Technische Universiteit **Eindhoven** University of Technology

Where innovation starts

Eindhoven University of Technology

MASTER

Predicting website visitor gender and age with mouse movement data

Hoogervorst, S.J.

Award date:
2016

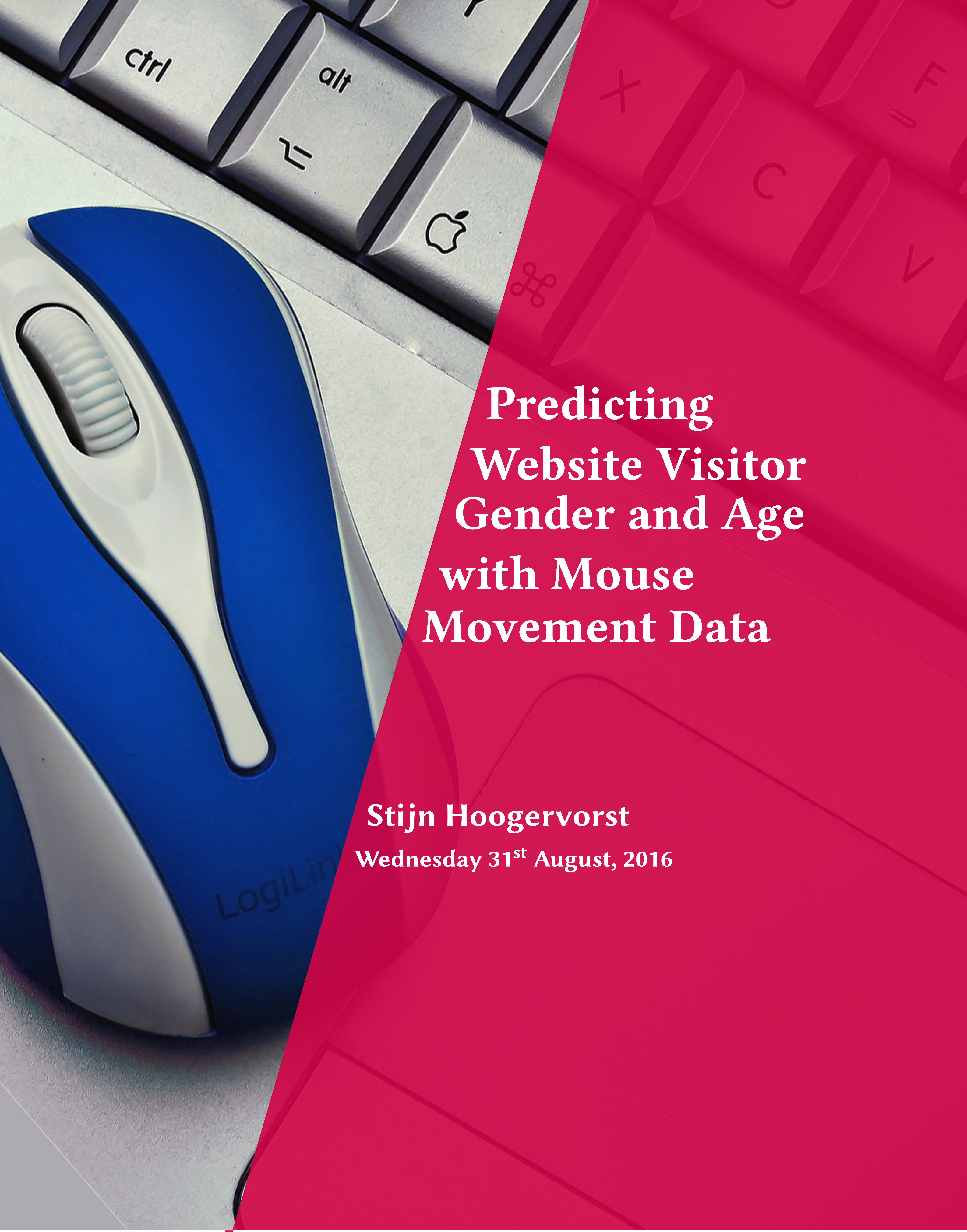
Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 09. Jan. 2018



Predicting Website Visitor Gender and Age with Mouse Movement Data

Stijn Hoogervorst

Wednesday 31st August, 2016

Technische Universiteit **Eindhoven** University of Technology

/ Department of Computer Science

Predicting Website Visitor Gender and Age with Mouse Movement Data

Master of Science Thesis

For obtaining the degree of Master of Science in Business Information Systems
at the department of Computer Science
of Eindhoven University of Technology.

Technische Universiteit **Eindhoven** University of Technology

Stijn Hoogervorst – 0852940
`mail@stijnhoogervorst.nl`

Tuesday 2nd August, 2016

Supervisors:

M. Pechenizkiy	Eindhoven University of Technology
R. van Eijk	Adversitement BV

Thank you to everyone who patiently waited for me to complete this work, to everyone that helped in collecting and processing data, to my supervisors for their support, to my friends and family for their endless questions and to the vast number of online users and communities who's fantastic products and help have been invaluable to this work. A special thanks to Adversitement B.V. for helping me in the realisation of this work and the Eindhoven University of Technology for teaching me the necessary skills to survive in this data-driven world.

Abstract

Online personalization is receiving increased attention from marketeers and many of the larger corporations with a web-presence use knowledge of the current user to offer better search results, advertisements or products. These recommendations are usually done based on extensive historical behavioural data for this specific user, data which is unavailable to the vast majority websites. Marketeers have resorted to using demographic attributes of the visitors to offer some degree of personalization when other information is not available, but even this information is not generally known when users visit a website for the first time. Recent work into behavioural biometrics look at how certain behavioural patterns that do not require domain-dependent information can uniquely identify a user or predict demographic attributes such as age and gender for this user. Previous work has shown that there is some power in cursor motion features to explain differences in gender and age for website visitors. This research aims to see if these generally available features can be used to make precise age and gender predictions for a part of the population that can be used on practically any website. In this work several input features are found based on previous work and analysed. These features are then collected unobtrusively for over 30.000 visitors to several different consenting websites. Several supervised machine learning algorithms are compared and optimized to find good one-class predictive models for gender. For predicting age a regression model is found and several techniques compared for binned classification. The accuracy, precision and recall of the models is evaluated as well as the general applicability of the models.

It is shown that there is some predictive power in mouse dynamic features when it comes to predicting gender and age. Using a Random Forest without normalizing input data was shown to yield the best models, with the model optimized for classifying male visitors being able to attain more than 85% class precision (with roughly balanced male and female instances) regardless of website visited, but only for less than 1% of all male visitors. The female model showed similar results for the training set but achieved a maximum of 60% precision when applied to instances from different websites, suggesting that this model is less universally applicable. Binned age prediction performs marginally better than guessing, which indicates that here too the chosen input features have some predictive power, achieving accuracies just under 40% instead of the 33% a guessing algorithm would achieve.

This research has confirmed that there is some merit in looking at mouse movement behavioural biometric features to derive features for the current visitor if no other information is available, but that the current state of the art is not precise and not generalizable enough to be useful in practical scenarios.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Background

Adaptive systems aim to offer the right content in the right way to a specific user and offer different information or a different interface depending on the situation. A definition given by Langley is *(emphasis added for this paper)*:

> An adaptive user interface is a software artifact that improves its ability to interact with a user by *constructing a user model* based on partial experience with that user. [26]

Recommender agents are an example of a way to implement such a 'personalized' web experience [17]. This agent recommends certain products, e.g. physical products in a webshop, articles, pages, friends, games, restaurants, *etc.* There are two main methods for offering such personalized content, with the system performing the recommendations being called the *recommender system*. One class of methods is known as *memory based* in which all previously observed data is used to make a recommendation. On the other hand there are *model based* agents that give recommendations based on a constructed model of the user [54]. At the core of this latter category of agents and adaptive applications lies a model of the current user, usually under the assumption that users with a similar model exhibit similar (and thus predictable) behaviour. Forming a model of the user while he/she is interacting with a web page is increasingly common for any adaptive system. Many of these models may keep track of behaviour specific to a website or interests (such as *likes* for Facebook or products for a web-shop) and many of these fall into the category of *domain-dependent* user models [9] whilst more general models capture demographic attributes such as age and gender, which are frequently classified as *stereotype based user models*. Stereotype-based user models offer a good start for an adaptive system although a more advanced model offers additional advantages as domain-dependent data is frequently more representative of a person's preferences than models based on demographic information [40][25]. Recommender systems based on these demographic user models are called *demographic recommender systems* and work by sorting users into stereotypical classes [17]. Many websites that require registering ask some basic demographic information from their users [24]. There is still a lot of merit in using them as the initial basis for web personalization, especially in situations where additional information is not provided by the user (also given that users may lie when knowingly providing information). [1] Furthermore, research based on demographic differences is abundantly present (the differences between men and women have been studied for a long time as well as the effects of ageing).

De Bock and Van den Poel identify several reasons for using demographic information to target advertisements to users [13]. Almost a third of respondents to an American Advertising Federation 2006 survey preferred their socio-demographic attributes to be the qualifying factor for targeting (even though most consumers still preferred being targeted by behaviour such as visited pages and products). De Bock and Van den Poel also build the case that many advertisers and companies define their products and services in terms of an intended audience, frequently based on a (fictional) user profile containing demographic information. Jansen et al. give an overview of previously conducted research and the effects of gender on the influence of advertisements [21]. A recent study by Sharma and Batra finds that age and gender are factors that lead to significant differences in online shopping behaviour [43]. This indicates that there are differences in how men and women are influenced by advertisements which suggests that taking different approaches in targeting them (by adapting to demographic attributes) could be beneficial to the business. Their analysis goes on to note that gender differences in online shopping have yielded inconsistent results across studies, which indicates that there is reason for more research into this area. They suggest that the integration of the internet in our daily lives has reduced gender differences, but these findings seem contradictory to the work of Sharma and Batra. Their own study finds that advertisements targeted at women or men specifically do not outperform gender-neutral advertisements. It is noteworthy that their class labels were obtained based on the gender score given to a specific search query, which may have introduced a lot of noise to the data. Targeting ads to women is suggested to be more effective than to men.

Recently, more work has been done on identifying users based on the behaviour they exhibit during browsing. The way we type can accurately identify us [32] (with results showing similar predictive properties of users' keystrokes on recognizing emotional state [15] and gender [47]). Not only the way we write, but also what we write can give us reliable indications of our age and gender. Schler et al. show that when trained and applied on texts found on blog posts, they can correctly predict the gender of the author with an accuracy between 70 and 80 % [42]. There are also indications that users interact with HTML elements differently based on gender [51]. Many different types of demographic inference techniques look at browsing behaviour whilst some others focus on linguistic differences associated with the different demographics. An overview of different experiments conducted that research these differences is given by Hsu et al [19]. Other techniques are trained by looking at specific products and pages visited, or by analysing search queries [3]. A large fraction of the last group of methods has as a downside that they require a large corpus of information, usually in the form of a high-dimensional index of web pages constructed by first solving a classification problem on the website content [45].

[1] It should be noted that some companies are moving away from demographic indicators in favour of user (domain specific) behaviour [39]. It speaks for itself that this is only possible if this domain specific data is available

Several methods have been proposed to overcome the difficulties with these usually sparsely populated vectors for user behaviour, such as SVD (singular value decomposition) [33] but they are not aimed at solving the source of the problem, which is the creation and maintenance of a large (and ever expanding) index of web pages.

Hu et al. have improved on predicting age and gender for website visitors by supervised learning and comparing such a large corpus of web-pages. They have done so without first classifying the content of the web-pages, but just looking at the browsing history of different users [20]. Unfortunately, to most website owners, browsing history and other visited pages is not information which is readily accessible. On the other hand, there are interesting developments in determining characteristics based on what is referred to as 'real-time' or 'dynamic' data. This is data that becomes available throughout a session with an application. Different methods exist for prediction which all have their roots in statistical models and are collectively frequently referred to as 'machine learning' techniques because they use observed instances to predict new instances.

An upcoming field of study that aims to improve authenticating that the current user is still who he or she claimed to be when logging in is investigating techniques for *continuous authentication*. This field is looking for unique characteristics in how applications are being used, which can preferably be collected unobtrusively. Since almost all applications that most users come into contact with on a day-to-day basis involve the use of pointing device such as a computer mouse, several papers have been published that look at how *mouse dynamics* can be used as a *behavioural biometric* attribute of a user [22]. These researchers use variation in cursor trajectory features to train their algorithms to identify users with varying degrees of success in different environments. As more research into this field emerges it is becoming clear that behavioural data can help to reach conclusions that existing methods may not yet allow.

Based on these findings there is reason to assume that users with different demographic attributes will interact differently on a website with regards to interaction data such as mouse motion, clicks and key input. Previous research has shown there to be statistical differences when performing the same controlled tasks, which begs the question if these results can be replicated outside of the experimental context.

These differences probably rose from the fact that part of the experimental group showed stereotypical behaviour for that demographic attribute, which does not mean to say that e.g. all male users performed better than all female users. In fact, if just a small fraction of the population demonstrated this stereotypical behaviour it could have still led to significant statistical differences in the distributions for the two genders. The expectation is that for visitors to a 'real' website this stereotypical behaviour that led to the differences will still be present for a similar fraction of the population. This means that if these differences are present in real systems they will be observable in only a part of our population. By increasing the sample size and by taking a more representative portion of the population, variation in users' features should be distinguishable for roughly similar percentages. Because the nature of the tasks executed will not be as uniform as in earlier experiments and other dependent variables will be hard to control, a small decrease in accuracy is to be expected. The most interesting application is the *accurate* identification of users *if possible*, *i.e.* precise user demographic classification. This implies that algorithms focusing mostly on that part of the population that demonstrates this aforementioned stereotypical behaviour will be of the most use in constructing user models. A model that is very accurate for a part of the population is more useful than one that performs marginally better than guessing for the general populace.

1.2 Motivation

Segmentation for marketing purposes on gender is perhaps the most common and frequently researched demographic attribute, backed by publications from the fields of psychology and marketing [16]. Pooler suggests that marketeers use consumer gender and age to create different segments for retail advertising purposes [38]. In his paper on 'Marketing and the Demographic Perspective', Pol focuses specifically on giving background perspective on age distributions and changes in the US populace and labour force over different years [37]. According to Dibb and Symkin characteristics like age, sex and personality are frequently used by businesses targeting consumers [14].

Other demographic attributes have been proposed, albeit that less research and attention has been paid to these features than to gender-related strategies and impact [16] [38]. A list of demographic variables developed to segment customers of a financial service provider is presented in the works of Clancy and Lou Roberts [10] and names amongst others Income, Occupation, Nationality, Education level, Geographical location, Marital Status, Religious and Political views. Although these attributes may prove to be interesting with regards to targeting purposes, they are harder to ascertain from normal browsing behaviour and will frequently require a form of survey or sample testing to get labels for visitors.

Although there have previously been critical statements regarding the use of demographics as a basis for segmentation or the benefits to marketing to segments as a whole [49], these arguments were quickly rebuffed [6]. Given the prevalence of segmentation as a marketing strategy and the fact that much-used analytics packages such as Google Analytics [2], Adobe Analytics [3] and IBM Analytics [4] all offer large degrees of insight into customer segments it is reasonable to state that, regardless of effectiveness of targeting and segmentation, there is a large market for it and technologies supporting it.

Age and gender are the most common used demographics for market segmentation and targeting. Because these attributes have been shown to be widely useful for many different kind of businesses and are additionally frequently requested from users during normal interaction (for example on any form where users are asked to fill in their preferred title or date of birth). This implies that these demographic attributes will also be the easiest to obtain from natural browsing behaviour. For these reasons, the focus of this study will be directed towards predicting these demographic attributes.

Other research has under experimental conditions shown that there is some explanatory power in mouse movement features to explain differences related to age and gender [51][2]. Since mouse movement features are nearly always available on any website, if these can be used to derive information about the visitor in a non-obtrusive manner, this opens up personalization

[2]https://analytics.google.com/
[3]http://www.adobe.com/marketing-cloud/web-analytics.html
[4]http://www.ibm.com/analytics/us/en/

options to websites that normally do not have access to attributes of their visitors. For business purposes, coming to a model that finds these demographics based on cursor features is the most desirable outcome, as it will allow for personalization and targeting where this might normally not be feasible. Machine learning techniques can be applied to this domain to derive a user model when trained on these differing features of interaction data. If a user model based on this interaction data can accurately be constructed, it would make indexing and classifying large numbers of internet pages for user profiling purposes superfluous and offer a good basis for stereotypical user profiling for websites. It would allow business owners who do not have access to extensive search and browsing history or user registration to offer personalization to visitors based on observed behaviour, even if their site has little other input information to classify users on. The motivation of this research is to determine how accurately demographic information for a user can be determined based on interaction tasks and to develop an application that will allow website owners to retrieve a user profile consisting of such demographic data for their current visitor in a timely manner.

In conclusions, many of the larger corporations with a large web-presence already incorporate a degree of personalization (one can think of targeted advertisements, product recommendations, holiday destinations, *etc...*). There is an increasing demand for offering personalized content on the web and underlying user models enable website owners to present this content to their visitors. For many websites however, domain-specific information is not readily available and the construction of user models infeasible at the current levels of maturity. Given the increased focus on offering personalized content to users, the usefulness that demographic user models can offer when no other characteristics are available and the information that can apparently be obtained from data-sources such as interaction and dynamics data which are always available on websites, this work focuses on studying if a demographic user model can be obtained from generally available mouse dynamic features collected during normal browsing behaviour.

1.3 Research Objective

In previous work it was shown that mouse dynamics and cursor trajectory features show promising results in identifying individual users and that there are indications that there are differences in these features for users with different gender and age demographics under controlled environmental conditions.

Because practical uses of finding user demographics for recommender agents will ultimately preferably work under a wider range of conditions and especially under varying environments seen in practice, this research will focus on making predictions based on attributes derived during normal browsing behaviour. Any conclusions should therefore be applicable to a wide range of websites, as input data will be available for any regular website visit. The main objective of this research is to answer the following question:

Is it possible to predict gender and age of website visitors using features of cursor movements during normal interaction?

1.4 Expectations

Earlier work has shown that under controlled experimental conditions, statistical gender differences existed in mouse motion features [5] collected for participants. Predicting for gender in these works showed overall accuracies of around 60% for a relatively small dataset. It is reasonable to assume that when replicating this study outside of the experimental conditions it was performed in and other environmental influences start factoring in, this percentage will drop. The accuracies reported in these works are overall measures for performance of the classifiers and aim at finding group-level distributions. It is likely that two one-class models will have better precision scores for their classes, but are only able to precisely identify a small part of the population due to gender and age-based differences not being present in all of the website visitors. In fact, stereotypical behaviour is likely only demonstrated by a small subset of the populace. For any practical application, a high degree of precision for a small part of the population is preferable to a globally 'better-than-guessing' predictor. This leads to a preference for training a unique classifier per attribute value for the categorical demographic attributes. This affects gender prediction where two classifiers will be sought that are trained to be precise in predicting for their respective positive classes rather than on achieving the highest overall accuracy. Under the assumption that interaction with a website involves mostly point-and-click tasks and that these tasks are similar in nature to those performed in experiments listed in earlier work under experimental settings, it is expected that gender differences will be present. Overall accuracies will be lower than 60%, but classifiers specific to a class will be able to attain higher precision scores for their classes for parts of the population. It is unknown at this point what part of the population would show this stereotypical behaviour.

Given this expected pattern in the input data, the classifiers for each individual gender are expected to have a relatively high confidence for the part of the population that shows this behaviour. As we investigate the effect of the threshold, setting a high confidence threshold should correctly identify these cases, reflected by the initial peak we see in the expected ROC curve in Figure 1.1. After this segment of the population is identified and the threshold for classification relaxed a little, the expected True Positive Rate will grow proportionally to the False Positive Rate, which is why the expected ROC curve flattens after that point.

For predicting age, users are expected to have age-related mouse motion attributes somewhere around their actual age average. When binning users into age groups, it is logical to expect more overlap especially around the edges of the bins. The previous described method of unique classifiers for different bins is not equally applicable as each classifier will most likely only have high confidence values for users that fall in the middle of their age band and not for users that are only just in this bin (for example, a user of age 29 would not fall in the 30-60 age bin, although attributes are likely similar to users falling into the same bin). Instead, the approach will be to see if regression models and random forests lead to good prediction results. Since distinctions here will likely prove more difficult, it is expected to see accuracies only a little upwards from guessing, between 33 and 50%. Since this will only show that there is predictive power in the input features, additional work will likely be needed before any useful application can be derived.

[5] Mouse motion features are assumed to be discern

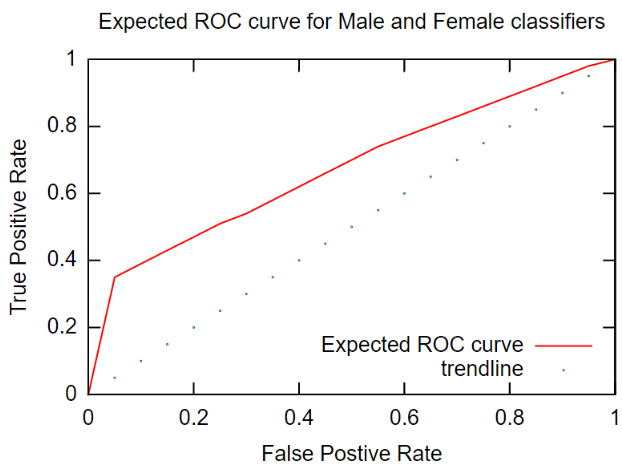


Figure 1.1: The expected shape of the Receiver Operator Characteristic curve that corresponds to the classifiers for gender that this research expects to find

Regression models are expected to yield better results, if relationships exist between several of the input features and age. Since performance in previous work was linked to age of the users, it is expected that the regression result will give relatively good indications of the age of a user. What may make calculations less accurate is that there is no external pressure to perform and users are not performing the exact same task. This could reduce the effect of age on performance-related input features such as movement time or overshoot.

1.5 Outline

To achieve the research objective the rest of this report is structured as follows:

1. Distil Research Questions from the research objective
2. Hypothesize answers
3. Outline Framework to test hypotheses
4. Perform analysis
5. Accept or refute hypotheses

1.6 Results

This research shows that there is some predictive power in mouse dynamic features collected during normal browsing interaction when classifying both user gender and age, but precision and recall are too low for many business applications as the cost of misclassification are assumed to quickly outweigh the benefits of personalization. Gender prediction is only very precise for less than 4% of all website visitors. The found models can be said to be fairly generally applicable after testing their performance on a dataset obtained from an independent website, although performance is shown to be slightly worse. Predicting age showed similar results that were better than guessing but not in any manner enabling business use cases. Regression results showed a RMSE of around 15 years, which is a fairly large error bound and predictions tended towards the input data average.

Chapter 2

Research Framework

The Research Framework further structures the steps in this research. The research objective is broken down into specific research questions, for which rough expectations are formulated based on earlier work and expectations. The subsequent steps to test the questions are outlined in this chapter and performed in the following chapters.

2.1 Research Questions

In order to answer the research objective, the following questions will have to be addressed:

Q1 What features of cursor movement are relevant for predicting age and gender?

Q2 How can Machine Learning be applied to the input features to answer the objective?

Q3 How precise and for what part of the population will prediction be feasible?

Q4 How generalizable are the results?

2.2 Expectations

Features that have explanatory power in cursor movement differences are likely a good place to start for predicting to which demographic a website visitor may belong. Since there is not a lot of existing research into these features, additional features that could be of interest are the ones used in identifying individual users during continuous authentication. It is hypothesised that these features will have predictive power with regards to gender and age.

The objective is to answer if prediction on the basis of these mouse movement features of normal website visits is feasible. From a labelled dataset with these features and the age and gender labels, supervised machine learning techniques can be applied and evaluated. The choice of technique, implementation and evaluation will depend on aspects of the input data such as noise and quantity, but it is likely that there is not necessarily one separating hyperplane that perfectly splits the input data. Earlier work into demographic differences show that accurate classifications for the whole dataset were not possible. It is plausible that this is a consequence of stereotypical behaviour only showing up in a small fraction of the population (where men behave 'manly' for example). For this part of the population, better than guessing accuracies should be obtained, possibly slightly upwards from accuracies seen for the entire population in earlier research. Since there may be combinations of features that together could be considered as stereotypical for a specific class, it is less likely that a single separating hyperplane will accurately classify users into their correct demographic group. The best performance likely comes from classifiers that are more robust to noise and find clusters, such as nearest neighbour or random forest approaches. As the most probable business cases have a need for precise classification, for the binary class distinction, it will be better to have two one-class classifiers that are precise for their respective classes than a single classifier that will need to have a good overall precision.

Previous research has frequently binned users in age groups, a tactic that will be deployed here as well. Most differences are likely perceivable in more distinct groups, therefore separating users into *young*, *middle-aged* and *elderly* makes sense. Since these groups are expected to have more or less similar behavioural features, predicting to which of these groups a user belongs should be more accurate than guessing. From this, the expectation **E3** is derived. Due to the nature of binning, it is expected that mostly users that fall close the middle values of the bin will be correctly classified. For this reason a regression model will also be trained to predict age of the user.

Because the features are derived from normal browsing behaviour and under the assumption that most website visitors behave roughly the same on different websites, it is likely that any found results will be applicable to any website for which cursor movement features can be collected. It is not assumed that this will also hold for web-applications that have different user interfaces than standard HTML websites (such as Google Maps for example).

E1 Mouse Dynamic features have predictive power with regards to gender and age

E2 Accuracy of models trained on observed *male* or *female* cases will score class precisions of upwards of 0.65

- **a)** Each of these models will only predict their positive class for a small proportion of the population

E3 Predicting to which of 3 evenly spaced age groups the user belongs will be possible with an accuracy of more than 0.33.

E4 Found classifiers will be applicable to most standard websites

2.3 Methodology

The Research Framework is meant to structure the approach to achieving the Research Goal and answering the questions that it raises. Overall, the following issues will be addressed:

1. What data will be needed to answer the research questions?

2. How will this data be obtained?
3. What analysis will be done on the data?
4. What results will allow us to answer the research questions?

To define relevant input, previous work using mouse movement features will be analysed to identify possible features of interest. From this a list of features will be established based on observations from earlier studies. These features, if discernible, will be extracted from logs of mouse motions made available from an earlier experiment in a 'Proof of Concept' to see if stereotypical behaviour is present in this subset of users under experimental conditions. This dataset was collected under a controlled environment and comprises a relatively small dataset with all participants having similar demographic attributes except for gender. This implies that conclusions based off this dataset will likely only be indicative. This Proof of Concept will shed further light on the features that will have to be collected and the quantity of observations needed in the actual experiment, conducted under normal browsing behaviour. To test what features yield good results, predictors for the demographic attributes will be trained and evaluated.

Translating the problem into one solvable in the Machine Learning space will occur based on experience with existing methods. Data will have to be collected from which the defined input features can be derived and this data collection ideally occurs unobtrusively during normal interaction with a website for users of differing gender and age. Methods of preprocessing the data, optimization, classification and evaluation will depend on the quantity and quality of the collected data. The prediction algorithms themselves will be identified from a literature study and past experiences. Their effectiveness will be further answered by comparing the performance of the different chosen algorithms on the dataset and taking the best models or methods. Because the desired output is precise classification, for the different genders, two one-class models will be trained and evaluated. The precision in classifying the corresponding class is ultimately the most interesting evaluation metric, which will be picked based on the best precision scores for varying threshold levels.

The class precision metric will clarify how precise age or gender can be predicted for that specific class. The part of the population it is applicable to will be reflected by the recall metric. Since no claims are made regarding what combination of precision and recall constitutes a 'best' solution, which is heavily dependent on the specific business case of the found models and the (likely varying) costs of misclassification to the use case at hand, a sensitivity analysis will be done. Because varying thresholds can be chosen, it is of interest to compare Precision-Recall curves to visualize the trade-offs between these two. It will allow answering for what part of the population precise classification is possible. These curves will be obtained by applying the model on a testing set and varying the threshold levels.

In order to answer the generalizability of the results, it is important to compare the same input features obtained from different sources, i.e. websites. The most accurate models found for one dataset can then be applied to the other dataset(s) to see if similar precision and recall scores are obtained. This would suggest that the results obtained will be generally applicable to all websites for which these features can be observed. This research will therefore involve data collection from more than 1 source and compare the results.

In conclusion, the steps that will be taken are:

1. Define input features
   (a) Derive from related work
   (b) Test features in 'Proof of Concept'
   (c) Conclude which input features are relevant
2. Perform analysis
   (a) Collect data from multiple sources
   (b) Define classification techniques
   (c) Process data
   (d) Train classifiers
   (e) Report results
3. Answer research questions

Chapter 3
Input Features

From related work, a list of relevant cursor motion features as input data will be derived. These features will be tested using data made available from a different earlier experiment conducted at the TU/e [5]. This anonymized dataset containing mouse positions, age and gender (as well as other features specific to the other experiment) for roughly 80 participants was created by having the participants perform high-level tasks on a website (such as browse products) on the same experimental machine and website. This dataset will be used to train a supervised learning algorithm to differentiate on gender. The reason to only attempt to predict gender is that the age range in the dataset is not very diverse (from 21 to 25). The mouse tracking software also registered keyboard presses, which will also be used in the feature extraction phase. The data consisted of session logs of varying lengths (between under 1 and over 10 minutes). Two different sets of input data are compared for their performance to decide the type of features which will be collected in the next chapter on data collection for the experiment looking at real-life website interaction data.

3.1 Related Work in Cursor Motion Features

Although the focus of the investigations into continuous authentication using mouse dynamics was on individual user identification, the mouse features used as input to these classification tasks can be a source of information for finding cursor motion attributes with explanatory power in gender differences too. The underlying assumption is that if individual users vary enough to allow authentication of the current user, users with similar demographic attributes will vary to a lesser extent but still vary somewhat. Distinction is made between different types of tasks, such as *point-and-click* and *movement* operations [52][44]. These experiments are performed in a controlled environment and collect features related to speed, accuracy, angle of movement, acceleration and time it takes to perform a click (or a double click). Based on these attributes, False Acceptance Rate (tested user incorrectly assumed to be the same user as the one the model was trained for) and False Rejection Rate (actual user not recognised) were both below 10%. These make these features interesting candidates for research, as the variation seen in individual users can depend on certain demographic attributes of the website visitors.

The work by De Bock and Van Den Poel sets out to create a demographic profile based on click-stream data [13]. Although their method attains accuracies up to 69% when predicting gender of website visitors, their input data requires information on sessions from several different tracked web-pages. This information may be available to large corporations who have their code present on a multitude of sites (such as Adwords, Facebook integration, Twitter share button, *etc.*) which could then use it to construct a demographic user profile [1]. However, for the owner of the website, such detailed logs are mostly unavailable, making the owners dependent on third parties for more information regarding their website visitors or necessitating them to rely on different methods to construct a user profile.

Recent work by De Andres et al. [2] shows that it is possible to infer demographic attributes for profiling web users based on interaction with a website. Through multivariate regression analysis on experimentally obtained GOMS (goals, operators, methods and selection rules) data they were able to differentiate between gender and age for certain interaction tasks. GOMS analysis is based on the assumption that the users are familiar with the type of task under observation and measures the performance in executing these tasks. The performance measure of choice is frequently the time needed to perform an action. GOMS splits up complex tasks in basic operations. The experiment was performed on 592 participants gathered online, meaning users were using their own systems (meaning their own pointing devices, monitors and browsers). Each user was presented with the same set of tasks, involving 5 different types of actions: selecting words, pointing to and clicking on elements, dragging elements, editing text and selecting an option from a menu. The times needed to perform the tasks were recorded and users were required to fill in a survey prior to the experiment in which they had to give information about their demographic attributes. They conclude that there are significant differences in relation to age and gender. Performance is decreasing with age, whilst for gender their results show that men generally perform better on point-and-click, drag-and-drop and menu selection tasks. Another noteworthy point is that most of the research subjects were from the same age group, suggesting that perhaps statistical differences in gender-based performance were age-dependent, meaning that they only apply to the age group studied. Further research is warranted to see if these variations persist outside of laboratory conditions and if the findings are applicable to the general population.

