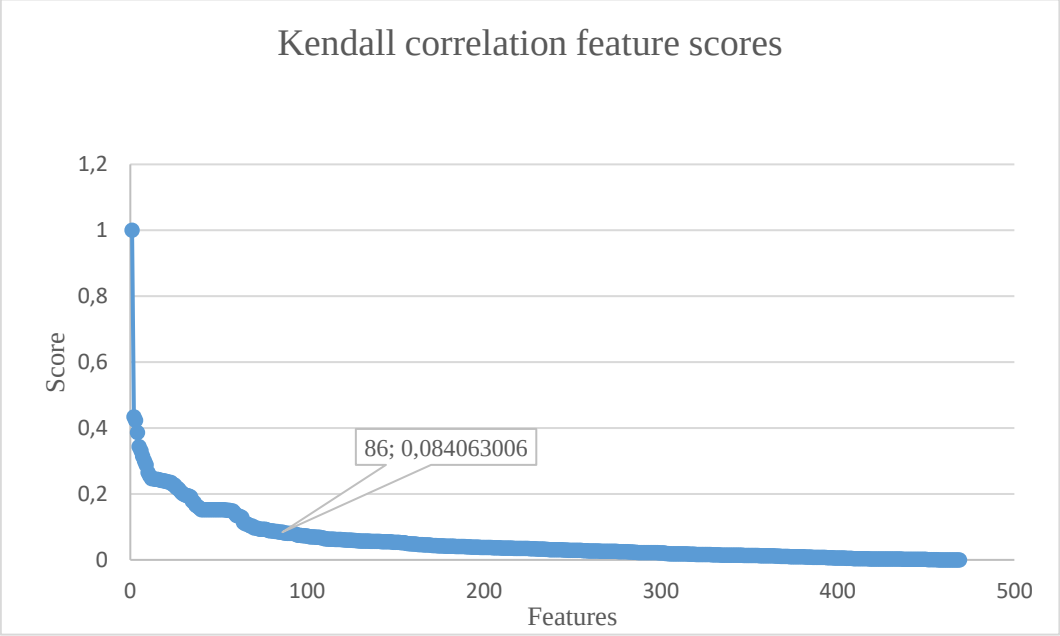


Figure 31: Results of the Kendall correlation algorithm.

According to the graph, 86 features are approximated to be statistically relevant.

Mutual Information

Mutual Information operates by measuring the contribution of one variable to another. Therefore, if the Mutual Information is zero, then it can be said that the two variables are statistically independent and one variable can be removed (Vergara *et al.*, 2014). Mutual Information is given by the following:

$$MI(x,y) = \sum_{i=1}^{n}\sum_{j=1}^{n} P\big(x(i),y(j)\big).\log\left(\frac{P\big(x(i),y(j)\big)}{P\big(x(i)\big).P\big(y(j)\big)}\right) \tag{24}$$

Where $P\big(x(i)\big)$ and $P\big(y(j)\big)$ are the marginal distributions of the random variables $x(i)$ and $y(j)$ (Mitra *et al.*, 2009). Each respective feature is given a mutual information score between 1 and 0. The features which score close to 0 on the elbow of the graph was removed from the experiment. According to the graph, 58 features were found to be statistically significant. The feature scores are shown in Figure 32.

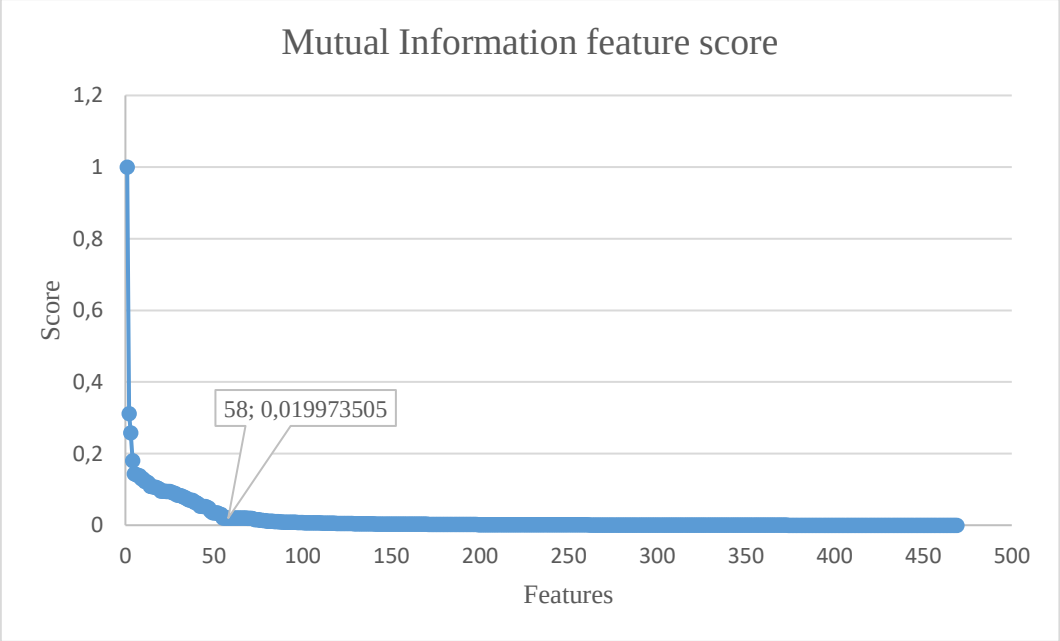


Figure 32: Mutual Information feature selection scores.

Chi-Squared

The Chi-squared measure aims to investigate the dependence and independence of the target and feature columns. The Chi-squared equation is defined as follows:

$$X^2 = \sum_{i=1}^{r}\sum_{j=1}^{n} \frac{\big(O_{ij} - E_{ij}\big)^2}{E_{ij}} \tag{25}$$

where O_{ij} is the observed frequency and E_{ij} is the expected theoretical frequency (Bidgoli *et al.*, 2012). The numerator magnifies the difference, while the result is weighted with the sum of the expected values. Therefore, the larger the Chi-squared value, the more dependent the variables and therefore, the more relevant to the model (Manning *et al*, 2008). Therefore, a Chi-Squared value of zero represents an independent feature and the feature can be removed from the dataset. The Chi-squared feature selection method, therefore, gives each feature in the dataset a dependency score. According to the graph in Figure 33, it is shown that 62 variables were found to be dependent.

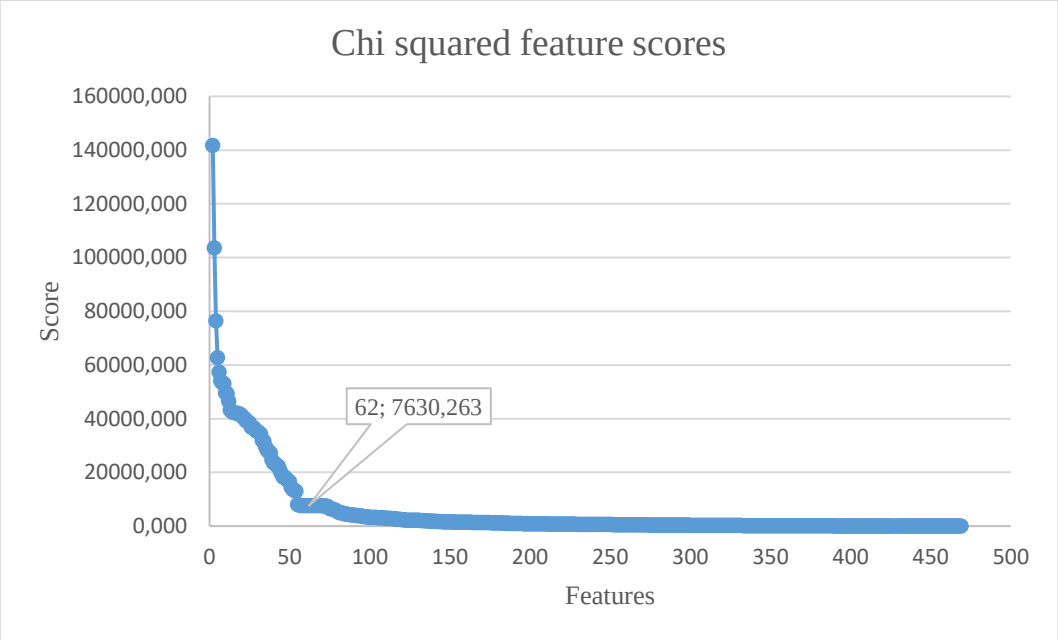


Figure 33: Chi-Squared feature selection dataset scores.

Table 10 summarises the number of features which was found by the different feature selection methods to contribute to the model. Each method was seen to produce a different number of contributing features. In addition to the number of features, the features found was not the same across the respective feature selection methods. Therefore, to find the best feature set within the classifier, each dataset produced from the respective feature selection method is used in the classifier selection experiment.

Table 10: Optimal number of features per feature selection method

Feature selection method	Number of features
Fisher score	36
Pearson correlation	83
Kendall correlation	86
Spearman correlation	78
Mutual information	58
Chi-Squared	62

5.5.4 Classifier selection

The machine learning problem is a supervised classification problem. For this type of problem, the Azure Multiclass Decision Forest (MDF) machine learning method was used (Astala *et al.*, 2018).

An MDF is an ensemble method meaning that a set of classifiers are constructed with subsets of the feature dataset, and a voting system is implemented to obtain the final prediction (Dietterich, 2007). A tree-based classifier was also seen to work well in Dharia *et al.* (Dharia *et al.*, 2018) for behaviour classification and is, therefore, a good starting point for the classification investigation. However, the MDF has a trade-off between bias, variance and computation time of the algorithm. The MDF is not limited to a number of trees in the forest, and therefore, multiple trees can be added to the method. However, adding a vast number of trees has diminishing returns and increases computation. In addition, the stricter splitting of nodes in the MDF makes the tree more biased, however, reduces the correlation between trees. Therefore, parameter changes affect the performance of the MDF (Kravitz, 2018).

To ensure that the most optimum performance of the MDF is achieved, the tuning of parameters of the algorithm was implemented. These parameters are:

- The number of decision trees.
- Maximum depth of decision trees.
- The number of random splits per node.

- The minimum number of samples per leaf node.

The *minimum number of samples per leaf node* refers to the minimum number of training samples required to generate a leaf node. The *Number of random splits per node* referred to the number of splits generated per node, and as the name suggests, the *maximum depth of the decision tree* referred to the depth of any decision tree. *The number of decision trees* referred to the number of decision trees used in the classification of the model (Astala *et al.*, 2018).

By varying these tuning parameters, the MDF module was optimised to produce a model which produced the best accuracy. Therefore, to identify the most optimum configuration the Tune Model Hyper-parameters (TMH) module was used. The TMH module tuned the MDF by sweeping through all the permutations of the parameters and selecting the configurations which resulted in the highest accuracy. The TMH experiment is shown in Figure 34.

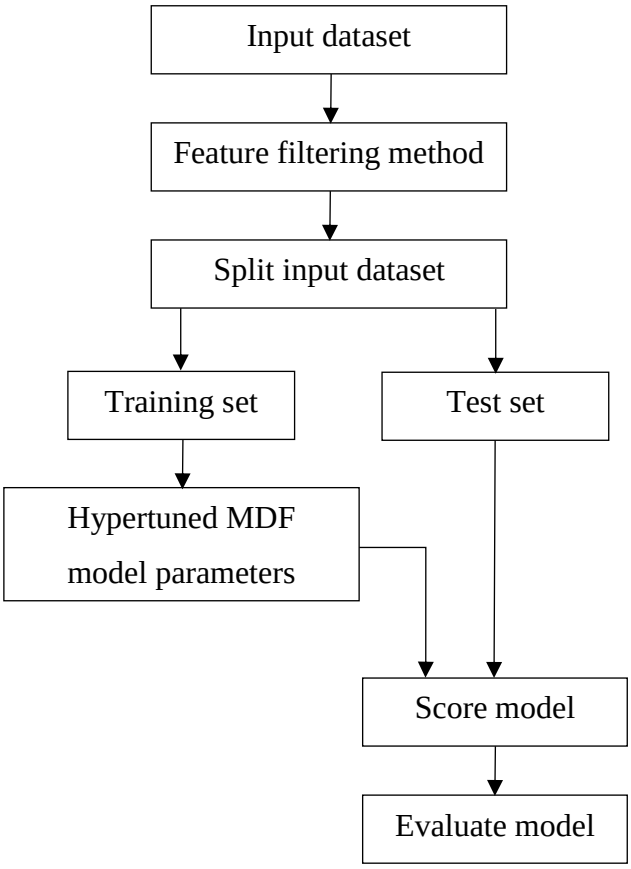


Figure 34: Individual experimental setup.

The TMH experiment is implemented on the output of each feature selection dataset. In addition, the entire dataset with all features was also evaluated to obtain a baseline performance. It should be noted that the dataset is split using an 80:20 ratio, where the model

was trained on 80% of the dataset and tested on the remaining 20%. The experiment setup is shown in Figure 35.

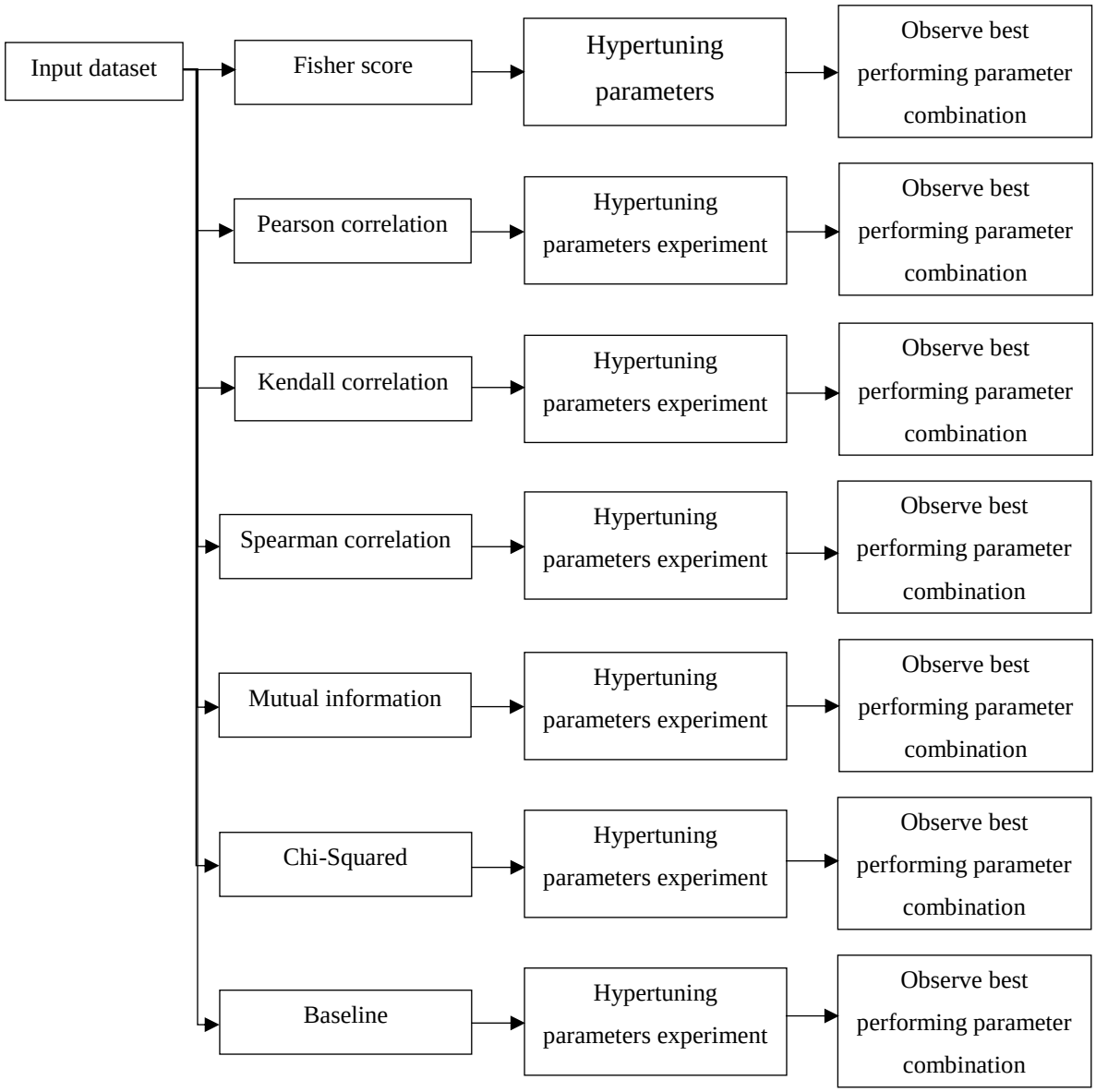


Figure 35: Hypertuning experimental setup.

The output of the TMH experiment is found in Table 11. The TMH experiment produced five models for each feature selection method, each with different model parameter configurations. The same parameter set was found across all the respective feature selection methods. These produced models form the basis of the model performance experiment where each model is trained on the respective feature selection dataset.

Table 11: Hypertuned model parameters

Hypertuned model	Minimum number of samples per leaf node	Number of random splits per node	Maximum depth of the decision trees	Number of decision trees
Model 1	5	1012	41	24
Model 2	11	539	59	22
Model 3	1	390	22	30
Model 4	8	868	63	2
Model 5	9	83	12	15

5.5.5 Model performance experiment

The configuration of the parameters found in Table 11 identifies model configurations which produces the highest accuracy. However, to identify the best performing model, additional metrics needed to be evaluated. These performance metrics were precision, recall and F1-score. These metrics were evaluated to ensure that the complete performance of the respective models was evaluated. A model was trained with each of the parameter configurations shown in Table 11. These configurations were imposed on each feature selection method and baseline dataset. The experimental setup for each feature selection method is shown in Figure 36.

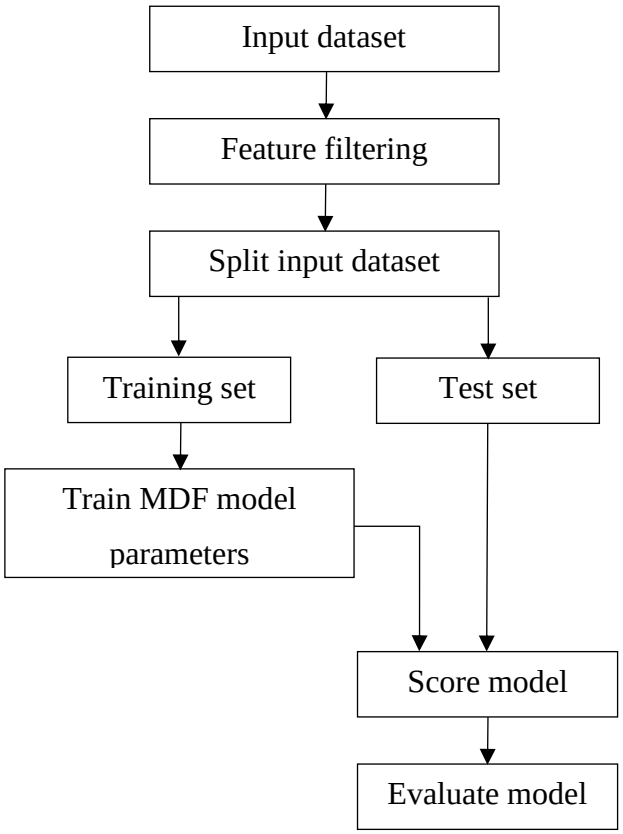


Figure 36: Classifier training configuration for one feature selection method.

The experimental setup was expanded to each dataset from the respective feature selection methods. The complete classifier training configuration is represented in Figure 37.

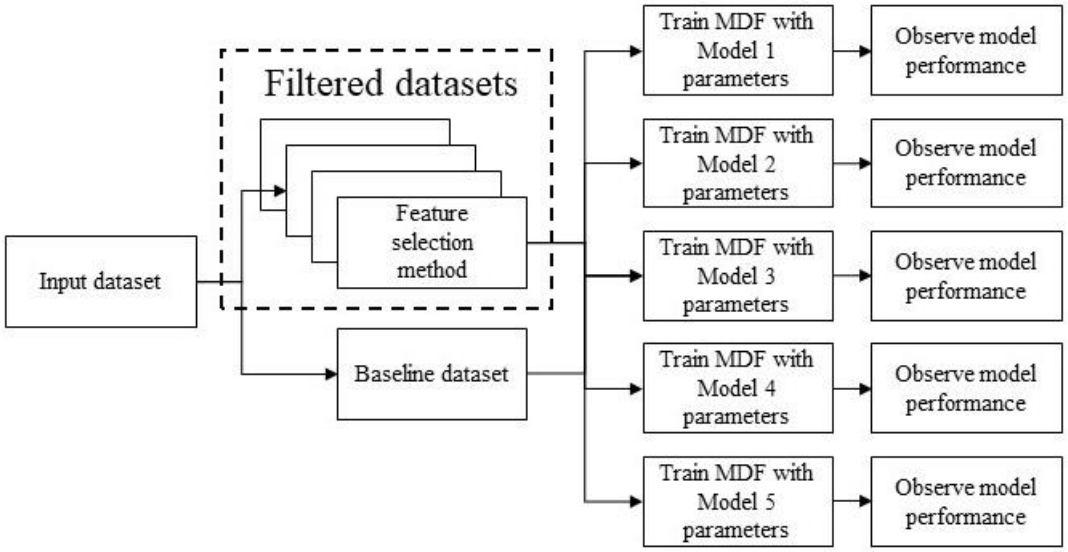


Figure 37: Complete feature selection classifier experiment for all model parameter configurations.

This configuration computes all the parameters found in the hypertuning experiment shown in Table 11. The experiment in Figure 37 was done to ensure that all metrics are evaluated for each feature selection dataset. In addition to these feature selection methods, the entire dataset is input into the experiment to create a baseline performance of the respective models for evaluation.

5.5.6 Generalisation Experiment

Once the best performing model was selected, the selected model is then tested using a cross-validation model. Cross-validation was done in order to obtain an insight into how well the classifier generalises on unseen data. The generalisation experimental setup is shown in Figure 38.

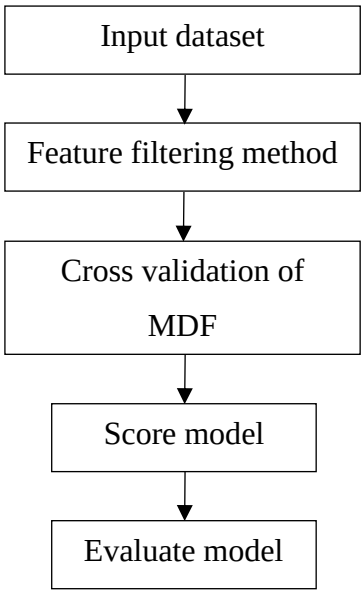


Figure 38: Cross-validation experiment.

The experiment splits the dataset into ten folds. Each fold is evaluated with regard to precision, log loss and recall. These metrics were implemented in each class in the dataset. On completion of the classifier experiment, the best performing model and the dataset were selected for further use in the recommender system.

5.6 Classification results

The results from the classification experiment are discussed in this section. The complete results table is seen in the C.1 Classifier training results in Appendix C. The results from each classification model and feature selection method were analysed in isolation, and the best performing configuration was chosen. Each model was evaluated for accuracy, precision, recall and F1-Score. The classification results summary of the best performing MDF is shown in Table 12.

Table 12: Summary of classification results

Filter algorithm	Number of features	Tuning parameters				Performance metrics			
		Minimum number of samples per leaf node	Number of random splits per node	Maximum depth of the decision trees	Number of decision trees	Average accuracy	Average precision	Average recall	F- score
Pearson Correlation	83	1	390	22	30	0.7755	0.7755	0.7755	0.7755
Kendall Correlation	86	1	390	22	30	0.7554	0.7554	0.7554	0.7554
Spearman Correlation	78	1	390	22	30	0.7556	0.7556	0.7556	0.7556
Chi Squared	62	5	1012	41	24	0.7744	0.7744	0.7744	0.7744
Fisher Evaluation	36	1	390	22	30	0.7642	0.7642	0.7642	0.7642
Mutual information	58	5	1012	41	24	0.7758	0.7758	0.7758	0.7758
Baseline	469	5	1012	41	24	0.7888	0.7888	0.7888	0.7888

The baseline dataset contained the entire dataset and produced a performance of 78% across the respective metrics. The Mutual Information produced a result closest to the baseline with an overall score of 77.58% and 58 features using a model 1 parameter configuration. Therefore, the Mutual Information feature selection method and model 1 parameter configuration were chosen and evaluated in the generalisation experiment. Model 1 parameter configuration was chosen as this model performed the best in comparison to the other configurations in the experiment. It was noted that the baseline produced a better performing classifier. However, the Mutual Information dataset produced a similar result with 12.37% of the dataset. This reduction in the size of the dataset resulted in a computationally less expensive classifier and personalised recommender system. The confusion matrix for the Mutual Information model is shown in Figure 39.

	Predicted Class					
Actual Class	Class	0	1	2	3	4
	0	81.2%	16.3%	1.5%	0.4%	0.5%
	1	17.9%	71.8%	6.1%	1.5%	2.7%
	2	1.9%	8.2%	74.1%	4.3%	11.4%
	3	0.2%	1.9%	6.3%	81.3%	10.3%
	4	0.8%	3.2%	13.3%	8.7%	74.0%

Figure 39: Mutual Information confusion matrix.

The confusion matrix shows the performance of the Mutual Information classification model for predicting each class. The Mutual Information classifier performed well in predicting class zero and three. The Mutual Information classifier does not perform as well on predicting class one. As mentioned, the Mutual Information classification model was input into a 10-fold cross-validation experiment to determine how well the chosen algorithm generalises on new data. The entire experiment can be seen in Appendix C. The summary showing the mean of the results of the generalisation experiment is shown in Table 13.

Table 13: Cross-validation model results

	Metric	Mean	Standard deviation
Class 0	Average log loss	0,498	0,015
	Precision	0,801	0,009
	Recall	0,830	0,005
Class 1	Average log loss	0,797	0,012
	Precision	0,703	0,006
	Recall	0,691	0,009
Class 2	Average log loss	0,799	0,015
	Precision	0,768	0,004
	Recall	0,738	0,009
Class 3	Average log loss	0,464	0,009
	Precision	0,848	0,004
	Recall	0,908	0,005
Class 4	Average log loss	0,658	0,012
	Precision	0,841	0,009
	Recall	0,800	0,007

For each class the log loss, precision and recall were evaluated. The log loss for class zero, three and four were the lowest. These values were seen to be 0.498, 0.4643 and 0.658, respectively. These results indicated adequate performance. However, class one and two had a log loss of 0.797 and 0.799, respectively. This result indicated poor performance amongst these respective classes.

The precision for class three was the highest at 0.848, followed by class four and zero with a mean value of 0.841 and 0.81, respectively. The performance for class one and two was consistent with the log loss metric as the precision was seen to have a poorer performance of 0.703 and 0.768, respectively. The recall for classes three, zero, four were seen to be the highest

with a performance value of 0.908, 0.830 and 0.800, respectively. Class two and three were again found to have a poorer performance of 0.691 and 0.738, respectively.

The Mutual Information model was seen to have consistent performance across the respective classes with a relatively low standard deviation, with the highest deviation being 0.015. Therefore, the performance metrics were said to be consistent across folds.

5.7 Summary of chapter

This chapter explores the various datasets and pre-processed steps which were implemented in order to obtain a single balanced aggregated dataset. The various feature selection methods were then implemented to identify the most prominent features within the dataset. Thereafter, the process for the development of an MDF classifier was defined, and the classifier results were shown and explored.

Chapter 6
Recommender system development

This chapter focuses on the development of the behaviour recommender system. All the steps implemented within the development of the behaviour recommendation system is explored in section 6 and 6.2. Section 6.3 focuses on the results obtained from the recommender system experiment. In this section, the mean and median-based recommender systems are evaluated, respectively. This evaluation is done to identify the best performing recommender. Thereafter, in section 6.4. the results are discussed, and the performance is evaluated.

6.1 Collaborative filtering

Collaborative filtering was used as the basis for the behaviour recommendation system which was implemented within the MIP. This method allowed users within a specific category group to find similar users within the same or higher category. This method does have a cold start problem. However, behaviour recommendations are made with current data from the respective target user, therefore reducing the need for historical data. An advantage of this method was that computation was limited because the MIP had limited recorded behaviours. These recorded behaviours were further reduced in the feature selection phase. This reduction reduced the sparsity problem which plagues collaborative filtering type recommenders. A disadvantage of this method was that recommendations are based mainly on similar users found in the dataset. For example, users within the same or lower status group could be identified as being the most similar. If these users were used as the basis for the recommendation, the recommended action would result in users descending or not moving within the MIP. Therefore, care was taken to ensure only similar users who are found to be in a higher category were used as the basis for behaviour recommendation.

6.2 Overall recommendation process

The overall personal recommendation process can be seen in Figure 40.

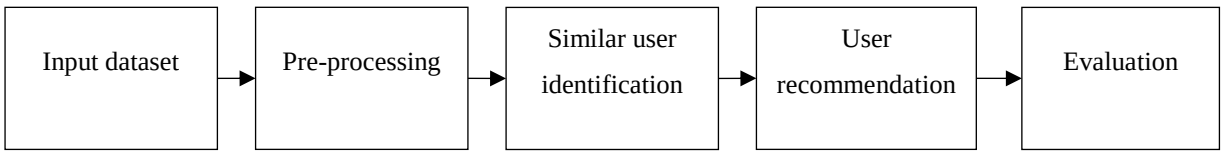


Figure 40: Recommendation process

The personal recommender process inputs the Mutual Information dataset which was found in the feature selection phase. The dataset was then pre-processed to transform the data into the form needed for the behaviour recommendation algorithm. Thereafter, an algorithm was developed to find similar users in the dataset. This dataset was then input into the recommendation engine where a recommendation was made based on the status of the user and the similarity neighbourhood. The recommender performance was then evaluated to explore the performance of the developed recommender.

6.2.1 Input dataset

The input dataset was found in the feature selection and classification phase. This dataset can be seen in Appendix B. The characteristics of the dataset can be seen in Table 14.

Table 14: Input Dataset Details

Metric	Details
Columns	59
Rows	231 980
Size	76.653 KB

From Table 14, it was seen that the dataset had been reduced from 483 to 59 features. These features were analysed in order to identify point contributor and non-point contributor features. The labelled columns are shown in Appendix B. The analysis of this dataset is shown in Table 15 and Table 16.

Table 15: Nonpoint-contributor summary

Nonpoint-contributor dataset	Total
Communication	4
Date	3
Entertainment	2
Location	5
Personal	2
Policy	1
Shopping	17

Table 15 shows the details of the nonpoint-contributor features within the filtered dataset. Features containing shopping information were the most common in this dataset. This finding

was plausible as seven of the 13 datasets contained shopping information. Although these columns did not contribute to the points of the user, it did contribute to identifying similar users within the dataset. The point contributor dataset summary can be seen in Table 16.

Table 16: Point-contributor summary

Point contributor dataset	Total
Health	10
MIP points	1
Policy	8
Safety	5
Status	1

From Table 16, the health type features appear most frequently within the features. This result was followed closely by policy features. Policy features was classified as additional policies taken by users. This result was plausible as it corresponded to the analysis of the MIP in Chapter 4.

6.2.2 Pre-processing

During the balancing of the dataset, invalid values are introduced within the ClientNo and PolicyFull columns. These values were introduced because of the nature of the SMOTE+TOMEK algorithm, where synthetic values did not correspond to a valid entry in the decoding phase. During the classification phase, this was not an issue as these columns were excluded from the analysis. However, during the recommendation phase, it was imperative that identifier columns be valid. Therefore, nulls within the ClientNo and PolicyFull columns are inspected, and rows with missing values were removed. Following the removal of missing value, all data types within the respective columns were inspected. This inspection was done to ensure all columns are of a numeric type. All non-numeric columns were encoded into numeric values using the *sci-kit-learn encoding* library. Encoded columns are shown below:

- ClientNo
- PolicyFull
- ResAddress3

All encoded values were decoded on completion of the recommendation. The dataset was then converted to a user-item matrix.

6.2.3 Recommendation algorithm

This section explores the steps within the recommendation algorithm. The details of the algorithm can be seen in Figure 41.