Their work suggests that a classification system using web interaction data as input could distinguish the gender and age of the user while they are interacting with a web page and offer personalized content on the basis of this model. GOMS features will therefore be investigated as input features and compared to mouse movement attributes.

In a study conducted by Yamauchi et al. cursor trajectories of 372 participants were analysed to predict gender, emotional state and the film clip that participants were shown before the mouse interaction [50]. These participants were asked to fill in surveys, watch a clip and perform several clicking tasks choosing from two images on screen each time on the same experimental set-up. Cursor trajectories were recorded every 20 milliseconds. Using dimensionality reduction through features selection, they utilized random forests on 159 input cursor motion features to predict gender correctly in 61% of cases. Future work from this experiment includes checking if

[1]Provided one is not already known due to the user is already registered to one of these services

the cursor motion features on which predictions are based are generalizable to web interaction tasks or specific to the research task. Although the task performed in this experiment can be similar in nature to tasks performed on a webpage, this requires further research. They also note the limitations of the chosen features when applied to devices with different user interfaces (such as touchdevices) for which other features would have to be investigated.

Different features are described in the works of Yamauchi et al., Pentel, Kaminsky et al. and Muthumari et al. from which feasible features will be used and tested[50][36] [23][34]. Based on these papers and the Proof of Concept, the full list of features that will be used in the actual experimental section of this research will be derived.

3.2 Testing Input Features in Exploratory Proof of Concept

From the mouse motion dataset a list of features will be derived. Initially, the claims made by De Andres et al. will be tested using a simple variation of their GOMS framework. This measures the time it took to complete specific actions. Because the raw data involves cursor positions from up to 10 minutes from general browsing behaviour and not from specific actions, features will be extracted from mouse motion behaviour during a specific interval. For example, the input data is split in periods of 1 minute and for each 1 minute session features are calculated. This work investigates if a decent prediction can be made on basis of these properties.

The next step is to see if the other features that led to promising results in continuous authentication and in the works of Yamauchi et al. [50][51] will be be good input features for our classification algorithms.

3.2.1 Description of POC Dataset

The data was stored a separate plain text file for each of 81 participants, with two participants having their data distributed over 2 files. Each line in these files contained information on the interaction data at a specific time interval. It could for example give the coordinates of the cursor, or which key was pressed or released. Since only the raw actions are recorded, there is no way of knowing what kind of element the cursor was hovering over at the time of clicking. Participants were told the study was a usability study of the website they were visiting. Users were asked to complete 5 search tasks on the test website for a maximum of 2 minutes per search task (10 minutes total) and two 'free surfing' tasks in which they were free to browse the website as they pleased for 5 minutes each (for a total of maximally 20 minutes).

3.2.2 Feature Extraction from POC Dataset

The GOMS features described by De Andres et al. were based on 5 different tasks. These could be divided into several categories: Point-And-Click tasks, Drag-And-Drop tasks, Text Editing, Text Selection and Menu Selection. Of these tasks, it was not feasible to see when a menu item was selected or when a drag and drop action actually involved text selection. The other tasks were calculated by looking at the interval between mouse down and mouse up, as well as the position to determine if the mouse was dragged for a period time. Text editing is also included, where the time it takes to type words is used. All these calculations were bound by certain thresholds and intervals, the values for which were determined heuristically and are shown in the source code, shown in Appendix A. Given that normal drag and drop actions are not very common during normal browsing behaviour, it is possible that these were actually mostly text selection tasks.

For the second part features similar to the ones used in the works of Yamauchi et al., Pentel and Kaminsky et al. were derived[50][36], [23][34]. This leads to the list of mouse dynamic and keystroke input features shown in Table 3.1 that can be calculated without additional contextual information, which is not present in the dataset.

Table 3.1: List of features extracted for the Proof of Concept

speed : the distance between two measurements
x velocity : the speed in the horizontal direction
y velocity : the speed in the vertical direction
x acceleration : the change in speed in the horizontal direction
y acceleration : the change in speed in the vertical direction
tangential acceleration : the overall change in speed between two measurements
path length : the distance travelled between two clicks
left click interval : time between clicks with left mouse button
right click interval: time between clicks with right mouse button
length of left clicks : time between mouse down and mouse-up event of left mouse button
length of right clicks : time between mouse down and mouse-up event of right mouse button
time between scrolling : time between two scroll events
precision : actual distance between two clicked points divided by path length
key-press length : time between key-down and key-up events
key-press interval: time between two key-presses
direction : the number of times moved in one of 8 wind directions
total left clicks
total right clicks
total key-presses
idle time : time spent not moving the cursor
total left double clicks

When looking at how these attributes are distributed for the different subjects, it can be seen that the means, standard deviations, minima and maxima roughly follow normal distributions. An example of this can be seen in Figure 3.1.

These features were calculated by dividing the mouse data in sessions of 1 minute, possibly creating multiple data entries for each participant. Sessions shorter than 1 minute were discarded. For each of the attributes that are not totals (such as the total left clicks) the minimum, maximum, mean and standard deviation was calculated to describe the feature. This gives a total feature space of 73 features. A Python script was used to calculate these features from the files, the full

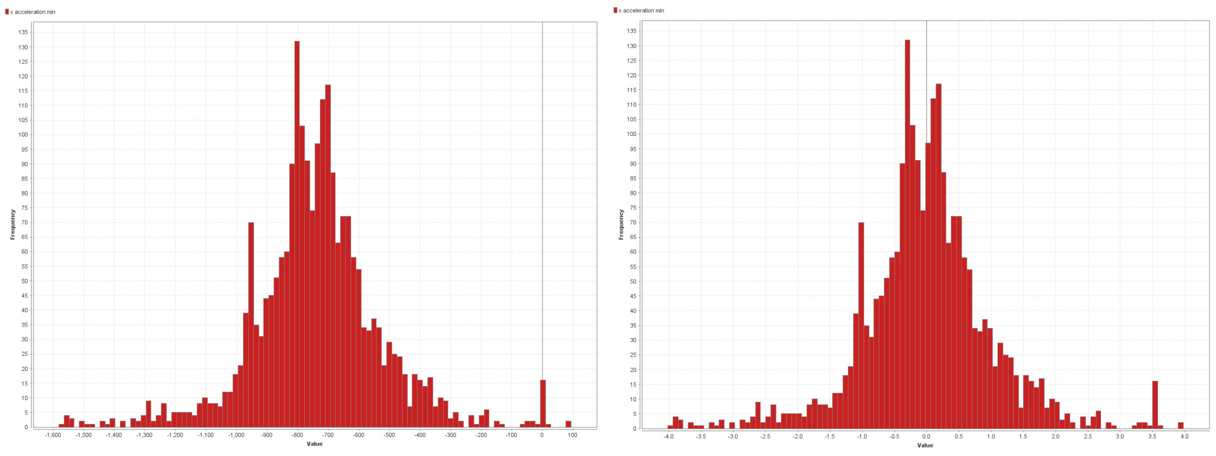


Figure 3.1: left: the distribution of the minimum acceleration in the x-axis for the different sessions, right: the same distribution after z-score transformation

script can be found in Appendix A.1.

3.2.3 Classification of Gender

Before training the classifiers, the input data was normalized so that all values fall roughly on the same scale. Because the data follows normal distributions, Z Score Transformation was used to scale the data. This method is not as sensitive to large outliers (when compared to for example Min-Max normalization), but the resulting range is a little larger than -1 to 1 (although almost 70% of observed cases will fall in that range). When applied to the distribution shown in Figure 3.1 the normalized distribution seen on the right-hand side of the figure is obtained.

In order to optimize classification results, two SVM's are trained with Radial Basis Function Kernels, one SVM trained for precisely classifying males and one for females. Training occurs by normalizing the data and splitting it into training and testing data. To prevent over-fitting for observed subjects, sessions originating from the same participant are all in the training or all in the testing set. The parameters ν and γ () are optimized for precision by using a grid-search for the optimum combination of these parameter values, tested on a 10 fold cross validation of the SVM classifiers. This process is used with both the different input features spaces, once with the GOMS and once with the other features.

A graphical overview of the flow is presented in Appendix C.

3.2.4 Results of POC

Results of GOMS Features

The results of performing classifications on the GOMS features can be seen in Tables 3.2 and 3.3. It can be seen that the Area Under the ROC curve (AUC value) is just below 0.5, indicating that the model found is actually slightly worse than guessing. Precision for male users is a little better than guessing.

Table 3.2: Confusion Matrix, Precision and Recall of Female SVM classifier trained on GOMS input features from Proof of Concept dataset

Model	poc_goms_svm_female		
AUC	0.495		
	True Female	True Male	Class Precision
Pred Female	246	308	44.40%
Pred Male	175	202	53.58%
Class Recall	58.43%	39.61%	

Table 3.3: Confusion Matrix, Precision and Recall of Male SVM classifier trained on GOMS input features from Proof of Concept dataset

Model	poc_goms_svm_male		
AUC	0.495		
	True Female	True Male	Class Precision
Pred Female	246	308	44.40%
Pred Male	175	202	53.58%
Class Recall	58.43%	39.61%	

These results show that accuracy is low for predicting gender. Looking at the ROC curve for these classifiers, shown in Figure 3.2, it can be seen that this classifier is marginally better than a guessing classifier (for which it would be expected to see roughly $x = y$). Compared to the ROC curve expected to see it is performing about as good as guessing. When doing the same computations but with different threshold values, the accuracy of the classifiers increases hardly, even at lower recall values.

Results of Mouse Dynamic and Keystroke Features

The confusion matrices for the male and female classifiers are shown in tables 3.4 and 3.5. Overall accuracy is a little better than 50%, but no good suitable hyperplane to predict the gender of the user is found. The AUC for both the models found is a little better than 0.5, suggesting that there is more information in these features than in the GOMS features.

The ROC plots for the two classifiers are shown in Figure 3.3. Especially the classifier trained to be precise for classifying female users seems to show some desired properties, whilst the male classifier performs closer to a guessing classifier.

These results indicate that it is not possible to differentiate between male and female internet users, based on the features described in 3.2.2 using Supervised Learning Mechanisms in a

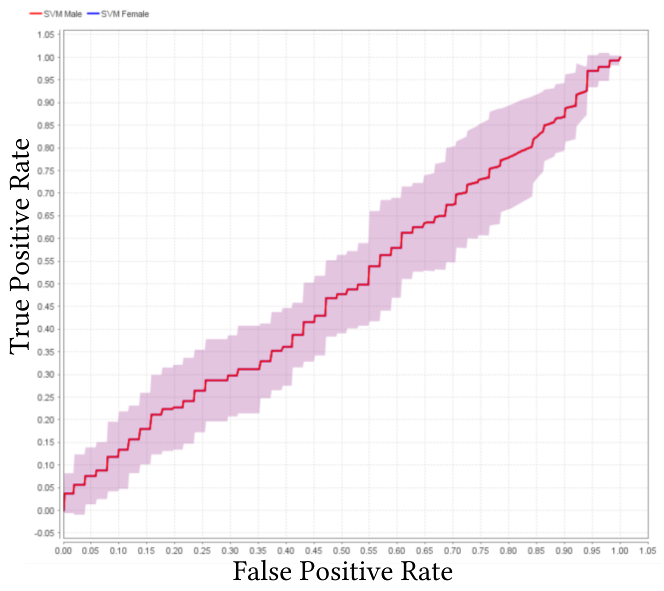


Figure 3.2: The Receiver Operator Characteristic curve for the SVM classifiers trained on the GOMS input features

Table 3.4: Confusion Matrix, Precision and Recall of Female SVM classifier trained on Mouse Dynamic and keystroke Features input data from Proof of Concept dataset

Model	poc_features_svm_female		
AUC	0.538		
	True Female	True Male	Class Precision
Pred Female	187	186	50.13%
Pred Male	134	185	57.99%
Class Recall	58.26%	49.87%	

Table 3.5: Confusion Matrix, Precision and Recall of Male SVM classifier trained on Mouse Dynamic and keystroke Features input data from Proof of Concept dataset

Model	poc_features_svm_male		
AUC	0.511		
	True Female	True Male	Class Precision
Pred Female	185	207	47.19%
Pred Male	136	164	54.67%
Class Recall	57.63%	44.20%	

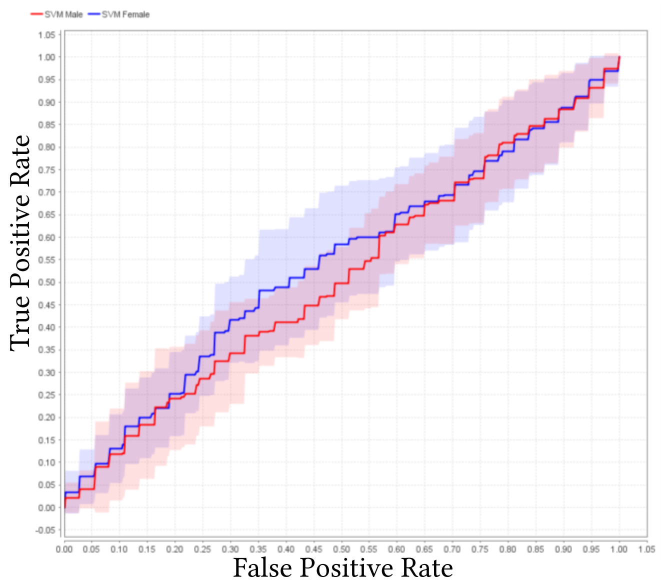


Figure 3.3: The Receiver Operator Characteristic curve for the SVM classifiers trained on the other input features derived from the POC data

manner significantly better than guessing, also not for small parts of the population. This seems to suggest that there is either no information in these features or more contextual information may be required. It should not be disregarded that both sets of input features showed some promise with regard to correctly classifying a small part of the population for which stereotypical behaviour could be present, especially the female model trained on cursor movement and keystroke features performed a little better than guessing which suggests that these features can be useful in predicting gender. The phenomenon observed was not as significant as sketched out in the expected ROC curve but still warrants further research.

Moving forward, the GOMS features show less promise than the mouse dynamic and keystroke features. To answer the research questions, more mouse dynamic features will be collected for the dataset looking at normal browsing behaviour. Although Class precision and Recall are given for the default threshold found by the classifiers, the Area Under the ROC curve gives an indication of overall performance of the classifier for different threshold levels. This is a better indication of classifier performance with regards to varying business applications.

3.2.5 Discussion of POC

From the initial results it seems that merely observing user behaviour when performing tasks in an experimental setting was not sufficient to make accurate or precise predictions pertaining to the gender of the user. It is also very conceivable that this difference was not observed due to the relatively small sample size. With 37 female participants, the algorithms were trained on only 26 women in the training set, which may not have been a large enough sample size to find stereotypical behaviour (which is possibly only present in a subset of subjects). Furthermore, all participants to the experiment had a similar background and age. The lack of diversity could also help to explain why no separating hyperplane was found. It is still possible that the same set of features with a larger dataset would have given different results as there will have been a higher absolute number of users for which their behaviour could be described as typically one way or the other.

In earlier works statistical differences were shown based on logs of mouse motion data of very specific tasks. Differences in cursor behaviour when executing the same lower level task (such as pointing and clicking the same element) could be more indicative than when observing normal web browsing behaviour. In a similar fashion, authenticating individual users using mouse motion or key presses usually requires the test subject to perform the exact same task (such as typing the same sentence as it was trained on or tracing the exact same figure).

Neither the GOMS features nor the other derived features could be used to accurately infer demographic attributes. The two main factors at play could have been the lack of similarity in the tasks performed (meaning that users were not required to click the same button from the same starting position, but rather the task itself was to browse naturally with environmental factors controlled) and the small sample size that was used. The next step is to collect more data (with more demographic attributes) to find if there is a group of users with mouse dynamic behavioural patterns for which predictions are feasible and to answer the question pertaining to predictive power of mouse movement features with regards to user age.

The GOMS features show less promise than the Mouse Dynamic features. This can be explained by the fact that GOMS features compare features related to the performance of specific tasks. It would appear that clicking of buttons on a website is not a atomic 'task' and different buttons and clicks are not directly comparable. It is possible that by looking at individual button clicks these GOMS features become more usable for predictive purposes.

Initial results indicate that, although insufficient for classification purposes, there is reason to believe that there is some explanatory power in mouse dynamics for gender distinction. More contextual information, using aspects of the button for example, and adding additional features is a logical next step. The number of unique test subjects was also on the low end for finding distinctions in stereotypical gender motor control and interaction strategies. The GOMS features are relatively limited in their application (since drag and drop operations were found to be infrequent) and text editing tasks are more intensive to log client-side.

3.3 Features Selected for Experiment

Behavioural differences between men and women are a common subject for studies from especially the psychology field, as well as in Human Computer Interaction [11][27][31]. In Business Information Systems, Information Retrieval and Adaptive Hypermedia more emphasis is placed on using web data to create accurate user models [17][8]. By looking at social media data, such as from Facebook or Twitter, many predictive models have been constructed with varying degrees of success [18]. Other researchers have focused more on search history and other contextual information, the practical applications of which seem apparent given the research is conducted by employees from the corporate world and patents are filed using these technologies [48][53].

In experiments involving Point and Click tasks with a mouse (or other pointing device) where users were encouraged to perform the task as fast as possible, women perform statistically slower than men [41][51][2] although an experiment involving a different pointing device found no such difference based on gender [19]. Women are more precise when clicking on targets (closer to the centre of the target) [41], although the work of Yamauchi et al. showed that women deviate more from the straight path towards the target[51]. There is no apparent difference in response time to a task for men and women [51][41]. Men tend to decelerate their cursors later than women [41]. Additionally movement time and static hand tremor are significantly different when distinguishing users in 3 age groups [19] and is shown to have negative effects on other performance measures [2]. The opposite was observed for average hours of internet usage per day.

Features pertaining to speed were 7 out of the 10 most distinctive features for predicting the emotional state of computer users and correspondingly factors that involved time explained the largest variance in the data [50]. Research involving what target a user intends to click on showed significant predictive capabilities based on the distance to the target, velocity, acceleration and angle [4].

In normal browsing behaviour, the 'task' users face is not necessarily the same for all users. Even if the same button is being clicked, starting positions of the cursor may vary as may the moment the user starts having the intention to click. Although the time to complete a task was an important basis for the research by De Andres et al. [2], it may be less relevant when the tasks at hand are not as controlled as in their experimental setting. Because some of the other metrics may be defined in terms of time, the total movement time is an important factor so that for the other features the *relative* time can be calculated (which may be more indicative than absolute times).

Since men tend to decelerate later than women, the moment peak velocity was reached must be found as well as the time it takes to complete the action after peak velocity was reached. The scalar quantity of peak speed is also included as an input feature.

Women are apparently more precise [2] than men when it comes to pointing and clicking tasks. Therefore *precision* will be calculated (based on the maximum distance from the centre) and an additional performance metric common in the HCI field will be calculated. This is Fitt's Index of Performance, which gives a scalar value of how well an element was clicked based on the movement time and the difficulty of the task (expressed in the distance to the button and the width). The distance from the starting point is used for this calculation and for the calculation of how closely the shortest path was followed. The calculation for Performance Index is shown in equations 3.1 and 3.2. The total distance moved is interesting to compare performance to actual moved pixels. This gives an indication of accuracy (how well the shortest path was followed by the user), but for good measure this statistic will also be included by client-side calculation.

$$\text{ID (index of difficulty)} = \log_2 \frac{2 \cdot \text{Distance to Target}}{min(\text{Goal Width, Goal Height})} \tag{3.1}$$

[2]the term *precision* is used differently in many different contexts. In this case, precision refers to the distance from the centre of the target

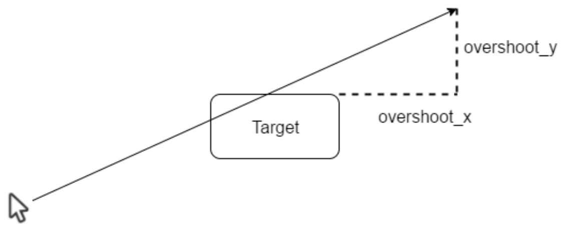


Figure 3.4: An example of how overshoot was calculated in the x and y directions. The arrow represents a hypothetical cursor trajectory

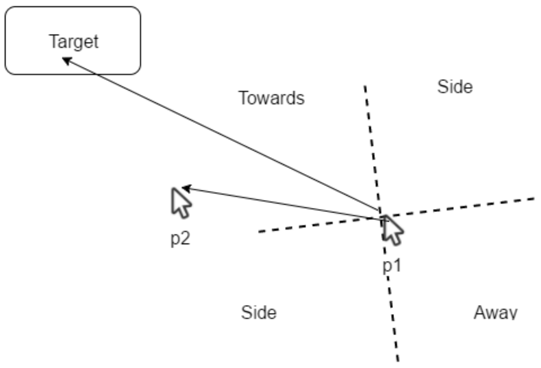


Figure 3.5: An example of how time in a certain direction was measured. *p1* is any point in time and *p2* is the position of the cursor at the next measured interval. The arrows show the direction from *p1* to the goal and the path taken from *p1* to *p2*. In this example the interval between *p1* and *p2* counts towards the time in the *towards* direction

$$\text{Performance Index} = \frac{\text{ID}}{\text{Movement Time}} \tag{3.2}$$

Other performance metrics that are considered are derived from the work of Yamauchi et al. [50][51] and involve the time moving in a particular direction (towards the goal, perpendicular to it or away from it). This feature was calculated by looking at the measurement prior and taking a $\frac{\pi}{2}$ angle as towards. In the example shown in figure 3.5, the position at *p2* was a movement globally towards the goal from point *p1*, so that time is added to the time moved in the *towards* direction. Supplementary performance measures include how far a goal was missed by the user in the x and y directions as overshoot (by comparing the path to the starting location and button). An example of how this is calculated is shown in figure 3.4. Finally in this category, the time spent with hovering over the button may be indicative of a level of decisiveness and is an additional feature under consideration.

The last group of features is related to speed and direction and measures the average speed, the deviation in the speeds measured at the intervals for this user, the distribution of the accelerations and the distribution of the angles between the measurements.

3.3.1 Changes in Features Based on POC

From the Proof of Concept it appeared that looking at the distributions of several promising features may not sufficient input data (or not enough test cases) to train classifiers that can identify visitor demographic attributes. In order to maximize the chances for finding patterns in practical scenarios additional features (and of course the labels) will be included and collected from a larger number of visitors during normal interaction with a website. Since it is shown that people behave differently when they know they are participating in an experiment, a phenomenon commonly known as the Hawthorne effect or the Observer effect, any distinguishing behaviour resulting from this fact will limit the utility a tool can have when applied to a regular website void of these experimental influences. For this reason, the choice was made to collect normal browsing behaviour of users consenting to general tracking which means they were unaware mouse features were being collected. The list of additional features can be seen in Table 3.6.

3.3.2 Features Excluded

Although earlier experiments suggest to use the time needed to perform different type of tasks such as Drag and Drop tasks [2], it turned out to be difficult to distinguish text selection tasks from drag and drop tasks in normal browsing scenario's. Regular point and click behaviour is more prevalent than the other interaction types. Furthermore, since the nature of these tasks can vary greatly (which text will be selected for example) it cannot be expected that the time to perform these tasks yields much information. Since there is no clear-cut starting moment for each task, reaction time is not a factor that can be measured.

Because of prerequisites set by the partner sites where the data collection framework would be hosted, some other features that would require more intensive client side operations have been excluded to minimize possible performance impact at the end users. Amongst these excluded features were key-presses as measuring these would mean performing calculations every time any text is edited anywhere and require more transfers between client and server. Specifically, if the same number of attributes was to be collected as during the Proof of Concept it would imply having to hijack all key press and key release events and storing data either locally and detecting whether navigation requests were made by the client's browser to first transmit data or have data being continuously sent after every key-up event. This was undesirable from both a performance aspect as well as for the customers. For similar reasons, mouse down and mouse up events were not handled in the same way as in the proof of concept stage.

Table 3.6: List of features that will be collected on each mouse click based on literature and the findings in the POC

id
gender
age
total movement time
peak velocity
precision (how close to the centre of the button the user clicked)
time after peak velocity
distance from starting point to goal
total distance cursor moved
performance index [a]
time on button
path accuracy
average speed (overall movement time divided by distance)
velocity at click
overshoot x-axis
overshoot y-axis
time moving in direction towards goal
time moving in a direction perpendicular to goal
time moving in direction away from goal
speed mean (the mean of the speeds calculated between every position 20 ms)
speed standard deviation
acceleration mean
acceleration standard deviation
angle (angle between mouse position every 20ms) mean
angle standard deviation

[a]Fitt's performance index was used, but instead of using the width of the target the smaller of the width and height, which was shown to yield comparable results for rectangular targets and makes the performance index applicable in 2-Dimensional context [29]

Chapter 4
Method

This chapter explains the method used to collect the features derived in Chapter 3. It aims to clarify which technologies were used and how these features were extracted from different websites. This chapter will answer how Machine Learning will be applied to the collected data, the preprocessing steps which are taken and how the classifiers were optimized and trained. Good parameters for various different classifiers are searched for by parameter optimization. Classification results, Receiver Operator Characteristic Curves are plotted and Precision-Recall curves for the best gender classification model shown and applied to the different datasets to test the generalizability of the found models. Plots of the regression results and metrics are found and presented and binned classification results presented.

4.1 Technologies Used

There are a combination of technologies used to collect, store, process and analyse the data.

DimML code is also used to process the incoming features, add additional features to the data such as the labels and the number of passengers, and stores the data in text files on the research server. To reduce the number of server calls and concurrent collection, the DimML platform collects data in buckets and aggregates write operations.

Preprocessing of the data is done using scripts written in Python [1]. Python scripts are used because they are easy to use, have extensive libraries available for us that are simple to import and help with processing files and calculations. Although it is commonly accepted that Python scripts are not the fastest way to process large volumes of data and that they tend to take up larger amounts of memories, this preprocessing step takes place only once per customer and speed and efficiency are not a prerequisite here.

The normalization, filtering of data, addition of attributes, discovery and optimization of models will initially be done in RapidMiner Community edition [2]. RapidMiner offers a comprehensive user interface to expose underlying algorithms that facilitates with setting up optimization and parameters search. Although it is not as (easily) extended as using libraries

[1] http://www.python.org
[2] https://rapidminer.com/

for existing programming languages, the benefits in the discovery phase are significant. The extra computational costs incurred do not outweigh the increased efficiency in manipulating the models and visualizing the results.

The RapidMiner platform does not lend itself well as an independent block in a chain of operations for the real life demo. Overhead is too big to run the entire platform every time a click with feature data comes in. Since the DimML platform itself is written in Java, if suitable models are found, they will be implemented in Java and integrate them in the DimML environment. DimML is already in place for these customers to collect their data and to send the mouse click data. Instead of storing the feature data to a server, the DimML script will run the data through the models found earlier. The outcome of the predictions will be used to place a User Model object on the page that the website owner can use for personalization or targeting. This Model will be updated when an interaction takes place.

4.2 Data Collection

4.2.1 Websites Where Collection Will Take Place

With the cooperation of two customers of Adversitement, data was collected from consenting users whilst they were browsing one of these client websites. No cross-domain tracking was performed, so if the same user visited multiple of these sites a new instance was observed. During interaction with the website, the customer will at some point specify the demographic attributes that will be used to train the algorithm. This was a requirement for client-website selection.

The benefit to collecting data from live actual websites is that it explores whether the results of earlier research can be replicated in actual normal websites and will allow a convenient integration with the business case that the owners of these websites have for using demographic information of non-registered visitors for personalizing content and targeting.

The collection framework will be placed on the websites of a Dutch airline [3] and health insurance sites.

Requirements from Website Owners

Because data collection takes place on a live website, the owners of the websites where the data collection framework will run have a set of demands that the experiment will not adversely affect active customers and visitors. At the same time, data collection can not violate local and European laws regarding data collection and storage.

Loading of the scripts that will calculate the features and send them to our server should not affect website load-time. Normal browsing behaviour should not be impacted by the script running in the background and any delays in interaction should not be considered hindering to the overall user experience.

From the fields of Human Computer Interaction and User Experience it is demonstrated that delays under 150 milliseconds do not hinder users [46]. The aim is to have almost all delays

[3] names of websites have been redacted

caused by the collection of data to be under 150 ms.

No personal data that could lead to identifiable individuals can be collected, data collection can only take place if consent for cookies and usage of data has been given as per European law. Data must be transmitted securely and stored in a location complying with European norms for storage of customer data.

Lastly, visitors to the mobile website are not a part of this experiment as website owners have concerns that mobile phones generally are less suited for client-side calculations and have slower internet connections. For this reason, no scripts should be loaded and no data collection should take place for the mobile sites or apps.

4.2.2 Assumptions Regarding Data Quality

It is common practice in web analytics to assume that not all obtained data is accurate and that results must always be used indicatively. Users can spoof information, have different locale settings, possibly use older or different engines interpreting the code, have extensions or plugins that alter the layout of the site or data sent and received, etc... Because of this, the possibility that certain behaviour seen in the data was not actually exhibited by the visitor can never be dismissed. This means that some levels of noise in the collected data are to be expected. The assumption is that the amount of traffic that behaves ‘normally’ will outweigh the number of data points that contain wrong information. Since a certain degree of noise will inevitably be present, techniques must be applied later on to minimize the effect this noise may have on our algorithms.

In both the websites for the Airline and Health Insurers, website visitors that move through the funnel of purchasing tickets or switching health provider have to fill in a date of birth and gender. This information can not be verified, so it is conceivable that customers fill in false information in this field. Although reasonably most real customers will fill this in truthfully to the best of their ability, potential customers that are just ‘browsing’ products and want to see an offer on a next page may pick an arbitrary birth date or gender. Since no information will be stored that makes the visitors identifiable in a different (back-end) data repository, there is no way of knowing if the information provided is accurate. To mitigate the effect of some users that may exhibit this ‘browsing’ behaviour early during a session but come back to ultimately purchase a ticket or switch provider using their real information, only the last known label for the users will be used.

Earlier insights into visitor data from the Health Insurance companies has shown that with the exception of IZZ (a health insurer aimed mostly at professionals in the health-care industry), most customers are male. On top of that, there are currently only a limited number of reasons why you can switch health care provider. This means that all customers fall into one of these categories.

You can switch provider in the Netherlands only if one or more of these conditions apply:

- you just turned 18
- the policy terms have changed
- you moved here from abroad

- you are retiring from military service and you need new insurance
- you are getting a divorce
- the collective insurance you are a part of is terminated or you are no longer part of the collective (such as when insured through employer)
- you are currently uninsured

Since mostly (young) adults that switch jobs or change marital status will change insurance, this group is expected to be more represented in the data. It is also possible for 1 person, the *main insurer*[4], to take out insurance for multiple people within the family. In this case, only the gender and age of the main insurer will be stored, under the assumption that this is also the person currently visiting the website.

In a similar fashion, it is also possible to book multiple tickets. In this case however, there is no specific main booker available to us. The assumption that the person who is booking the tickets fills out their own information first is less strong than with the health insurance sites. Because booking of flights is not limited to constraints and in continuous demand, there is a lot more traffic to the airline website and a lot more purchases. For this reason it is acceptable to disregard visitors that purchase multiple tickets and leave enough data for subsequent analyses. The assumption for single ticket purchases is that they are frequently purchased by the people that will be using them. Unfortunately this might still include ticket purchases done by third parties (employees booking for colleagues for example). Since business-related travel accounts for only roughly 30% of air travel in the United States, this number is taken as an upper bound for the number of visitors to the airline website booking for business purposes [35]. Since business related travel is frequently more luxurious and this specific airline is a low-cost airline, it is plausible that the actual percentage is in fact lower. On top of that, in small companies that may be more budget-sensitive, employees may be expected to acquire their own tickets and get refunded by their company later, incentivizing them to look for lower-cost air travel. This implies that the number of single tickets booked by people with different features than the passenger info suggest will therefore be strictly less than 30%. It should be noted though, that as far as noise levels go, it is not unreasonable to expect as much as up to 20% of data-points to be incorrectly labelled. This signals that algorithms that are robust to noise will be best suited for predictive purposes.