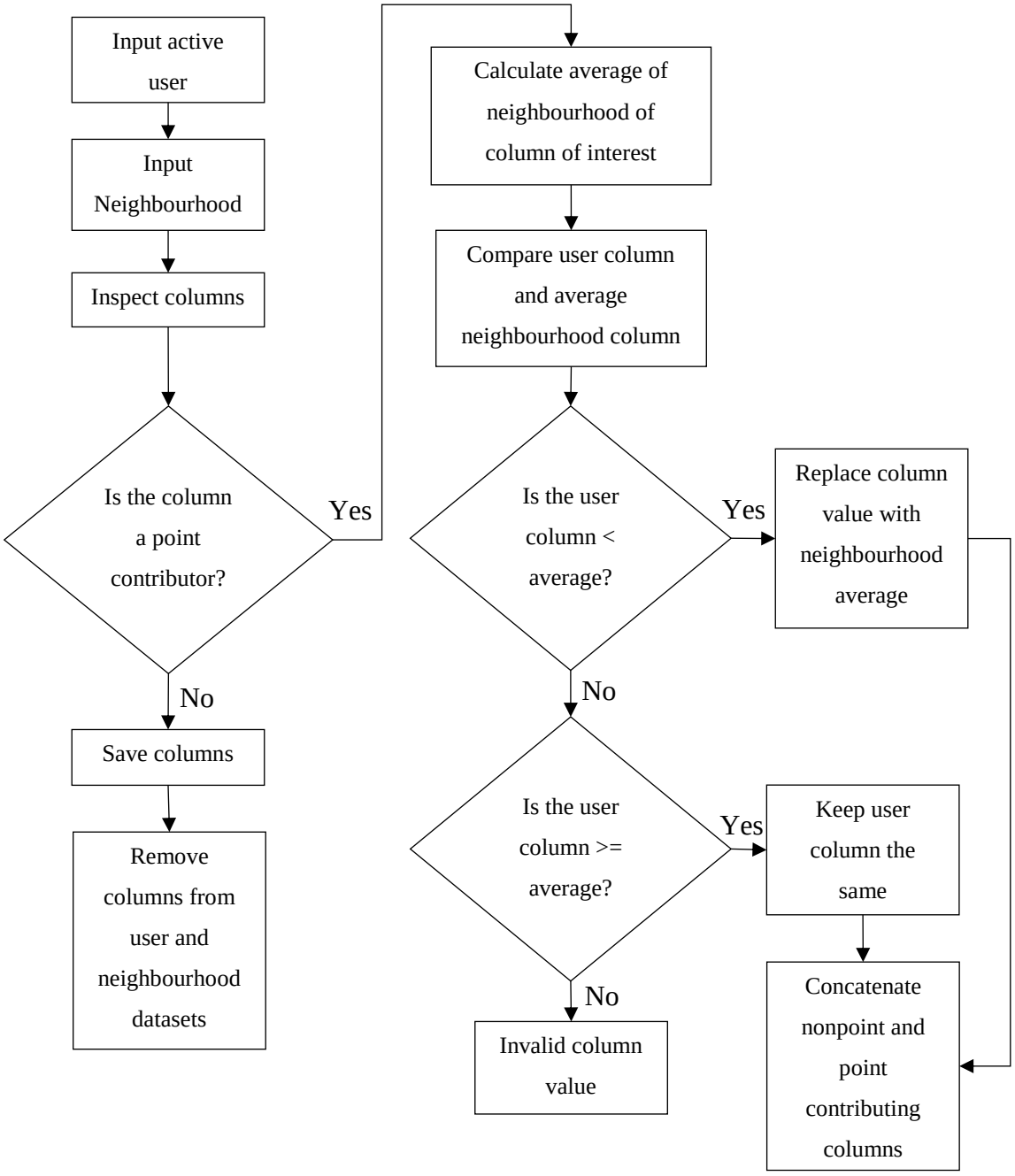


Figure 41: Recommender algorithm flowchart.

The algorithm inputs an active user and similarity neighbourhood. The similarity neighbourhood is selected in section 6.2.4. The respective features were then analysed to identify point contributor features. Nonpoint-contributor columns were removed from the recommender as only actions that contributed to the movement of the user of interest (UOI) within the program was kept. The average of the users within the neighbourhood was then calculated. The recommendation equation for the UOI was defined by the following:

$$u_{col} = \begin{cases} neig_{ave}, & u_{col} < neig_{ave} \\ u_{col}, & u_{col} \geq neig_{ave} \end{cases} \quad (26)$$

Where u_{col} is the active user value within the dataset identity column and $neig_{ave}$ is the average column value of the neighbourhood for each point contributing column. Equation 26 compares the values within the respective columns to that of the neighbourhood group. Columns which have a value which is less than the average was replaced with the average of the neighbourhood. If the value in the user column was more than or equal to the average of the neighbourhood value the initial value was maintained. The algorithm ensures that only areas which require improvement is recommended. Once the recommendation had been made, the result was concatenated with the original nonpoint columns.

The recommender algorithm was not without its shortcomings. The basis of the recommender was made on the average value of the neighbourhood. This links to the neighbourhood size issue where larger neighbourhoods may include users who are outliers and therefore would result in a skewed recommendation. This skew in recommendation could be reduced by using the median value in the similarity neighbourhood as the comparison value. Using the median value would reduce the effect of outliers on the reference result. Another issue with the algorithm was that recommendations were made on the entire point-contributing dataset and did not consider recommending increased activity in actions already performed by the user. Therefore, the recommender may advise additional actions to the user which may be unnecessary. The recommender could be made more efficient by first recommending an improvement on actions already being undertaken by the respective user and then recommending additional actions.

6.2.4 Similar user investigation

The effectiveness of the recommender was highly dependent on the neighbourhood of similar users. Within the MIP, users were placed into categories. These were individuals who had a similar number of points based on their point contributing actions. However, this does not mean

that users in the same group are similar. Within the input dataset, there are point contributing behaviours and non-point contributing behaviours. Similar users across all the different categories can be found using the complete collection of behaviours. This collection of behaviours forms the basis of the similarity algorithm.

The similarity algorithm implemented a KNN clustering algorithm which identified similar users to a chosen user. The identification of similar users was made by computing the Euclidean distance of respective users to the UOI. Therefore, closest users to the UOI was said to be the most similar. The similarity algorithm can be seen in Figure 42.

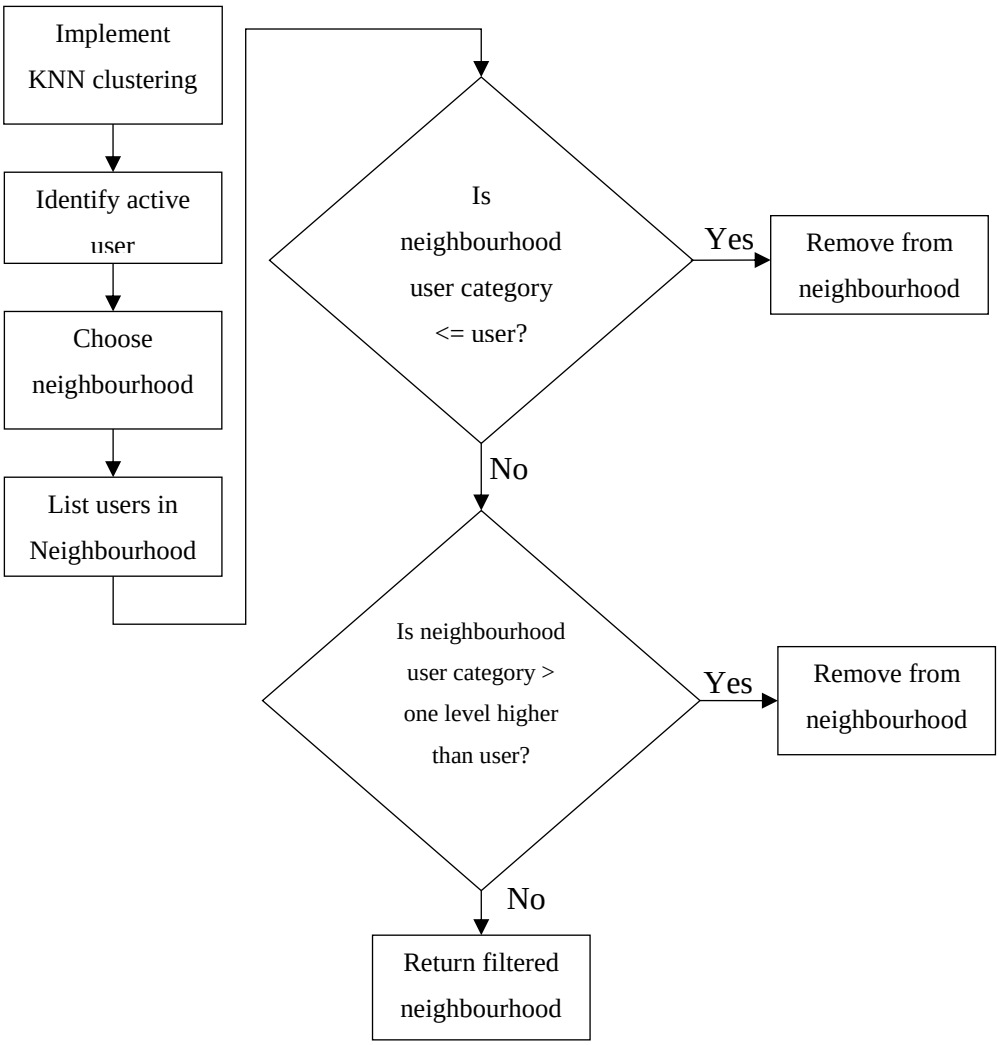


Figure 42: User similarity algorithm.

Neighbourhood size defined the number of similar users to be included in a similarity neighbourhood. Once the neighbourhood size has been chosen, the users were listed in ascending order from most similar to least similar. The neighbourhood was then filtered to remove similar users in the same or lower category of the UOI. Filtering was done to ensure

that the recommender did not recommend actions which would result in a user remaining in the same category or decreasing to a lower category. The neighbourhood was then filtered further for users who were not immediately above the user's category. This filter was imposed to ensure that the proposed recommendation did not require an action which requires a large amount of effort. Users within the Private category did not filter users from the same category, as the Private category was seen to be the highest. Therefore, Private users were recommended to maintain their position.

The similarity algorithm was not without its disadvantages. The main drawback was that the size of the neighbourhood could have a significant effect on the recommendation. The similarity algorithm uses a Euclidean distance to find the most similar users. Where similar users were found to be a short distance away, and less similar users were further away. Therefore, the size of the chosen neighbourhood affects the quality of users within the neighbourhood. The different size of neighbourhoods can be seen in Figure 43.

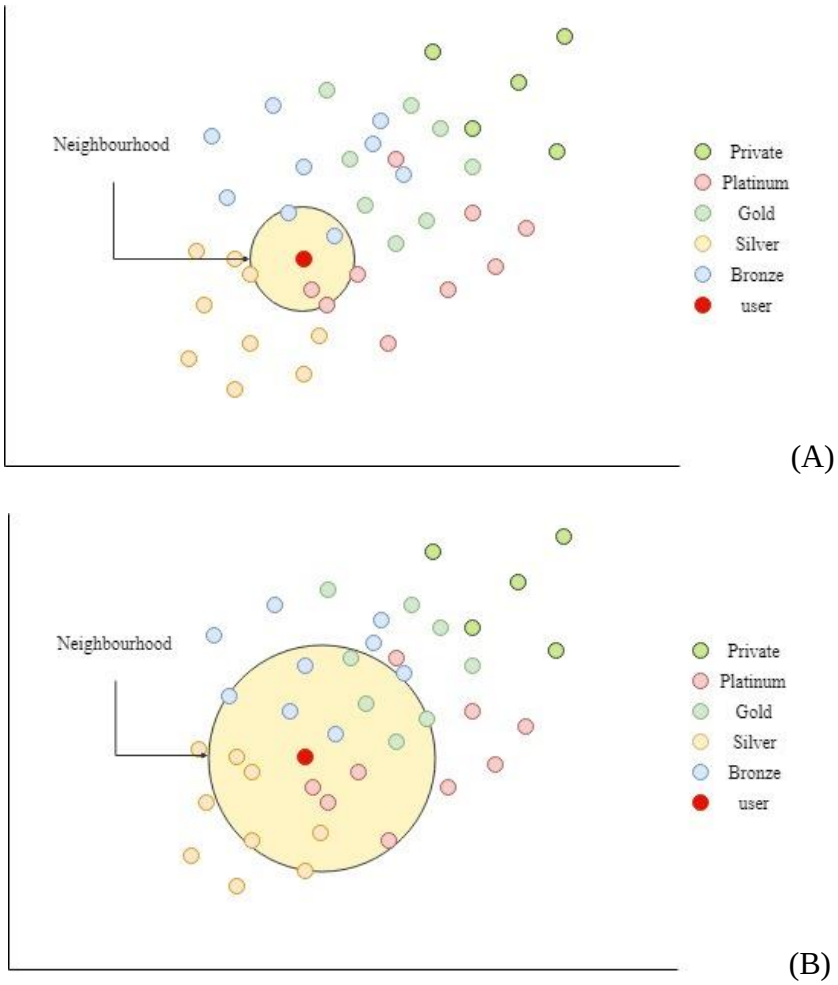


Figure 43: Neighbourhood selection.

The objective of the neighbourhood selection was to identify similar users in the category above the UOI. Therefore, in the case where a small neighbourhood is selected as shown in A, the users in the neighbourhood may not be from one category higher, but all from the same similarity group. This neighbourhood would result in a recommendation which would not promote behaviour change. In the case of B, a larger neighbourhood is selected. Although there is a higher probability of covering users from different groups, some of the users within this group may not be the most similar and skew the recommendation.

6.2.5 Recommender performance investigation

To investigate the dynamics of the recommender, several experiments were conducted. For each of the experiments, the neighbourhood size was varied to size 10, 30 and 50 users. The neighbourhood was varied to identify the best performing size for the recommender. A popular recommender was implemented to serve as a baseline for the performance of the personalised recommender system. For this experiment, the recommendation type was also varied between the mean and median value of the neighbourhood. The effect of the actions recommended was also explored, where recommendations were made on the complete set of point contributing actions and a user's current actions. The two different actions sets were investigated to identify whether a user could change existing behaviour as opposed to adopting new behaviour to advance within the MIP. The experiment configurations can be seen in Table 17.

Table 17: Recommender investigation parameter configuration

Experiment	Neighbourhood size	Recommender type	Recommended action
Baseline	All	Mean	Complete action
Baseline	All	Mean	Existing action
Baseline	All	Median	Complete action
Baseline	All	Median	Existing action
One	10	Mean	Complete action
One	10	Mean	Existing action
One	10	Median	Complete action
One	10	Median	Existing action
Two	30	Mean	Complete action

Experiment	Neighbourhood size	Recommender type	Recommended action
Two	30	Mean	Existing action
Two	30	Median	Complete action
Two	30	Median	Existing action
Three	50	Mean	Complete action
Three	50	Mean	Existing action
Three	50	Median	Complete action
Three	50	Median	Existing action

6.2.6 Evaluation

The recommender is evaluated in order to validate its effectiveness. The evaluation process is shown in Figure 44.

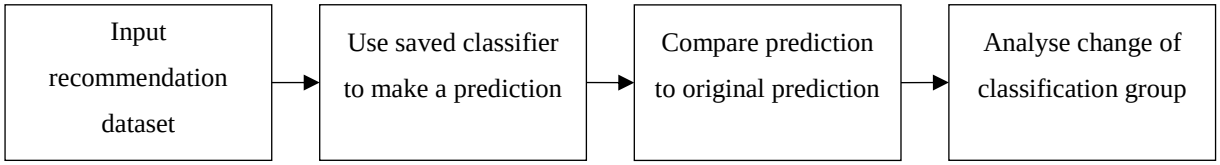


Figure 44: Recommender evaluation process.

The basis of the evaluation phase was to use the trained classifier model developed in section 5.5.4 to determine whether the recommendation resulted in movement within the MIP. The classifier was used to make a prediction on the recommended dataset and this prediction was compared against the initial dataset prior to recommendation.

Classification of recommendation dataset

The trained classifier formed a significant component in the evaluation of the recommender. The evaluations were implemented within the AMLS, the flowchart of this process is seen in Figure 45.

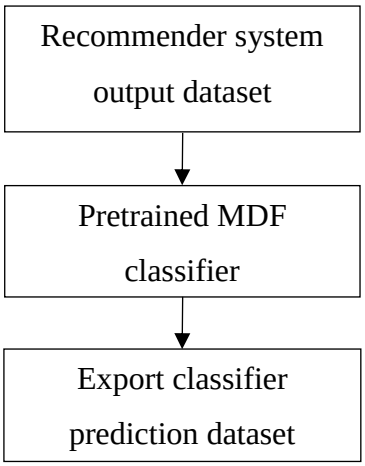


Figure 45: Classification of the recommended dataset.

Prediction comparison

In this phase, the comparison between the initial classification and the classification after the recommendation was examined. The initial classification predictions were used to reduce the propagation of inaccuracies within the classifier. The movement of the respective uses was obtained by subtracting the new prediction from the initial classification. The aim was to obtain an insight into whether users moved to a higher category or was unaffected by the recommendation. The description of the movement class values is shown in Table 18.

Table 18: Prediction comparison key

Movement class	Description
-4	The user moved four categories down
-3	The user moved three categories down
-2	The user moved two categories down
-1	The user moved one category down
0	User never moved
1	The user moved one category up
2	The user moved two categories up
3	The user moved three categories up
4	The user moved four categories up

A negative movement class implies that the recommendation had the opposite effect on the user and the action would decrease the user's category. A zero-movement class implies that the recommendation had no effect on the user and that the user remained in the same category. Finally, a positive movement class implies that the user advanced in category. The aim of the recommender is for the majority of the users to have a comparison score of one, which indicates that the user moved one category up.

The distribution of the respective results of each recommender was analysed in order to obtain the recommender and parameters which performed the best in this context. This evaluation was done by implementing a histogram of the frequency of the comparison scores. A well-performing recommender was defined as having minimal negative and high positive scores and had the majority of users within movement class one.

6.3 Recommender results

This section explores the results of the recommender algorithm. The comprehensive results of the experiment can be seen in Appendix D. In this section the mean-based, and median-based recommender results are explored. Thereafter, the best performing algorithm is selected, and a comparison between these recommenders are made.

6.3.1 Mean-based recommender

The summarised results showing the best performing mean-based results from the recommender are shown in Table 19.

Table 19: Summarized mean-based recommender results

		Category movement								
Experiment	Neighbour-hood	-4 (%)	-3 (%)	-2 (%)	-1 (%)	0 (%)	1 (%)	2 (%)	3 (%)	4 (%)
Baseline	Popular (all)	0.00	0.00	0.06	2.56	11.38	17.51	29.91	17.96	20.62
One	10	0.00	0.00	0.46	3.23	14.63	25.80	29.20	12.50	14.17
Two	30	0.00	0.00	0.50	2.92	14.01	25.83	28.40	12.51	15.83
Three	50	0.00	0.00	0.49	2.77	13.53	25.55	27.79	13.10	16.76

Table 19 shows the percentage movement of the individuals in the MIP after the recommendation. The distribution of these results is shown in the sections below.

Popular mean-based recommender

The distribution of the popular mean-based recommender is shown in Figure 46.

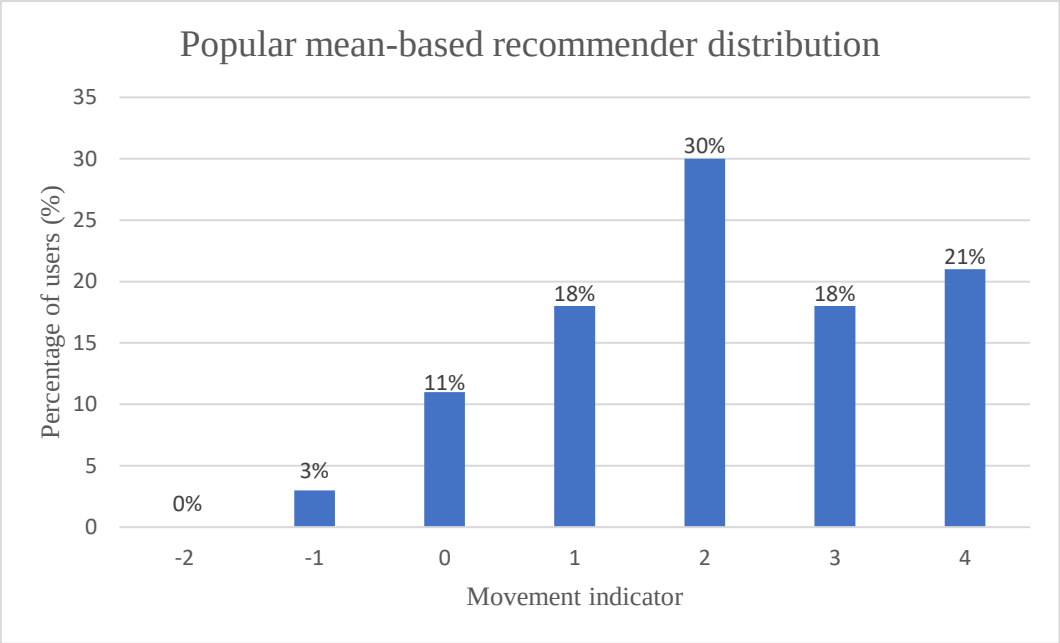
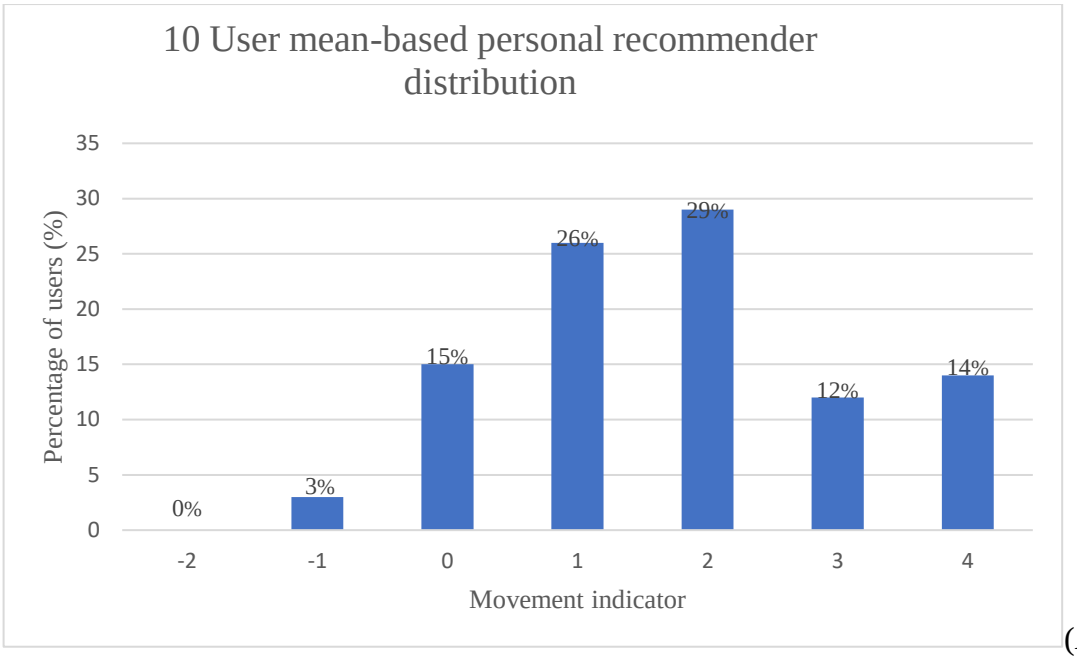


Figure 46: Popular mean-based recommender results

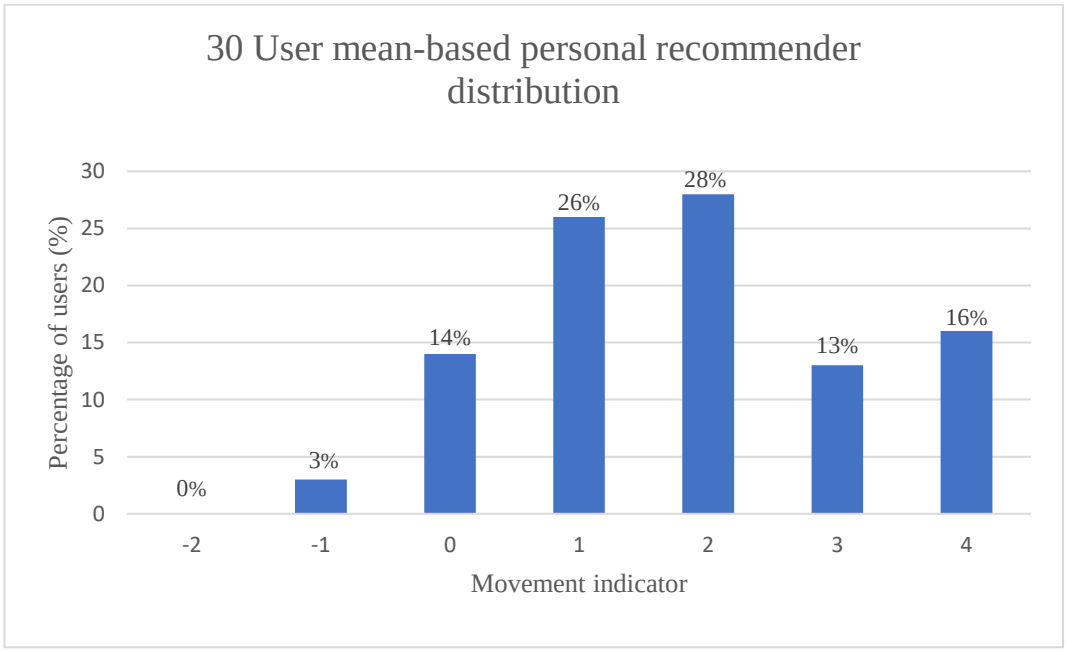
The popular based recommender was computed as a baseline to evaluate the performance of the personalised recommender compared to a popular-based recommender. From Figure 46 it is seen that the popular-based recommender performed well with 87% of users ascending the MIP. It was also noted that the data skewed to the right with 18% in the movement class of one.

Personalised mean-based recommender

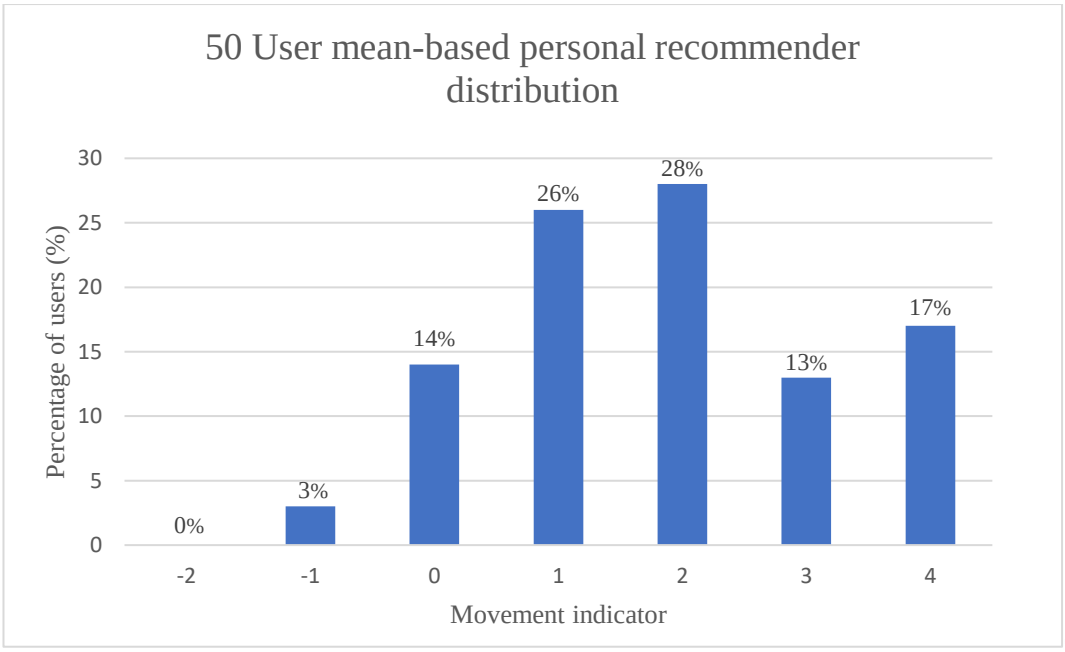
In this section, the results of the personalised mean-based recommender are shown. Figure 47 shows a personalised mean recommender of neighbourhood size 10, 30 and 50, respectively.


10 User mean-based personal recommender distribution
Percentage of users (%)
35
30
25
20
15
10
5
0
0%
3%
15%
26%
29%
12%
14%
-2
-1
0
1
2
3
4
Movement indicator

(A)


30 User mean-based personal recommender distribution
Percentage of users (%)
30
25
20
15
10
5
0
0%
3%
14%
26%
28%
13%
16%
-2
-1
0
1
2
3
4
Movement indicator

(B)



(C)

Figure 47: Personalised mean-based recommender distributions.

It was noted that across all graphs the individuals within class one remained unchanged at 26% for the different neighbourhood sizes. The size of the neighbourhood was shown to have a marginal effect on the movement, with 3% more users moving up in category in graph C than in graph A.

Mean-Based recommender experiment comparison

In this section, the popular and personalised recommenders' performance were evaluated to identify the effectiveness of the personalised recommender verses a general popular based recommender. The different personalised recommenders were also compared with each other to identify the best neighbourhood size. The comparison of the different recommenders is shown in Figure 48.

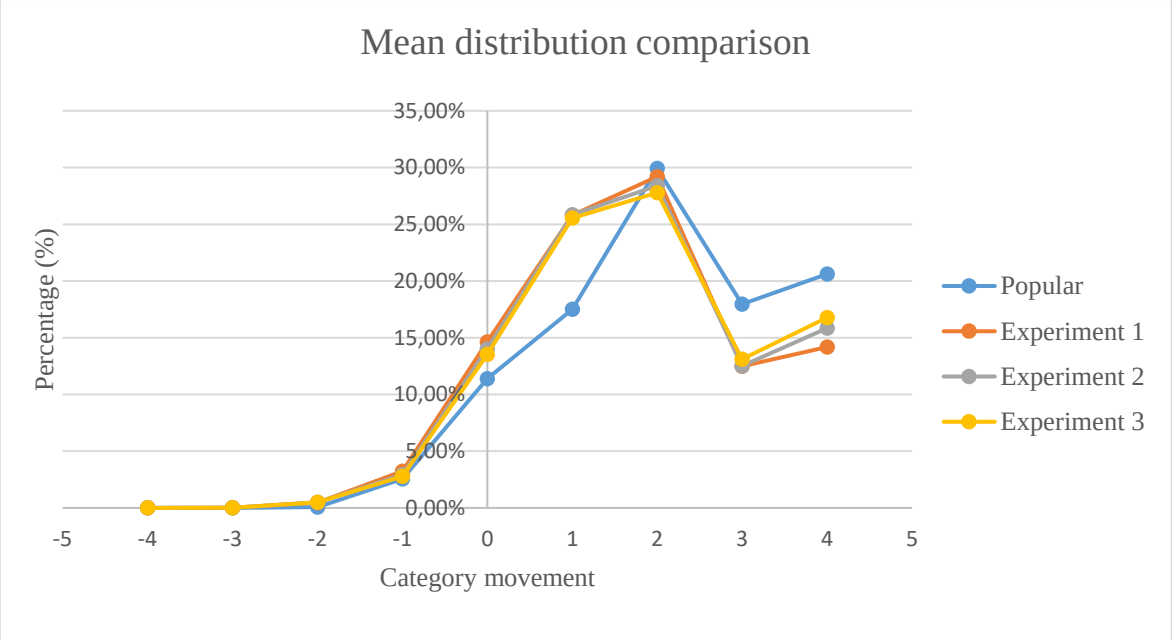


Figure 48: Mean-based recommender comparison.

Both the popular and personal recommender was seen to recommend actions which resulted in successful category movement. The popular recommender in Figure 48 moved an overall 87% of users to higher categories, as opposed to the personalised recommender which moved a maximum of 84% to higher categories. However, the personalised recommender outperformed the popular recommender by moving 8% more user to the desired category, one class. The personalised recommender result also outperformed the popular recommender by moving fewer users to class three and four. The personalised recommender experiments were similar in performance with marginal differences across the different movement classes. The size of the neighbourhood marginally affected the movement of users, with more users moving up in the program. However, the users within class one are unaffected by the larger neighbourhood size. Therefore, experiment one was selected as the best performing personalised recommender as it moved fewer users to class three and four.