With subsequent steps in mind, the attribute *number of passengers* was also collected for visitors to the airline website so that multiple ticket purchases can be disregarded later (as well as click data from users for which no labels were ultimately sent to the server).

4.2.3 Overview of Data Collection method

Our experimental set-up will first involve a data collection step where features of cursor motion from many different visitors from several websites within the limits set by the customers (outlined in Section 4.2.1) will be amassed. A graphical depiction of this process is shown in Figure 4.1.

[4]in Dutch: hoofdverzekerde

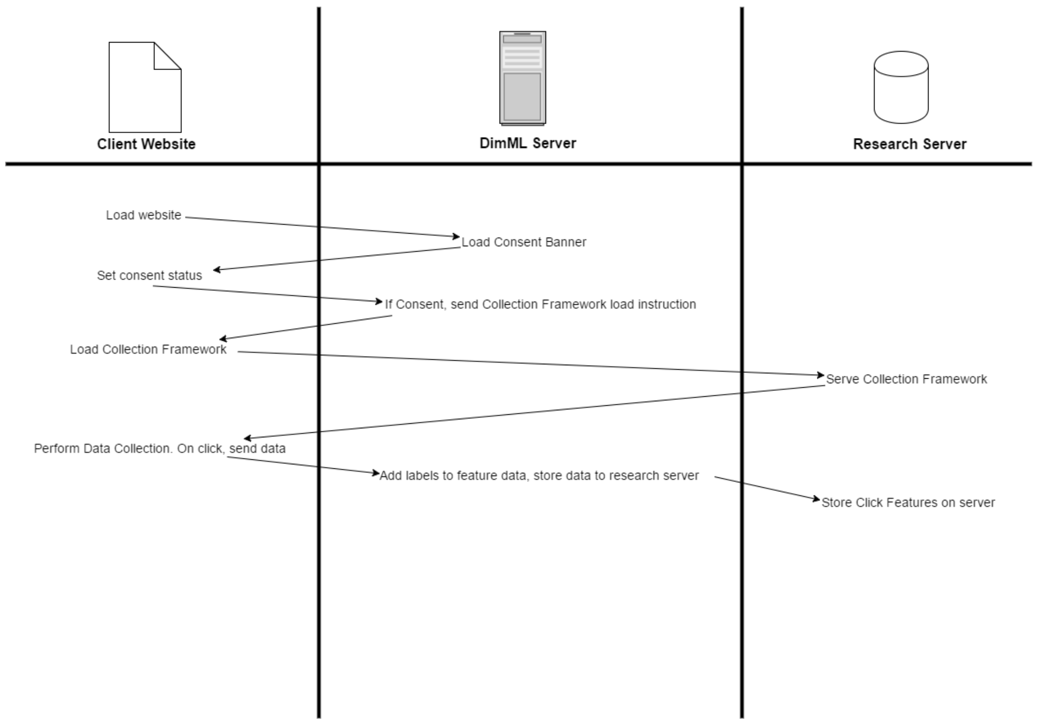

Client Website
DimML Server
Research Server
Load website
Load Consent Banner
Set consent status
If Consent, send Collection Framework load instruction
Load Collection Framework
Serve Collection Framework
Perform Data Collection. On click, send data
Add labels to feature data, store data to research server
Store Click Features on server

Figure 4.1: An overview of how the process flow of data collection for this experiment was set up

4.2.4 Client-Side Data Collection Framework

Under the assumption that users once they have the intention to click a button do this in the fastest way possible, matching the set-up described in the experiments by Yamauchi et al. [50][51] certain elements on the web page will be marked (elements are selected using jQuery selectors such as `$('.button')`) and when one of these elements is clicked a series of calculations takes place in the user's browser to compute our features.

The features are collected using JavaScript run on the users browser during the website visit. The framework is hosted on a server set-up solely for this experiment. The instruction to load the framework is performed by a DimML [5] script. DimML stands for *Declarative in motion Machine Language* and is a propriety technology by O2MC [6] (part of the Adversitement group). It is a coding framework that allows collection and processing of data on a client site. It has its own syntax and allows for both client and server-side manipulation and collection of data.

The full source code of the collection framework can be seen in Appendix A.2.

Because of the size of the logs which increases rapidly, even for shorter session times, it was infeasible to store all mouse positions for each user (which would allow us to calculate additional features retroactively). Storing this information would greatly increase the storage requirements of the server and have too much negative impact on delays occurring the moment an element is clicked (as even asynchronous calls will have to block navigating away from the page until confirmation that all data is sent is received). Since network calls are usually slower than local processing, to minimize impact to live environments the choice was made to only calculate features client side and store only these features for each click.

The framework works by collecting the cursor position every 20ms. Additionally, indicated elements on the page are 'marked' and a new click handler is added that takes place *before* the normal action is executed. When one of these elements is clicked, it starts a loop over the cursor position data and calculates the features of interest based on which button was clicked, up to a point defined as *the start of the point and click action.* Identifying the moment when a user *intends to click* on an element proved not easily solvable. Preliminary research has not shown when this moment occurs exactly. Although this could be a topic in and of itself that merits further research, since no suitable starting point based on literature could be found, a backwards heuristic approach was taken to find a point that was good enough to function as a starting point: as soon as a user has clicked a button, the framework looks at the data collected between the click and the last moment the cursor was (near) stationary. This moment is assumed to be the last time when users started the movement towards the button they wanted to click. This implies that users who, for example, use their cursor to 'read along' when going through text bodies are seen to have much larger movement times and likely lower accuracies.

Using this heuristic method it was found that if the cursor is moving less than 100 pixels per second for a period of 300ms (except when the cursor is over the button that is going to be clicked) it is practically stationary. This was taken as the starting point of the action.

The way the precision feature is defined in our framework is shown in equation 4.1. This method works best for symmetrical elements (where the middle of the element is also the middle

[5] http://www.dimml.io
[6] http://www.o2mc.io

of the visual element, something that does not happen when a larger margin is added to one side). Due to the way certain browsers handle focus of the window, it is possible for users to click buttons without having cursor positions available. In this case, many of the fields will be ∞ or otherwise incorrect. Due to the way precision is calculated, it is possible to get lower than zero values for this feature. This happens for example if due to styling or client side button interactions a button click occurs even though the cursor is not over the button in question.

$$1 - \frac{\text{distance from centre}}{\text{half the hypotenuse of width and height}} \tag{4.1}$$

A rough indication of the network speed of the visitor is also performed every now and then to determine if the framework should send code. On slow connections, to limit usability impact, no data is transmitted.

Several additional features were stored and collected. The type of element that was clicked and the path of this element on the page (in combination with the URL) can be used to identify a unique button on the web-page. This information could theoretically be used to train individual classifiers per button. The position and dimensions of the button are also stored so that features can be added to the dataset later. Although a test is performed to load the framework only on desktop clients, it remains possible (although unlikely) that a mobile device was used. There are also laptops and desktops that feature touch screens. Since many of the features identified in this work will not be applicable if the user clicked the button directly (without having to move the cursor), these wrong entries will have to be excluded from the rest of our analysis. For similar reasons, users who make use of a touchpad rather than a mouse as a pointing device will have differing features and will be filtered out. The framework used an adapted version of a Stackoverflow answer[7] to detect whether a touchpad or mouse pointer was being used. The list of additional collected features is:

Table 4.1: List of additional features collected on each mouse click that may later be relevant

element
path of the clicked element (DOM Traversal path)
x position of clicked element
y position of clicked element
height of clicked element
width of clicked element
device is touch device
whether user is using a touchpad
url at the time of clicking
browser user agent string

After performing several tests on the impact loading the framework has using different constraints and different browsers, the delays were almost always between 12 and 160 mil-

[7] http://stackoverflow.com/questions/10744645/detect-touchpad-vs-mouse-in-javascript

liseconds. However, since the script is loaded asynchronously, longer load times do not affect website usability at all. The delay when clicking on buttons is between 10 and 200 milliseconds depending on both the client's resources and the amount of motion. The high end of the spectrum is only reached if the visitor vigorously moves their mouse around for 2 minutes (if no suitable starting point was found 2 minutes prior to clicking, the entry was disregarded). Most of the times the delays were between 10 and 30 milliseconds which was an acceptable and imperceptible delay.

4.2.5 Data Storage

The full code used to serve the framework and store data can be seen in A.3.

Before determining whether the data collection framework should be loaded on the client page, it must be determined whether consent was given for cookies and web analytics. If this is the case and the client is not visiting the website on a mobile browser, the framework is loaded using the DimML line of code already present on the website.

The loader file also contains a parameter specifying which elements will be 'tagged', which is to say: which jQuery selector will be used to determine on which elements can be clicked to send feature data to the server.

For every click that occurs on a client website that is running the collection framework, the DimML script receives the features of that click. It attempts to add label data to that click if it is known for the current visitor (for example if a user has already filled in the box with date of birth). On the DimML server, data is aggregated and written to the research server after a short time interval. This reduces the number of concurrent connections to the research server as it would not have been feasible to open a connection for every click of every visitor. This also means that if a user never fills out the gender selection or date of birth, clicks will be stored for this user for which no corresponding label is known. This is done purposefully, as for some of these entries the user will fill in demographic attributes later on during their interactions with the site. This will retroactively make the label available from earlier clicks.

There is no data validation taking place at this stage other than that a value for all the features has to be received. Even if the input data is not correctly formatted or contains erroneous values they are still stored on the research server. The mouse logs are saved to a different file for every hour, to limit the size of each file (as it is harder to append to larger files over sFTP).

4.2.6 Description of Collected Data

Data was collected over a period of 14 days (from 26th of April 2016 to 8 May 2016) for all sites.

For the Health insurance sites, data from 3809 labelled visitors was collected, with a total of 15187 clicks from these visitors (averaging roughly 4 clicks on tracked buttons per session). The airline website yielded 29884 labelled visitors [8] with a total of 153615 clicks from these visitors (averaging a little over 5 clicks on buttons being tracked by the framework per user per session).

The logs were spread out over 585 files and composed a total of roughly 500 MB.

[8]This is only a small fraction of the total visitor because labels are only added once an actual booking takes place. The vast majority of users simply browses for tickets and prices without proceeding to booking

4.3 Prediction Techniques

4.3.1 Literature on Prediction Techniques

In the field of user modelling, traditional domain-knowledge is increasingly swapped out for more data-driven statistical analysis. Instead of assumptions regarding which customers with certain demographics prefer certain products, these conclusions are drawn based on ever-growing datasets. The prevalence of statistical approaches to user models is closely related to Machine Learning (also called Predictive Modelling) and opens the door for finding User Models by comparing new users to data in the dataset using different statistical methods [56].

Machine Learning techniques have been in existence since the early 1980's (and statistical methods have their origins even further back) but have received renewed attention and widespread application due to the huge amount of data that is now being collected and stored and the improved and accessible computing power. Caruana and Niculescu-Mizil offer a concise overview of several different machine learning techniques and compare their performances in different applications [7]. Xavier Amatriain also offers his views on different techniques in an online answer that helps explain some of the up and downsides [9] [1].

Since the goal is to construct user models, the prediction techniques will be evaluated according to their ability to accurately predict new individual instances based on past observations. Models are preferable to in-memory solutions when the number of training examples grows.

Support Vector Machines are a mathematical model that aim to find a separating hyperplane between datapoints with a binomial class. This is done based on certain defining points that are called the 'support vectors'. It is usual to transform the data using a kernel (sometimes referred to as a 'kernel trick'). The advantage of this manipulation is that it can allow the algorithm to find a linear model to fit non-linear input data. SVM's are used in many different fields and many applications. Examples of current uses can be seen on `http://www.clopinet.com/SVM.applications.html`. Downsides of SVM's include the difficulty of choosing a kernel, finding correct parameters for the model and the risk of overfitting for observed test cases (due to the nature of minimizing the error). Especially the *gamma* (γ) parameter which roughly specifies the individual influence of training examples and *nu* (ν) which roughly binds the number of support vectors used and serves as an upper bound to the margin of error. Experimentally deriving the best values for these parameters is still the most common practice. More importantly, SVM's are computationally heavy and alternatives with better performance are better suited to industry applications [1].

Neural Networks work by linking input features to 'neurons' (based on the nervous systems of animals in nature) and having these neurons fire a signal to other neurons if a certain activation requirement is met. The most common of these networks are the Multi-Layer Perceptrons (MLP's) although various other variants exist. In MLP's there can be multiple in-between layers, called *hidden layers*, and ultimately one neuron in the last layer, the output layer can fire. Neural networks are often used in image and character recognition tasks (pattern

[9] `https://www.quora.com/What-are-the-advantages-of-different-classification-algorithms`

recognition for classification) and for approximating functions for regression tasks. For the classification problem space, Neural Networks with back propagation for the error works well for many cases.

k-Nearest Neighbour (kNN) is an example of a frequently used instance based learning technique. It does not work by creating a model, but instead compares new datapoints (instances) to the k nearest datapoints from the labelled set. Distance is usually measured as Euclidean distance in all feature dimensions. k-Nearest Neighbour is a fairly straightforward algorithm with decreasing sensitivity to noise with larger values for K, although it can have problems finding a good decision boundary. Naive kNN implementations can be very inefficient for large training sets as all points are kept in memory and must be compared for distance. Adaptations of the algorithm exist that improve efficiency of finding the nearest neighbours (for example, by using indexed trees). kNN can also be used to solve regression tasks by for example interpolating the k nearest instances in the training set.

Decision Trees work by creating a model by recursively splitting the training set on different attributes and looking at which splits yield the most 'gain' (as is common with these algorithms, the definition of gain can depend on the implementation or desired outcome). At the leaves of the tree come the classification result. Decision Trees do not require data to be normalized on forehand. Benefits of decision trees include the fact that the resulting model is often relatively easy to comprehend, although the number of branches and decision points can quickly increase as the input feature space increases. They are also good with categorical data, whereas other algorithms prefer numerical input.

Multiple Trees can be used in a **Random Forest**, in a form of ensemble classification. The advantage over single trees is that trees are trained from random subsets of the input data, reducing the chance that the classifiers are training a tree to too specifically fit the training data (overfitting). Instead of splitting on training examples, the different trees can also only use a subset of the input features to construct a model.

Linear Regression is a classification algorithm that works best for features that are roughly linear. It is an approach to model a numerical outcome value based on a set of scalar input variables. It can be extended to polynomial functions on the input vector in **Polynomial Regression**. Kernels can also be used on the input features to try to find a linear separation of the data.

With these being only a selection of methods available to us, this research will be limited to comparing several of the popular methods and compare their effectiveness to the case at hand.

Although the literature suggests that there are statistical differences in the features extracted from mouse behaviour, there may be a large segment of the population for which the mouse features do not reflect their demographic attributes. For these website visitors no *stereotypical gender behaviour with regard to mouse cursor motion* patterns are present. As one can imagine, it is very likely that if gender differences exist, there are multiple men that have female attribute values and vice versa. Because of this, it will be difficult to find a crisp decision boundary between male and female. This means that a single model is unlikely to be very accurate for classifying both male and female users.

To solve this problem two one-class classifiers can be trained, in an attempt to shift the decision boundary to the point where the algorithms can hopefully accurately identify one class (disregarding the number of wrong classifications) [10]. This means that a unique model will need to be trained for each possible label value (one for predicting male instances and one for female instances in our gender example). These models are expected to have a higher degree of precision in their class prediction (as it will be the performance metric they will be trained for), but a low overall accuracy. This is not a bad thing, as it is more beneficial to the customers to have an accurate user model for a part of the population than an inaccurate one for everyone.

4.3.2 Conclusions Regarding Prediction Techniques

4.4 Data Analysis

This section explains the steps of processing the data to make it ready for the different learning algorithms, and how those algorithms will be optimized.

4.4.1 Data Preprocessing

Because the collection framework stores clicks of users before their labels are known and in separate files, some preprocessing has to take place to remove the entries with invalid data and add labels to data points for which the labels became known later. The Python scripts iterate through all the files and store the last values for age and gender for the website visitor. These values get assigned to every click from that user. A separate result file is built in which only clicks with label attributes are placed. The mouse features for users for which no labels were known were disregarded for the rest of the experiment. During this pre-processing step validation of the visitor id takes place as well as a check for the presence of labels.

The full script can be found in Appendix B

4.4.2 Additional Features

Since the starting position and target vary greatly amongst the different clicks, some features such as the ’overshoot’ (the distance moved further than the target) and the ’time after peak speed’ can depend greatly on the total distance moved. It is very conceivable that users who both have 20% of their movement take place after they have reached their maximum speed have very differing absolute values. For this reason, some of the features will also be expressed as percentage of total movement time or distance.

The overshoot (the amount of pixels that the user went further than the target that was ultimately clicked) likely depends on starting position and distance. The percentage of overshoot may be a better indication of performance when compared to the total movement distance. Similarly, percentages are calculated for the time spent moving in a certain direction (towards the goal, perpendicular to it or away from it) and the percentage of time the user hovered the cursor over the clicked target. These measures give an indication of how much time was

[10] A multi-class classifier that includes an ‘unknown’ option could also be used in combination with a performance metric defined to be weighted so that an ‘unknown’ classification incurs less of a penalty than a wrong result. However, this might quickly lead to ‘unknown’ classifications being the preferred class

relatively spent on each part of the interaction, in the hopes of reducing the variation in tasks may have on the observed features.

For an additional exploration step the browser which was used, taken from the User Agent string, was also recorded. This is not perfectly accurate, as the user agent string tends to name the rendering engine that was used rather than the actual browser. The browser classification rules are explained below.

- time after peak percentage

  calculated as $\frac{\text{time after peak velocity}}{\text{total movement time}}$

- time on button percentage

  calculated as $\frac{\text{time on button}}{\text{total movement time}}$

- time towards percentage

  calculated as $\frac{\text{time towards}}{\text{total movement time}}$

- time side percentage

  calculated as $\frac{\text{time side}}{\text{total movement time}}$

- time away percentage

  calculated as $\frac{\text{time away}}{\text{total movement time}}$

- overshoot x percentage

  calculated as $\frac{\text{x overshoot}}{\text{total distance}}$

- overshoot y percentage

  calculated as $\frac{\text{y overshoot}}{\text{total distance}}$

- overshoot total

  calculated as $\sqrt{(\text{x overshoot})^2 + (\text{y overshoot})^2}$

- overshoot total percentage

  $\frac{\text{overshoot total}}{\text{total distance}}$

- browser

  calculated as *firefox if* "firefox" appeared in the UA string, *safari if* "safari" appeared in the UA string and "chrom" did not, *chrome if* "chrom" appeared in the UA string, *opera if* "opera" appeared in the UA string and *internet explorer* if "MSIE" appeared.

4.4.3 Data Cleansing

After importing the data, examples are filtered for which there are missing attributes. Since there are enough instances left, no attempt is made to replace missing or incorrect values instead. If this data was not sent and/or stored correctly (something that should not be possible for normal scenarios), the other non-missing features of this instance can not be trusted to be accurate, so the entire observation was disregarded.

Next is a data sanitation step in which values that can not occur are filtered. For example, due to the way that some browsers automatically fill form fields or process the client-side script, some users are shown with an age of 2016. The example set is filtered according to the following rules:

- age must be between 18 and 100
- the number of passengers (in the case of the airline) must be 1
- the user must *not* be using a touchpad or device that supports touch input
- precision must be between 0 and 1 (inclusive)
- accuracy must be between 0 and 1
- peak speed must be non-negative
- overshoot is not greater than total distance

After this step the additional features described in 4.4.2 are added. The data is then normalized using the Z-score transformation, as the distributions of the attributes are largely normally distributed with a few outliers that are much larger or smaller than the other values. This makes Min-Max normalization impractical due to it's sensitivity to noise (as the vast majority of examples would all get very similar values). [11]

The benefit of using the z-score method is that it does not actually transform an attribute to the $[-1, 1]$ range but in fact goes from $[-\infty, +\infty]$. However, since the data is roughly normally distributed, most values will lie within 2 or 3 standard deviations of 0. Instances that, after normalization, contain attribute values smaller than -3 or greater than 3 are removed as they are likely erroneous [12].

4.5 Gender Classification

Because individual users generally have multiple interactions during a website visit, more than one click per user is recorded (averaging between 4 and 5). To reduce the risk of overfitting our classifiers for specific users, all clicks from the same user all either fall in the training or testing set. Users are split based on the user id's and label. Since many algorithms will work best if

[11]The interquartile range as a normalization method would also have been acceptable

[12]Skipping the normalization step was also performed for classification using Random Forest in one execution. These algorithms are fairly robust against noise, so the effect of skipping the filtering step was also tested

there is a roughly equal prevalence of labels in the training set, class imbalance is a problem that needs to be addressed. Down-sampling was chosen (only take as much instances of the most-common class as there are in the least-seen class) instead of bootstrapping (which would rely on generating additional datapoints of the less seen class, whether or not by intelligently creating new instances). Although it is commonly accepted that down-sampling 'wastes' valid data, the large number of clicks present in the data will leave enough instances to train the models.

The choice was made to train a separate model for each of the two genders, each aiming to find a good decision boundary for which for their respective classes they will be able to predict when instances belong to this class with a high degree of precision.

For each of the algorithms, first only the features are selected that will be used to train the algorithm with and not the additional features captured for alternate purposes. The training set is duplicated and in one of these sets the *male* label is set to be the positive class and in the other *female* is the positive class. For each algorithm an optimization step takes place where different parameters are compared for the algorithm using a grid-search.

In order to optimize the parameters, a good quantification of classifier performance must be defined. the goal of this step is to create two models that are precise in their predictions for the positive class. This leads us to optimizing for precision in the positive class as the performance measure of choice. [13]

4.5.1 Results Gender Classification

Classification accuracies and class precision for all the classifiers were mostly around 50%. This is reflected in the ROC curves for the models trained on the airline input data seen in Figure 4.2. All the curves roughly follow the straight line that is expected from a classifier picking labels at random [14].

On first glance, none of the algorithms produce results that are significantly better than guessing. This suggests that based on the selected features it is not possible to predict the gender of the visitor during normal interaction. *However*, the Random Forest classifier without normalizing input features, seen in Figure 4.3, shows better than guessing performance. This classifier has the largest AUC [15]. An overview of the class precision of several classifiers is shown in Table 4.2. From this, it can be seen that although some classifiers appear precise, when applied to a different testing dataset the performance frequently drops back to levels more consistent with random labelling. It is expected that a good classifier for this class scores better than guessing regardless of the dataset to which it is applied. This suggests that both the airline and insurer SVM models for detecting female users are slightly better than guessing. The male classifier for Random Forests, especially the one trained for the larger airline dataset, seems to be fairly precise in identifying male users for the given threshold values.

[13]Because of the way the test data were sampled earlier, our classes are more or less balanced. This means that optimizing for the Area Under the ROC (Receiver Operator Curve) can also be used as a good indication of classifier performance without risking specific situations described by Davis and Goadrich [12]

[14]This only applies if class distribution is equal, which is what was used in this research

[15]Although some researches dispute its value as a good performance indicator, it is fairly useful in our case [28]

Table 4.2: Class Precision scores for SVM, Random Forest and Random Forest without normalization trained and applied to different training and testing sets

SVM model		applied to airline	applied to insurer
airline	male	48.78%	45.27%
	female	51.55%	58.73%
insurer	male	-	*100.00%* [a]
	female	*50.67%*	*54.96%*[b]

RF model		applied to airline	applied to insurer
airline	male	-	-
	female	49.17%	42.69%
insurer	male	*47.37%*	*64.52%*
	female	*49.24%*	*53.21%*

RF (no normalization) model		applied to airline	applied to insurer
airline	male	66.67%	100.00%
	female	47.90%	41.23%
insurer	male	*51.86%*	*60.61%*
	female	*47.88%*	*100.00%*

[a]Only two cases were, albeit correctly, identified as male. Although this could be very precise, it may also be a fortunate accident. Class Recall for male was 1.82%

[b]Since the insurer dataset was much smaller, the testing set was also smaller. The value of 54% corresponds to a randomly introduced class imbalance in the testing set and was not significant

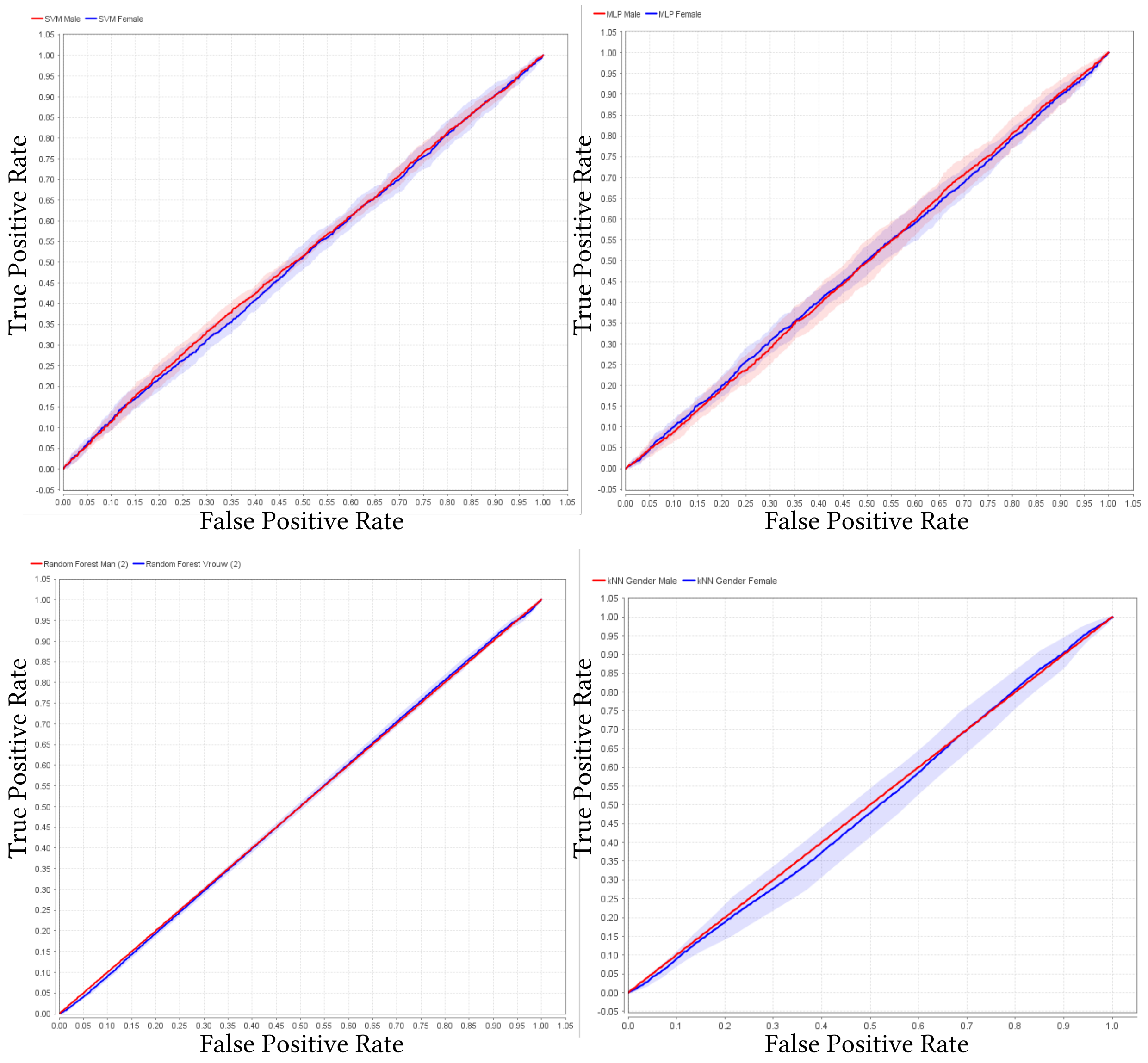


Figure 4.2: The Receiver Operator Characteristic curves plotting False Positive Rate (x-axis) against True Positive Rate (y-axis) for the different classifiers trained on the airline input. left-top: SVM , right-top: MLP, left-bottom: Random Forest, right-bottom: kNN

When applying the models to the smaller (unbalanced) set of health insurer data, occasionally better performing models are found. Unfortunately, applying these models to a larger test set shows the precision scores dropping rapidly. The best performances were obtained by the Random Forest Classifier applied without normalizing the data, although the high performing classifiers tended to only be precise when labelling the minority classes or were only classifying a very small sample of the positive cases (such as the Random Forest classifier identifying male

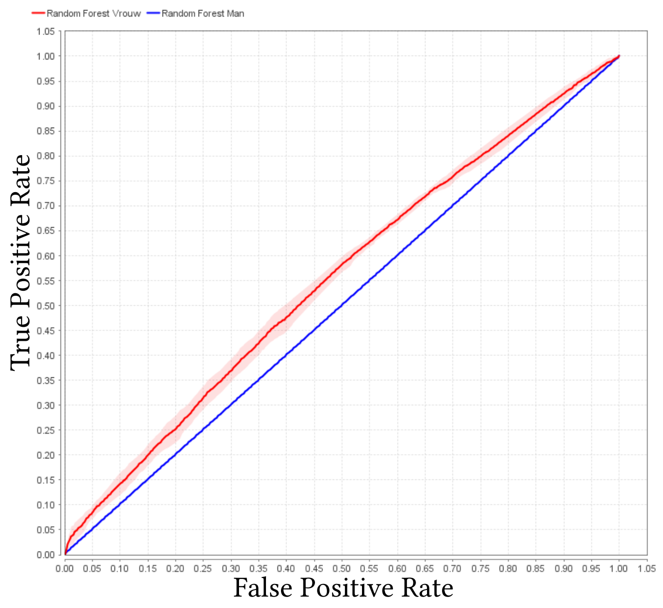


Figure 4.3: The Receiver Operator Characteristic curves plotting False Positive Rate (x-axis) against True Positive Rate (y-axis) for the Random Forest classifier trained and applied to non-normalized input data

users only predicted 7 of the 2625 cases as male. The precision of 71.43% is hard to confirm as it may have been chance with such small numbers). The overview can be seen in 4.3. Interesting to note is that when this male model is applied to the dataset obtained from the other website, it has a class precision of 100% by correctly labelling 6 users as male out of the total click example set. This result can be seen in Table 4.4 and it suggests that the found model was not simply a case of overfitting on certain behavioural outliers in the airline dataset.

When applying this RF model on the airline input data, the results shown in Table 4.3 are obtained. Although at first glance, these numbers seem indistinguishable from guessing, when the proper threshold values are applied accuracy for the whole population increases to around 55%. This comes closer to the accuracy obtained by earlier experimental results, but still falls short of the predictive power hypothesized. This is however only when applying the classifiers to the whole population, and not to only smaller subsets.