6.3.2 Median-based recommender

The Summarised results showing the best performing median-based results from the recommender are shown in Table 20.

Table 20: Summarised median-based recommender results.

Experiment	Similarity	-4 (%)	-3 (%)	-2 (%)	-1 (%)	0 (%)	1 (%)	2 (%)	3 (%)	4 (%)
Baseline	Popular	0.00	0.03	0.30	1.94	43.25	42.90	7.75	2.94	0.89
One	10	0.00	0.01	0.48	3.24	20.58	31.85	21.67	11.44	10.72
Two	30	0.00	0.02	0.54	2.92	25.34	36.02	17.45	9.48	8.23
Three	50	0.00	0.02	0.57	2.70	27.65	37.76	15.47	8.50	7.33

The median-based recommender experiment follows the same process as the mean-based recommender. A popular recommender is implemented to obtain a baseline performance. Thereafter, the neighbourhood was varied from to 10, 30 and then 50 users within the dataset. The distributions of the respective experiments are expanded upon in the various sections below.

Popular Median-based Recommender

The distribution of the popular median-based recommender is shown in Figure 49.

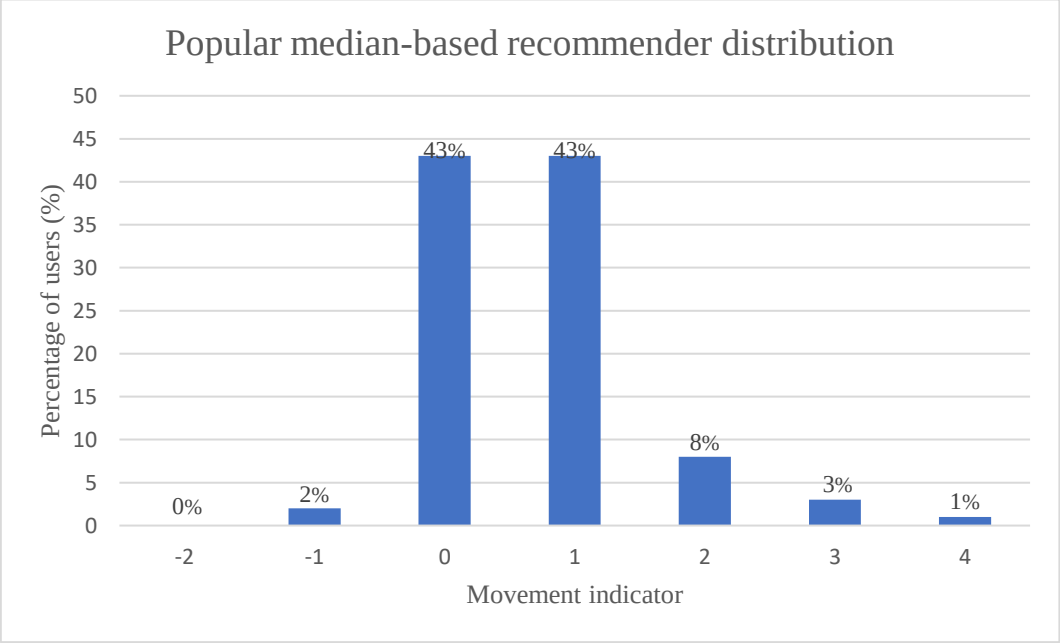
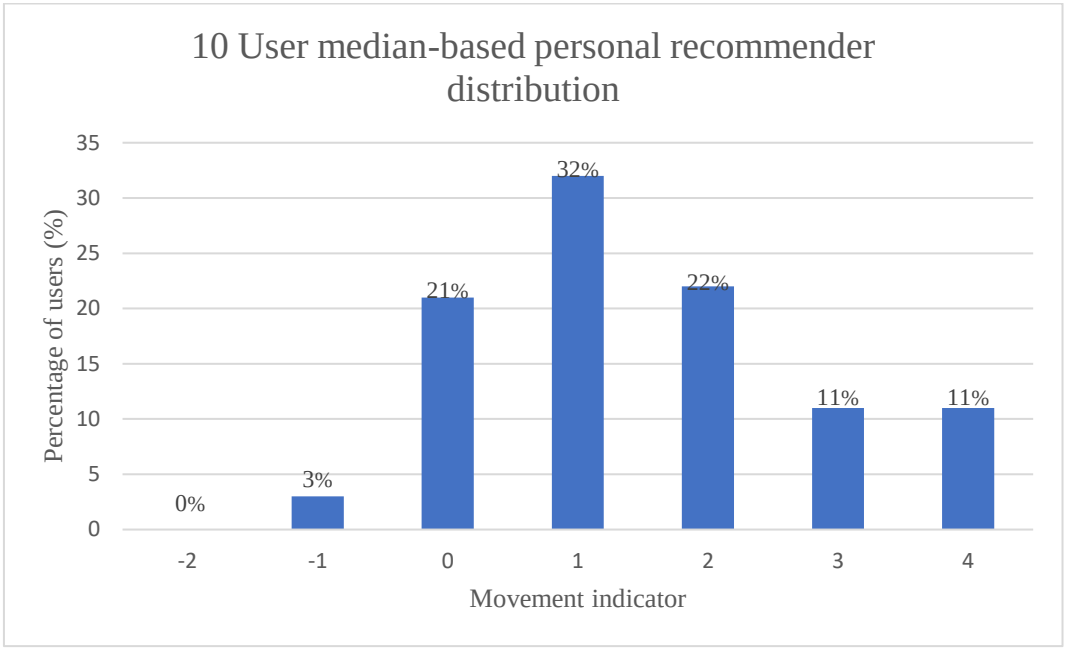


Figure 49: Popular median based recommender distribution.

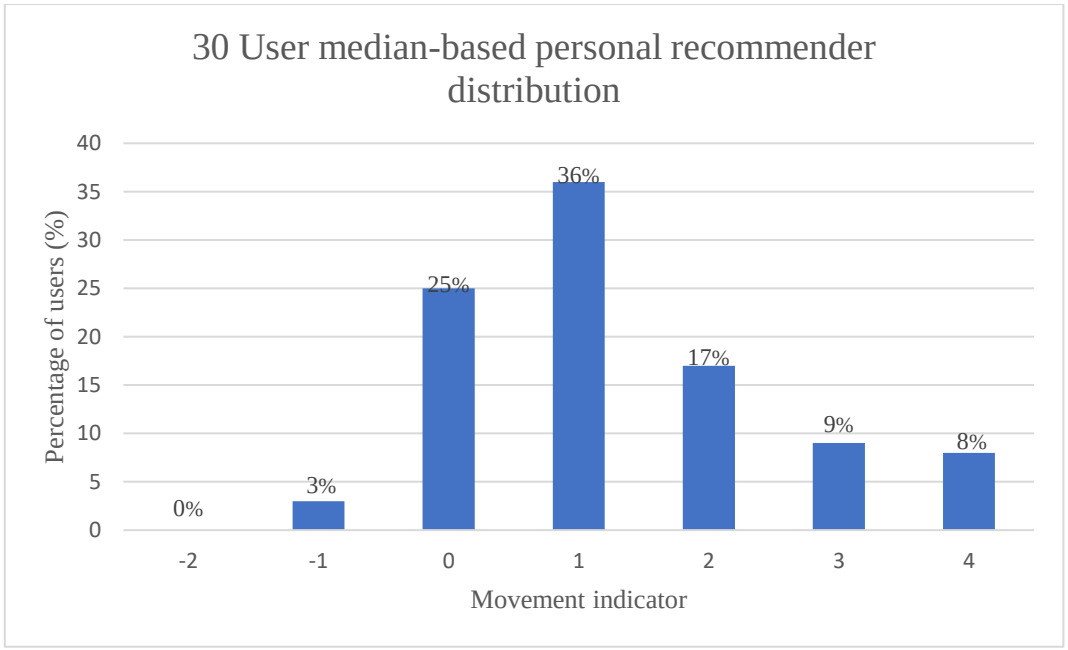
From the graph in Figure 49, 43% of users moved to the desired class one. However, the recommender did not affect the additional 43% of users. An advantage of this result was that a smaller number of users were moved to class two to four.

Personalised Median-based recommender

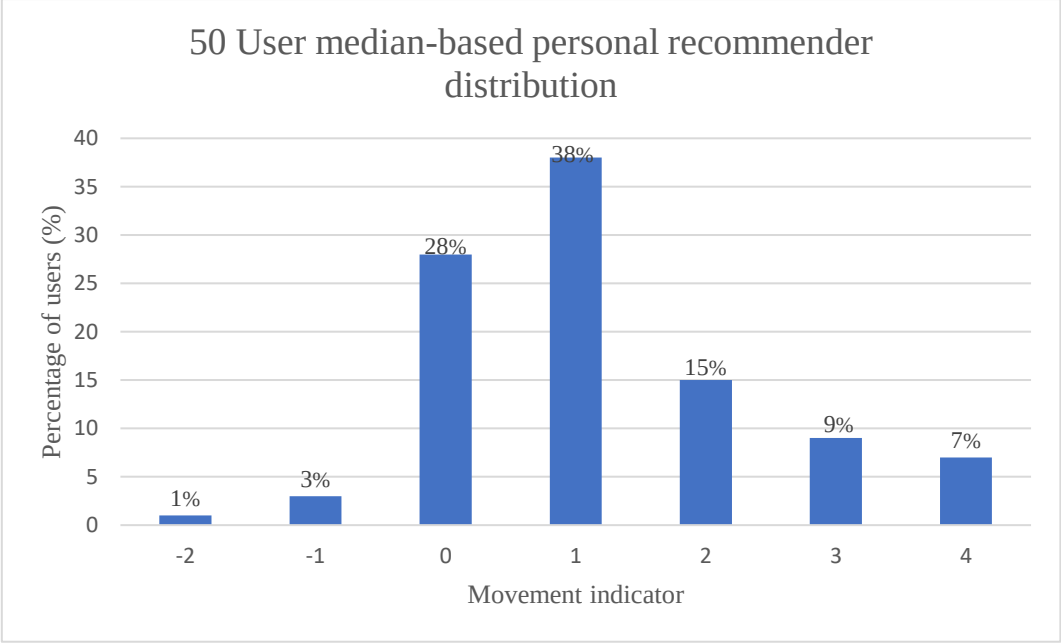
In this section, the results of the personalised median-based recommender are shown. In Figure 50, a personalised median-based recommender of neighbourhood size 10, 30 and 50 are shown, respectively.



(A)



(B)



(C)

Figure 50: Median-based personalized recommender.

Most users within the respective graphs moved to the desired class one, with the user class size ranging from 32% to 38%, respectively for graphs A to C. It was observed that the neighbourhood size affected the number of users which moved to the desired class, where a 6% increase was observed between the 10 and 50 user neighbourhoods.

Comparison of Median-based recommender

In this section, the results of both the popular and personalised median-based recommenders are shown in Figure 51.

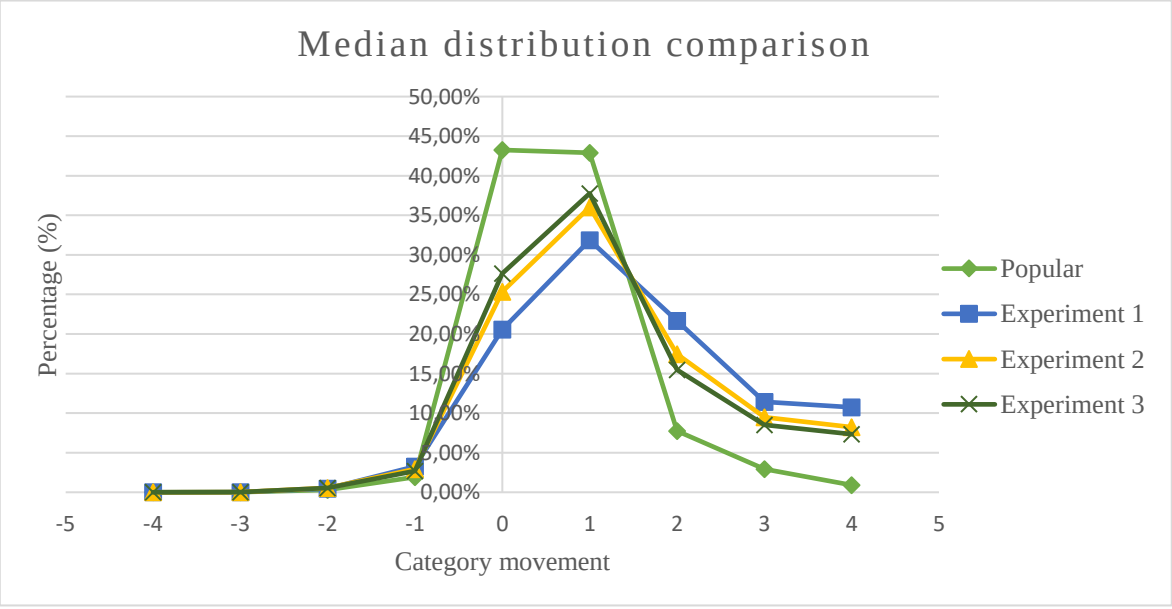


Figure 51: Median-based recommender comparison.

The popular and personal median-based recommenders both showed to be effective in moving users to the desired class. The popular recommender did, however, move the most users to the desired class, which was found to be 5% higher than the closest personal recommender. However, the overall user movement for the popular recommender was seen to be a lower 55%, therefore, being ineffective for 45% of users. The personal recommenders outperformed the popular recommender in moving more users up within the MIP, as well as placing most users into the desired class. The size of the neighbourhood did affect the performance of the personalised recommender, in that, an increase of 6% was observed when the neighbourhood was increased from 10 to 50 users. Experiment three, with a neighbourhood of 50, yielded the best results, with this recommender producing the highest number of users in class one, as well as fewer users assigned to movement class two, three and four.

6.3.3 Overall recommender comparison

In this section, the comparison between the personalised mean-based recommender with a neighbourhood size of 10 and personalised median-based recommender with a neighbourhood size of 50 explored. The comparison between the respective personalised recommenders is shown in Figure 52.

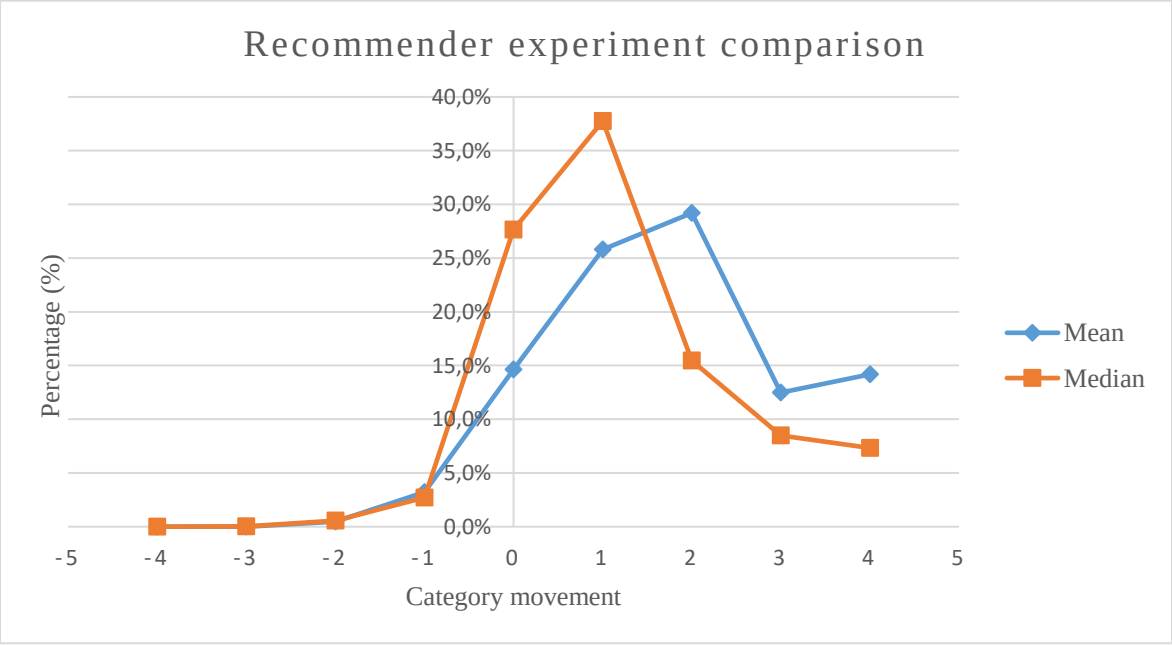


Figure 52: Mean based recommender vs median based recommender.