Because the Random Forest without normalization outperforms the other classifiers for most configurations of threshold values, optimal configuration of thresholds are found for this RF Classifier. It is only expected that accurate classification can take place for a subsection of the populace, so different thresholds for the Random Forest Classifier are compared and an optimal one selected. The found male classifier will not attain higher precisions by changing the threshold as can be seen in Figure 4.5. The classifier trained to be precise for identifying female users shows interesting patterns when varying the confidence value needed to classify, as is plotted in Figure 4.4. It is seen that the algorithm can be very precise if confidence is high, but it becomes applicable only to a small part of the population. What becomes important for these cases is the class recall. From these numbers, a feeling for how many female users show

Table 4.3: Result of the Random Forest Classifiers without applying normalization to the input features (includes more noise as no filtering based on distance from the mean after Z-score normalization was performed).Top: Female classifier, Bottom: Male Classifier

Train	airline		
Test	airline		
Model	rf_nonorm_airline_vrouw		
	True Female	True Male	Precision
Pred Female	846	920	**47.90%**
Pred Male	404	455	52.97%
Class Recall	67.68%	33.09%	

Train	airline		
Test	airline		
Model	rf_nonorm_airline_man		
	True Female	True Male	Precision
Pred Female	1194	1229	49.28%
Pred Male	1	6	**85.71%**
Class Recall	99.92%	0.49%	

Table 4.4: Result of the male Random Forest Classifier without applying normalization trained on the airline dataset applied to the insurer dataset

	True Female	True Male	Precision
Pred Female	446	664	41.24%
Pred Male	0	6	100%
Class Recall	100%	0.90%	

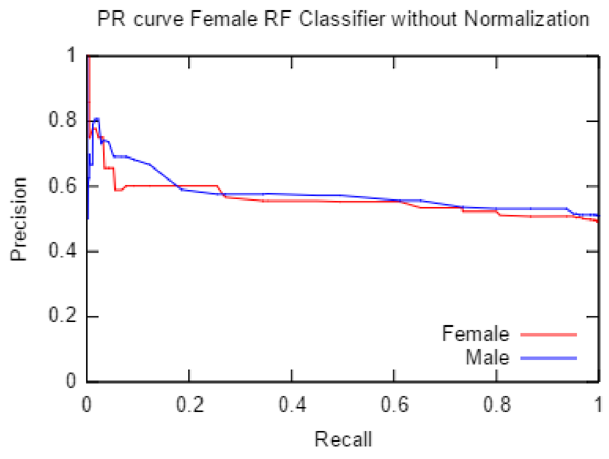


Figure 4.4: Graph showing the trade-off between precision (for classifying both male and female users) and recall when varying the threshold for the female Random Forest Classifier without Normalization

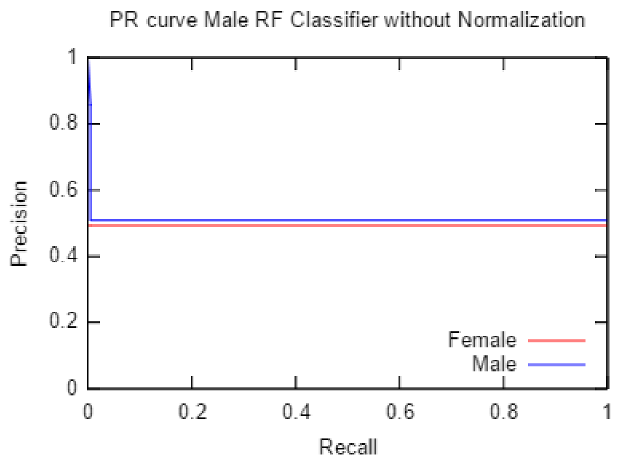


Figure 4.5: Graph showing the trade-off between precision (for classifying both male and female users) and recall when varying the threshold for the male Random Forest Classifier without Normalization

stereotypical behaviour can be obtained. Accuracies of roughly two thirds can be obtained for around 4% of the population. The optimal value depends on the business proposition and the cost of misclassification, but the numbers drop rapidly which shows that gender-specific cursor motions apply to only small fractions of the populace. Especially the female classifier shows consistent higher precision than guessing for gender classification, although precision drops rapidly as recall grows.

For the male classifier this is less applicable. Although accuracy values are high regardless of the testing set on which it is applied for a threshold of 0.5, the recall is below 1%. This means that it can correctly identify male users, but only in rare cases. When we apply these models to the Health Insurer dataset, the Precison-Recall curves seen in Figure ?? are obtained. Although the models follow a similar trend, performance is a little worse. The male classifier is still very precise for only a few examples in the testing set and is not very useful beyond that, slightly less than 1% of males get labelled, but with a precision of 100%. The best model for female classification performs worse on the other dataset, achieving a maximum precision of 60% but actually wrongly labelling instances as female with high confidence, leading to performance worse than guessing when setting high threshold values.

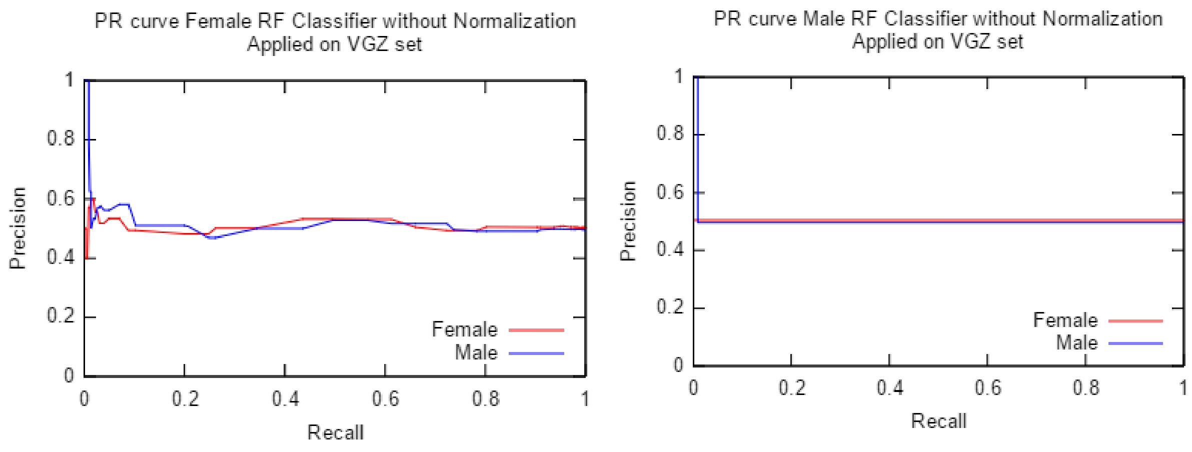


Figure 4.6: The Precision-Recall curves for the female Model (left) and male model (Right) applied to the Health Insurer dataset

4.6 Age Prediction

For age prediction two different methods were utilized. One is a classification problem similar to the one attempted for gender as described in the section on Gender Prediction 4.5. For this the data was discretized into different age bins and sampled these bins to contain roughly the same amount of instances.

Additionally performance of regression to predict the age of users will be evaluated. Regression will be evaluated on the Root Mean Squared Error statistic. This statistic is one of the most commonly used ones in comparing the result of regression analysis. To make the results easier to interpret, a discretized overview of the applied model and the corresponding plot of the prediction against the labelled value in a scatter plot will be generated.

4.6.1 Results Age Prediction

Age prediction took place in two ways, i) using 3 predefined bins in which users fell, shown in Table 4.5 and ii) by using all the examples and regression to predict the age of the test subject.

Table 4.5: The different age bins in which users fell together with the number of clicks in each bin

age		airline
0-33	610	5654
33-66	471	3406
67-100	55	985

The results of applying different techniques on the bins is shown in Table 4.6. Classification results show that the algorithms perform marginally better than guessing after balancing the test data classes (random performance would be 33.33%). For all cases, the algorithms have a good recall when it comes to predicting users between 60 and 90 years of age. Upwards of 43% of users in this age bin are correctly identified. When it comes to precision in predictions, accuracies are best for predicting the youngest and oldest age bins. Interesting to see is that the found Random Forest classifier never predicts users to be in the 30-60 range. Class precision is always higher than 33%, what a guessing algorithm would be expected to score. In Table 4.7 the same models are applied to the dataset obtained from the Health Insurance websites. Accuracies for all models drop slightly, as do class precisions. Especially precision in identifying users in the range 0-30 falls significantly to levels worse than guessing for all but the Random Forest.

As for predicting the numerical age (rather than the bins), it can be seen that the regression results tend to deviate quite far from the observed cases. The graphical overview shown in Figure 4.7 clearly shows that predictions do not follow the optimal linear line[16]. The best regression result shows a Root Mean Squared Error of roughly 16 years. On a population with an average of 43 years, this is a rather large error which makes the result not very practical for targeting purposes. It can be seen that precision is just a little better than guessing for some of the less frequently occurring classes, but that recall is only good for values around the average.

[16]A discretized version of the regression result which gives a more intuitive sense of how accurate the regression classifier is also given in Appendix G, Table G.15. This gives only a rough overview as it is possible for examples to be fairly accurate but falling just at the cut-off point to the wrong bin

Table 4.6: Result of three different classification algorithms on the binned input dataset from airline

Model	Random Forest			
Train	airline			
Test	airline			
Overall Accuracy	41.27%			
	true 0 - 30	true 30 - 60	true 60 - 90	*precision*
pred 0 -30	267	244	174	*38.98%*
pred 30 - 60	0	0	0	*0.00%*
pred 60 - 90	219	285	381	*43.05%*
recall	*54.94%*	*0.00%*	*68.65%*	

Model	SVM			
Train	airline			
Test	airline			
Overall Accuracy	38.15%			
	true 0 - 30	true 30 - 60	true 60 - 90	*precision*
pred 0 -30	199	168	154	*38.20%*
pred 30 - 60	130	161	162	*35.54%*
pred 60 - 90	157	200	239	*40.10%*
recall	*40.95%*	*30.43%*	*43.06%*	

Model	kNN			
Train	airline			
Test	airline			
Overall Accuracy	38.92%			
	true 0 - 30	true 30 - 60	true 60 - 90	*precision*
pred 0 -30	170	151	142	*36.72%*
pred 30 - 60	133	153	125	*37.23%*
pred 60 - 90	183	225	288	*41.38%*
recall	*34.98%*	*28.92%*	*51.89%*	

Table 4.7: Result of three different classification algorithms trained on airline input and applied on the binned input dataset from the Health Insurance Sites

Model	Random Forest			
Train	airline			
Test				
Overall Accuracy	39.18%			
	true 0 - 30	true 30 - 60	true 60 - 90	*precision*
pred 0 -30	32	30	35	*32.99%*
pred 30 - 60	0	0	0	*0.00%*
pred 60 - 90	17	36	44	*45.36%*
recall	*65.31%*	*0.00%*	*55.70%*	

Model	SVM			
Train	airline			
Test				
Overall Accuracy	35.14%			
	true 0 - 30	true 30 - 60	true 60 - 90	*precision*
pred 0 -30	5	24	18	*10.64%*
pred 30 - 60	17	42	35	*44.68%*
pred 60 - 90	9	17	18	*40.91%*
recall	*16.13%*	*50.60%*	*25.35%*	

Model	kNN			
Train	airline			
Test				
Overall Accuracy	27.57%			
	true 0 - 30	true 30 - 60	true 60 - 90	*precision*
pred 0 -30	15	40	40	*15.79%*
pred 30 - 60	10	16	11	*43.24%*
pred 60 - 90	6	27	20	*37.74%*
recall	*48.39%*	*19.28%*	*28.17%*	

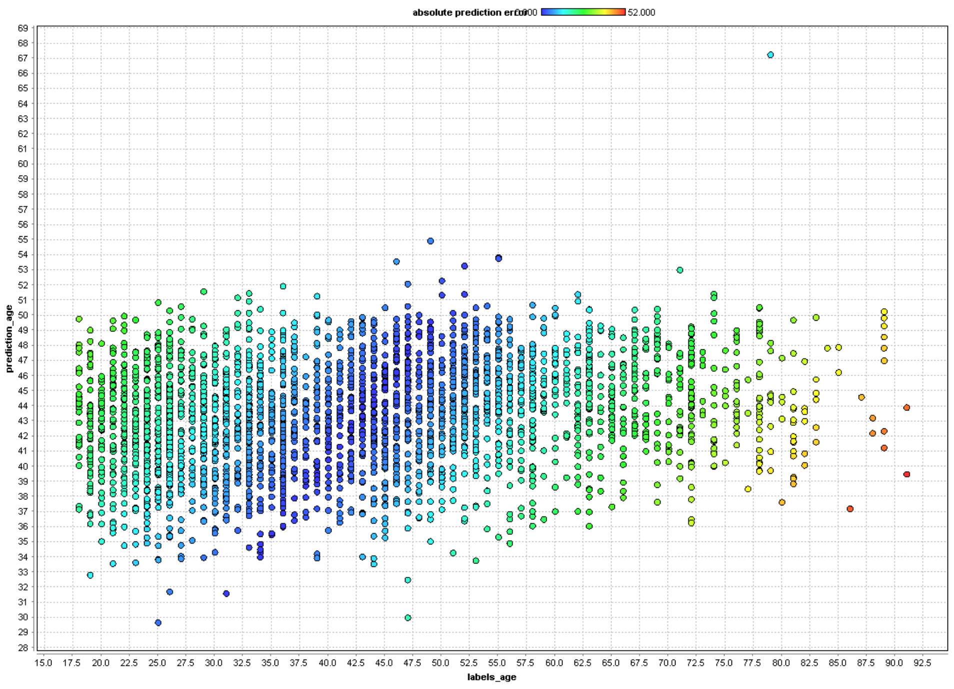


Figure 4.7: A scatter plot of the linear regression result showing predicted age on the y-axis and actual age on the x-axis. Colour of the points encodes for the absolute error

Chapter 5 Conclusions

Conclusions based on the results presented in the previous chapter are given in this chapter and implications for the research questions stipulated in Chapter 2 are explored. The conclusions will be presented as how this research contributes to the field overall and the answers to specific research questions are presented too.

The implications of the findings for the scientific community and the businesses involved, as well as an analysis of the applicability of some of the claims made in related research relating to gender differences in cursor behaviour under experimental conditions, are documented in this chapter as well.

5.1 Main Contributions

This research has set out to answer the following question:

> *Is it possible to predict gender and age of website visitors using features of cursor movements during normal interaction?*

Based on a large dataset of cursor movements collected during normal interaction from different websites, it was shown that gender prediction can be done significantly better than guessing, but not near perfect and only for a very small part of the population. The optimal threshold depends on the costs and benefits of misclassification and correct classification respectively which can vary with every business case, but overall it can be said that for any application targeting large fractions of the population performance of the best models will not be good enough for any practical implementations.

Predicting age revealed comparable results. The obtained models performed better than guessing and with (in most cases) significantly better recall than the gender-demographic predictor. It turns out that older users and younger users are easier to identify than the middle age range. Although these models could possibly be applied to identify when a user falls in a special category that might have less financial means or different needs in health insurance (younger age group) or different luxury requirements for a flight or wishes to have more optional modules on top of the basic health care package (older age group), precision and recall of this model has the same problems as the gender models. For any real-life applications the results are not significant enough. The regression result is frequently too far off to be useful. Since demographics may be available on group-level through third-party analytics, indicating for example that most of the visitors will usually be in a certain age range, the added benefit of slightly better than guessing accurate age classification or a regression model that will frequently fall in the 40-50 range has only limited uses. The results were more or less replicable on different websites, indicating that the results are generalizable over many different sites, although applicability is low.

Overall, it was shown that unobtrusively obtained features of mouse dynamics on websites, independent of domain-dependent data, have some predictive power with regard to visitor gender and age. It was shown that precision levels were better than guessing but too low for any business purposes or only available for a very small fraction of the population.

5.2 Input Features

This research has shown that despite differences being observed in cursor motion features for different genders and age groups under experimental situations, practical demographic attribute prediction on a live website is not yet feasible at levels accurate enough to be of interest to the business. Models that perform marginally better than guessing may in fact do more harm than good, as offering wrongly personalized content may in fact reduce engagement more than non-personalized content. For the customers that cooperated in this research it also plays a role that they are already aware of some of the common statistics regarding their visitor demographics, making added benefit to certain predictions marginal.

When comparing claims made regarding explanatory power of cursor motion features gender differences in previous work with the Kolmogorov-Smirnov analysis on the feature-set collected for this experiment, the results of which can be seen in Table 5.1, it can be seen that certain differences shown in experimental settings were not reproduced in this research. Note that this does not necessarily imply that their conclusions are invalidated, but it gives cause for further research.

Females deviated more from the straight path
Not shown, although path accuracy was slightly higher for women than for men, the distributions were not significantly different

Females showed more motion in the away direction
This is not supported by our data. If anything, both in absolute terms as well as the percentage of the total time spent in the away direction, male users seem to spend more time in this direction. It can not conclusively be stated that the distributions for men and women are different with a confidence p-value of them being the same having a value close to 0.3.

Females showed more forward motion
(possibly to compensate for the movement in the away direction). This is also not supported by the data. Men seem to spend more time moving towards the goal, and the

distributions are significantly different. This is only a measure of the time spent however, not the distance moved in a certain direction.

Sidewards deviation was the same for men and women
At first glance this is supported by the data as the values for both men and women are similar, although the distributions vary a little in shape. However, the *p*-value is around 0.073 which could mean that the sidewards deviation distribution is slightly different for men and women.

Men had shorter movement times than women
This is not supported by our data, overall it shows that male visitors have longer movement times and that the movement time distribution for men is significantly different from that of women.

Women were more accurate than men
This finding is not supported by the data.

men spend less time decelerating
This conclusion can also be drawn from this experiment. Although the difference is only about 1%, men do seem to be a little faster and also more consistent and the distributions are significantly different.

Some of the earlier work done is corroborated by our findings, which suggest that those findings are widely applicable and not a result of the experimental setting under which they were obtained. Since prediction results were consistently better than guessing for both gender and age prediction (for certain classifiers) and that class precision for especially the oldest age bin is always higher than 33% (precision of guessing classifier), it follows that there is some predictive power in the chosen set of input features, albeit that it is very limited.

5.3 Gender and Age Prediction

5.3.1 Gender Prediction

Although most of the models were ultimately unable to identify gender demographic accurately for the majority of visitors, the results show that there is some predictive value in using mouse dynamics. Although only a small part of the population displayed behavioural patterns for which the algorithms could discern users with around 65% accuracy, these results are promising when it comes to creating user models for websites that do not have access to user specified information or have fewer contextual input data that can be used. There are good indications that these models are applicable to a wide range of different websites, as similar classification results were obtained when applying the model to a different dataset. This work establishes that prediction with roughly two thirds precision is possible for 4% of all female visitors and around 1% of male website visitors. This was in the line with expectation **E2**. It is shown that precisions of 65% and upwards are obtainable and from the Precision-Recall curves it could be determined that the best male model found actually can achieve precisions of 85% and upwards regardless

Table 5.1: Comparison of male and female attributes using Kolmogorov-Smirnov statistic. Null-hypothesis (same distribution) not significantly enough shown for entries in bold

Value	KS	KS p	Male μ	Male σ	Female μ	Female σ
acceleration_mean	**0.0304**	**0.0195**	**0.0058**	**0.1092**	**0.0054**	**0.0698**
acceleration_stdev	**0.0272**	**0.0492**	**0.1119**	**0.9583**	**0.1062**	**0.5802**
actual_distance	0.0246	0.0957	439.13	335.29	432.23	330.24
angle_mean	0.0229	0.1424	0.8473	0.3511	0.8494	0.3523
angle_stdev	0.0181	0.3864	0.7165	0.2645	0.7177	0.2700
average_speed	0.0180	0.3917	0.3970	0.2854	0.3911	0.2818
movement_time	**0.0287**	**0.0318**	**2417.2**	**3514.9**	**2330.2**	**2010.0**
overshoot_total	0.0225	0.1574	78.044	130.426	71.306	116.757
overshoot_total_percentage	0.0158	0.5572	0.1403	0.4433	0.1417	0.6234
overshoot_x	0.0211	0.2150	46.016	117.848	40.138	103.556
overshoot_x_percentage	0.0152	0.6042	0.0973	0.4439	0.0967	0.6236
overshoot_y	0.0167	0.4850	40.749	73.727	39.878	69.222
overshoot_y_percentage	0.0207	0.2315	0.0557	0.0809	0.0580	0.0840
path_accuracy	0.0149	0.6290	0.6390	0.2775	0.6431	0.2706
peak_velocity	0.0226	0.1535	8.0358	13.2612	8.1058	10.9735
performance_index	0.0122	0.8509	0.0017	0.0010	0.0018	0.0010
precision	0.0194	0.3003	0.7310	0.1874	0.7283	0.1863
speed_mean	0.0178	0.4054	0.4133	0.3291	0.4046	0.3019
speed_stdev	0.0205	0.2426	1.1214	1.4930	1.1248	1.2506
time_after_peak_percentage	**0.0318**	**0.0123**	**0.6904**	**0.1757**	**0.6998**	**0.1694**
time_after_peak_velocity	0.0186	0.3484	1702.2	3150.0	1695.4	1839.4
time_away	0.0196	0.2868	224.37	344.69	216.68	339.62
time_away_percentage	0.0196	0.2905	0.0877	0.0845	0.0865	0.0848
time_on_button	0.0161	0.5359	1341.0	3421.1	1287.6	1861.4
time_on_button_percentage	**0.0272**	**0.0485**	**0.4877**	**0.2262**	**0.4943**	**0.2223**
time_side	0.0257	0.0733	193.01	269.60	184.52	200.00
time_side_percentage	0.0153	0.5962	0.0870	0.0674	0.0850	0.0655
time_towards	**0.0398**	**0.0007**	**567.89**	**351.14**	**548.93**	**362.41**
time_towards_percentage	0.0235	0.1235	0.2768	0.1280	0.2715	0.1270
total_distance	0.0263	0.0615	840.38	875.83	803.24	745.88
velocity_at_click	0.0160	0.5414	2.2319	3.5830	2.1994	3.4526

of the dataset it is applied to, but with recall values under 1%. The female classifier can be more precise than 65% when applied to data obtained from the site it was trained on but peaks at 60% when applied to a different website. However, the expectation was that although these scores would not apply to the general population but only to a segment of it, the segments are smaller than originally anticipated. Precision of 65% and upwards are not consistently obtained for the female classifier, but instead is closer to results observed in experimental settings in earlier work. Both classifiers only operate in the 60% and more precision range for low recall values of 4% and under, implying that only a very small part of the population has stereotypical behaviour that can be observed from the mouse dynamic features that were collected.

Interestingly, the model found for female users is also quite useful for identifying male users. When looking at the effect of adjusting the threshold of the female classifier to label male users, shown as the blue line in Figure 4.4, it is interesting to see that the highest confidence male classifications are not that precise. This is not unexpected as it was not trained for these cases. However, fairly good precision scores are obtained when predicting with a confidence of more than 0.6 for which 80% precision can be seen for 2% of all male users. By applying this optimal female model, 12% of the male users can be correctly classified 65% of the time, even using a classifier that was not specifically trained to be as precise as possible for this class.

These results show that gender prediction can be done for a part of the population, but the part of the population for which stereotypical behaviour is reflected in the input features is rather small. A small proportion of women seem to show more consistent behaviour with regard to mouse dynamic features, but more male users have similar patterns (for around 65% accuracy 4% of women as opposed to 12% of men can be correctly classified). This research supports **E2** that gender prediction can take place with precision of around 65% and upwards for a part of the population, but the part of the population for which it is applicable is very small.

It is not strange to see that the Random Forest Classifier is one of the best performers, as stereotypical behaviour may manifest itself in multiple ways which could possibly be distinguished based on several rules. This as opposed to for example the SVM classifier which tries to find a linearly separable hyperplane, assuming all entries are ultimately linearly separable (after application of the kernel trick).

This research indicates that mouse dynamics are a promising means to establish a rough user model on any website for these fractions of the population. As it stands the applicability is limited due to the small percentage of users that get picked out using these methods, but a system that constructs a user model on a website is not limited to cursor motion data alone. The accuracy of the predictions can likely be improved upon by introducing additional mouse dynamic features (such as static hand tremor) and by expanding our input feature space to include for example typing behaviour. Furthermore, a user model does not necessarily require crisp attributes, so a model only giving confidence scores could leave deciding the threshold to the website-owner based on the cost of misclassification or the importance of finding demographic attributes.

There are several reasons why precise predictions may only occur for a small part of the population. Some of the factors that may have been at play are that:

1. input data is highly dependent on local configuration and user environment

2. algorithms were not trained on features for the exact same tasks
3. no external stress factor was introduced

For point 1, users have different systems from which they are accessing the web. Different browsers, resolutions, mouse devices (and device settings such as sensitivity) all influence the behaviour of the cursor during normal interaction. Zoom levels on the website, custom plugins that alter layout could also have influenced the dataset. A large number of noisy values will therefore be present in our dataset which may have had big adverse effects on the precision and accuracy of the derived predictive models.

Another difference is that in our literature, the studies that identified differences between genders and age groups by their very nature kept independent variables constant. This means that the nature of the task had to remain the same. However, in normal browsing behaviour, users are expected to employ different strategies. For example: in Yamauchi's experiments, users were asked to click one of two buttons (which were always on a fixed location) as fast as possible from the same starting point [50]. This makes comparing the data easier when all input features are derived from the same task. The same can not be said for normal browsing behaviour when clicking buttons on a webpage. The behaviour of the user preceding the actual click moment may depend on a large number of variables. The user could have been 'reading along' with text, moved the mouse up and down out of frustration of load times or clicked the button as fast as possible. The nature of these action is unbeknownst from the web interaction data that was collected. These factors will also have played a role in the obtained models. Although this research does not irrefutably prove it, it stands to reason that realistic browsing behaviour is not similar to following specific experimental instructions. The results found by Yamauchi et al. may be closer related to the nature of the selection task than the actual pointing strategies and motor control exhibited by the test subjects.

Lastly, it may be possible that the differences arose only because of external stress factors. The fact that the subjects knew they were in an experiment may have affected their behaviour (with men and women reacting to these circumstances differently). The fact that tasks had to be performed within a short time period can also have factored in.

5.3.2 Age Prediction

Using different algorithms, overall accuracy when predicting website visitor age in 3 bins was consistently higher than 33%, suggesting that here too there is predictive power in the input features. The same conclusions regarding applicability can be drawn too, showing that although there is some promise in using cursor motion features, these features by themselves are not enough for current business demands. Age prediction could benefit from a more varied dataset obtained from multiple sources. Especially regression results tend to deviate towards the modal value to minimize the error metric. Using more bins and training individual classifiers in a similar manner to how gender was classified could help identify the trade-off between precision and accuracy. Precision scores per age bin were not consistently higher than 33%. Precision in identifying the older users was around 40% as was prediction for the users up to 30 years of age. This can be explained by reasoning that users that fall in the range 0-30 or 60-90 exhibit more extreme and/or stereotypical behaviour, whilst users in the range 30-60 demonstrate less common mouse dynamic features. Precision levels just barely above guessing level are not well-suited for business applications if the cost of misclassification is not offset by a significant increase in the benefits of proper classification. In the case of age prediction for either of the customers, this is not the case, with no known current targeting campaigns directed at specific age groups.

When applying the models to the data collected from the Health Insurance websites, scores drop and the identification of younger users is no longer possible with precision better than guessing. The precision score for users in the middle range increases for the SVM and kNN classifiers, but the scores for users in the 60-90 bin are still pretty consistent.

Overall accuracies of the models applied to the airline testing dataset were between 37 and 40%. When applying the models to the other dataset, these scores dropped a little but especially the Random Forest was still able to classify user age with overall accuracy of 39%. The expectation that binned age prediction would be better than a guessing algorithm (33%) seems supported by these consistently higher values. This goes to show that there is some predictive power in determining rough user age based on cursor motion features (as per expectation **E3**), although practical applications are limited.

5.4 Applicability

Although the number of websites from which data was collected was not high enough to unilaterally confirm or reject the expectation that found models will be generalizable across most standard websites, by comparing the performance of the best models to a different dataset indicative results are obtained. These show that for both gender prediction and age, the models found from the airline dataset performed a little worse when applied to users from another website. Although this could indicate that the model is specific to the website it was trained for and not directly applicable to other websites, the performance only dropped by a small amount. Results show that also on other websites, precision scores are higher than guessing levels, which indicates that the models are not only applicable to the website for which they were trained. It may be too soon to claim general applicability for most websites, but these results show that it is not unlikely that the models will perform consistently better than guessing on any website for which these input features can be collected (albeit that they will not be very useful). Overall expectation **E4** seems to hold and the found models hint towards a degree of generalizability over different websites.

Chapter 6
Recommendations & Future Work

Recommendations regarding future work based on the results of this research are suggested in this chapter, with regard to more scientific research that is warranted as well as next steps that will be beneficial to the companies that supported this paper.

6.1 Research into Mouse Dynamic Input Features

Although it was shown that there is predictive power in mouse dynamics collected during normal website interaction, the applications of the current input feature set have proven to be limited. Further research into additional features could have raise precision levels and recall so that accurate models can be found for enough visitors to justify personalization and targeting business cases. It may be interesting to develop experiments to determine how many people show behaviour ‘stereotypical’ for their demographic attributes. Furthermore, some of the claims made regarding explanatory power in mouse dynamic features seem to only be applicable in controlled environments when users are performing the same task with an external stress factor. Further investigation into task specific pointing strategies may reveal more features of interest or identify for which part of the visitors precise classification can occur. These visitors may have other traits in common that are worth investigating, for example they may be more susceptible to advertisements as an ‘external’ stimulus.

6.1.1 Recent Developments

A recently published paper looking at continuous user authentication has set out to solve some of the problems with using mouse dynamics for classification purposes [55]. To minimize the effect the specific environment of the user can have on the input data they focus input data on attributes that are less affected by for example pointing device sensitivity and screen resolution. They show that looking at the Cumulative Distribution Functions of the angles the user moves in and computing the difference from test cases yields great results for authentication. A similar type of solution for the other variables that looks more at the distributions of the attributes of user movement may be used to increase accuracy. However, their collection process spanned two hours per user, which is impractical for the purposes listed in this research as targeting usually occurs earlier on in a session.

Another recent paper was able to predict gender by using features derived from the swipes on smartphones [30]. This experiment, although performed under controlled conditions, could greatly complement the findings in this work as the use of mobile devices is increasingly common for browsing behaviour.

6.2 Research into Demographic Predictions

Accurate user models are the foundation of many Recommender Systems. In situations where obtaining a model through conventional methods is difficult cursor dynamic features can be used to predict with reasonable certainty, for a part of the population, what their gender is. From this research it is clear that there is merit in using cursor dynamics as a basis of inferring demographics. The domain of motor control and web element interaction is a field that merits further research and could prove to be a great source of information when composing a user profile. Aside from the demographic attributes that this research has focused on, performance may also indicate the current emotional state of the user (excited, angry). It is important to establish whether the results found in experimental settings are applicable when users are browsing normally and it is shown that to a certain extent this holds. The presence of stress-factors such as a time-limit or simply the fact that subjects' performance is being measured should be further investigated as a possible source for statistical differences in different genders.

The effects of age have been touched upon, but this field needs more work before any conclusions can be drawn for which the applications are clear. The effects of age on mouse dynamics should be investigated further in experimental settings. Furthermore, any claims done in this work need to be tested on different websites to see if the models that were found are truly as universal as was assumed. Unlikely as it is, the fact that users were assumed to be mostly from the Netherlands has had impact on the features as well. It is worthwhile to identify which other variables may have played a role and to what extent they have influenced the results.

Introducing more input features, and especially coming up with features that will work regardless of the local environment of the user, can help to reduce the amount of noise in the data and greatly improve the accuracy of gender prediction. More research into what fraction of the population shows gender-specific cursor behaviour and what kind of patterns correspond to this kind of behaviour would make for interesting follow-up questions that are worth researching.

6.3 Business Applications

With the work laid out in this paper, it is shown that a predictor can be built that will construct a user model with decent accuracy for a very small part of the population regardless of the website the user is visiting. This research lays the foundation for a general purpose user-model that can be found on any website which will not require domain-dependent information, offering the opportunity for websites to offer personalized content to a small fraction of the population. The next steps are twofold: i) improve the precision and recall of the models to useful levels and ii) the actual implementation of a service, preferably using the DimML technology, that will return a user profile based on click interaction on any website. This first step can likely be achieved by collecting additional features and minimizing noise in the input data. On top of that, users usually click more than once. The performance is currently linked to classification of individual click features, research needs to be done to see how using multiple clicks can improve scoring for individual users. Several options that may be researched include using voting for all past clicks, adjusting the prediction by using the current prediction weighted with the past confidence score or by using the maximum confidence value found thus far during the session. On top of that, mouse dynamics are not the only generally available information. More details such as the time of visit, keystrokes, time on a page or software may be collected and used to improve the accuracy of the models. More experimentation is desirable to obtain additional features and investigate if this will lead to better prediction levels. If other demographic attributes can be incorporated in this research it could reveal other traits that are good to market to.

For any website owner it is important to define these segments of users that require specific targeting strategies and identify the benefits and costs of classification of users. For example, most visitors to the airline website already fall in a specific age range and most visitors to the Health Insurer sites are male. It may be beneficial to identify outliers that are not part of the largest segments and target these varying users differently. Future work could involve identifying which segments are in need of being targeted differently and how a user model that incorporates these features can be inferred from the feature data listed in this research along with additional relevant, possibly more contextual, features.

Bibliography

[1] Amatriain, X. (2015). What are the advantages of different classification algorithms? Accessed: 2016-06-03.

[2] Andrés, J. D., Pariente, B., Gonzalez-Rodriguez, M., and Lanvin, D. F. (2015). Towards an automatic user profiling system for online information sites: Identifying demographic determining factors. *Online Information Review*, 39(1):61–80.

[3] Bi, B., Shokouhi, M., Kosinski, M., and Graepel, T. (2013). Inferring the demographics of search users: Social data meets search queries. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 131–140, New York, NY, USA. ACM.

[4] Biswas, P., Aydemir, G. A., Langdon, P., and Godsill, S. (2013). Intent recognition using neural networks and kalman filters. In *Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data*, pages 112–123. Springer.

[5] Bremmers, L. (2011). What behavior predicts banner effectiveness? : a mouse and eye tracking study.

[6] Cahill, D. J. (1997). Target marketing and segmentation: valid and useful tools for marketing. *Management Decision*, 35(1):10–13.

[7] Caruana, R. and Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM.

[8] Castillejo, E., Almeida, A., and de Ipiña, D. L. (2014). Modelling users, context and devices for adaptive user interface systems. *International Journal of Pervasive Computing and Communications*, 10(1):69–91.

[9] Castillejo, E., Almeida, A., and López-de Ipiña, D. (2013). User, context and device modeling for adaptive user interface systems. In Urzaiz, G., Ochoa, S., Bravo, J., Chen, L., and Oliveira, J., editors, *Ubiquitous Computing and Ambient Intelligence. Context-Awareness and Context-Driven Interaction*, volume 8276 of *Lecture Notes in Computer Science*, pages 94–101. Springer International Publishing.

[10] Clancy, K. J. and Lou Roberts, M. (1984). Toward an optimal market target: a strategy for market segmentation. *Journal of Consumer Marketing*, 1(1):64–73.

[11] Coley, A. and Burgess, B. (2003). Gender differences in cognitive and affective impulse buying. *Journal of Fashion Marketing and Management: An International Journal*, 7(3):282–295.

[12] Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM.

[13] De Bock, K. and Van den Poel, D. (2010). Predicting website audience demographics forweb advertising targeting using multi-website clickstream data. *Fundamenta Informaticae*, 98(1):49–70.

[14] Dibb, S. and Simkin, L. (1991). Targeting, segments and positioning. *International Journal of Retail & Distribution Management*, 19(3).

[15] Epp, C., Lippold, M., and Mandryk, R. L. (2011). Identifying emotional states using keystroke dynamics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 715–724. ACM.

[16] FitzGerald, M. and Arnott, D. (1996). Understanding demographic effects on marketing communications in services. *International Journal of Service Industry Management*, 7(3):31–45.

[17] Godoy, D., Schiaffino, S., and Amandi, A. (2010). Integrating user modeling approaches into a framework for recommender agents. *Internet Research*, 20(1):29–54.

[18] Harald, S., Daniel, G.-A., Panagiotis, T. M., Eni, M., Markus, S., and Peter, G. (2013). The power of prediction with social media. *Internet Research*, 23(5):528–543.

[19] Hsu, S. H., Huang, C. C., Tsuang, Y. H., and Sun, J. S. (1999). Effects of age and gender on remote pointing performance and their design implications. *International Journal of Industrial Ergonomics*, 23(5):461–471.

[20] Hu, J., Zeng, H.-J., Li, H., Niu, C., and Chen, Z. (2007). Demographic prediction based on user's browsing behavior. In *Proceedings of the 16th international conference on World Wide Web*, pages 151–160. ACM.

[21] Jansen, B. J., Moore, K., and Carman, S. (2013). Evaluating the performance of demographic targeting using gender in sponsored search. *Information Processing & Management*, 49(1):286–302.

[22] Jorgensen, Z. and Yu, T. (2011). On mouse dynamics as a behavioral biometric for authentication. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 476–482. ACM.

[23] Kaminsky, R., Enev, M., and Andersen, E. (2008). Identifying game players with mouse biometrics. *University of Washington, Tech. Rep.*

[24] Karuppan, C. M. (2001). Web-based teaching materials: a user's profile. *Internet Research*, 11(2):138–149.

[25] Lamche, B., Pollok, E., Wörndl, W., and Groh, G. (2014). Evaluating the effectiveness of stereotype user models for recommendations on mobile devices. In *Proceedings of the Joint Workshop on Personalized Information Access (PIA 2014), in conjunction with the 22nd conference on User Modeling, Adaptation and Personalization (UMAP 2014).*

[26] Langley, P. (1999). *User modeling in adaptive interface.* Springer.

[27] Liu, Z. and Huang, X. (2008). Gender differences in the online reading environment. *Journal of Documentation*, 64(4):616–626.

[28] Lobo, J. M., Jiménez-Valverde, A., and Real, R. (2008). Auc: a misleading measure of the performance of predictive distribution models. *Global ecology and Biogeography*, 17(2):145–151.

[29] MacKenzie, I. S. and Buxton, W. (1992). Extending fitts' law to two-dimensional tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 219–226. ACM.

[30] Miguel-Hurtado, O., Stevenage, S. V., Bevan, C., and Guest, R. (2016). Predicting sex as a soft-biometrics from device interaction swipe gestures. *Pattern Recognition Letters*, 79:44–51.

[31] Mitchell, V.-W. and Boustani, P. (2015). The effects of demographic variables on measuring perceived risk. In *Proceedings of the 1993 Academy of Marketing Science (AMS) Annual Conference*, pages 663–669. Springer.

[32] Monrose, F. and Rubin, A. D. (2000). Keystroke dynamics as a biometric for authentication. *Future Generation computer systems*, 16(4):351–359.

[33] Murray, D. and Durrell, K. (2000). Inferring demographic attributes of anonymous internet users. In *Web Usage Analysis and User Profiling*, pages 7–20. Springer.

[34] Muthumari, G., Shenbagaraj, R., and Blessa Binolin Pepsi, M. (2014). Mouse gesture based authentication using machine learning algorithm. In *Advanced Communication Control and Computing Technologies (ICACCCT), 2014 International Conference on*, pages 492–496. IEEE.

[35] Nensel, M. (2016). A4a: Vacations eclipse business trips as primary us air travel purpose. Accessed: 2016-06-03.

[36] Pentel, A. (2015). Patterns of confusion: Using mouse logs to predict user's emotional state. *Personalization Approaches in Learning Environments.*

[37] Pol, L. G. (1986). Marketing and the demographic perspective. *Journal of Consumer Marketing*, 3(1):57–65.

[38] Pooler, J. (2004). Demographic targeting: The essential role of population groups in retail marketing. *Journal of Consumer Marketing*, 21(1):67–69.

[39] PSFK (2015). In artificial intellignece, anything is possible. Accessed: 2015-11-03.

[40] Rich, E. (1989). Stereotypes and user modeling. In Kobsa, A. and Wahlster, W., editors, *User Models in Dialog Systems*, Symbolic Computation, pages 35–51. Springer Berlin Heidelberg.

[41] Rohr, L. E. (2006). Gender-specific movement strategies using a computer-pointing task. *Journal of motor behavior*, 38(6):431–137.

[42] Schler, J., Koppel, M., Argamon, S., and Pennebaker, J. W. (2006). Effects of age and gender on blogging. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, volume 6, pages 199–205.

[43] Sharma, P. and Batra, M. V. (2016). A study of demographic determinants of online shopping behaviour of consumers. *International Education and Research Journal*, 2(3).

[44] Shen, C., Cai, Z., Guan, X., Du, Y., and Maxion, R. A. (2013). User authentication through mouse dynamics. *IEEE Transactions on Information Forensics and Security*, 8(1):16–30.

[45] Srivastava, J., Cooley, R., Deshpande, M., and Tan, P.-N. (2000). Web usage mining: Discovery and applications of usage patterns from web data. *ACM SIGKDD Explorations Newsletter*, 1(2):12–23.

[46] Tolia, N., Andersen, D. G., and Satyanarayanan, M. (2006). Quantifying interactive user experience on thin clients. *Computer*, (3):46–52.

[47] Tsimperidis, I., Katos, V., and Clarke, N. (2015). Language-independent gender identification through keystroke analysis. *Information and Computer Security*, 23(3):286–301.

[48] White, R. W., Bailey, P., and Chen, L. (2009). Predicting user interests from contextual information. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 363–370. ACM.

[49] Wright, M. (1996). The dubious assumptions of segmentation and targeting. *Management Decision*, 34(1):18–24.

[50] Yamauchi, T. and Bowman, C. (2014). Mining cursor motions to find the gender, experience, and feelings of computer users. In *Data Mining Workshop (ICDMW), 2014 IEEE International Conference on*, pages 221–230. IEEE.

[51] Yamauchi, T., Seo, J. H., Jett, N., Parks, G., and Bowman, C. (2015). Gender differences in mouse and cursor movements. *International Journal of Human-Computer Interaction*, 0(ja):null.

[52] Yoon, C.-H. and Kim, D. D. (2005). User authentication based on behavioral mouse dynamics biometrics.

[53] Zhang, B., Dai, H., Zeng, H., Qi, L., Najm, T., Mah, T., Shipunov, V., Li, Y., and Chen, Z. (2007). Predicting demographic attributes based on online behavior. US Patent App. 11/366,526.

[54] Zhang, H.-R. and Min, F. (2016). Three-way recommender systems based on random forests. *Knowledge-Based Systems*, 91:275–286.

[55] Zheng, N., Paloski, A., and Wang, H. (2016). An efficient user verification system using angle-based mouse movement biometrics. *ACM Transactions on Information and System Security (TISSEC)*, 18(3):11.

[56] Zukerman, I. and Albrecht, D. W. (2001). Predictive statistical models for user modeling. *User Modeling and User-Adapted Interaction*, 11(1):5–18.

APPENDIX A

Appendix: Code

This section contains all the code that was used to process, collect and store data. This code is also available online at http://www.stijnhoogervorst.nl/thesis/Code

A.1 Appendix: Data Preprocessing POC (Python)

../Code/poc/process_data.py

```python
# This script expects mousedata, currently in the form of  a .m2s file with the following syntax:
##   action values
# where action is a number according to the following classification:
## 1 : position
## 2 : Mouse action
## 3 : Key press
## 4 : Key release
## 5 : delay
# The value says something about what happens and depends on the action.
## 1 : "x:.. y:.." coordinates
## 2 : "Left Down" | "Left Up" | "Right Down" | "Right Up" | "Scroll Up" | "Scroll Down"
## 3 : normalkey | {Shift}, {Right}, {Backspace}, {...}
## 4 : same as 3
## 5 : time (ms)
#
# Using this information, we will create a csv document that contains for each dataset the following
    features a mean and a standard deviation:
# id ["distance", "x velocity", "y velocity", "x acceleration", "y acceleration", "path length", "path
    time",
#            "leftclick interval", "rightclick interval", "leftclick length", "rightclick length", "scroll
     interval",
#            "keypress length", "keypress interval"]
# where clickrate is the time between two different clicks and clicklength is the time between mouse down
    and mouse up


import csv, os, sys, getopt, numpy, math, re

inputdir = ''
outputfile = ''
rawdir = ''
raw = False
normalized = False
genders = [""] * 81
features = ["distance", "x velocity", "y velocity", "x acceleration", "y acceleration", "tangental
    acceleration",
            "path length", #"path time",
            "leftclick interval", "rightclick interval",
            "leftclick length", "rightclick length", "scroll interval", "precision",
            "keypress length", "keypress interval"]
derived_features = ["id", "subid", "gender"]
static_features = ["westsouthwest", "southsouthwest", "southsoutheast", "eastsoutheast", "eastnortheast",
              "northnortheast", "northnorthwest", "westnorthwest", "leftclick total", "rightclick total",
               "keypress total", "idle time", "left double clicks"]
for feat in features:
    derived_features.append(feat + " mean")
    derived_features.append(feat + " stdev")
    derived_features.append(feat + " min")
    derived_features.append(feat + " max")

derived_features += static_features
```

```

def getAngle(angle):
    res = -1
    stepsize = math.pi / 4

    # Angle: positive = upper half, negative = lower half (-1,0) being -pi and 0

    if(angle < -1 * math.pi + stepsize):
        res = 0
    elif(angle < -1 * math.pi + 2 * stepsize):
        res = 1
    elif(angle < -1 * math.pi + 3 * stepsize):
        res = 2
    elif(angle < -1 * math.pi + 4 * stepsize):
        res = 3
    elif(angle < -1 * math.pi + 5 * stepsize):
        res = 4
    elif(angle < -1 * math.pi + 6 * stepsize):
        res = 5
    elif(angle < -1 * math.pi + 7 * stepsize):
        res = 6
    elif(angle <= -1 * math.pi + 8 * stepsize):
        res = 7
    else:
        print('dit is niet de bedoeling')

    return static_features[res]

def createGenderArray(dir):
    # create the array with the genders
    file = open(dir + "/genders.txt", 'r')

    # the first line contains no information we need
    for line in file:
        s_line = line.split('\t')
        s_id = int(s_line[0])
        s_gender = re.sub('[^A-Za-z0-9: ]+', '', s_line[1].rstrip('\n'))
        genders[s_id] = s_gender



def writeresult(result, writer):
    res = []
    for feat in derived_features:
        res.append(result[feat])
    writer.writerow(res)

# Only write to results if the time constraint was reached (i.e. the user session is split in equal length
     sessions)
def doWrite(id, counter, distributions, calculatedfeatures, writer):
    print "this should be called only once per file. Current file {}".format(id)


    for x in range(len(distributions['x velocity'])-1):
        distributions['x acceleration'].append(distributions['x velocity'][x+1] - distributions['x
    velocity'][x])
        distributions['y acceleration'].append(distributions['y velocity'][x+1] - distributions['y
    velocity'][x])
        distributions['tangental acceleration'].append(distributions['distance'][x+1] - distributions['
    distance'][x])



    # if the raw flag was specified, create a csv file for each of the mouse data id's and
    # create a copy of the population data in there
    if raw:
        maxlength = max({k: len(v) for k, v in distributions.iteritems()}.values())
        rawfile = open(rawdir + '/' + id + '_raw.csv', 'wb')
        raw_writer = csv.writer(rawfile, delimiter=';')
        raw_writer.writerow(features)
        for k in range(1, maxlength):
            row = []
            for j in features:
                try: row.append(distributions[j][k])
                except LookupError: row.append('')
            raw_writer.writerow(row)


    result = {"id": id, "subid" : id + "-" + str(counter), "gender" : genders[int(id)]}
    # calculate the derived features from the sets here
    for feat in features:
        if len(distributions[feat])>1:
            arr = numpy.array(distributions[feat])
            # If we want to normalize the array to -1 and 1 here already instead of in the mining
            # we pass the --normalized flag
            # This is linear range normalization
            if(normalized):
                arrmin = numpy.min(arr)
                arrmax = numpy.max(arr)
                arr = (arr - arrmin)**(2/(arrmax-arrmin)) - 1
            mean = numpy.mean(arr, axis=0)
            stdev = numpy.std(arr, axis=0)
            minimum = min(arr)
            maximum = max(arr)

        else:
            mean = ''
            stdev = ''

        result[feat + " min"] = minimum
        result[feat + " max"] = maximum
        result[feat + " mean"] = mean
        result[feat + " stdev"] = stdev


    result.update(calculatedfeatures)
    writeresult(result, writer)

def createDistributions(input_dir, input_file, writer):
    id = os.path.splitext(input_file)[0]


    file = open(input_dir + "/" + input_file, 'r')

    # the first line contains no information we are using
    file.readline()

    counter = 0
    write = True
    print "this should be called only once per file before entering the while loop. Current file {}".
    format(id)
    while write:
        write = False

        # distributions will contain all the derived values from the dataset
        distributions = {}
        for x in features:
            distributions[x] = []
        pathlength = 0.0
        pathtime = 0.0
        totaltime = 0.0
        lastleftclick = 0.0
        lastrightclick = 0.0
        lastleftdown = 0.0
        lastrightdown = 0.0
        lastscroll = 0.0
        coords = [0,0]
        lastcoords = list(coords)
        keys = {}
        lastkeypress = 0.0
        lastclickposition = [0,0]
        distancebetweenclicks = 0
        calculatedfeatures = dict(zip(static_features, [0] * len(static_features)))

        for line in file:
            s_line = line.split('\t')
            action = s_line[0]
            value = re.sub('[^A-Za-z0-9: ]+', '', s_line[1].rstrip('\n'))

            # 1 : position
            if action == '1':

                temp_coords = value.split(' ')
                coords[0] = int(temp_coords[0].split(':')[1])
                coords[1] = int(temp_coords[1].split(':')[1])

                # add computed values to the distributions
                tempx = coords[0] - lastcoords[0]
                tempy = coords[1] - lastcoords[1]

                distributions["x velocity"].append(float(tempx))
                distributions["y velocity"].append(float(tempy))

                distance = round(math.hypot(tempx, tempy))

                distributions["distance"].append(distance)
                distancebetweenclicks += distance

                # Get the direction of the movement and append a score to that direction
                angle = math.atan2(tempy, tempx)
                calculatedfeatures[getAngle(angle)]+=1


                pathlength+=distance

                lastcoords = list(coords)

            # 2 : Mouse action
            elif action == '2':
                if value == "Left Down":
                    lastleftdown = totaltime
                elif value == "Left Up":
                    #same as left click interval
                    #distributions["path time"].append(totaltime - pathtime)
                    distributions["path length"].append(pathlength)
                    distributions["leftclick interval"].append(totaltime - lastleftclick)
                    distributions["leftclick length"].append(totaltime - lastleftdown)
                    if (totaltime - lastleftclick) < 500:
                        calculatedfeatures["left double clicks"] += 1
                    lastleftclick = totaltime

                    # update the precision and reset the path
                    actualdistance = (math.hypot(lastcoords[0] - lastclickposition[0], lastcoords[1] -
    lastclickposition[1]))
                    if actualdistance > 0:
                        distributions["precision"].append(distancebetweenclicks / actualdistance)
                    lastclickposition = lastcoords
                    distancebetweenclicks = 0

                    calculatedfeatures["leftclick total"] += 1
                    pathtime = totaltime
                    pathlength = 0
                elif value == "Right Down":
                    lastrightdown = totaltime
                elif value == "Right Up":
                    distributions["rightclick interval"].append(totaltime - lastrightclick)
                    distributions["rightclick length"].append(totaltime - lastrightdown)
                    lastrightclick = totaltime
                    calculatedfeatures["rightclick total"]+=1

                elif value == "Scroll Down" or value == "Scroll Up":
                    distributions["scroll interval"].append(totaltime - lastscroll)
                    lastscroll = totaltime
                else:
                    pass
            # 3 : Key press
            elif action == '3':
                distributions["keypress interval"].append(totaltime - lastkeypress)
                keys[value] = totaltime
            # 4 : Key release
            elif action == '4':
                if value in keys: distributions["keypress length"].append(totaltime - keys[value])
                lastkeypress = totaltime
                calculatedfeatures["keypress total"]+=1
            # 5 : delay
            elif action == '5':
                totaltime += int(value)
                calculatedfeatures["idle time"]+= int(value)
            else:
                print('unknown action encountered: ' + action)

            # We assume that after each entry, 15 ms passes
            totaltime += 15

            # split the documents by time passed
            if totaltime >= 1 * 60 * 1000: # currently set to 1 minute
                counter += 1
                write = True
                doWrite(id, counter, distributions, calculatedfeatures, writer)
                break #Break out of line by line loop of the file

    print "this should be called only once per file after the while loop."
    doWrite(id, counter, distributions, calculatedfeatures, writer)
```

```


# proces input options

def main(argv):

    global normalized

    try:
        opts, args = getopt.getopt(argv, "hi:o:r:", ["idir=", "ofile=", "raw=", "normalized"])
    except getopt.GetoptError:
        print 'process_data.py -i <inputdirectory> -o <outputfile> -r <rawdirectory>'
        sys.exit(2)
    for opt, arg in opts:
        if opt == '-h':
            print '''process_data.py -i <inputdirectory> -o <outputfile>  -r
      if left blank, assume working in the current directory. Default outputfilename is mousedata.csv. Run
     this script to convert a directory of
      .m2s files into a single CSV dataset describing for each file the velocities in x and y direction,
    acceleration, path lengths, clickrates and scrollspeeds. If -r or --raw flag is given, also create a
    directory with a file with the raw data for each file. NOTE: Currently, the program expects
      the input file names to be the id value.
     '''
            sys.exit()
        elif opt in ("-i", "--idir"):
            inputdir = arg
        elif opt in ("-o", "--ofile"):
            outputfile = arg
        elif opt in ("-r", "--raw"):
            global raw
            raw = True
            global rawdir
            rawdir = arg
            if (not os.path.exists(rawdir)):
                os.makedirs(rawdir)
        elif opt in ("--normalized"):
            normalized = True

    print 'Beginning processing files'

    outfile = open(outputfile, 'wb')
    raw_writer = csv.writer(outfile, delimiter=';')
    raw_writer.writerow(derived_features)

    createGenderArray(inputdir)

    for file in os.listdir(inputdir):
        if file.endswith(".m2s"):
            print "Working on file: {} \r".format(file),
            createDistributions(inputdir, file, raw_writer)

if __name__ == "__main__":
    main(sys.argv[1:])
```

../Code/poc/process_data_goms.py

```
# This script expects mousedata, currently in the form of  a .m2s file with the following syntax per line:
##   action values
# where action is a number according to the following classification:
## 1 : position
## 2 : Mouse action
## 3 : Key press
## 4 : Key release
## 5 : delay
# The value says something about what happens and depends on the action.
## 1 : "x:.. y:.." coordinates
## 2 : "Left Down" | "Left Up" | "Right Down" | "Right Up" | "Scroll Up" | "Scroll Down"
## 3 : key | {Shift}, {Right}, {Backspace}, {...}
## 4 : same as 3
## 5 : time (ms)
#
# Using this information, we will create a csv document that contains for each dataset the following
    features a mean and a standard deviation:
# id ["distance", "x velocity", "y velocity", "x acceleration", "y acceleration", "path length", "path
    time",
#          "leftclick interval", "rightclick interval", "leftclick length", "rightclick length", "scroll
     interval",
#          "keypress length", "keypress interval"]
# where clickrate is the time between two different clicks and clicklength is the time between mouse down
    and mouse up
```

```

# These features are taken under the assumption that they describe the mouse motions of a user to some
    extent. We assume each of the features to be described more or less by a normal distribution.

import csv, os, sys, getopt, numpy, math, re

inputdir = ''
outputfile = ''
rawdir = ''
raw = False
normalized = False
genders = [""] * 81
features = ["pac", "dad", "tx", "te", "ms"]
derived_features = ["id", "subid", "gender"]
for feat in features:
    derived_features.append(feat + " mean")
    derived_features.append(feat + " stdev")
    derived_features.append(feat + " min")
    derived_features.append(feat + " max")

def getAngle(angle):
    res = -1
    stepsize = math.pi / 4

    # Angle: positive = upper half, negative = lower half (-1,0) being -pi and 0

    if(angle < -1 * math.pi + stepsize):
        res = 0
    elif(angle < -1 * math.pi + 2 * stepsize):
        res = 1
    elif(angle < -1 * math.pi + 3 * stepsize):
        res = 2
    elif(angle < -1 * math.pi + 4 * stepsize):
        res = 3
    elif(angle < -1 * math.pi + 5 * stepsize):
        res = 4
    elif(angle < -1 * math.pi + 6 * stepsize):
        res = 5
    elif(angle < -1 * math.pi + 7 * stepsize):
        res = 6
    elif(angle <= -1 * math.pi + 8 * stepsize):
        res = 7
    else:
        print('dit is niet de bedoeling')

    return static_features[res]


def createGenderArray(dir):
    # create the array with the genders
    file = open(dir + "/genders.txt", 'r')

    # the first line does not contain actual information
    for line in file:
        s_line = line.split('\t')
        s_id = int(s_line[0])
        s_gender = re.sub('[^A-Za-z0-9: ]+', '', s_line[1].rstrip('\n'))
        genders[s_id] = s_gender


def writeresult(result, writer):
    res = []
    for feat in derived_features:
        res.append(result[feat])
    writer.writerow(res)

# Only write to results if the time constraint was reached
def doWrite(id, counter, distributions, calculatedfeatures, writer):
    print "this should be called only once per file. Current file {}".format(id)

    # if the raw flag was specified, create a csv file for each of the mouse data id's and
    # create a copy of the population data in there
    if raw:
        maxlength = max({k: len(v) for k, v in distributions.iteritems()}.values())
        rawfile = open(rawdir + '/' + id + '_raw.csv', 'wb')
        raw_writer = csv.writer(rawfile, delimiter=';')
        raw_writer.writerow(features)
        for k in range(1, maxlength):
            row = []
            for j in features:
                try: row.append(distributions[j][k])
                except LookupError: row.append('')
            raw_writer.writerow(row)


    result = {"id": id, "subid" : id + "-" + str(counter), "gender" : genders[int(id)]}
    # calculate the derived features from the sets here
    for feat in features:
        if len(distributions[feat])>1:
            arr = numpy.array(distributions[feat])
            # If we want to normalize the array to -1 and 1 here already instead of in the mining
            # we pass the --normalized flag
            if(normalized):
                arrmin = numpy.min(arr)
                arrmax = numpy.max(arr)
                arr = (arr - arrmin)**(2/(arrmax-arrmin)) - 1
            mean = numpy.mean(arr, axis=0)
            stdev = numpy.std(arr, axis=0)
            minimum = min(arr)
            maximum = max(arr)

        else:
            mean = ''
            stdev = ''
            minimum = ''
            maximum = ''

        result[feat + " min"] = minimum
        result[feat + " max"] = maximum
        result[feat + " mean"] = mean
        result[feat + " stdev"] = stdev


    writeresult(result, writer)

def createDistributions(input_dir, input_file, writer):
    id = os.path.splitext(input_file)[0]


    file = open(input_dir + "/" + input_file, 'r')

    # the first line contains nonsense
    file.readline()

    counter = 0
    write = True
    while write:
        write = False

        # distributions will contain all the derived values from the dataset
        distributions = {}
        for x in features:
            distributions[x] = []

        totaltime = 0.0
        lastleftclick = 0.0
        lastleftdown = 0.0
        coords = [0,0]

        # needed to calculate Drag and Drop actions
        start_drag = [0,0]
        drag_threshold = 5 # mouse must move at least this number of pixels to count as drag

        # needed to calculate text editing actions
        typing = False
        start_typing = 0.0
        type_time = 0.0
        word_interval_threshold = 2000

        for line in file:
            s_line = line.split('\t')
            action = s_line[0]
            value = re.sub('[^A-Za-z0-9: ]+', '', s_line[1].rstrip('\n'))

            addTime = 15

            # 1 : position
            if action == '1':

                temp_coords = value.split(' ')
                coords[0] = int(temp_coords[0].split(':')[1])
                coords[1] = int(temp_coords[1].split(':')[1])
```

```python

            # 2 : Mouse action
            elif action == '2':
                if value == "Left Down":
                    lastleftdown = totaltime
                    start_drag = coords[:]
                    dragging = True
                elif value == "Left Up":
                    dragging = False
                    if math.sqrt((coords[0] - start_drag[0])**2 + (coords[1] - start_drag[1])**2 ) > drag_threshold:
                        # the mouse has dragged more than the threshold, measure a drag operation
                        distributions["dad"].append(totaltime - lastleftdown)
                    else:
                        # it was a regular click interaction
                        distributions["pac"].append(totaltime - lastleftclick)

                    lastleftclick = totaltime

                else:
                    pass
            # 3 : Key press
            elif action == '3':
                # if a space has been pressed or it has been too long since the last keypress, we start a new word
                if totaltime - start_typing > word_interval_threshold:
                    typing = False
                    distributions["te"].append(type_time)

                if not typing:
                    start_typing = totaltime
                    typing = True

                type_time = totaltime - start_typing

                if  value == " ":
                    typing = False
                    distributions["te"].append(type_time)
            # 4 : Key release
            elif action == '4':
                pass
            # 5 : delay
            elif action == '5':
                addTime = int(value)
            else:
                print('unknown action encountered: ' + action)

            # After each entry (except for if there was a wait action), 15 ms passes
            totaltime += addTime

            # split the documents by time passed
            if totaltime >= 5 * 60 * 1000:
                counter += 1
                write = True
                doWrite(id, counter, distributions, [], writer)
                break #Break out of line by line loop of the file

    #print "this should be called only once per file after the while loop."
    #doWrite(id, counter, distributions, [], writer)


# Process input options

def main(argv):

    global normalized

    try:
        opts, args = getopt.getopt(argv, "hi:o:r:", ["idir=", "ofile=", "raw=", "normalized"])
    except getopt.GetoptError:
        print 'process_data.py -i <inputdirectory> -o <outputfile> -r <rawdirectory>'
        sys.exit(2)
    for opt, arg in opts:
        if opt == '-h':
            print '''process_data.py -i <inputdirectory> -o <outputfile>  -r
      if left blank, assume working in the current directory. Default outputfilename is mousedata.csv. Run this script to convert a directory of
      .m2s files into a single CSV dataset describing for each file the distributions of the times for point and click, drag and drop and text editing actions.
     '''
            sys.exit()
        elif opt in ("-i", "--idir"):
            inputdir = arg
        elif opt in ("-o", "--ofile"):
            outputfile = arg
        elif opt in ("-r", "--raw"):
            global raw
            raw = True
            global rawdir
            rawdir = arg
            if (not os.path.exists(rawdir)):
                os.makedirs(rawdir)
        elif opt in ("--normalized"):
            normalized = True

    print 'Beginning processing files'

    outfile = open(outputfile, 'wb')
    raw_writer = csv.writer(outfile, delimiter=';')
    raw_writer.writerow(derived_features)

    createGenderArray(inputdir)

    for file in os.listdir(inputdir):
        if file.endswith(".m2s"):
            print "Working on file: {} \r".format(file),
            createDistributions(inputdir, file, raw_writer)

if __name__ == "__main__":
    main(sys.argv[1:])
```

A.2 Appendix: Data Collection Framework (Javascript)

The collected and processed data is also available online at http://www.stijnhoogervorst.nl/thesis/Data