From the graph, the mean-based recommender moved 12% more users within the MIP. However, the median-based recommender was shown to outperform the mean-based recommender in moving users to the desired categories with the median-based recommender

moving 9% more users to the desired category. The median-based recommender was also seen to be more precise and therefore moving fewer users to class two, three and four than the mean-based recommender. Therefore, from this analysis, the best recommender for the MIP was the median-based recommender with a neighbourhood size of 50.

6.4 Discussion

In this section, the results obtained in section 6.3 is explained and critically evaluated. The results of this work are then evaluated against the objective of the dissertation. From the recommender experiment, the best performing recommender configuration is shown in Table 21.

Table 21: Chosen recommender

Recommender Parameters	Details
Similarity algorithm	KNN
Neighbourhood size	50
Recommender type	Median
Features to be recommended on	All point contributors

The personalised median-based recommender with a neighbourhood size of 50 was chosen as the best performing recommender system. This personal recommender was chosen as it assigned the highest number of users to movement class one for all of the personal recommenders. The chosen recommender moved 69% of users up in categories within the MIP, and 38% of users moved to movement category one. The recommender was evaluated against a baseline popular median-based recommender. The popular median-based recommender did move 5% more users to the movement class one. However, this recommender was not effective on 15% more users than the chosen personalised median-based recommender. The personalised mean-based recommender was shown to move 12% more users up in the MIP than the personalised median-based recommender. However, the personalised mean-based recommender moved 12% fewer users to movement class one than the personalised median-based recommender.

As mentioned, the size of the neighbourhood did influence the performance of the recommender, with the larger neighbourhoods yielding better performance. However, the

largest neighbourhood size investigated was of size 50. As mentioned, the recommendation is based on the median user in the similarity subset. This method reduces the effect of outliers within the group on the recommendation. However, a drawback of this method is that the recommendation of respective users is effectively made on a single user.

Within the results, it is observed that users moved more than one category up as well as moved down in category. Recommendations were made on the values of users within the category higher than the UOI. Therefore, it is implausible for users to move more than one category higher, or to drop in level as the MIP is a points-based program with clear point thresholds. The classifier trained in the classification phase was used to evaluate the recommender in this phase and was trained on both the point and nonpoint features. However, the recommendation is only made on point contributing features. Therefore, the resultant movement of users was exaggerated because of the presence of the nonpoint contributing features of the UOI within the training dataset. The classifier is also seen to have an approximate accuracy of 77%, which, therefore, leaves an error of 23%. This error is propagated into the recommendation evaluation system. The initial prediction values of the classifier were compared with the produced recommendation values to mitigate the effect of the classifier error.

It is noted that bias was introduced during the pre-processing phase, where the text labelling was done manually using a visualisation tool. Another source of bias within the analysis was introduced within the feature selection method. Each feature selection curve was plotted, and the features on the elbow of the curve were chosen manually and included in the development of the recommender system.

The objective of this dissertation was to create a personalised recommendation system within the MIP in order to aid users in rising in the respective categories by providing a lost of actions to be implemented by the user. These recommended actions were also to recommend the minimum action which would result in moving up one category. These objectives have been met as the personalised recommender would provide a UOI with a list of actions which would result in the user moving up in the MIP. The personalised recommender is also designed to recommend actions which would result in a user moving one class up. Therefore, requiring minimum action from the user.

The personal recommender developed in this dissertation was effective in that it provided a recommendation without historical data as was the case in Farrell *et al.* (2012) and Yürüten, (2017). The recommendation method also did not require feedback as was the case in Yürüten,

(2017). The collaborative filter type recommender worked well in the MIP because all desired behaviours were already defined, tracked and the various users labelled. Therefore, enabling the development of a supervised behaviour classification model within the MIP. Areas for improvement would be the reduction of biases within the text classification and feature selection methods. The evaluation of the recommender can be improved by using a model trained only on the point contributing columns.

6.5 Summary

In this section, the behaviour recommendation system is developed. The chapter initially defines the pre-processing steps, experiment process and evaluation criteria for the recommender system. Thereafter, the various recommendation systems are implemented, and the results are shown. The results are then analysed and discussed to evaluate the effectiveness of the recommendation system.

Chapter 7
Conclusion

In this section, the summary of this dissertation is elaborated on in section 7. Thereafter, the additional avenues of work are expanded upon in section 7.2.

7.1 Summary of work

The MIP classifies users into respective incentive categories. These categories are used as an indication of the desired behaviour of users within the program. This dissertation aimed to develop a personalised behaviour recommender which aimed to help individuals within the MIP make healthy choices. These healthy choices were linked to actions within the MIP. These choices are categorised as health, safety, financial and policies taken within the program. For this work, data was provided by Multiply in 13 respective datasets. The datasets were explored in order to identify the data types and information provided within the various datasets. It was found that the data was transactional and therefore required pre-processing which reduced and aggregated the dataset. The resulting dataset was found to have 483 features with 136535 rows. The resultant dataset was found to be highly sparse with missing values. Once aggregated, the complete dataset was used as input for a classification algorithm.

A supervised learning classification was implemented with the individual's category as the target variable. A random forest classifier was used, as it performs well with missing values and is capable of handling high dimensional data. Before classification, the dataset was input into a feature selection algorithm to remove features which did not contribute statistically to the model. The classifier and resultant dataset were then used within the recommender algorithm.

The recommender algorithm aimed to recommend an action which would result in the ascension of users or individuals within the MIP. Recommendations were made to ensure that minimum actions would result in a movement of the user within the MIP. The recommended actions, therefore, were to ensure users moved one category up within the MIP with each recommendation. A collaborative filtering approach was implemented where similar users were identified using a KNN clustering algorithm. The algorithm selected users within different sized neighbourhoods around the user of interest. The similarity between these users was quantified by their Euclidean distance away from the user of interest, i.e. closer users were said

to be more similar than further users. Recommended actions were user-focused, and therefore recommendations were only made on point contributing actions. The recommendation was made by recommending the median value of the identified user neighbourhood. Once the recommendation was made, the dataset was input into the developed classifier. The classifier was used to observe the movement of users within the program. It was found that the personalised recommender resulted in 69% of users ascending the MIP and 38% of users moving one category up.

7.2 Opportunities for future work

In this work, a KNN algorithm was implemented for the identification of similar users. However, the maximum neighbourhood used consisted of 50 users. It is, therefore, suggested that a larger neighbourhood group be investigated to identify the effect on the performance of the personalised recommender. In addition, alternative similarity methodologies should be investigated such as Pearson and Cosine correlations. The recommendation system developed in this dissertation uses the median user value of the neighbourhood as the basis of the recommendation. To improve on this system, an average window around the median value should be computed, thereby reducing the effect of an individual user. To improve on the evaluation of the recommender system, an additional classifier should be trained only on the point contributing features to remove the effect of nonpoint contributing features to improve on this evaluation method. Future work should also explore reducing the bias created within the pre-processing by implementing a text labelling method such as K-means clustering for all text-based datasets. Bias can also be reduced in the feature selection phase by creating standard thresholds to ensure all models are evaluated equally.

An additional avenue for future work is the merging of wearable technology with existing datasets in order to obtain higher resolution data of the performance and habits of individuals. Another avenue is implementing an Introspective retrospective recommendation (IRR) algorithm to users in the higher tiers of the program. It is assumed that it takes an individual some years to ascend to the highest category within the MIP. Therefore, once the user reaches the higher tiers within the program, additional health goals can be made as data from a user's past can be accessed and used within the recommender.

Bibliography

El Aboudi, N. and Benhlima, L. (2016) ‘Review on wrapper feature selection approaches’, in *Proceedings - 2016 International Conference on Engineering and MIS, ICEMIS 2016*, pp. 1–6. doi: 10.1109/ICEMIS.2016.7745366.

Almási, A. D. *et al.* (2016) ‘Review of advances in neural networks: Neural design technology stack’, *Neurocomputing*, 174, pp. 31–41. doi: 10.1016/j.neucom.2015.02.092.

Astala, R. *et al.* (2018) *Filter Based Feature Selection*. Available at: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/filter-based-feature-selection (Accessed: 18 October 2018).

Astala, R. *et al.* (2018) *Multiclass Decision Forest, Microsoft Azure*. Available at: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/multiclass-decision-forest (Accessed: 18 October 2018).

Auria, L. and Moro, R. A. (2009) *Support Vector Machines (SVM) as a Technique for Solvency Analysis, SSRN*. doi: 10.2139/ssrn.1424949.

Baeza-Yates, R. and Ribeiro-Neto, B. (1999) *Modern Information Retrieval, Information Retrieval*. doi: 10.1080/14735789709366603.

Balicki, J. *et al.* (2015) ‘Collective citizens’ behavior modelling with support of the Internet of Things and Big Data’, in *Proceedings - 2015 8th International Conference on Human System Interaction, HSI 2015*, pp. 61–67. doi: 10.1109/HSI.2015.7170644.

Bentley, R. A., O’Brien, M. J. and Brock, W. A. (2014) ‘Mapping collective behavior in the big-data era’, *Behavioral and Brain Sciences*, pp. 63–119. doi: 10.1017/S0140525X13000289.

Berwick, R. (2003) ‘An Idiot’s guide to Support vector machines (SVMs)’, *Z - NO JOURNAL*, pp. 1–28. doi: 10.5194/amt-2-55-2009.

Bidgoli, A.-M. and Parsa, M. N. (2012) ‘A Hybrid Feature Selection by Resampling, Chi squared and Consistency Evaluation Techniques’, *Computer and Information Engineering*, 6(8), pp. 957–966.

Boulesteix, A. L. *et al.* (2012) ‘Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics’, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(6), pp. 493–507. doi: 10.1002/widm.1072.

Breiman, L. (2001) ‘Random forests’, *Machine Learning*, 45(1), pp. 5–32. doi: 10.1023/A:1010933404324.

Chen, C., Liaw, A. and Breiman, L. (2004) *Using random forest to learn imbalanced data*, *Department of Statistics, University of Berkely*. doi: ley.edu/sites/default/files/tech-reports/666.pdf.

Chomboon, K. *et al.* (2015) ‘An Empirical Study of Distance Metrics for k-Nearest Neighbor Algorithm’, in *The Proceedings of the 2nd International Conference on Industrial Application Engineering 2015*, pp. 280–285. doi: 10.12792/iciae2015.051.

Crespin, D. J., Abraham, J. M. and Rothman, A. J. (2016) ‘The effect of participation in an incentive-based wellness program on self-reported exercise’, *Preventive Medicine*, 82, pp. 92–98. doi: 10.1016/j.ypmed.2015.11.001.

Cui, P. *et al.* (2016) ‘Uncovering and Predicting Human Behaviors’, *IEEE Intelligent Systems*, 31(2), pp. 77–88. doi: 10.1109/MIS.2016.37.

Cunningham, P. and Delany, S. J. (2007) ‘k-Nearest Neighbour Classifiers’, pp. 1–17.

Dharia, S. *et al.* (2018) ‘Social recommendations for personalized fitness assistance’, *Personal and Ubiquitous Computing*, 22(2), pp. 245–257. doi: 10.1007/s00779-017-1039-8.

Dietterich, T. G. (2007) ‘Ensemble Methods in Machine Learning’, in. doi: 10.1007/3-540-45014-9_1.

Dwivedi, S. and Roshni, V. S. K. (2017) ‘Recommender system for big data in education’, in *2017 5th National Conference on E-Learning & E-Learning Technologies (ELELTECH)*, pp. 1–4. doi: 10.1109/ELELTECH.2017.8074993.

Engelbrecht, A. P. (2007) *Computational Intelligence: An Introduction: Second Edition*, *Computational Intelligence: An Introduction: Second Edition*. doi: 10.1002/9780470512517.

Farrell, R. G. *et al.* (2012) ‘Intrapersonal retrospective recommendation: Lifestyle change recommendations using stable patterns of personal behavior’, in *CEUR Workshop Proceedings*, pp. 24–28. doi: 10.1145/2365952.2366045.

Gibson, T. B. *et al.* (2017) ‘Engagement in health and wellness: An online incentive-based program’, *Preventive Medicine Reports*, 7, pp. 86–90. doi: 10.1016/j.pmedr.2017.05.013.

Gneezy, U., Meier, S. and Rey-Biel, P. (2011) ‘When and Why Incentives (Don’t) Work to Modify Behavior’, *Journal of Economic Perspectives*, 25(4), pp. 191–210. doi: 10.1257/jep.25.4.191.

Goswami, S. and Chakrabarti, A. (2014) ‘Feature Selection: A Practitioner View’, *International Journal of Information Technology and Computer Science*, 6(11), pp. 66–77. doi: 10.5815/ijitcs.2014.11.10.

Gu, Q., Li, Z. and Han, J. (2012) ‘Generalized Fisher Score for Feature Selection’, *arXiv.org*, pp. 1–8.

Guy, I. *et al.* (2009) ‘Personalized recommendation of social software items based on social relations’, in *Proceedings of the third ACM conference on Recommender systems - RecSys ’09*. doi: 10.1145/1639714.1639725.

He, R. and McAuley, J. (2017) ‘Fusing similarity models with markov chains for sparse sequential recommendation’, in *Proceedings - IEEE International Conference on Data Mining, ICDM*, pp. 191–200. doi: 10.1109/ICDM.2016.88.

van Heerden, W. S. (2017) ‘Self-Organizing Feature Maps for Exploratory Data Analysis and Data Mining: A Practical Perspective’, *University of Pretoria, Pretoria*.

Hors-Fraile, S. *et al.* (2018) ‘Analyzing recommender systems for health promotion using a multidisciplinary taxonomy: A scoping review’, *International Journal of Medical Informatics*, 114, pp. 143–155. doi: 10.1016/j.ijmedinf.2017.12.018.

Jiang, S. *et al.* (2016) ‘Personalized Travel Sequence Recommendation on Multi-Source Big Social Media’, *IEEE Transactions on Big Data*, 2(1), pp. 43–56. doi: 10.1109/TBDATA.2016.2541160.

Kesorn, K., Juraphanthong, W. and Salaiwarakul, A. (2017) ‘Personalized Attraction Recommendation System for Tourists Through Check-In Data’, *IEEE Access*, 5, pp. 26703–26721. doi: 10.1109/ACCESS.2017.2778293.

Khatwani, S. and Chandak, M. B. (2017) ‘Building Personalized and Non Personalized recommendation systems’, in *International Conference on Automatic Control and Dynamic Optimization Techniques, ICACDOT 2016*, pp. 623–628. doi: 10.1109/ICACDOT.2016.7877661.

Kotsiantis, S. B. (2013) ‘Decision trees: A recent overview’, *Artificial Intelligence Review*. doi: 10.1007/s10462-011-9272-4.

Kotsiantis, S., Kanellopoulos, D. and Pintelas, P. (2006) ‘Handling imbalanced datasets : A review’, *International Transactions on Computer Science and Engineering*, pp. 25–36. doi: 10.1007/978-0-387-09823-4_45.

Kravitz, R. (2018) *Why is tuning needed in a random forest algorithm?* Available at: https://www.quora.com/Why-is-tuning-needed-in-a-random-forest-algorithm (Accessed: 26 February 2018).

Lani, J. (2018) *Correlation (Pearson, Kendall, Spearman)*, *Complete Dissertation*. Available at: http://www.statisticssolutions.com/correlation-pearson-kendall-spearman/.

Lemaıtre, G., Nogueira, F. and Aridas, C. K. (2017) ‘Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning’, *Journal of Machine Learning Research*, 18(17), pp. 1–5. Available at: http://jmlr.org/papers/v18/16-365.html.

Liaw, A. and Wiener, M. (2002) ‘Classification and Regression by randomForest’, *R news*, 2(December), pp. 18–22. doi: 10.1023/A:1010933404324.

Lim, C. G., Kim, Z. M. and Choi, H. J. (2017) ‘Developing a mobile wellness management system for healthy lifestyle by analyzing daily living activities’, in *Studies in Health Technology and Informatics*, pp. 146–150. doi: 10.3233/978-1-61499-830-3-146.

Manning, C. D., Raghavan, P. and Schütze, H. (2008) *An Introduction to Information Retrieval*. Available at: https://nlp.stanford.edu/IR-book/html/htmledition/feature-selectionchi2-feature-selection-1.html.