../Code/thesis_tracking_lib.js

```
/**
 * Last edited by sjhoo on 28/6/2016.
 */

/**
 * This library is meant to perform tracking of cursor motions. It requires jQuery to work.
 * It tracks the mouse position client side at regular intervals and calculates features
 * of how a click was performed. This library is loaded through a DimML loader file
 * (for more about DimML, see: http://dimml.io)
 */

var _adv_predict = _adv_predict || {};


_adv_predict.MouseTracking = function(mainObject) {

    if(_adv_predict.hasRan) return;
    _adv_predict.hasRan = true;

    var debugging = mainObject["debugging"] === true;
    var performance = {
        start_time : Date.now()
    };

    var recordsString = "";

    // Adapted from: http://www.quirksmode.org/js/cookies.html
    var createCookie = function(name, value, days) {
        var expires;

        if (days) {
            var date = new Date();
            date.setTime(date.getTime() + (days * 24 * 60 * 60 * 1000));
            expires = "; expires=" + date.toGMTString();
        } else {
            expires = "";
        }
        document.cookie = encodeURIComponent(name) + "=" + encodeURIComponent(value) + expires + "; path=/
    ";
    };

    var readCookie = function (name) {
        var nameEQ = encodeURIComponent(name) + "=";
        var ca = document.cookie.split(';');
        for (var i = 0; i < ca.length; i++) {
            var c = ca[i];
            while (c.charAt(0) === ' ') c = c.substring(1, c.length);
            if (c.indexOf(nameEQ) === 0) return decodeURIComponent(c.substring(nameEQ.length, c.length));
        }
        return null;
    };

    var eraseCookie = function(name) {
        createCookie(name, "", -1);
    };

    // returns speed in bytes/second to the callback
    var getConnectionSpeed = function(cb) {

        var speed = readCookie('con_s');
        if ($.isNumeric(speed)) return cb(speed);

        var testsize = mainObject['speed_test_size'];
        var url = mainObject['speed_test_url'];
        var starttime = (new Date()).getTime();
        $.get(url + "?t=" + starttime, function (res) {
            var endtime = (new Date()).getTime();
            speed = testsize / ((endtime - starttime)/1000); // bytes / s
            createCookie('con_s', speed, 1/(60*24)); // save for 1 minute
            cb(speed);
        });
    };

    var log = function(arg){
        if(debugging){
            console.log('[adv_predict]: ' + arg);
        }
    };

    getConnectionSpeed(function(speed){
        var MBs = (speed/(1000*1000));
        log('connection speed estimate (MB/s): ' + MBs.toFixed(2));

        if(speed < .5) mainObject['fast_connection'] = false;

    });

    log("initialize tracking library");

    var isTouchPad;
    var eventCount = 0;
    var eventCountStart;

    // Adapted from: http://derickbailey.com/2014/09/21/calculating-standard-deviation-with-array-map-and-
    array-reduce-in-javascript/
    var average = function(data){
        var sum = data.reduce(function(sum, value){
            return sum + value;
        }, 0);

        return sum / data.length;
    };

    var distribution = function(values) {
        var avg = average(values);

        var squareDiffs = values.map(function (value) {
            var diff = value - avg;
            return diff * diff;
        });

        var avgSquareDiff = average(squareDiffs);

        return {
            'mean': avg,
            'stdev': Math.sqrt(avgSquareDiff)
        };
    }


    /**
     * This function determines whether the user is using a mouse or a trackpad
     *
     * Based on the answer from: http://stackoverflow.com/questions/10744645/detect-touchpad-vs-mouse-in-
    javascript
     *
     * @param evt
     */
    var mouseHandle = function (evt) {
        var isTouchPadDefined = isTouchPad || typeof isTouchPad !== "undefined";
        if (!isTouchPadDefined) {
            if(isTouchPad == null || typeof isTouchPad === "undefined") {

                if (eventCount === 0) {
                    eventCountStart = new Date().getTime();
                }

                eventCount++;

                if (new Date().getTime() - eventCountStart > 50) {
                    if (eventCount > 5) {
                        isTouchPad = true;
                    } else {
                        isTouchPad = false;
                    }
                    createCookie('touchpad', isTouchPad, 1 / 24);
                }

            }
            else{
                isTouchPadDefined = true;
            }
        }
    };

    document.addEventListener("mousewheel", mouseHandle, false);
    document.addEventListener("DOMMouseScroll", mouseHandle, false);


    mainObject = mainObject || {};


    var elements = [];
    var records = [];
    var calculated = [];


    // For support, if not defined, define Math.hypot
    Math.hypot = Math.hypot || function(x, y){ return Math.sqrt(x*x + y*y) };


    var getId = function(){
        var id = readCookie('prediction_id');
        if(id !== null){
            return id;
        }

        var chars = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXTZabcdefghiklmnopqrstuvwxyz";

        id = "_pred_";
        for (var i = 0; i < 20; i++){
            id += chars.charAt(Math.floor(Math.random() * chars.length));
        }

        createCookie('prediction_id', id, 1);
        return id;
    };

    var custom_features = mainObject["custom_features"] || {};

    var id = mainObject['visitor_id'] || getId();

    var predictions = {
        'id': id,
        'custom_features': custom_features
    };


    var point_and_click_elements = mainObject["point_and_click_elements"] || [];


    var isOverElement = function(tocheck, elem){
        var $this = $(elem);
        var offset = $this.offset();
        var width = $this.outerWidth();
        var height = $this.outerHeight();

        return (tocheck.x > offset.left && tocheck.x < offset.left + width && tocheck.y > offset.top &&
    tocheck.y < offset.top + height);
    };

    var getCenter = function(elem){
        var $this = $(elem);
        var offset = $this.offset();
        var width = ($.isNumeric($this.outerWidth())) ? $this.outerWidth() : $this.width();
        var height = ($.isNumeric($this.outerHeight())) ? $this.outerHeight() : $this.height();

        return {'x': offset.left + width / 2 , 'y':offset.top + height / 2};
    };

    var cursorX;
    var cursorY;
    document.onmousemove = function(e){
        cursorX = e.pageX;// + $(window).scrollLeft();
        cursorY = e.pageY;// + $(window).scrollTop();

    };

    var getAngle = function(vector1, vector2){

        return Math.atan2(vector1.y, vector1.x) - Math.atan2(vector2.y, vector2.x);

    };



    /**
     * Start measuring only once first cursor position is known.

     * At every step (20 ms interval), measure these things:
     * timestamp, position, speed, direction, acceleration, distance since last point
     *
     */

    var interval = 20;
    var is_tracking = false;

    // semi-blocking wait until mouseposition is known
    var start_tracking = function(){
        if(cursorX == null || cursorY == null){
            setTimeout(start_tracking, 100);
        }
        else {is_tracking = true}
    };

    start_tracking();

    var tracking = setInterval(function(){
        var calc = {
            'distance':0,
            'angle':0,
            'speed':0,
            'acceleration':0,
            'vector' : {'x' : 0, 'y' : 0}
        };
        var result = {
            'position': { 'x':cursorX, 'y':cursorY},
            'timestamp': Date.now()
        };
        if( records.length>0){
            var previous = records[records.length-1];
            var previous_calc = calculated[records.length-1];
            calc['distance'] = getDistance(result.position, previous.position);
            calc['angle'] = getAngle(result.position, previous.position);
            calc['speed'] = (calc.distance / (result.timestamp - previous.timestamp));
            calc['acceleration'] = ((calc.speed - previous_calc.speed) / (result.timestamp - previous.
    timestamp));
            calc['vector'] = {
                x: result.position.x - previous.position.x,
                y: result.position.y - previous.position.y
            };

        }

        if(is_tracking){
            records.push(result);
            calculated.push(calc);
            if(calc.speed > 0) recordsString += result.timestamp + " " + result.position.x + " " + result.
    position.y + "\n"; // filter none movements
        }

    }, interval);

    var getDistance= function(pos1, pos2){
        return Math.hypot(pos1.x-pos2.x, pos1.y-pos2.y);
    };


    // This function was used in debugging and places a circle on a specified point on the screen
    var add_token = function(position, id, color, radius){

        if(!debugging) return;

        radius = radius || 20;
        color = color || "blue";

        $('#'+id).remove();

        var $div = $("<div>", {id: id, class: "token", css: {
            position: "absolute",
            marginLeft: 0, marginTop: 0,
            top: position.y, left: position.x,
            width: radius, height: radius, "border-radius":"50%",
            background:color
            }
        });

        $('body').append($div);
    };

    var calculateMeasurements = function(event, goal){
        if(records.length==0) return;

        is_tracking = false;
        var clicktime = Date.now();
        var clickX = event.pageX;
        var clickY = event.pageY;



        var getDistanceBetweenAngles = function(angle1, angle2){
            var temp_d = (angle1>angle2)?angle1-angle2 : angle2 - angle1;
            temp_d %= Math.PI * 2;
            return (temp_d > Math.PI) ? 2 * Math.PI - temp_d : temp_d;
        };

        var standing_still = false;
        var standing_still_timestamp;

        var isIntentionStart = function(goal, instant, calculated){

            // if the cursor moves less than 100 pixels per second for at least 300ms, it is basically stationary
            // we assume the last stationary point to be the starting point. Also, if it is standing still over the
            // target button, it does not count as restarting the movement from there
            if(calculated.speed < 100/1000 && !isOverElement(instant.position, goal['domid'])){
                if(!standing_still){
                    standing_still = true;
                    standing_still_timestamp = instant.timestamp;
                }

                return standing_still_timestamp - instant.timestamp >= 300;
            }

            standing_still = false;

            return standing_still;

        };

        var peak_velocity = 0;
        var start_timestamp = clicktime;
        var peak_velocity_timestamp = clicktime;
        var peak_velocity_point = $.extend({}, goal.position);
        var total_distance = 0;
        var starting_point = $.extend({}, goal.position);
        var over_button_timestamp = clicktime;
        var overshoot_x, overshoot_y;
        var max_x = max_y = -1;
        var min_x = min_y = 999999999;

        var angles = [];
        var accelerations = [];
        var speeds = [];
        var time_angle = {
            'towards':0,
            'side':0,
            'away':0
        };


        // traverse the records backwards until we find the moment we intended to start
        var instant;
        for(var i = records.length-1; i >= 0 && !isIntentionStart(goal, records[i], calculated[i]); i--){
            instant = records[i];
            start_timestamp = instant.timestamp;
            starting_point = instant.position;

            // calculate peak velocity
            if (Math.abs(calculated[i].speed) > peak_velocity){
                peak_velocity = Math.abs(calculated[i].speed);
                peak_velocity_timestamp = instant.timestamp;
                peak_velocity_point = instant.position;
            }

            // Calculate time on button
            if( isOverElement(instant.position, goal['domid'])){
                over_button_timestamp = instant.timestamp;
            }

            max_x = Math.max(instant.position.x, max_x);
            max_y = Math.max(instant.position.y, max_y);
            min_x = Math.min(instant.position.x, min_x);
            min_y = Math.min(instant.position.y, min_y);

            total_distance += calculated[i].distance;

            accelerations.push(calculated[i]['acceleration']);
            speeds.push(calculated[i]['speed']);
            var vector_from_previous_to_goal = {
              'x' : goal.position.x - records[Math.max(0,i-1)].position.x,
              'y' : goal.position.y - records[Math.max(0,i-1)].position.y
            };

            if(calculated[i]['speed'].toFixed(2) > 0) {
                var angle_to_goal = Math.abs(getAngle(calculated[i]['vector'], vector_from_previous_to_goal));
                if (angle_to_goal > Math.PI) angle_to_goal = Math.abs(angle_to_goal - 2 * Math.PI);
                angles.push(angle_to_goal);

                // Add the time in the direction
                var direction;
                if (angle_to_goal >= 0 && angle_to_goal < Math.PI / 4) direction = 'towards';
                else if (angle_to_goal < Math.PI / 2) direction = 'side';
                else direction = 'away';

                time_angle[direction] += instant.timestamp - records[Math.max(0, i - 1)].timestamp;
            }

        }
        overshoot_x = Math.max(0,(goal.position.x - starting_point.x < 0) ? goal.position.x - min_x : max_x - goal.position.x);
        overshoot_y = Math.max(0,(goal.position.y - starting_point.y < 0) ? goal.position.y - min_y : max_y - goal.position.y);

        if(max_x == -1 || min_x == 999999999) overshoot_x = 0;
        if(max_y == -1 || min_y == 999999999) overshoot_y = 0;


        var actual_distance = getDistance(goal.position, starting_point);

        var movement_time = clicktime - start_timestamp;
        var average_speed = (total_distance / movement_time);
        var path_accuracy = (actual_distance / total_distance);
        var time_after_peak_velocity = clicktime - peak_velocity_timestamp;
        var velocity_at_click = (getDistance(goal.position, records[records.length-1].position) / ( clicktime - records[records.length-1].timestamp));

        // we take precision as 1 minus the ratio of the distance from the center to the maximum distance from the center
        var precision = 1 - Math.hypot(goal['position']['x'] - clickX, goal['position']['y'] - clickY) / ( Math.hypot(goal['width'], goal['height'])/2) ;

        // for debugging (only placed if 'debugging'==true)
        add_token(starting_point, 'start', "green");
        add_token(peak_velocity_point, 'peak', "orange");
        add_token({'x':clickX, 'y':clickY}, 'end', "red");

        // if we are logging all movement and there is a fast connection, perform send_all
        if(mainObject['log_all'] === true && mainObject['fast_connection'] === true) send_all(goal);

        calculated = [];
        records = [];
        is_tracking = true;

        // We use Fitts's law Index of Performance (with MacKenzie's Index of Difficulty) using the smallest side as the width of the target as
        // per http://www.billbuxton.com/fitts92.html
        var performance_index = (Math.log2(actual_distance / Math.min(goal['width'], goal['height']) + 1)) / movement_time;

        var speed = distribution(speeds);
        var acceleration = distribution(accelerations);
        var angle = distribution(angles);

        var result = {
            'id':predictions.id,
            'element_path':goal['path'],
            'element':goal['pacid'],
            'precision':precision.toFixed(6),
            'performance_index': performance_index.toFixed(6),
            'actual_distance':actual_distance,
            'total_distance':total_distance,
            'average_speed':average_speed.toFixed(6),
            'movement_time':movement_time,
            'path_accuracy':path_accuracy.toFixed(6),
            'time_on_button':(clicktime - over_button_timestamp),
            'peak_velocity':peak_velocity.toFixed(6),
            'time_after_peak_velocity':time_after_peak_velocity,
            'element_position_x':goal['position'].x,
            'element_position_y':goal['position'].y,
            'element_width':goal['width'],
            'element_height':goal['height'],
            'velocity_at_click':velocity_at_click.toFixed(6),
            'overshoot_x' : overshoot_x,
            'overshoot_y' : overshoot_y,
            'time_towards' : time_angle['towards'],
            'time_side' : time_angle['side'],
            'time_away' : time_angle['away'],
            'speed_mean' : speed.mean,
            'speed_stdev' : speed.stdev,
            'acceleration_mean' : acceleration.mean,
            'acceleration_stdev': acceleration.stdev,
            'angle_mean' : angle.mean,
            'angle_stdev': angle.stdev
        };

        result['touchdevice'] = ('ontouchstart' in window);
        result['url']= window.location.href.slice(0);
        if(typeof(isTouchPad)=="boolean"){
            result['touchpad']= isTouchPad;
        }
        else{
            result['touchpad']= (readCookie('touchpad') == null) ? "unknown" : readCookie('touchpad');
        }

        result['browser']= navigator.userAgent;

        return result;

    };

    var send_data = function(data){

    dimml.event('process_features', flatten(data));

    };


    var nth_ocurrence_from_back = function(str, needle, nth) {
        for (var i=str.length;i>=0;i--) {
            if (str.charAt(i) == needle) {
                if (!--nth) {
                    return i;
                }
            }
        }
        return -1;
    };

    var send_all = function(goal){
        // Do not send all the position data if it is more than 20 seconds worth of browsing for
    performance reasons
        if(records.length > 50 * 10) recordsString.slice(Math.max(nth_ocurrence_from_back(recordsString,"\
    n", 50 * 10)+1,0));

        $.post('//dynamic.dimml.io/flow/code.js?dimml.concept='+mainObject['log_all_url']+'&dimml.cors', {
    'id':predictions.id,
            'element_path':goal['path'], 'clicked':goal['pacid'], 'position_x':goal['position'].x, '
    position_y':goal['position'].y, 'width':goal['width'],
            'height':goal['height'], 'url':window.location.href, 'records': recordsString});

    };

    // This code was adapted from:
    // http://stackoverflow.com/questions/19098797/fastest-way-to-flatten-un-flatten-nested-json-objects
    var flatten = function(data){

```

```
    var result = {};
    function _recurse (cur, prop) {
        if (Object(cur) !== cur) {
            result[prop] = cur;
        } else if (Array.isArray(cur)) {
            for(var i=0, l=cur.length; i<l; i++)
                _recurse(cur[i], prop ? prop+"."+i : ""+i);
            if (l == 0)
                result[prop] = [];
        } else {
            var isEmpty = true;
            for (var p in cur) {
                isEmpty = false;
                _recurse(cur[p], prop ? prop+"."+p : p);
            }
            if (isEmpty)
                result[prop] = {};
        }
    }
    _recurse(data, "");
    return result;
};


/**
 * When losing focus (when switched to another tab for example), destroy the current mouse track records. Otherwise, after long absence, the delay may be quite big
 */
$(window).blur(function(){
    is_tracking = false;
});

window.onfocus = (function() {
    is_tracking = true;
    records = [];
    calculated = [];
});

var getFullPath = function(el){
   return $(el)
        .parentsUntil('body')
        .andSelf()
        .map(function(i, val) {
            return val.nodeName + '[' + $(val).index() + ']';
        }).get().join('>');
};

_adv_predict.updateListeners = function () {

    //since page views do not guarantee that the page is done loading, give it 500 ms
    setTimeout(function() {

        log("updateListeners called");
        // ==================================================== \\
        // ======   Registered Point and Click Elements  ====== \\
        // ==================================================== \\

        for (var i in point_and_click_elements) {
            if (point_and_click_elements.hasOwnProperty(i)) {
                var pac_selector = point_and_click_elements[i];

                if ($(pac_selector).length == 0) continue;

                $.each($(pac_selector), function (ix, pac) {
                    $(pac).unbind('click.predict');
                    $(pac).attr('cpredict-measured', 'true');
                    var new_pac = {};
                    new_pac['path'] = getFullPath(pac);
                    new_pac['domid'] = pac;
                    new_pac['pacid'] = pac_selector;
                    new_pac['position'] = getCenter(pac);
                    new_pac['width'] = ($.isNumeric($(pac).outerWidth()) ? $(pac).outerWidth() : $(pac).width();
                    new_pac['height'] =($.isNumeric($(pac).outerHeight()) ? $(pac).outerHeight() : $(pac).height();

                    var original_click_function = $(pac).onclick;
                    $(pac).onclick = null;
                    var executeOnce = true;
                    (function (elem, id, execute_function) {

                            $(elem['domid']).bind("click.predict", function (event) {
                                if(!executeOnce) return;
                                executeOnce = false;

                                performance.click_time = Date.now();
                                var res = calculateMeasurements(event, elem);
                                if(res && mainObject['fast_connection'] === true) send_data(res);
                                // because buttons may move, and we need to update what we know (such as the center)
                                _adv_predict.updateListeners();
                                log('performance delay when clicking: ' + (Date.now() - performance.click_time ));
                            });
                            if (typeof execute_function === "function") $(elem['domid']).click(execute_function);

                        })(new_pac, i, original_click_function);
                        elements.push(new_pac);
                    });
                }
            }
        }, 1000);
    };
    _adv_predict.updateListeners();

    log('Performance - initialization time ' + (Date.now() - performance.start_time));


};

// wait for jquery to load
if(typeof(window._adv_predict && _adv_predict.MouseTracking) === "function" && window._adv_predict.config){
    var counter = 20;
    var waitForJquery = function(){
        if(!window.jQuery && counter > 0){
            counter--;
            setTimeout(waitForJquery, 500)
        }
        else{
            if(window.jQuery) _adv_predict.MouseTracking(window._adv_predict.config);
            if(counter == 0)  log("Could not load tracking framework. Missing jQuery dependency");
        }
    };
    waitForJquery();
};
```

A.3 Appendix: Data Storage Solution (DimML)

../Code/research/main.dimml

```
import 'lib/functions.dimml'
import 'lib/trackmouse.dimml'
import 'lib/storedata.dimml'

const FEATURE_HEADER=''


// This concept has to extend Logmouse and be references in the config
// By extending it here, we make sure that all files get writen to the right server
concept LogAll extends 'lib/Logmouse' {

}
```

../Code/research/lib/functions.dimml

```
def isTop = `
  try {
    if (window.self === window.top) {
      return '1';
    }
  } catch(e) {}

  return '0';
`

def getVisitId = `
  return getValue('o2vtId', function() {
    return randomId();
  }, 0);
`

def getReferrerHost = `
  return !document.referrer?"":new dimml.uri(document.referrer).host();
`

def getValue = { name, fun, lifetime => `
  var value = getCookie(name);

  if (!value) {
    value = fun();

    if (value !== null) {
      setCookie(name, value, lifetime);
    }
  }

  return value;
` }

def getCookie = { name => `
  if (!String.prototype.trim) {
    (function(){
      var rtrim = /^[\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g;
      String.prototype.trim = function () {
        return this.replace(rtrim, "");
      }
    })();
  }
  var i, l, x, y, cookies = document.cookie.split(';');

  for (i = 0, l = cookies.length; i < l; i++) {
    x = cookies[i].substr(0, cookies[i].indexOf('='));
    y = cookies[i].substr(cookies[i].indexOf('=') + 1);

    if (x.trim() == name) {
      return unescape(y);
    }
  }

  return null;
` }

def setCookie = { name, value, lifetime => `
    var d = new Date();
  var e = '';

  if (lifetime > 0) {
    d.setSeconds(d.getSeconds() + lifetime);
    e = '; expires=' + d.toUTCString();
  }

  document.cookie = name + '=' + escape(value) + e + '; path=/';
` }

def randomId = `
  return 'xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx'.replace(
    /[xy]/g,
    function(c) {
      var r = Math.random() * 16 | 0;
      var v = c == 'x' ? r : (r & 0x3 | 0x8);
      return v.toString(16);
    }
  )
`

def getBannerId = `
  if (typeof dimml.telegraaf !== 'undefined' && typeof dimml.telegraaf.bannerid !== 'undefined' && dimml.
    telegraaf.bannerid != -1) {
    return {bannerid: dimml.telegraaf.bannerid};
  } else {
    return false;
  }
`

def addScript = {src, callback => `

  if(window.jQuery){
    $.getScript(src, callback);
  }
  else{
    var s,
        r,
        t;
      r = false;
      s = document.createElement('script');
      s.type = 'text/javascript';
      s.src = src;
      s.onload = s.onreadystatechange = function() {
        //console.log( this.readyState ); //uncomment this line to see which ready states are called.
        if ( !r && (!this.readyState || this.readyState == 'complete') )
        {
          r = true;
          callback();
        }
      };
      t = document.getElementsByTagName('script')[0];
      t.parentNode.insertBefore(s, t);

    }
  `}

// This code was adapted from:
// http://stackoverflow.com/questions/19098797/fastest-way-to-flatten-un-flatten-nested-json-objects
def flattenObject = {data =>`
  var result = {};
    function _recurse (cur, prop) {
        if (Object(cur) !== cur) {
            result[prop] = cur;
        } else if (Array.isArray(cur)) {
             for(var i=0, l=cur.length; i<l; i++)
                 _recurse(cur[i], prop ? prop+"."+i : ""+i);
            if (l == 0)
                result[prop] = [];
        } else {
            var isEmpty = true;
            for (var p in cur) {
                isEmpty = false;
                _recurse(cur[p], prop ? prop+"."+p : p);
            }
            if (isEmpty)
                result[prop] = {};
        }
    }
    _recurse(data, "");
    return result;
  `@dimml
```

```
}
```

../Code/research/lib/storedata.dimml

```
concept ProcessFeatures {

  flow
    => filter['id','element','nr_passengers','element_path','labels_gender','labels_age', 'movement_time',
    'peak_velocity', 'precision',
    'time_after_peak_velocity','actual_distance','total_distance','performance_index','time_on_button','
    path_accuracy',
    'average_speed','velocity_at_click','overshoot_x','overshoot_y','time_towards','time_side','time_away'
    ,'speed_mean','speed_stdev','acceleration_mean',
    'acceleration_stdev','angle_mean','angle_stdev', 'touchdevice','touchpad','url', 'browser', '
    element_position_x', 'element_position_y','element_height', 'element_width' ]
    => csv
    => time[?filename="origin:'thesis-train-'yyyyMMddHH'.log'"]
    => sftp[
      sync='true',
         host="^^SFTP_HOST",
         user="^^SFTP_USER",
         password="^^SFTP_PASSWORD",
         dir="^^SFTP_DIR",
         header="^^FEATURE_HEADER"
      ]

  plugin debug
}

concept Logmouse {

  //const FEATURE_HEADER = "'id','element','visitor_id','element_path','labels_gender','labels_age','
    movement_time','peak_velocity','precision','time_after_peak_velocity','total_distance','
    performance_index','time_on_button','path_accuracy','average_speed','velocity_at_click', 'touchdevice
    ','touchpad','url', 'browser'\n"
  //const FEATURE_HEADER=''

  // for debugging (if sftp does not work because ip blocked)
  //const FTP_DIR = ''thesis/all_movements/' + (new Date().format('yyyyMMdd'))`@groovy


  flow
    => select[`id != null`@groovy]
    => code[filename=`(new Date().format('yyyyMMddHH'))+"-${id}-all.log"`@groovy]
    => property[?filename = 'filename']
    => code[serialized = `getOutput(records, clicked, element_path, position_x, position_y, width, height,
     url)`@groovy]
    => nop(writeToCSV)


  flow (writeToCSV)
    => filter['serialized']
      => sftp[
       //sync='true',
         host="^^SFTP_HOST",
         user="^^SFTP_USER",
         password="^^SFTP_PASSWORD",
         dir="^^SFTP_DIR_ALL"
      ]

    def getOutput = {data, clicked_id, path, position_x, position_y, width, height, url => `
      data = "${data}clicked url|${url} pacid|${clicked_id} path|${path} position|${position_x},${
    position_y} width|${width} height|${height}"
      return data
    `@groovy}

}
```

../Code/research/lib/trackmouse.dimml

```
import 'storedata.dimml'

concept TrackMouse {

  // client-side data collection framework gets loaded from external source
  // It needs to be called afterwards with the right configuration object

  // This is the configuration for the cursor tracking library. It needs variably declared labels, a
    static list of elements
  // val config = ``

  // It does not load on mobile and only sends limited data on slow connections (very basic checks for
    those are in place)

  // Additional load check: onlyl for single tickets
    script '
    window._adv_predict = window._adv_predict || {};
  window._adv_predict.loaded = window._adv_predict.loaded || false;
  window._adv_predict.load = function(){

    var attempts = 2;

       if(!window._adv_predict.loaded){
        if(!dimml.consent || (dimml.consent && dimml.consent.get() > 1)){

          var isMobile = window.matchMedia("only screen and (max-width: 760px)");

          if (!isMobile.matches){
                addScript("https://research-stijn.adversitement.com/adv_mouse_predict.js", function(){
                  if(window._adv_predict.config.debugging) console.log("[adv_predict]: script loaded")
                })
                window._adv_predict.loaded = true;
                }
              }

    }
    if(attempts > 0){
      attempts--;
      setTimeout(window._adv_predict.load, 1000);
    }
  }
    window._adv_predict.load();
  '


}

def $preMatch = {doMatch => '
    var handle = setInterval(function(){
        if (window.jQuery) {
            clearInterval(handle);
            doMatch();
        }
    },100);
'}
```

A.4 Appendix: Research Loader airline (DimML)

../Code/airline–research–loader.dimml

```
import '../researchStijn/main.dimml'

concept AddMouseTracking extends '../research/lib/TrackMouse' {
  match '*'

  script `

  window._adv_predict = window._adv_predict || {};
  window._adv_predict.config = {
            'point_and_click_elements':[
              '.button'
              ],
            'debugging':false,
            'log_all':false,
            'speed_test_url':"https://research-stijn.adversitement.com/img_speed.jpg",
            'speed_test_size':597677
          }

  ^^super
          `@javascript


  on process_features include 'AddLabels','../research/lib/ProcessFeatures'

}

// This concept adds client-specific features to every set of features of a click
concept AddLabels {

  val nr_passengers = `
    if(!tmParam) return null;
    return parseInt(tmParam['nr_adults']) + parseInt(tmParam['nr_children']) + parseInt(tmParam['nr_babies
    ']);
  `

  val labels_gender = `
        // if not on ticket page, try to get last label value
        var g;
        var _g = getCookie('pred_label_g');
        if($('#MaleGender0').length > 0){
          g = $('input[name="newBookingPassengers[0].Gender"]:checked').val();
        }
        g =  g || _g || "";
        setCookie('pred_label_g',g,60*60*24*2); //2 days

        return g;
        `@javascript

  val labels_age = `
        // if not on ticket page, try to get last label value
        var a;
        var _a = getCookie('pred_label_a');
        if($('input[name="newBookingPassengers[0].DateOfBirth.Year"]').length > 0){
          a = $('input[name="newBookingPassengers[0].DateOfBirth.Year"]').val();
        }

        a = ($.isNumeric(a)) ? a = new Date().getFullYear() - a : _a || "";
        setCookie('pred_label_a',a,60*60*24*2); //2 days

        return a;
        `@javascript

}
```

A.5 Appendix: Research Loader Health Insurers (DimML)

../Code/insurer−research−loader.dimml

```
import '../../researchStijn/main.dimml'

concept AddMouseTracking extends '../../research/lib/TrackMouse' {
  match '*'

  script `

  window._adv_predict = window._adv_predict || {};
  window._adv_predict.config = {
            'point_and_click_elements':[
              'a[data-toggle-state-target!="body"]',
              '.btn'
              ],
            'debugging':true,
            'log_all':false,
            'speed_test_url':"https://research-stijn.adversitement.com/img_speed.jpg",
            'speed_test_size':597677
          }
  ^^super
          `@javascript


  on process_features include 'AddLabels','../../research/lib/ProcessFeatures'

}

// This concept adds client-specific features to every set of features of a click
concept AddLabels {


  val labels_gender = `
        // try to get last label value
        var g;
        var _g = getCookie('pred_label_g');

        if (wa && wa.verzekerden && wa.verzekerden[0] && wa.verzekerden[0].geslacht && (typeof(wa.
    verzekerden[0].geslacht)!="undefined")){
          g = wa.verzekerden[0].geslacht;
        }
        else if (wa && wa.verzekerdobject && wa.verzekerdobject[0] && wa.verzekerdobject[0].objecten
          && wa.verzekerdobject[0].objecten[0] && wa.verzekerdobject[0].objecten[0].geslacht && typeof(wa.
    verzekerdobject[0].objecten[0].geslacht)!= "undefined")
        {
          g = wa.verzekerdobject[0].objecten[0].geslacht
        } else if ($('#persoongegevensHoofdverzekerde>div>div.form-part-content>div.output-info-wrap>div.
    left>table>tbody>tr:nth-child(4)>td.content-dice').length>0){
            g = $('#persoongegevensHoofdverzekerde>div>div.form-part-content>div.output-info-wrap>div.left
    >table>tbody>tr:nth-child(4)>td.content-dice').text().replace(/( |\n)/g,'');
          }
        g =  g || _g || "";
        setCookie('pred_label_g',g,60*60*24*2); //2 days

        return g;
        `@javascript

  val labels_age = `
        // if not on ticket page, try to get last label value
        var a;
        var _a = getCookie('pred_label_a');

        if (wa && wa.verzekerden && wa.verzekerden[0] && wa.verzekerden[0].leeftijd && (typeof(wa.
    verzekerden[0].leeftijd)!="undefined")){
          a = wa.verzekerden[0].leeftijd;
        }
        else if (wa && wa.verzekerdobject && wa.verzekerdobject[0] && wa.verzekerdobject[0].objecten
          && wa.verzekerdobject[0].objecten[0] && wa.verzekerdobject[0].objecten[0].leeftijd && typeof(wa.
    verzekerdobject[0].objecten[0].leeftijd)!= "undefined")
        {
          a = wa.verzekerdobject[0].objecten[0].leeftijd
        }
        else if( $('#persoongegevensHoofdverzekerde>div>div.form-part-content>div.output-info-wrap>div.
    left>table>tbody>tr:nth-child(3)>td.content-dice').length>0){
          var year = $('#persoongegevensHoofdverzekerde>div>div.form-part-content>div.output-info-wrap>div
    .left>table>tbody>tr:nth-child(3)>td.content-dice').text();
          a = new Date().getFullYear() - parseInt(year.split('-').slice(-1)[0])
        }

        a = ($.isNumeric(a)) ? a : _a || "";
        setCookie('pred_label_a',a,60*60*24*2); //2 days

        return a;
        ‘@javascript

}
```