Maroulis, S., Boutsis, I. and Kalogeraki, V. (2016) ‘Context-aware point of interest recommendation using tensor factorization’, in *2016 IEEE International Conference on Big Data (Big Data)*, pp. 963–968. doi: 10.1109/BigData.2016.7840694.

Massa, P. and Avesani, P. (2007) ‘Trust-aware recommender systems’, in *Proceedings of the 2007 ACM conference on Recommender systems - RecSys ’07*. doi: 10.1145/1297231.1297235.

Maxwell, F. and Konstan, J. A. (2015) ‘The MovieLens Datasets: History and Context’, *ACM Transactions on Intelligent Systems and Technology (TIST)*, pp. 1–20. doi: 10.1145/2827872.

McComb, S. *et al.* (2016) 'Designing Incentives to Change Behaviors: Examining College Student Intent Toward Healthy Diets', *Western Journal of Nursing Research*, 38(9), pp. 1094–1113. doi: 10.1177/0193945916644705.

Mitra, P., Bokil, H. and Mitra, P. P. (2009) 'Entropy and Mutual Information', in *Observed Brain Dynamics*. doi: 10.1093/acprof:oso/9780195178081.003.0014.

More, A. (2016) 'Survey of resampling techniques for improving classification performance in unbalanced datasets', *arXiv.org*, pp. 878–887. Available at: https://arxiv.org/pdf/1608.06048.pdf.

Ng, A. Y. (2000) 'CS229 Lecture notes', *CS229 Lecture notes*, 1(1), pp. 1–3. doi: 10.1016/j.aca.2011.07.027.

Noble, W. S. (2006) 'What is a support vector machine?', *Nature Biotechnology*. doi: 10.1038/nbt1206-1565.

Ravn, M. O. and Uhlig, H. (2002) 'On adjusting the Hodrick-Prescott filter for the frequency of observations', *Review of Economics and Statistics*, pp. 371–380. doi: 10.1162/003465302317411604.

Ren, Y., Li, G. and Zhou, W. (2015) 'A survey of recommendation techniques based on offline data processing', in *Concurrency Computation*, pp. 3915–3942. doi: 10.1002/cpe.3370.

Resnick, P. *et al.* (1994) 'GroupLens: An Open Architecture for Collaborative Filtering of Netnews', in *Proceedings of the 1994 ACM conference on Computer supported cooperative work - CSCW '94*. doi: 10.1145/192844.192905.

Safavian, S. R. and Landgrebe, D. (1991) '(29) A Survey of Decision Tree Classifier Methodology', *Electrical Engineering*, 21(3), pp. 660–674. doi: 10.1109/21.97458.

Sahu, U. *et al.* (2015) 'Personalized recommendation engine using HADOOP', in *2015 International Conference on Technologies for Sustainable Development (ICTSD)*, pp. 1–6. doi: 10.1109/ICTSD.2015.7095901.

Sarwat, M. *et al.* (2017) 'Database system support for personalized recommendation applications', in *Proceedings - International Conference on Data Engineering*, pp. 1320–1331. doi: 10.1109/ICDE.2017.174.

Shong, N. (2010) ‘Pearson’s versus Spearman’s and Kendall’s correlation coefficients for continuous data’, *Graduate School of Public Health*.

Simović, A. (2018) ‘A Big Data smart library recommender system for an educational institution’, *Library Hi Tech*, pp. 498–523. doi: 10.1108/LHT-06-2017-0131.

Sztyler, T. (2017) ‘Towards real world activity recognition from wearable devices’, in *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 97–98. doi: 10.1109/PERCOMW.2017.7917535.

Tang, X. and Zhou, J. (2013) ‘Dynamic personalized recommendation on sparse data’, *IEEE Transactions on Knowledge and Data Engineering*, 25(12), pp. 2895–2899. doi: 10.1109/TKDE.2012.229.

Teknomo, K. (2017) *Strength and Weakness of K-Nearest Neighbor Algorithm*. Available at: https://people.revoledu.com/kardi/tutorial/KNN/Strength and Weakness.htm.

Tikk, D. (2016) *What is a non-personalized recommender system?* Available at: https://www.quora.com/What-is-a-non-personalized-recommender-system (Accessed: 29 July 2018).

Touw, W. G. *et al.* (2013) ‘Data mining in the life science swith random forest: A walk in the park or lost in the jungle?’, *Briefings in Bioinformatics*, 14(3), pp. 315–326. doi: 10.1093/bib/bbs034.

Trang Tran, T. N. *et al.* (2018) ‘An overview of recommender systems in the healthy food domain’, *Journal of Intelligent Information Systems*, 50(3), pp. 501–526. doi: 10.1007/s10844-017-0469-0.

Utgoff, P. E. (1989) ‘Incremental Induction of Decision Trees’, *Machine Learning*, 4(2), pp. 161–186. doi: 10.1023/A:1022699900025.

Vergara, J. R. and Estevez, P. A. (2014) ‘A review of feature selection methods based on mutual information’, *Springer*, 24, pp. 175–186.

Verma, A. and Ranga, V. (2018) ‘Statistical analysis of CIDDS-001 dataset for Network Intrusion Detection Systems using Distance-based Machine Learning’, in *Procedia Computer Science*. doi: 10.1016/j.procs.2017.12.091.

Westerlund, M. L. (2005) ‘Classification with Kohonen Self-Organizing Maps’, *Soft Computing*, pp. 39–48.

Wójcik, P. I. and Kurdziel, M. (2018) ‘Training neural networks on high-dimensional data using random projection’, *Pattern Analysis and Applications*, pp. 1–11. doi: 10.1007/s10044-018-0697-0.

Yürüten, O. (2017) ‘Recommender Systems for Healthy Behavior Change’, *ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE*, pp. 1–135.

Yuruten, O., Zhang, J. and Pu, P. (2014) ‘Decomposing Activities of Daily Living to Discover Routine Clusters’, *Aaai 2014*, pp. 1348–1354.

Zhou, T. *et al.* (2007) ‘Bipartite network projection and personal recommendation’, *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*. doi: 10.1103/PhysRevE.76.046115.

Appendix A

A.1 Dataset exploration

Table 22 shows the details of each data type within the 13 datasets provided. Within the table, the dataset name, feature name, data type and additional notes are shown.

Table 22: Exploration of fields in the dataset.

Dataset name	Feature name	Type of feature	Data type	Feature category	Note
1_Personal_Data.csv	Cryptoset	Identifier	numeric	categorical	
1_Personal_Data.csv	ClientNo	Identifier	Alphanumeric	categorical	
1_Personal_Data.csv	PolicyFull	Identifier	Alphanumeric	categorical	
1_Personal_Data.csv	Gender	Predictor	character	categorical	
1_Personal_Data.csv	Age	Predictor	numeric	Continuous	
1_Personal_Data.csv	ResAddress3	Predictor	character	categorical	Incomplete column
1_Personal_Data.csv	Role	Predictor	character	categorical	
2_Membership_Information	Cryptoset	Identifier	numeric	categorical	

Dataset name	Feature name	Type of feature	Data type	Feature category	Note
2_Membership_Information	PolicyFull	Identifier	Alphanumeric	categorical	
2_Membership_Information	InceptionDate	Date	numeric	Continuous	
2_Membership_Information	MultiplyMemberStatusDesc	Target	character	categorical	Target Column
2_Membership_Information	FinancialDateKey	Date	numeric	Continuous	
2_Membership_Information	TotPts	Predictor	numeric	Continuous	Incomplete column
3_Healthy_Heart	Cryptoset	Identifier	numeric	categorical	
3_Healthy_Heart	ClientNo	Identifier	Alphanumeric	categorical	
3_Healthy_Heart	EffectiveDateKey	Date	numeric	Continuous	
3_Healthy_Heart	ExpiryDateKey	Date	numeric	Continuous	
3_Healthy_Heart	RagStatusDescriptionDD	predictor	character	categorical	
4_Activity_Information	Cryptoset	Identifier	numeric	categorical	
4_Activity_Information	ClientNo	Identifier	Alphanumeric	categorical	
4_Activity_Information	ScoreValue	predictor	numeric	categorical	
4_Activity_Information	MultiplyActiveDayEventDesc	predictor	character	categorical	
4_Activity_Information	EventValue	predictor	numeric	Continuous	
4_Activity_Information	EventDateKey	Date	numeric	Continuous	
5_Partner_Data	Cryptoset	Identifier	numeric	categorical	
5_Partner_Data	PolicyFull	Identifier	Alphanumeric	categorical	

Dataset name	Feature name	Type of feature	Data type	Feature category	Note
5_Partner_Data	EventDateKey	Date	numeric	Continuous	
5_Partner_Data	EventTimeKey	Time	numeric	Continuous	
5_Partner_Data	NumberOfTicketsDD	Predictor	numeric	Continuous	
5_Partner_Data	MovieTitleDD	Predictor	character	categorical	
5_Partner_Data	CinemaDD	Predictor	character	categorical	
5_Partner_Data_CarRental	Cryptoset	Identifier	numeric	categorical	
5_Partner_Data_CarRental	PolicyFull	Identifier	Alphanumeric	categorical	
5_Partner_Data_CarRental	StartDateKey	Date	numeric	Continuous	
5_Partner_Data_CarRental	EndDateKey	Date	numeric	Continuous	
5_Partner_Data_CarRental	PickUpPoint	Location	character	categorical	
5_Partner_Data_CarRental	DropPoint	Location	character	categorical	
5_Partner_Data_CarRental	SupplierPrice	Price	numeric	Continuous	Zero for all these columns
5_Partner_Data_CarRental	SurplusDeficit	Price	numeric	Continuous	Zero for all these columns
5_Partner_Data_CarRental	Total	Price	numeric	Continuous	Zero for all these columns
5_Partner_Data_CarRental	VAT	Price	numeric	Continuous	Zero for all these columns

Dataset name	Feature name	Type of feature	Data type	Feature category	Note
5_Partner_Data_CarRental	AmountPaidByClient	Price	numeric	Continuous	Zero for all these columns
5_Partner_Data_Netsense	Cryptoset	Identifier	numeric	categorical	
5_Partner_Data_Netsense	PolicyFull	Identifier	Alphanumeric	categorical	
5_Partner_Data_Netsense	EventDateKey	Date	numeric	Continuous	
5_Partner_Data_Netsense	TreatmentAmount	Price	numeric	Continuous	
5_Partner_Data_Netsense	TreatmentDescriptionDD	Description	character	categorical	
5_Partner_Data_onlineShopping	Cryptoset	Identifier	numeric	categorical	
5_Partner_Data_onlineShopping	ClientNo	Identifier	Alphanumeric	categorical	
5_Partner_Data_onlineShopping	ItemPrice	Price	numeric	Continuous	
5_Partner_Data_onlineShopping	ItemDescription	Description	character	categorical	
5_Partner_Data_Dischem	Cryptoset	Identifier	numeric	categorical	
5_Partner_Data_Dischem	PolicyFull	Identifier	Alphanumeric	categorical	
5_Partner_Data_Dischem	TransactionDateKey	Date	numeric	Continuous	
5_Partner_Data_Dischem	StoreNameDD	Name	character	categorical	
5_Partner_Data_Dischem	StoreCityDD	Name	character	categorical	
5_Partner_Data_Dischem	StoreProvinceDD	Name	character	categorical	
5_Partner_Data_Dischem	ItemCategoryDD	Description	numeric	categorical	
5_Partner_Data_Dischem	ItemDD	Description	character	categorical	

Dataset name	Feature name	Type of feature	Data type	Feature category	Note
5_Partner_Data_Dischem	ItemDiscount	Price	numeric	Continuous	
5_Partner_Data_Dischem	ItemPrice	Price	numeric	Continuous	
5_Partner_Data_Dischem	ItemVAT	Price	numeric	Continuous	
5_Partner_Data_Dischem	TotalAllMonthlyBenefitEarned	points	numeric	Continuous	
5_Partner_Data_Dischem	TotalDischemMonthlySpent	Price	numeric	Continuous	
5_Partner_Data_P&P	Cryptoset	Identifier	numeric	categorical	
5_Partner_Data_P&P	PolicyFull	Identifier	Alphanumeric	categorical	
5_Partner_Data_P&P	TransactionDateKey	Date	numeric	Continuous	
5_Partner_Data_P&P	StoreNameDD	Name	character	categorical	
5_Partner_Data_P&P	Points	points	numeric	Continuous	
5_Partner_Data_P&P	SpendAmount	Price	numeric	Continuous	
6_Product_Reward	Cryptoset	Identifier	numeric	categorical	
6_Product_Reward	PolicyFull	Identifier	Alphanumeric	categorical	
6_Product_Reward	SourceProductHouseDescription	Description	character	categorical	
6_Product_Reward	TotalSpent	Price	numeric	Continuous	
7_DrivingData	Cryptoset	Identifier	numeric	categorical	
7_DrivingData	ClientNo	Identifier	Alphanumeric	categorical	
7_DrivingData	PassengerDD	Description	numeric	categorical	0 for all data points

Dataset name	Feature name	Type of feature	Data type	Feature category	Note
7_DrivingData	TripBatchIDDD	TBD	numeric	Continuous	
7_DrivingData	JourneyIDDD	TBD	numeric	Continuous	
7_DrivingData	BatchIDDD	TBD	numeric	Continuous	
7_DrivingData	ShortTripDD	TBD	numeric	categorical	
7_DrivingData	FromDate	Date	numeric	Continuous	
7_DrivingData	ToDate	Date	numeric	Continuous	
7_DrivingData	Distance	Description	numeric	Continuous	
7_DrivingData	Duration	Time(minutes)	numeric	Continuous	
7_DrivingData	Samples	TBD	numeric	Continuous	
7_DrivingData	DistanceScore	Points	numeric	Continuous	
7_DrivingData	AccelerationScore	Points	numeric	Continuous	
7_DrivingData	DecelerationScore	Points	numeric	Continuous	
7_DrivingData	SmoothnessScore	Points	numeric	Continuous	
7_DrivingData	SpeedScore	Points	numeric	Continuous	
7_DrivingData	TimeOfDayScore	Points	numeric	Continuous	
7_DrivingData	RelativeSpeedScore	Points	numeric	Continuous	
7_DrivingData	FamiliarScore	Points	numeric	Continuous	
7_DrivingData	ConfidenceScore	Points	numeric	Continuous	
7_DrivingData	Average	TBD	numeric	Continuous	

Dataset name	Feature name	Type of feature	Data type	Feature category	Note
8_Engagement_Data	Cryptoset	Identifier	numeric	categorical	
8_Engagement_Data	PolicyFull	Identifier	Alphanumeric	categorical	
8_Engagement_Data	InteractionDateKey	Date	numeric	Continuous	
8_Engagement_Data	InteractionSystemDD	Description	character	categorical	
8_Engagement_Data	InteractionTypeDD	Description	character	categorical	

Appendix B

B.1 Feature selected dataset

In Table 23, the chosen features which were output from the feature selection algorithm is shown. The features are arranged in descending order from most important to least important.

Table 23: Features which was found to be of most importance.