A.6 Appendix: Data Preprocessing (Python)

../Code/ProcessRaw.py

```python
import csv, os, sys, getopt

feature_header = ['id','element','nr_passengers','element_path','labels_gender','labels_age', '
    movement_time','peak_velocity', 'precision', 'time_after_peak_velocity','actual_distance','
    total_distance','performance_index','time_on_button','path_accuracy','average_speed','
    velocity_at_click','overshoot_x','overshoot_y','time_towards','time_side','time_away','speed_mean','
    speed_stdev','acceleration_mean','acceleration_stdev','angle_mean','angle_stdev', 'touchdevice','
    touchpad','url', 'browser', 'element_position_x', 'element_position_y','element_height', '
    element_width']

# MAIN FUNCTION, PROCESS OPTIONS

def main(argv):

    try:
        opts, args = getopt.getopt(argv, "hi:o:r:s:", ["idir=", "odir=", "raw=", "source="])
    except getopt.GetoptError:
        print 'ProcessRaw.py -i <inputdirectory> -o <outputfile> -r <rawdirectory> -s <source>'
        sys.exit(2)

    outdir = 'processed'

    for opt, arg in opts:
        if opt == '-h':
            print '''ProcessRaw.py -i <inputdirectory> -o <outputfile>  -r -s <source:transavia/vgz>
            if left blank, assume working in the current directory. Default outputfiledir is processed/
    merged_output.log. Run this script to convert a directory of
            .log files containing click data for users into a single CSV dataset. If a label is available
    for one/some of the click data, use the latest one for all
            previous clicks.
         '''
            sys.exit()
        elif opt in ("-i", "--idir"):
            inputdir = arg
        elif opt in ("-o", "--odir"):
            outdir = arg
            if (not os.path.exists(outdir)):
                os.makedirs(outdir)
        elif opt in ("-s", "--source"):
            if (arg == "vgz"):
                if 'nr_passengers' in feature_header: feature_header.remove('nr_passengers')

    print 'Beginning processing files'

    outputfile = outdir+'/merged_output.log'

    outfile = open(outputfile, 'wb')
    writer = csv.writer(outfile, delimiter='|')
    writer.writerow(feature_header)
    mapping_writer = csv.writer(open(outdir+'/mapping.file', 'wb'), delimiter='|')

    id_label_map = {}

    # First, create a dictionary that contains all the labels for observed cases. We will only use those
    # cases later on

    for filename in os.listdir(inputdir):
        if filename.endswith(".log"):
            # These files can contain 100.000's rows, so the whole process may take a while...

            print "Getting labels from file: {} \r".format(filename)

            file = open(inputdir + "/" + filename, 'rb')
            csvfile = csv.reader((l.replace(',"', '|"') for l in file), delimiter='|')


            try:
                for i, line in enumerate(csvfile):
                    if(len(line) == len(feature_header)):
                        id = line[0].strip('"')
                        gender = line[3].strip('"?')
                        age = line[4].strip('"')
                        touchpad = line[17].strip('"')
                        if (id and gender and age and len(id) == 26 and gender!="undefined" and gender!="
    Onbekend" and age!="NaN") :
                            id_label_map[id] = {
                                'id':id,
                                'gender' : gender,
                                'age' : age,
                                'touchpad' : touchpad
                            }
                        print "\r Genders found of {}".format(len(id_label_map))
            except csv.Error:
                print "Error reading line {} \r".format(i)
                pass
    # Once we have the masterlist of all subjects for which we have labels, we aggregate their features into the outputfile

    for m in id_label_map:
        mapping_writer.writerow([id_label_map[m]['id'],id_label_map[m]['gender'],id_label_map[m]['age'],id_label_map[m]['touchpad']])

    for filename in os.listdir(inputdir):
        if filename.endswith(".log"):

            print "Processing features of file: {} \r".format(filename)

            file = open(inputdir + "/" + filename, 'rb')
            csvfile = csv.reader((l.replace(',"', '|"') for l in file), delimiter='|')

            try:
                for i, line in enumerate(csvfile):
                    if(len(line) == len(feature_header)):

                        line = [x.strip('"') for x in line]

                        id = line[0]

                        if(id in id_label_map):
                            line[3] = id_label_map[id]['gender']
                            line[4] = id_label_map[id]['age']
                            line[17] = id_label_map[id]['touchpad']
                            writer.writerow(line)

            except csv.Error:
                print "Error reading line {} \r".format(id)
                pass
if __name__ == "__main__":
    main(sys.argv[1:])
```

Appendix B

Appendix: Data Preprocessing (Python)

../Code/ProcessRaw.py

```
import csv, os, sys, getopt

feature_header = ['id','element','nr_passengers','element_path','labels_gender','labels_age', '
    movement_time','peak_velocity', 'precision', 'time_after_peak_velocity','actual_distance','
    total_distance','performance_index','time_on_button','path_accuracy','average_speed','
    velocity_at_click','overshoot_x','overshoot_y','time_towards','time_side','time_away','speed_mean','
    speed_stdev','acceleration_mean','acceleration_stdev','angle_mean','angle_stdev', 'touchdevice','
    touchpad','url', 'browser', 'element_position_x', 'element_position_y','element_height', '
    element_width']

# MAIN FUNCTION, PROCESS OPTIONS

def main(argv):

    try:
        opts, args = getopt.getopt(argv, "hi:o:r:s:", ["idir=", "odir=", "raw=", "source="])
    except getopt.GetoptError:
        print 'ProcessRaw.py -i <inputdirectory> -o <outputfile> -r <rawdirectory> -s <source>'
        sys.exit(2)

    outdir = 'processed'

    for opt, arg in opts:
        if opt == '-h':
            print '''ProcessRaw.py -i <inputdirectory> -o <outputfile>  -r -s <source:transavia/vgz>
            if left blank, assume working in the current directory. Default outputfiledir is processed/
    merged_output.log. Run this script to convert a directory of
            .log files containing click data for users into a single CSV dataset. If a label is available
    for one/some of the click data, use the latest one for all
            previous clicks.
         '''
            sys.exit()
        elif opt in ("-i", "--idir"):
            inputdir = arg
        elif opt in ("-o", "--odir"):
            outdir = arg
            if (not os.path.exists(outdir)):
                os.makedirs(outdir)
        elif opt in ("-s", "--source"):
            if (arg == "vgz"):
                if 'nr_passengers' in feature_header: feature_header.remove('nr_passengers')

    print 'Beginning processing files'

    outputfile = outdir+'/merged_output.log'

    outfile = open(outputfile, 'wb')
    writer = csv.writer(outfile, delimiter='|')
    writer.writerow(feature_header)
    mapping_writer = csv.writer(open(outdir+'/mapping.file', 'wb'), delimiter='|')

    id_label_map = {}

    # First, create a dictionary that contains all the labels for observed cases. We will only use those
    # cases later on

    for filename in os.listdir(inputdir):
        if filename.endswith(".log"):
            # These files can contain 100.000's rows, so the whole process may take a while...
```

```python

            print "Getting labels from file: {} \r".format(filename)

            file = open(inputdir + "/" + filename, 'rb')
            csvfile = csv.reader((l.replace(',"', '|"') for l in file), delimiter='|')


            try:
                for i, line in enumerate(csvfile):
                    if(len(line) == len(feature_header)):
                        id = line[0].strip('"')
                        gender = line[3].strip('"?')
                        age = line[4].strip('"')
                        touchpad = line[17].strip('"')
                        if (id and gender and age and len(id) == 26 and gender!="undefined" and gender!="Onbekend" and age!="NaN") :
                            id_label_map[id] = {
                                'id':id,
                                'gender' : gender,
                                'age' : age,
                                'touchpad' : touchpad
                            }
                        print "\r Genders found of {}".format(len(id_label_map))
            except csv.Error:
                print "Error reading line {} \r".format(i)
                pass
    # Once we have the masterlist of all subjects for which we have labels, we aggregate their features into the outputfile

    for m in id_label_map:
        mapping_writer.writerow([id_label_map[m]['id'],id_label_map[m]['gender'],id_label_map[m]['age'],id_label_map[m]['touchpad']])

    for filename in os.listdir(inputdir):
        if filename.endswith(".log"):

            print "Processing features of file: {} \r".format(filename)

            file = open(inputdir + "/" + filename, 'rb')
            csvfile = csv.reader((l.replace(',"', '|"') for l in file), delimiter='|')

            try:
                for i, line in enumerate(csvfile):
                    if(len(line) == len(feature_header)):

                        line = [x.strip('"') for x in line]

                        id = line[0]

                        if(id in id_label_map):
                            line[3] = id_label_map[id]['gender']
                            line[4] = id_label_map[id]['age']
                            line[17] = id_label_map[id]['touchpad']
                            writer.writerow(line)

            except csv.Error:
                print "Error reading line {} \r".format(id)
                pass
if __name__ == "__main__":
    main(sys.argv[1:])
```

B.1 Appendix: Data Preprocessing Distributions (Python)

../Code/ProcessRawDistributions.py

```python
import csv, os, sys, getopt, numpy, math

# for airline, add these after element: 'nr_passengers',
feature_header = ['id','element','nr_passengers','element_path','labels_gender','labels_age', 'movement_time','peak_velocity', 'precision', 'time_after_peak_velocity','actual_distance','total_distance','performance_index','time_on_button','path_accuracy','average_speed','velocity_at_click','overshoot_x','overshoot_y','time_towards','time_side','time_away','speed_mean','speed_stdev','acceleration_mean','acceleration_stdev','angle_mean','angle_stdev', 'touchdevice','touchpad','url', 'browser', 'element_position_x', 'element_position_y','element_height', 'element_width']

# MAIN FUNCTION, PROCESS OPTIONS

def main(argv):

    try:
        opts, args = getopt.getopt(argv, "hi:o:r:s:", ["idir=", "odir=", "raw=", "source=", "normalized"])
    except getopt.GetoptError:
        print 'ProcessRaw.py -i <inputdirectory> -o <outputfile> -r <rawdirectory> -s <source>'
        sys.exit(2)

    outdir = 'processed'

    for opt, arg in opts:
        if opt == '-h':
            print '''ProcessRaw.py -i <inputdirectory> -o <outputfile>  -r -s <source:transavia/vgz>
            if left blank, assume working in the current directory. Default outputfiledir is processed/merged_output.log. Run this script to convert a directory of
            .log files containing click data for users into a single CSV dataset. If a label is available for one/some of the click data, use the latest one for all
            previous clicks.
         '''
            sys.exit()
        elif opt in ("-i", "--idir"):
            inputdir = arg
        elif opt in ("-o", "--odir"):
            outdir = arg
            if (not os.path.exists(outdir)):
                os.makedirs(outdir)
        elif opt in ("-s", "--source"):
            if (arg == "vgz"):
                if 'nr_passengers' in feature_header: feature_header.remove('nr_passengers')
        elif opt in ("--normalized"):
            normalized = True

    print 'Beginning processing files'

    outputfile = outdir+'/merged_output.log'

    outfile = open(outputfile, 'wb')
    writer = csv.writer(outfile, delimiter='|')
    writer.writerow(feature_header)
    mapping_writer = csv.writer(open(outdir+'/mapping.file', 'wb'), delimiter='|')

    id_label_map = {}

    # First, create a dictionary that contains all the labels for observed cases. We will only use those
    # cases later on

    for filename in os.listdir(inputdir):
        if filename.endswith(".log"):
            # These files can contain 100.000's rows, so the whole process may take a while...

            print "Getting labels from file: {} \r".format(filename)

            file = open(inputdir + "/" + filename, 'rb')
            csvfile = csv.reader((l.replace(',"', '|"') for l in file), delimiter='|')


            try:
                for i, line in enumerate(csvfile):
                    if(len(line) == len(feature_header)):
                        id = line[0].strip('"')
                        gender = line[3].strip('"?')
                        age = line[4].strip('"')
                        touchpad = line[17].strip('"')
                        if (id and gender and age and len(id) == 26 and gender!="undefined" and gender!="Onbekend" and age!="NaN") :
                            id_label_map[id] = {
                                'id':id,
                                'gender' : gender,
                                'age' : age,
                                'touchpad' : touchpad
                            }
                        print "\r Genders found of {}".format(len(id_label_map))
            except csv.Error:
                print "Error reading line {} \r".format(i)
                pass
    # Once we have the masterlist of all subjects for which we have labels, we aggregate their features into the outputfile

    for m in id_label_map:
        mapping_writer.writerow([id_label_map[m]['id'],id_label_map[m]['gender'],id_label_map[m]['age'],id_label_map[m]['touchpad']])

    for filename in os.listdir(inputdir):
        if filename.endswith(".log"):

            print "Processing features of file: {} \r".format(filename)

            file = open(inputdir + "/" + filename, 'rb')
            csvfile = csv.reader((l.replace(',"', '|"') for l in file), delimiter='|')

            try:
                for i, line in enumerate(csvfile):
                    if(len(line) == len(feature_header)):

                        line = [x.strip('"') for x in line]

                        id = line[0]

                        if(id in id_label_map):
                            line[3] = id_label_map[id]['gender']
                            line[4] = id_label_map[id]['age']
                            line[17] = id_label_map[id]['touchpad']
                            writer.writerow(line)

            except csv.Error:
                print "Error reading line {} \r".format(id)
                pass
if __name__ == "__main__":
    main(sys.argv[1:])
```

Appendix C

Appendix: RapidMiner flow Proof of Concept

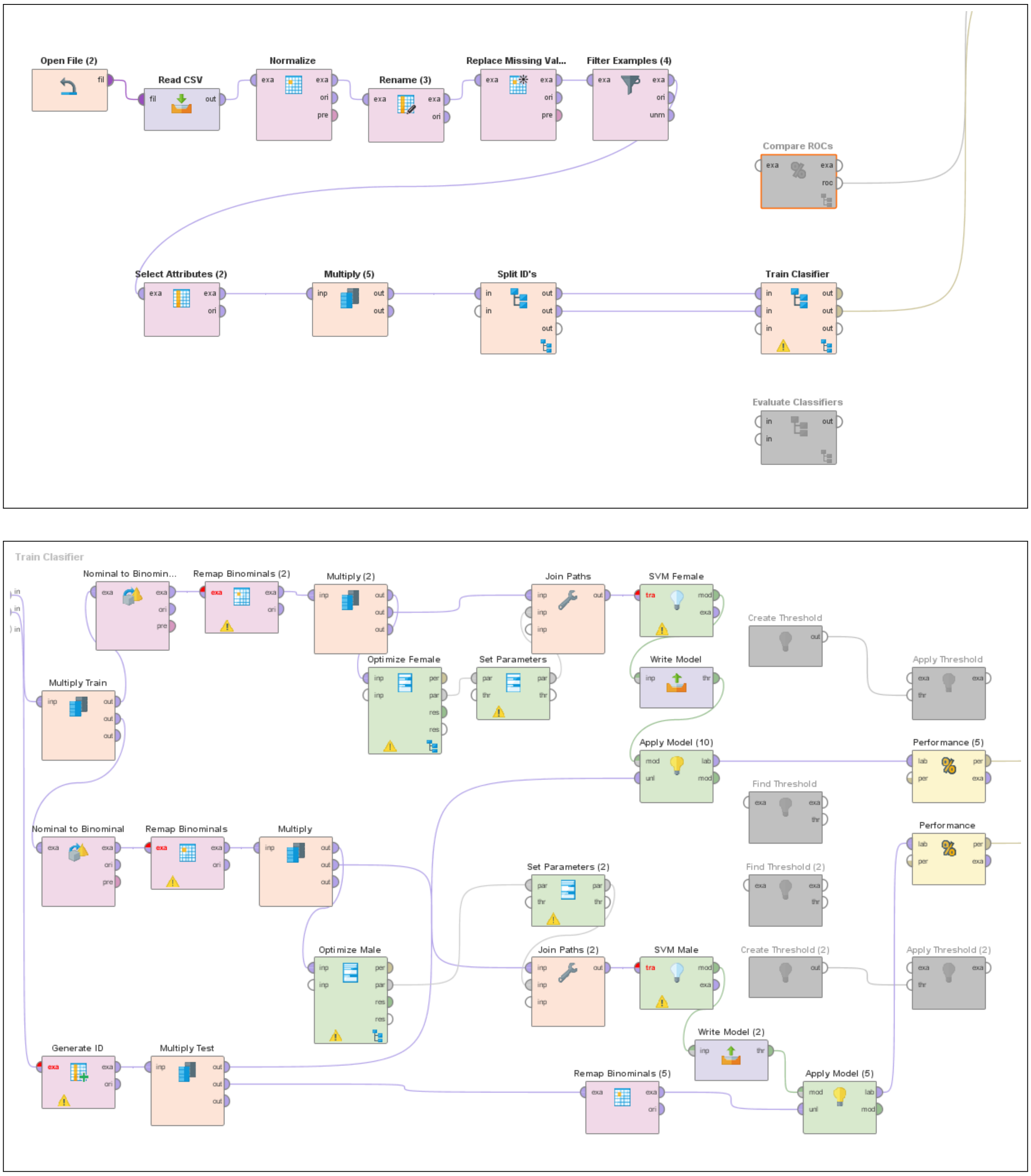


Figure C.1: An overview of the RapidMiner process to predict gender from the preprocessed proof of concept input data. Top: high level overview, Bottom: the sub-process named ’Train Clasifier’

APPENDIX D

Appendix: RapidMiner models

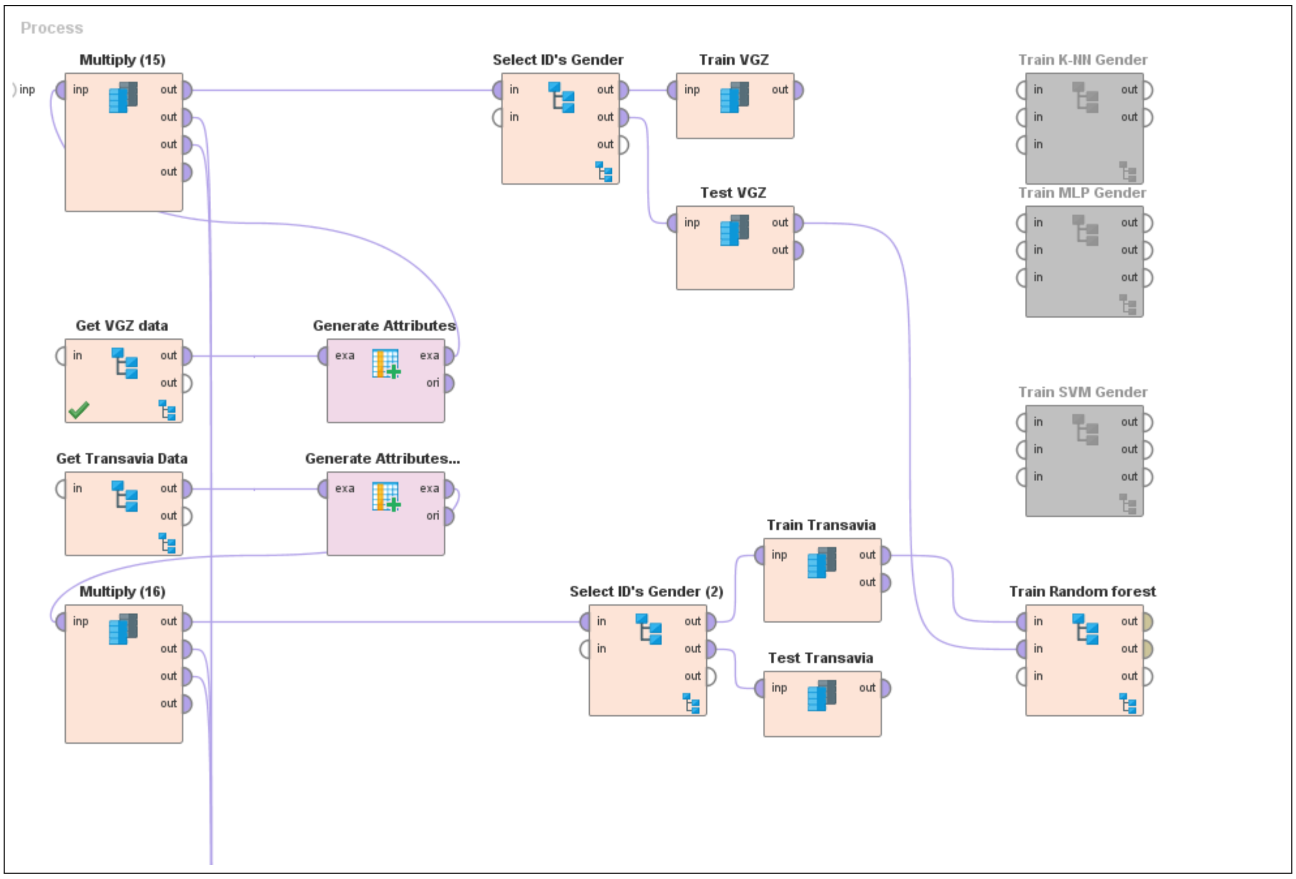


Figure D.1: An overview of the RapidMiner process to predict gender using different algorithms. Only the random forest classifier is being shown, the other algorithms follow a similar process

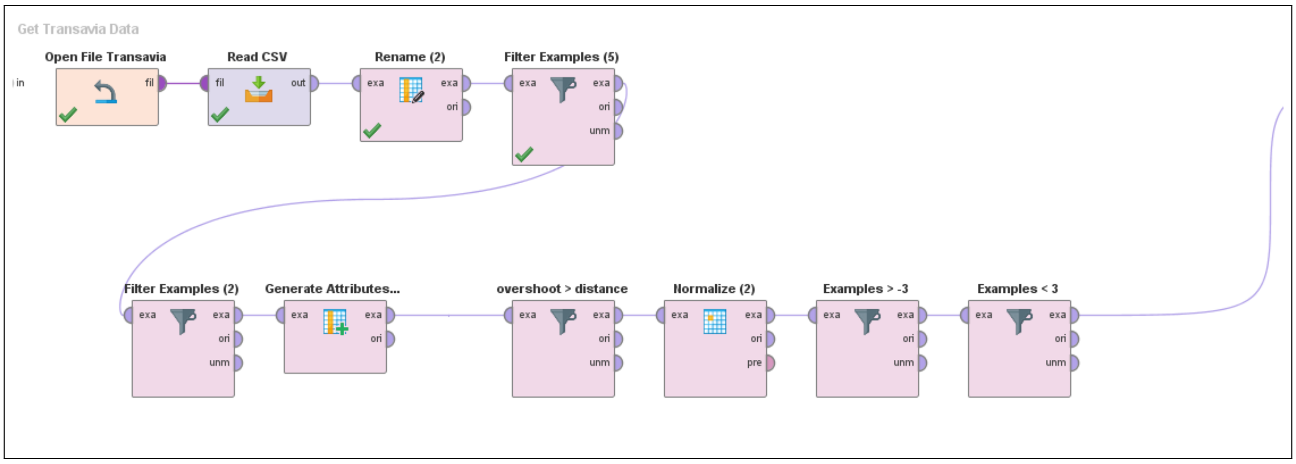


Figure D.2: The data import step for airline. This import step is the same for regression and classification

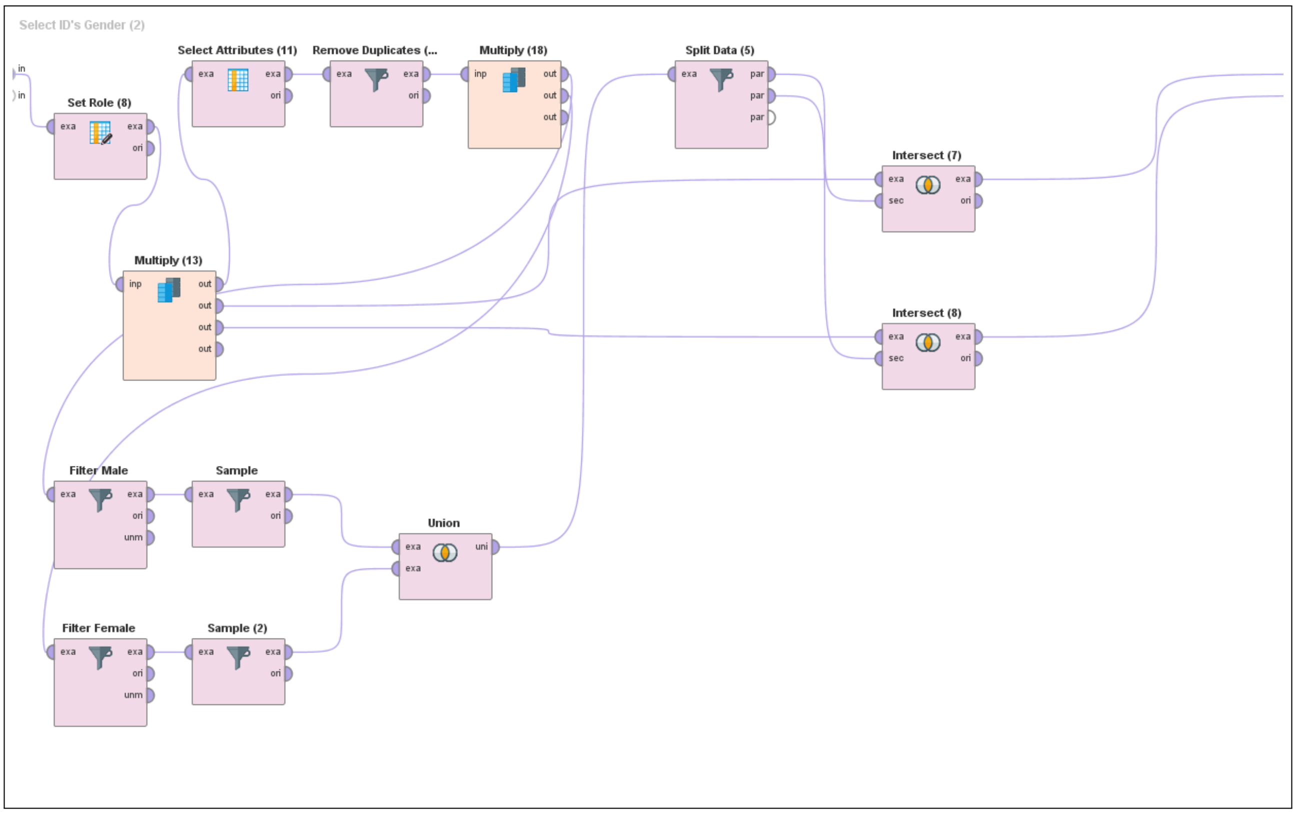


Figure D.3: The process for splitting the data into training and testing sets

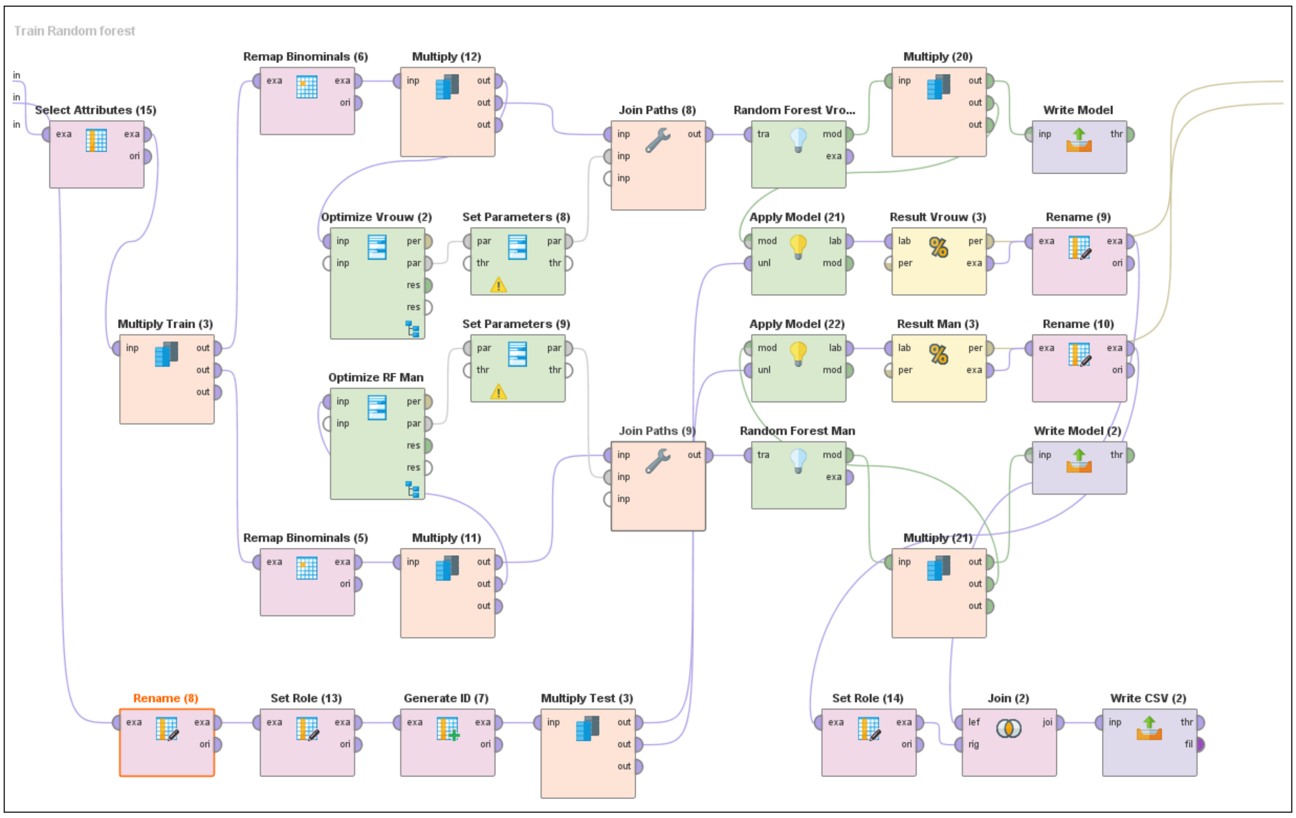


Figure D.4: The process for optimizing two classifiers for Male and Female. The classifier in this case is the Random Forest classifier. The other algorithms follow the same set-up