No.	Features	Point contributor	Score	Category
1	MultiplyMemberStatusDesc	yes	1	Status
2	TotPts	yes	0.311487	Points
3	GREEN RagStatusDescriptionDD_ RagStatusDescriptionDDexploded.csv	yes	0.258428	Health
4	EventValue	yes	0.180131	Health
5	MOMENTUM HEALTH SourceProductHouseDescription_x_ SourceProductHouseDescription_xexploded.csv	yes	0.143848	Policy
6	InceptionDate	no	0.141793	Date
7	ScoreValue	yes	0.138934	Health
8	ResAddress3	no	0.138634	Personal
9	NO_RAG RagStatusDescriptionDD_ RagStatusDescriptionDDexploded.csv	yes	0.130605	Health
10	MYRIAD SourceProductHouseDescription_x_ SourceProductHouseDescription_xexploded.csv	yes	0.129026	Policy
11	Gender	no	0.121882	Personal
12	step_MultiplyActiveDayEventDescexploded.csv	yes	0.121672	Health
13	Investo SourceProductHouseDescription_x_ SourceProductHouseDescription_xexploded.csv	yes	0.118899	Policy
14	EMAIL_ InteractionTypeDDexploded.csv	no	0.108712	Communication
15	Role	no	0.107539	Communication
16	EB SourceProductHouseDescription_x_ SourceProductHouseDescription_xexploded.csv	yes	0.107451	Policy

No.	Features	Point contributor	Score	Category
17	calorie_activity_MultiplyActiveDayEventDescexploded.csv	yes	0.106191	Health
18	WEALTH SourceProductHouseDescription_x_ SourceProductHouseDescription_xexploded.csv	yes	0.104121	Policy
19	ItemDiscount	no	0.10164	Shopping
20	ItemVAT	no	0.095893	Shopping
21	2_ ItemCategoryDDexploded.csv	no	0.095864	Shopping
22	ItemPrice	no	0.09544	Shopping
23	BPM_ InteractionSystemDDexploded.csv	no	0.09493	Communication
24	TotalAllMonthlyBenefitEarned	no	0.09486	Shopping
25	TotalDischemMonthlySpent	no	0.09466	Shopping
26	medication_ItemDDexploded.csv	no	0.092347	Shopping
27	other_ItemDDexploded.csv	no	0.090434	Shopping
28	detergent_ItemDDexploded.csv	no	0.089313	Shopping
29	1_ ItemCategoryDDexploded.csv	no	0.084408	Shopping
30	PDS SourceProductHouseDescription_x_ SourceProductHouseDescription_xexploded.csv	yes	0.083764	Policy
31	food_ItemDDexploded.csv	no	0.083272	Shopping
32	cosmetics_ItemDDexploded.csv	no	0.081427	Shopping
33	TotalSpent	no	0.078207	Shopping
34	AMBER RagStatusDescriptionDD_ RagStatusDescriptionDDexploded.csv	yes	0.077068	Health
35	AWD_ InteractionSystemDDexploded.csv	no	0.072821	Communication
36	MSTIPersonal SourceProductHouseDescription_x_ SourceProductHouseDescription_xexploded.csv	yes	0.070661	Policy
37	misc_ItemDDexploded.csv	no	0.070226	Shopping
38	Points	yes	0.069142	Health
39	0 RagStatusDescriptionDD_ RagStatusDescriptionDDexploded.csv	yes	0.064058	Health
40	METRET SourceProductHouseDescription_x_ SourceProductHouseDescription_xexploded.csv	yes	0.063246	Policy
41	TELEPHONE_ InteractionTypeDDexploded.csv	no	0.059622	Communication
42	FinancialDateKey	no	0.053047	Date
43	year	no	0.053047	Date

Appendix B: Feature selected dataset

No.	Features	Point contributor	Score	Category
44	SpendAmount	no	0.052544	Shopping
45	GP_ StoreProvinceDDexploded.csv	no	0.052257	Location
46	medical_equip_ItemDDexploded.csv	no	0.050625	Shopping
47	water_ItemDDexploded.csv	no	0.04775	Shopping
48	Gym_activity_MultiplyActiveDayEvent Descexploded.csv	yes	0.040363	Health
49	westernCape_PickUpPointexploded.csv	no	0.035281	Location
50	(no genres listed)_genreexploded.csv	no	0.03504	Entertainment
51	westernCape_DropPointexploded.csv	no	0.034685	Location
52	KZN_ StoreProvinceDDexploded.csv	no	0.03427	Location
53	WC_ StoreProvinceDDexploded.csv	no	0.031095	Location
54	NumberOfTicketsDD	no	0.029841	Entertainment
55	TimeOfDayScore	yes	0.020003	Safety
56	RelativeSpeedScore	yes	0.019983	Safety
57	ConfidenceScore	yes	0.019977	Safety
58	TripBatchIDDD	yes	0.019974	Safety
59	JourneyIDDD	yes	0.019974	Safety

Appendix C

C.1 Classifier training results

The results from the classifier training experiment are shown in Table 24.

Table 24: Complete results from the decision forest algorithm

Model	Filter algorithm	Number of features	Tuning parameters				Performance metrics			
			Minimum number of samples per leaf node	Number of random splits per node	Maximum depth of the decision trees	Number of decision trees	Average accuracy	Average precision	Average recall	F1 score
model 1	baseline	469	5	1012	41	24	0.788834531	0.788834531	0.788834531	0.788834531
model 2	baseline	469	11	539	59	22	0.738786908	0.738786908	0.738786908	0.738786908
model 3	baseline	469	1	390	22	30	0.784375271	0.784375271	0.784375271	0.784375271
model 4	baseline	469	8	868	63	2	0.65990562	0.65990562	0.65990562	0.65990562
model 5	baseline	469	9	83	12	15	0.527188501	0.527188501	0.527188501	0.527188501
model 1	Pearson Correlation	83	5	1012	41	24	0.765499177	0.765499177	0.765499177	0.765499177
model 2	Pearson Correlation	83	11	539	59	22	0.747705429	0.747705429	0.747705429	0.747705429
model 3	Pearson Correlation	83	1	390	22	30	0.775478396	0.775478396	0.775478396	0.775478396

Model	Filter algorithm	Number of features	Tuning parameters				Performance metrics			
			Minimum number of samples per leaf node	Number of random splits per node	Maximum depth of the decision trees	Number of decision trees	Average accuracy	Average precision	Average recall	F1 score
model 4	Pearson Correlation	83	8	868	63	2	0.667590268	0.667590268	0.667590268	0.667590268
model 5	Pearson Correlation	83	9	83	12	15	0.653563079	0.653563079	0.653563079	0.653563079
model 1	Kendall Correlation	86	5	1012	41	24	0.747921898	0.747921898	0.747921898	0.747921898
model 2	Kendall Correlation	86	11	539	59	22	0.724694779	0.724694779	0.724694779	0.724694779
model 3	Kendall Correlation	86	1	390	22	30	0.755411724	0.755411724	0.755411724	0.755411724
model 4	Kendall Correlation	86	8	868	63	2	0.652653909	0.652653909	0.652653909	0.652653909
model 5	Kendall Correlation	86	9	83	12	15	0.627348688	0.627348688	0.627348688	0.627348688
model 1	Spearman Correlation	78	5	1012	41	24	0.747337432	0.747337432	0.747337432	0.747337432
model 2	Spearman Correlation	78	11	539	59	22	0.72510607	0.72510607	0.72510607	0.72510607
model 3	Spearman Correlation	78	1	390	22	30	0.755584899	0.755584899	0.755584899	0.755584899
model 4	Spearman Correlation	78	8	868	63	2	0.647523595	0.647523595	0.647523595	0.647523595

Model	Filter algorithm	Number of features	Tuning parameters				Performance metrics			
			Minimum number of samples per leaf node	Number of random splits per node	Maximum depth of the decision trees	Number of decision trees	Average accuracy	Average precision	Average recall	F1 score
model 5	Spearman Correlation	78	9	83	12	15	0.627565157	0.627565157	0.627565157	0.627565157
model 1	Chi Squared	62	5	1012	41	24	0.774417699	0.774417699	0.774417699	0.774417699
model 2	Chi Squared	62	11	539	59	22	0.722400208	0.722400208	0.722400208	0.722400208
model 3	Chi Squared	62	1	390	22	30	0.772534419	0.772534419	0.772534419	0.772534419
model 4	Chi Squared	62	8	868	63	2	0.643973504	0.643973504	0.643973504	0.643973504
model 5	Chi Squared	62	9	83	12	15	0.494934626	0.494934626	0.494934626	0.494934626
model 1	Fisher Evaluation	36	5	1012	41	24	0.754783964	0.754783964	0.754783964	0.754783964
model 2	Fisher Evaluation	36	11	539	59	22	0.73863538	0.73863538	0.73863538	0.73863538
model 3	Fisher Evaluation	36	1	390	22	30	0.764200364	0.764200364	0.764200364	0.764200364
model 4	Fisher Evaluation	36	8	868	63	2	0.675816088	0.675816088	0.675816088	0.675816088
model 5	Fisher Evaluation	36	9	83	12	15	0.664234999	0.664234999	0.664234999	0.664234999
model 1	Mutual information	58	5	1012	41	24	0.775781453	0.775781453	0.775781453	0.775781453
model 2	Mutual information	58	11	539	59	22	0.724759719	0.724759719	0.724759719	0.724759719
model 3	Mutual information	58	1	390	22	30	0.771343839	0.771343839	0.771343839	0.771343839

Appendix C: Classifier results

Model	Filter algorithm	Number of features	Tuning parameters				Performance metrics			
			Minimum number of samples per leaf node	Number of random splits per node	Maximum depth of the decision trees	Number of decision trees	Average accuracy	Average precision	Average recall	F1 score
model 4	Mutual information	58	8	868	63	2	0.649731578	0.649731578	0.649731578	0.649731578
model 5	Mutual information	58	9	83	12	15	0.489566196	0.489566196	0.489566196	0.489566196
model 1	count based	406	5	1012	41	24	0.798618928	0.798618928	0.798618928	0.798618928
model 2	count based	406	11	539	59	22	0.771084077	0.771084077	0.771084077	0.771084077
model 3	count based	406	1	390	22	30	0.809052732	0.809052732	0.809052732	0.809052732
model 4	count based	406	8	868	63	2	0.706381505	0.706381505	0.706381505	0.706381505
model 5	count based	406	9	83	12	15	0.656593644	0.656593644	0.656593644	0.656593644

C.2 Cross-validation results

Table 25 shows the complete results from the 10-fold cross-validation experiment.

Table 25: Complete result from the cross-validation analysis

	Class 0			Class 1			Class 2			Class 3			Class 4		
Fold number	Average Log Loss	Precision	Recall	Average Log Loss	Precision	Recall	Average Log Loss	Precision	Recall	Average Log Loss	Precision	Recall	Average Log Loss	Precision	Recall
0	0,496	0,782	0,824	0,817	0,695	0,669	0,788	0,772	0,739	0,461	0,852	0,910	0,663	0,834	0,801
1	0,484	0,802	0,834	0,789	0,705	0,696	0,794	0,769	0,736	0,460	0,851	0,913	0,649	0,852	0,806
2	0,500	0,796	0,825	0,788	0,704	0,689	0,809	0,765	0,735	0,457	0,845	0,910	0,643	0,848	0,802
3	0,521	0,801	0,835	0,784	0,713	0,697	0,785	0,775	0,751	0,479	0,845	0,903	0,673	0,847	0,799
4	0,491	0,809	0,830	0,807	0,705	0,694	0,783	0,762	0,744	0,473	0,851	0,903	0,647	0,842	0,802
5	0,507	0,800	0,825	0,808	0,693	0,691	0,831	0,769	0,727	0,454	0,845	0,913	0,653	0,848	0,806
6	0,521	0,799	0,827	0,802	0,703	0,686	0,789	0,770	0,746	0,454	0,849	0,910	0,654	0,845	0,801
7	0,478	0,811	0,838	0,791	0,707	0,702	0,809	0,770	0,722	0,472	0,843	0,904	0,650	0,827	0,800
8	0,481	0,811	0,834	0,799	0,704	0,697	0,807	0,762	0,741	0,469	0,844	0,901	0,681	0,828	0,782
9	0,499	0,802	0,829	0,781	0,700	0,694	0,794	0,767	0,740	0,465	0,855	0,907	0,665	0,840	0,800
Mean	0,498	0,801	0,830	0,797	0,703	0,691	0,799	0,768	0,738	0,464	0,848	0,908	0,658	0,841	0,800
Standard Deviation	0,015	0,009	0,005	0,012	0,006	0,009	0,015	0,004	0,009	0,009	0,004	0,005	0,012	0,009	0,007

Appendix D

D.1 Recommender results

Table 26 shows the overall results of the recommender algorithms. The highlighted columns show the best performing recommender in each experiment category.

Table 26: Complete results from the recommender algorithm

Experiment	Similarity	Recommender type	Recommended action	-4	-3	-2	-1	0	1	2	3	4
Baseline	All	Mean	Complete action	0.00%	0.00%	0.06%	2.56%	11.38%	17.51%	29.91%	17.96%	20.62%
Baseline	All	Mean	Existing action	0.01%	0.18%	0.46%	1.19%	67.46%	27.88%	1.92%	0.79%	0.11%
Baseline	All	Median	Complete action	0.00%	0.03%	0.30%	1.94%	43.25%	42.90%	7.75%	2.94%	0.89%
Baseline	All	Median	Existing action	0.00%	0.15%	0.24%	1.29%	74.29%	22.15%	1.35%	0.46%	0.07%
One	10	Mean	Complete action	0.00%	0.00%	0.46%	3.23%	14.63%	25.80%	29.20%	12.50%	14.17%
One	10	Mean	Existing action	0.00%	0.05%	0.10%	0.33%	81.86%	16.78%	0.66%	0.16%	0.07%
One	10	Median	Complete action	0.00%	0.01%	0.48%	3.24%	20.58%	31.85%	21.67%	11.44%	10.72%

Experiment	Similarity	Recommender type	Recommended action	-4	-3	-2	-1	0	1	2	3	4
One	10	Median	Existing action	0.00%	0.04%	0.08%	0.35%	82.24%	16.36%	0.69%	0.18%	0.06%
Two	30	Mean	Complete action	0.00%	0.00%	0.50%	2.92%	14.01%	25.83%	28.40%	12.51%	15.83%
Two	30	Mean	Existing action	0.00%	0.08%	0.21%	0.63%	77.54%	20.06%	1.10%	0.31%	0.07%
Two	30	Median	Complete action	0.00%	0.02%	0.54%	2.92%	25.34%	36.02%	17.45%	9.48%	8.23%
Two	30	Median	Existing action	0.00%	0.07%	0.17%	0.62%	78.69%	18.97%	1.14%	0.29%	0.06%
Three	50	Mean	Complete action	0.00%	0.00%	0.49%	2.77%	13.53%	25.55%	27.79%	13.10%	16.76%
Three	50	Mean	Existing action	0.00%	0.09%	0.28%	0.77%	75.59%	21.45%	1.35%	0.38%	0.08%
Three	50	Median	Complete action	0.00%	0.02%	0.57%	2.70%	27.65%	37.76%	15.47%	8.50%	7.33%
Three	50	Median	Existing action	0.00%	0.09%	0.21%	0.74%	77.03%	20.11%	1.39%	0.37%	0.06%

Appendix D: Recommender results

Appendix E

Acronyms

MIP	: Multiply incentive program
IRR	: Introspective retrospective recommendation
TLR	: TOMEK link removal
SMOTE	: Synthetic Minority Oversampling Technique
KNN	: K-Nearest Neighbour
SOM	: Self-organising maps
ANN	: Artificial neural network
SVM	: Support vector machine
AMLS	: Azure machine learning studio
MDF	: Multiclass decision forest
TMH	: Tune model hyper-parameters
UOI	: User of interest
IoT	: Internet of things
BMU	: Best matching unit
SN	: Single neuron
FFNN	: Feedforward neural networks

Appendix F

Symbols

I_{xy} : Rating set all users and items in the respective sets

r_{xi} : Rating of user

u_x : Respective user in user set

u_y : Respective user in user set

r_x : Rating of u_x

r_y : Rating of u_y

r_{ai} : Recommendation value

r_{xi} : Recommendations for user u_x

ρ : Normalisation factor

u_a : Respective user in user set

v_i : Content vector

p_a : Preference vector

v_{ik} : Kth element of content vector

p_{ak} : Kth element of preference vector

μ : Overall average rating

$b_i(t)$: Time changing item bias

$b_x(t)$: Time changing user bias

q_i : Item vector within the factor space

y_i : Item vector within the factor space

$R(u_x)$: Rated item set of user u_x

p_x : Variable to capture changes over time

pr_{ui} : Preference score

a_{ui} : Number of times a user performed as specific activity

$n_k(u_x)$: Neighbourhood function of user u_x

k : Number of rated items

N : Dimensionality of a respective item

p_+ : positive values of

p_- : Negative values of

S : Subset of data

S_v : Value of subset S

v : Value of the attribute A

A : Attribute in a table

γ : positive regularization parameter

$\tilde{S}_b$: Between class matrix

$\tilde{S}_t$: Total scatter matrix

$\tilde{\mu}_i$: Mean vector of the reduced space

$\tilde{\mu}$: Overall mean of reduced data

n_i : Size of reduced space

$rank(x_i)$: Rank of x_i in the dataset

$rank(y_i)$: Rank of y_i in the dataset

E_{ij} : Expected theoretical frequency

O_{ij} : Observed frequency

$P(x(i))$: Marginal distributions of random variable $x(i)$

$P(y(i))$: Marginal distributions of random variable $y(i)$