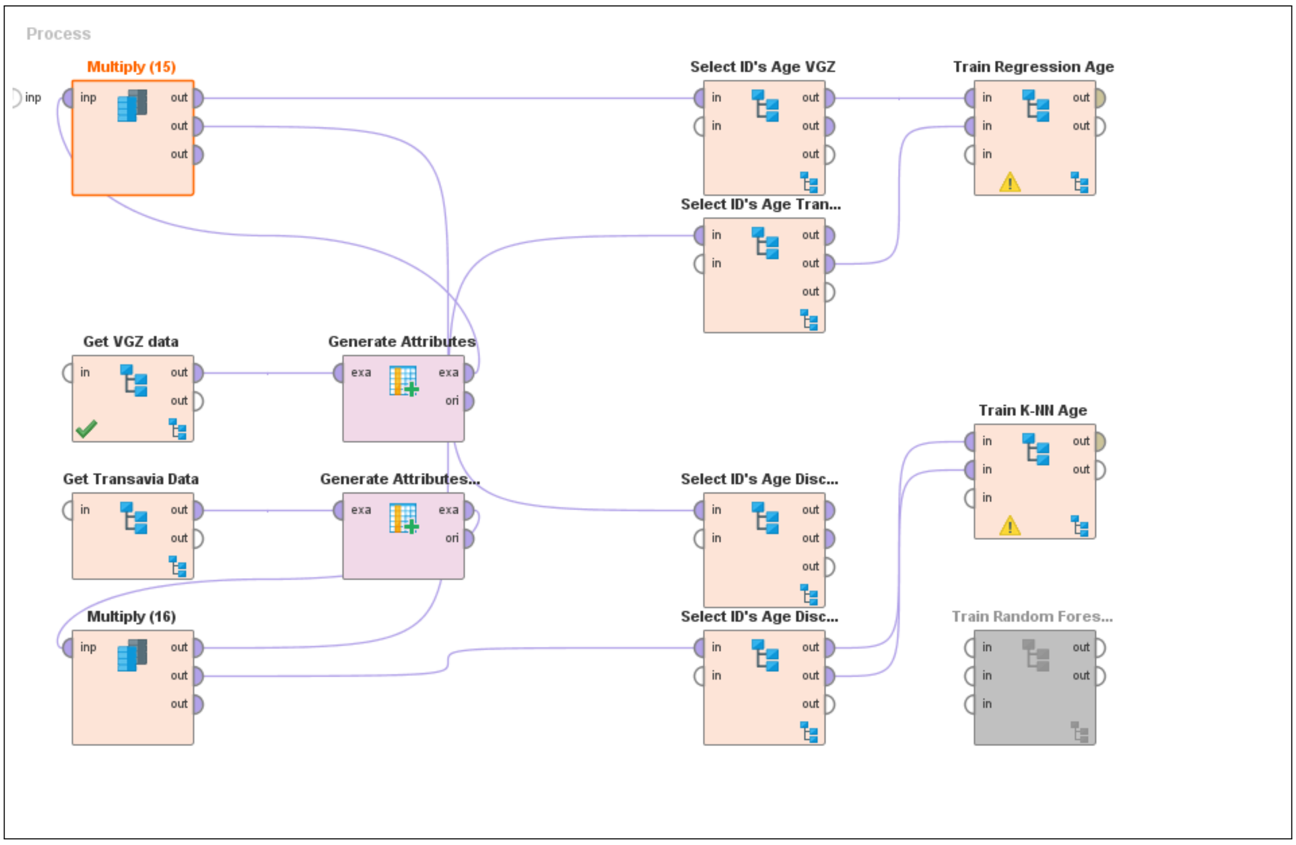


Figure D.5: An overview of the RapidMiner process to predict age, once as regression and once by discretizing age labels into bins. Different algorithms were used to but only the random forest classifier is being shown, the other algorithms follow a similar process. For the random forest classifier, no normalization was performed

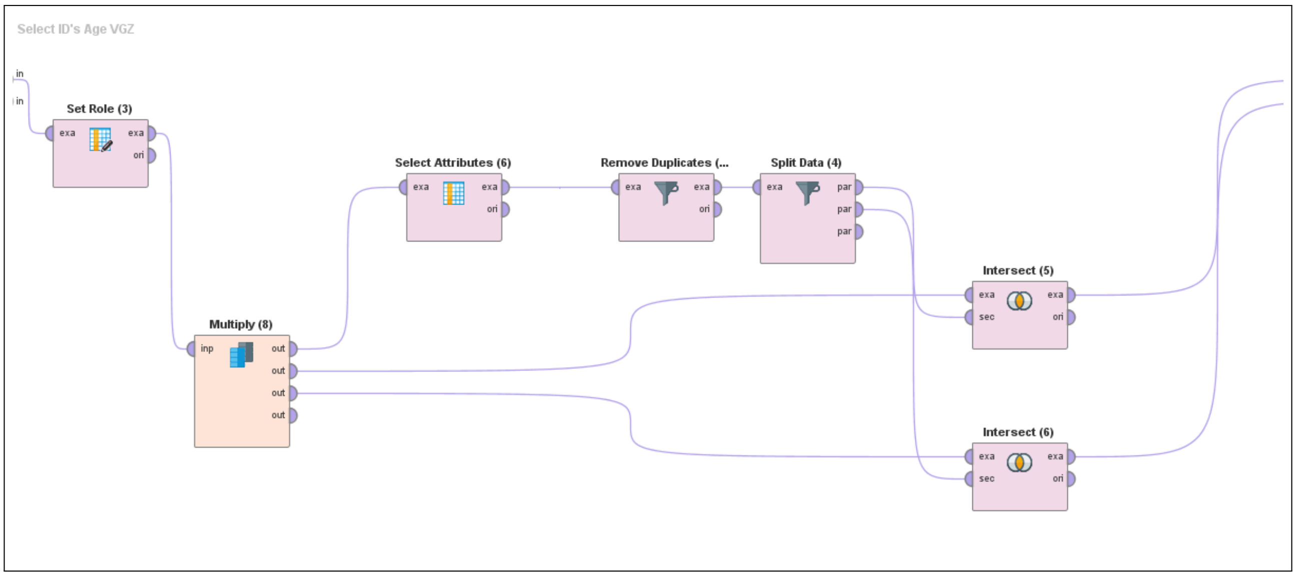


Figure D.6: The entries are split per user so as to prevent over-fitting

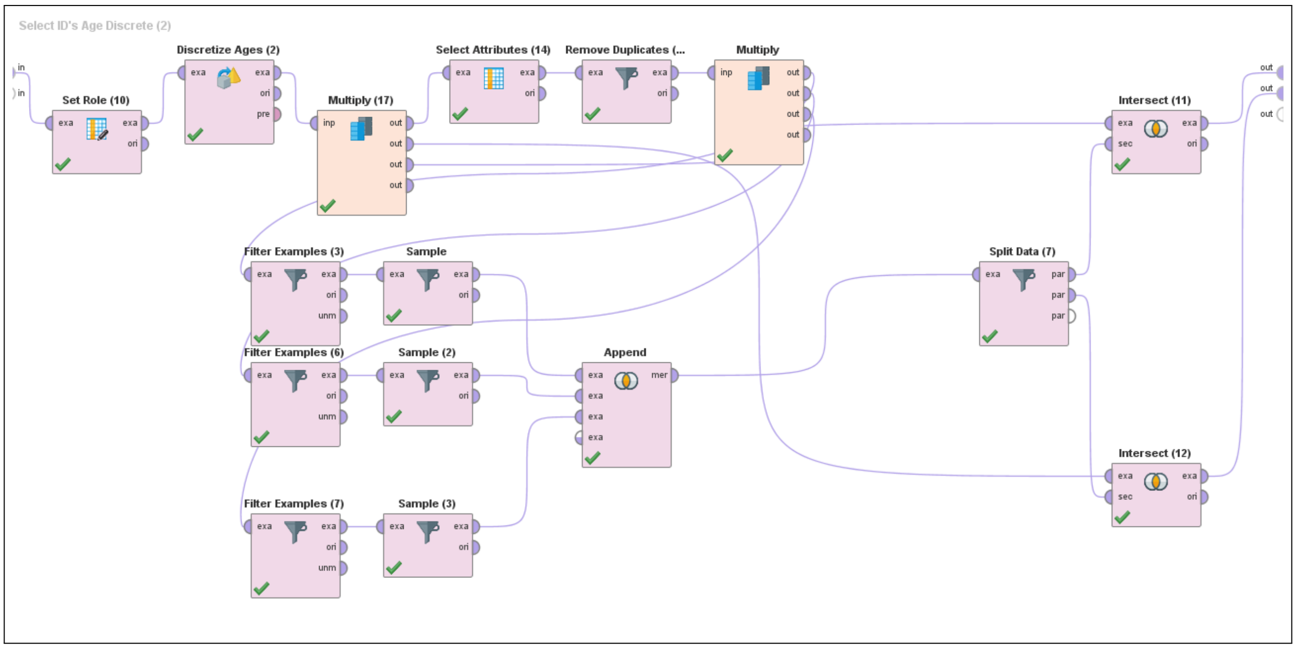


Figure D.7: The process to split the users into training and testing set per id, with the age labels discretized into bins

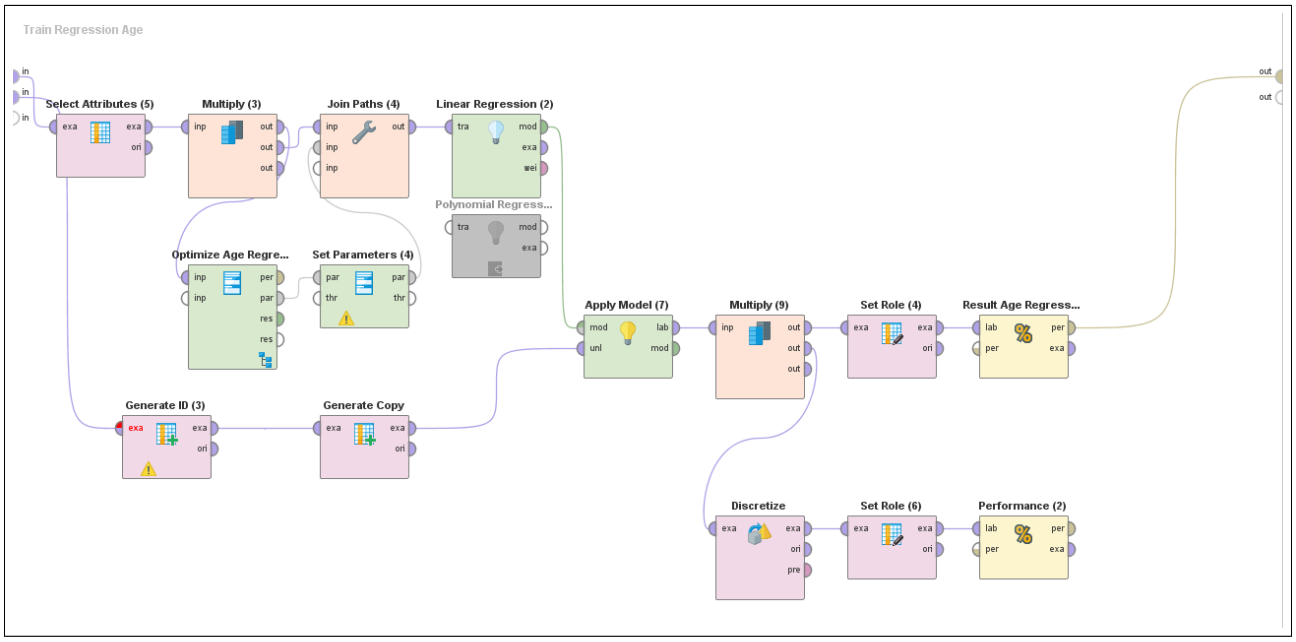


Figure D.8: The process for optimizing two classifiers for Male and Female. The classifier in this case is the Random Forest classifier. The other algorithms follow the same set-up

APPENDIX E

Appendix: Research Server Specifications

The research server is an Amazon s3 server in Ireland with the following specifications:

- 1 CPU-core
- 3.75 GB RAM
- Unlimited storage[1]
- Moderate Network Performance

The server is running Ubuntu Server 14 and Apache was installed to serve the data collection framework. Connections are made through sFTP to write files to an isolated directory.

[1]with individual files up to 5TB https://aws.amazon.com/s3/faqs/

Appendix F

Appendix: Collected Data

Data and code can be downloaded from `http://www.stijnhoogervorst.nl/thesis`

Appendix G

Appendix: Classification Results

Table G.1: SVM classifier result

Train	airline		
Test	airline		
Model			svm_1_vrouw
	True Female	True Male	Precision
Pred Female	814	765	51.55%
Pred Male	1231	1226	49.90%
Class Recall	39.80%	61.58%	

Train	airline		
Test	airline		
Model			svm_1_man
	True Female	True Male	Precision
Pred Female	936	935	50.03%
Pred Male	1109	1056	48.78%
Class Recall	45.77%	53.04%	

Train	airline		
Test	insurer		
Model			svm_1_vrouw
	True Female	True Male	Precision
Pred Female	74	52	58.73%
Pred Male	59	58	49.57%
Class Recall	55.64%	52.73%	

Train	airline		
Test	insurer		
Model			svm_1_man
	True Female	True Male	Precision
Pred Female	0	0	#DIV/0!
Pred Male	133	110	45.27%
Class Recall	0.00%	100.00%	

Train	insurer		
Test	insurer		
Model			svm_insurer_vrouw
	True Female	True Male	Precision
Pred Female	133	109	54.96%
Pred Male	0	1	100.00%
Class Recall	100.00%	0.91%	

Train	insurer		
Test	insurer		
Model			svm_insurer_man
	True Female	True Male	Precision
Pred Female	133	108	55.19%
Pred Male	0	2	100.00%
Class Recall	100.00%	1.82%	

Train	insurer		
Test	airline		
Model			svm_insurer_vrouw
	True Female	True Male	Precision
Pred Female	2045	1991	50.67%
Pred Male	0	0	#DIV/0!
Class Recall	100.00%	0.00%	

Train	insurer		
Test	airline		
Model			svm_insurer_man
	True Female	True Male	Precision
Pred Female	2045	1991	50.67%
Pred Male	0	0	#DIV/0!
Class Recall	100.00%	0.00%	

Table G.2: Classification result of an SVM Classifier trained with touchdevices included in the training and testing data

Train	airline		
Test	airline		
Model			svm_touch_airline_vrouw
	True Female	True Male	Precision
Pred Female	26	20	56.52%
Pred Male	0	0	#DIV/0!
Class Recall	100.00%	0.00%	

Train	airline		
Test	airline		
Model			svm_touch_airline_man
	True Female	True Male	Precision
Pred Female	0	0	#DIV/0!
Pred Male	26	20	43.48%
Class Recall	0.00%	100.00%	

Train	airline		
Test	insurer		
Model			svm_touch_airline_vrouw
	True Female	True Male	Precision
Pred Female	878	995	46.88%
Pred Male	404	366	47.53%
Class Recall	68.49%	26.89%	

Train	airline		
Test	insurer		
Model			svm_touch_airline_man
	True Female	True Male	Precision
Pred Female	134	142	48.55%
Pred Male	1148	1219	51.50%
Class Recall	10.45%	89.57%	

Train	insurer		
Test	insurer		
Model			svm_touch_insurer_vrouw
	True Female	True Male	Precision
Pred Female	26	20	56.52%
Pred Male	0	0	#DIV/0!
Class Recall	100.00%	0.00%	

Train	insurer		
Test	insurer		
Model			svm_touch_insurer_man
	True Female	True Male	Precision
Pred Female	26	20	56.52%
Pred Male	0	0	#DIV/0!
Class Recall	100.00%	0.00%	

Train	insurer		
Test	airline		
Model			svm_touch_insurer_vrouw
	True Female	True Male	Precision
Pred Female	1282	1360	48.52%
Pred Male	0	1	100.00%
Class Recall	100.00%	0.07%	

Train	insurer		
Test	airline		
Model			svm_touch_insurer_man
	True Female	True Male	Precision
Pred Female	1282	1361	48.51%
Pred Male	0	0	#DIV/0!
Class Recall	100.00%	0.00%	

Table G.3: Classification result of an SVM Classifier trained to predict the use of a touchdevice

Train	airline		
Test	airline		
Model	svm_airline_predict_touch_vrouw		
	True	False	Precision
Pred True	3077	112	96.49%
Pred False	9	2	18.18%
Class Recall	99.71%	1.75%	

Train	airline		
Test	airline		
Model	svm_airline_predict_touch_man		
	True	False	Precision
Pred True	3067	110	96.54%
Pred False	19	4	17.39%
Class Recall	99.38%	3.51%	

Train	airline		
Test	insurer		
Model	svm_airline_predict_touch_vrouw		
	True	False	Precision
Pred True	278	3	98.93%
Pred False	0	0	#DIV/0!
Class Recall	100.00%	0.00%	

Train	airline		
Test	insurer		
Model	svm_airline_predict_touch_man		
	True	False	Precision
Pred True	278	3	98.93%
Pred False	0	0	#DIV/0!
Class Recall	100.00%	0.00%	

Train	insurer		
Test	insurer		
Model	svm_predict_touch_insurer_vrouw		
	True	False	Precision
Pred True	278	3	98.93%
Pred False	0	0	#DIV/0!
Class Recall	100.00%	0.00%	

Train	insurer		
Test	insurer		
Model	svm_predict_touch_insurer_man		
	True	False	Precision
Pred True	278	3	98.93%
Pred False	0	0	#DIV/0!
Class Recall	100.00%	0.00%	

Train	insurer		
Test	airline		
Model	svm_predict_touch_insurer_vrouw		
	True	False	Precision
Pred True	3086	114	96.44%
Pred False	0	0	#DIV/0!
Class Recall	100.00%	0.00%	

Train	insurer		
Test	airline		
Model	svm_predict_touch_insurer_man		
	True	False	Precision
Pred True	3086	114	96.44%
Pred False	0	0	#DIV/0!
Class Recall	100.00%	0.00%	

Table G.4: Classification result of an SVM Classifier trained on Principal Component Analysis of input features

Train	airline			Train	airline		
Test	airline			Test	airline		
Model		svm_pca_airline_vrouw		Model		svm_pca_airline_man	
	True Female	True Male	Precision		True Female	True Male	Precision
Pred Female	811	849	48.86%	Pred Female	283	237	54.42%
Pred Male	434	432	49.88%	Pred Male	962	1044	52.04%
Class Recall	65.14%	33.72%		Class Recall	22.73%	81.50%	

Train	airline			Train	airline		
Test	insurer			Test	insurer		
Model		svm_pca_airline_vrouw		Model		svm_pca_airline_man	
	True Female	True Male	Precision		True Female	True Male	Precision
Pred Female	31	41	43.06%	Pred Female	0	0	#DIV/0!
Pred Male	0	0	#DIV/0!	Pred Male	31	41	56.94%
Class Recall	100.00%	0.00%		Class Recall	0.00%	100.00%	

Train	insurer			Train	insurer		
Test	insurer			Test	insurer		
Model		svm_insurer_vrouw		Model		svm_insurer_man	
	True Female	True Male	Precision		True Female	True Male	Precision
Pred Female	31	41	43.06%	Pred Female	31	41	43.06%
Pred Male	0	0	#DIV/0!	Pred Male	0	0	#DIV/0!
Class Recall	100.00%	0.00%		Class Recall	100.00%	0.00%	

Train	insurer			Train	insurer		
Test	airline			Test	airline		
Model		svm_insurer_vrouw		Model		svm_insurer_man	
	True Female	True Male	Precision		True Female	True Male	Precision
Pred Female	1225	1262	49.26%	Pred Female	1245	1281	49.29%
Pred Male	20	19	48.72%	Pred Male	0	0	#DIV/0!
Class Recall	98.39%	1.48%		Class Recall	100.00%	0.00%	

Table G.5: Result of the Random Forest Classifier

Train	airline		
Test	airline		
Model	rf_airline_vrouw		
	True Female	True Male	Precision
Pred Female	894	924	49.17%
Pred Male	351	357	50.42%
Class Recall	71.81%	27.87%	

Train	airline		
Test	airline		
Model			rf_airline_man
	True Female	True Male	Precision
Pred Female	1245	1281	49.29%
Pred Male	0	0	#DIV/0!
Class Recall	100.00%	0.00%	

Train	airline		
Test	insurer		
Model			rf_airline_vrouw
	True Female	True Male	Precision
Pred Female	73	98	42.69%
Pred Male	393	572	59.27%
Class Recall	15.67%	85.37%	

Train	airline		
Test	insurer		
Model			rf_airline_man
	True Female	True Male	Precision
Pred Female	466	670	41.02%
Pred Male	0	0	#DIV/0!
Class Recall	100.00%	0.00%	

Train	insurer		
Test	insurer		
Model			rf_insurer_vrouw
	True Female	True Male	Precision
Pred Female	18	20	47.37%
Pred Male	13	21	61.76%
Class Recall	58.06%	51.22%	

Train	insurer		
Test	insurer		
Model			rf_insurer_man
	True Female	True Male	Precision
Pred Female	20	21	48.78%
Pred Male	11	20	64.52%
Class Recall	64.52%	48.78%	

Train	insurer		
Test	airline		
Model			rf_insurer_vrouw
	True Female	True Male	Precision
Pred Female	420	433	49.24%
Pred Male	825	848	50.69%
Class Recall	33.73%	66.20%	

Train	insurer		
Test	airline		
Model			rf_insurer_man
	True Female	True Male	Precision
Pred Female	953	949	50.11%
Pred Male	292	332	53.21%
Class Recall	76.55%	25.92%	

Table G.6: Result of the Random Forest Classifier without applying normalization to the input features (includes more noise as no filtering based on distance from the mean after Z-score normalization was performed)

Train	airline		
Test	airline		
Model	rf_nonorm_airline_vrouw		
	True Female	True Male	Precision
Pred Female	846	920	47.90%
Pred Male	404	455	52.97%
Class Recall	67.68%	33.09%	

Train	airline		
Test	airline		
Model	rf_nonorm_airline_man		
	True Female	True Male	Precision
Pred Female	1193	1216	49.52%
Pred Male	2	4	66.67%
Class Recall	99.83%	0.33%	

Train	airline		
Test	insurer		
Model	rf_nonorm_airline_vrouw		
	True Female	True Male	Precision
Pred Female	308	439	41.23%
Pred Male	158	231	59.38%
Class Recall	66.09%	34.48%	

Train	airline		
Test	insurer		
Model	rf_nonorm_airline_man		
	True Female	True Male	Precision
Pred Female	466	664	41.24%
Pred Male	0	6	100%
Class Recall	100.00%	0.90%	

Train	insurer		
Test	insurer		
Model	rf_nonorm_insurer_vrouw		
	True Female	True Male	Precision
Pred Female	4	0	100.00%
Pred Male	13	21	61.76%
Class Recall	23.53%	100.00%	

Train	insurer		
Test	insurer		
Model	rf_nonorm_insurer_man		
	True Female	True Male	Precision
Pred Female	4	1	80.00%
Pred Male	13	20	60.61%
Class Recall	23.53%	95.24%	

Train	insurer		
Test	airline		
Model	rf_nonorm_insurer_vrouw		
	True Female	True Male	Precision
Pred Female	961	1046	47.88%
Pred Male	289	329	53.24%
Class Recall	76.88%	23.93%	

Train	insurer		
Test	airline		
Model	rf_nonorm_insurer_man		
	True Female	True Male	Precision
Pred Female	667	747	47.17%
Pred Male	583	628	51.86%
Class Recall	53.36%	45.67%	

Table G.7: Result of the Random Forest Classifier using Browser as additional input feature

Train	airline		
Test	airline		
Model			rf_browser_airline_vrouw
	True Female	True Male	Precision
Pred Female	870	953	47.72%
Pred Male	380	422	52.62%
Class Recall	69.60%	30.69%	49%

Train	airline		
Test	airline		
Model			rf_browser_airline_man
	True Female	True Male	Precision
Pred Female	1250	1375	47.62%
Pred Male	0	0	#DIV/0!
Class Recall	100.00%	0.00%	48%

Train	airline		
Test	insurer		
Model			rf_browser_airline_vrouw
	True Female	True Male	Precision
Pred Female	12	17	41.38%
Pred Male	5	4	44.44%
Class Recall	70.59%	19.05%	42%

Train	airline		
Test	insurer		
Model			rf_browser_airline_man
	True Female	True Male	Precision
Pred Female	17	21	44.74%
Pred Male	0	0	#DIV/0!
Class Recall	100.00%	0.00%	45%

Train	insurer		
Test	insurer		
Model			rf_browser_insurer_vrouw
	True Female	True Male	Precision
Pred Female	4	0	100.00%
Pred Male	13	21	61.76%
Class Recall	23.53%	100.00%	66%

Train	insurer		
Test	insurer		
Model			rf_browser_insurer_man
	True Female	True Male	Precision
Pred Female	10	5	66.67%
Pred Male	7	16	69.57%
Class Recall	58.82%	76.19%	68%

Train	insurer		
Test	airline		
Model			rf_browser_insurer_vrouw
	True Female	True Male	Precision
Pred Female	409	429	48.81%
Pred Male	841	946	52.94%
Class Recall	32.72%	68.80%	52%

Train	insurer		
Test	airline		
Model			rf_browser_insurer_man
	True Female	True Male	Precision
Pred Female	813	899	47.49%
Pred Male	437	476	52.14%
Class Recall	65.04%	34.62%	49%

Table G.8: Different classification results for different Thresholds for the Female Random Forest classifier trained on Non-normalized input features from the airline Training dataset, applied to the airline Testing set

threshold	true_female	false_male	false_female	true_male	Precision	Recall
1	0	1195	0	1235		0%
0.75	0	1195	0	1235		0%
0.74	1	1194	0	1235	100.00%	0%
0.72	1	1194	0	1235	100.00%	0%
0.71	2	1193	0	1235	100.00%	0%
0.65	3	1192	0	1235	100.00%	0%
0.64	6	1189	0	1235	100.00%	1%
0.62	6	1189	1	1234	85.71%	1%
0.61	6	1189	2	1233	75.00%	1%
0.60	14	1181	4	1231	77.78%	1%
0.59	21	1174	6	1229	77.78%	2%
0.58	27	1168	9	1226	75.00%	2%
0.57	42	1153	22	1213	65.63%	4%
0.56	66	1129	46	1189	58.93%	6%
0.55	92	1103	61	1174	60.13%	8%
0.54	323	872	247	988	56.67%	27%
0.53	412	783	330	905	55.53%	34%
0.52	592	603	479	756	55.28%	50%
0.51	779	416	679	556	53.43%	65%
0.50	877	318	800	435	52.30%	73%
0.49	964	231	921	314	51.14%	81%
0.48	1036	159	1006	229	50.73%	87%
0.47	1119	76	1083	152	50.82%	94%
0.46	1142	53	1121	114	50.46%	96%
0.45	1157	38	1150	85	50.15%	97%
0.44	1166	29	1170	65	49.91%	98%
0.43	1176	19	1182	53	49.87%	98%
0.42	1181	14	1195	40	49.71%	99%
0.41	1182	13	1200	35	49.62%	99%
0.40	1190	5	1214	21	49.50%	100%
0.39	1191	4	1220	15	49.40%	100%
0.38	1191	4	1227	8	49.26%	100%
0.37	1192	3	1228	7	49.26%	100%
0.36	1192	3	1230	5	49.22%	100%
0.35	1193	2	1232	3	49.20%	100%
0.34	1193	2	1233	2	49.18%	100%
0.33	1194	1	1235	0	49.16%	100%
0	1195	0	1235	0	49.18%	100%

Table G.9: Different classification results for different Thresholds for the Female Random Forest classifier when applied to Male confidence, trained on non-normalized input features from the airline Training dataset, applied to the airline Testing set

Threshold	true_male	false_male	false_female	true_female	Precision	Recall
0	1235	1195	0	0	50.82%	100%
0.25	1235	1195	0	0	50.82%	100%
0.26	1235	1194	0	1	50.84%	100%
0.27	1235	1194	0	1	50.84%	100%
0.28	1235	1194	0	1	50.84%	100%
0.29	1235	1193	0	2	50.86%	100%
0.30	1235	1193	0	2	50.86%	100%
0.31	1235	1193	0	2	50.86%	100%
0.32	1235	1193	0	2	50.86%	100%
0.33	1235	1193	0	2	50.86%	100%
0.34	1235	1193	0	2	50.86%	100%
0.35	1235	1192	0	3	50.89%	100%
0.36	1235	1189	0	6	50.95%	100%
0.37	1235	1189	0	6	50.95%	100%
0.38	1234	1189	1	6	50.93%	100%
0.39	1233	1189	2	6	50.91%	100%
0.40	1231	1181	4	14	51.04%	100%
0.41	1229	1174	6	21	51.14%	100%
0.42	1226	1168	9	27	51.21%	99%
0.43	1213	1153	22	42	51.27%	98%
0.44	1189	1129	46	66	51.29%	96%
0.45	1174	1103	61	92	51.56%	95%
0.46	988	872	247	323	53.12%	80%
0.47	905	783	330	412	53.61%	73%
0.48	756	603	479	592	55.63%	61%
0.49	556	416	679	779	57.20%	45%
0.50	435	318	800	877	57.77%	35%
0.51	314	231	921	964	57.61%	25%
0.52	229	159	1006	1036	59.02%	19%
0.53	152	76	1083	1119	66.67%	12%
0.54	114	53	1121	1142	68.26%	9%
0.55	85	38	1150	1157	69.11%	7%
0.56	65	29	1170	1166	69.15%	5%
0.57	53	19	1182	1176	73.61%	4%
0.58	40	14	1195	1181	74.07%	3%
0.59	35	13	1200	1182	72.92%	3%
0.60	21	5	1214	1190	80.77%	2%
0.61	15	4	1220	1191	78.95%	1%
0.62	8	4	1227	1191	66.67%	1%
0.63	7	3	1228	1192	70.00%	1%
0.64	5	3	1230	1192	62.50%	0%
0.65	3	2	1232	1193	60.00%	0%
0.66	2	2	1233	1193	50.00%	0%
0.67	0	1	1235	1194	0.00%	0%
1	0	0	1235	1195	0.00%	0%

Table G.10: Different classification results for different Thresholds for the Female Random Forest classifier when applied to Female confidence, trained on non-normalized input features from the airline Training dataset, applied to the Health Insurer set

Threshold	true_male	false_male	false_female	true_female	Precision	Recall
0.00	0	466	0	460		0%
0.68	0	466	0	460		0%
0.67	0	466	1	459	0.00%	0%
0.66	0	466	1	459	0.00%	0%
0.65	0	466	1	459	0.00%	0%
0.64	1	465	1	459	50.00%	0%
0.63	2	464	2	458	50.00%	0%
0.62	2	464	3	457	40.00%	0%
0.61	4	462	3	457	57.14%	1%
0.6	6	460	4	456	60.00%	1%
0.59	10	456	8	452	55.56%	2%
0.58	14	452	13	447	51.85%	3%
0.57	23	443	20	440	53.49%	5%
0.56	33	433	29	431	53.23%	7%
0.55	41	425	42	418	49.40%	9%
0.54	93	373	100	360	48.19%	20%
0.53	122	344	121	339	50.21%	26%
0.52	203	263	178	282	53.28%	44%
0.51	263	203	232	228	53.13%	56%
0.5	308	158	302	158	50.49%	66%
0.49	337	129	346	114	49.34%	72%
0.48	374	92	365	95	50.61%	80%
0.47	421	45	413	47	50.48%	90%
0.46	443	23	428	32	50.86%	95%
0.45	452	14	442	18	50.56%	97%
0.44	455	11	445	15	50.56%	98%
0.43	457	9	448	12	50.50%	98%
0.42	459	7	452	8	50.38%	98%
0.41	460	6	454	6	50.33%	99%
0.4	463	3	455	5	50.44%	99%
0.39	465	1	456	4	50.49%	100%
0.38	466	0	457	3	50.49%	100%
0.37	466	0	458	2	50.43%	100%
0.36	466	0	460	0	50.32%	100%
1.00	466	0	460	0	50.32%	100%

Table G.11: Different classification results for different Thresholds for the Female Random Forest classifier trained on Non-normalized input features

threshold	true_female	false_male	false_female	true_male	Female Precision	Male Precision
0	0	1195	0	1235		49.18%
0.25	0	1195	0	1235		49.18%
0.26	1	1194	0	1235	100.00%	49.16%
0.27	1	1194	0	1235	100.00%	49.16%
0.28	1	1194	0	1235	100.00%	49.16%
0.29	2	1193	0	1235	100.00%	49.14%
0.3	2	1193	0	1235	100.00%	49.14%
0.31	2	1193	0	1235	100.00%	49.14%
0.32	2	1193	0	1235	100.00%	49.14%
0.33	2	1193	0	1235	100.00%	49.14%
0.34	2	1193	0	1235	100.00%	49.14%
0.35	3	1192	0	1235	100.00%	49.11%
0.36	6	1189	0	1235	100.00%	49.05%
0.37	6	1189	0	1235	100.00%	49.05%
0.38	6	1189	1	1234	85.71%	49.07%
0.39	6	1189	2	1233	75.00%	49.09%
0.4	14	1181	4	1231	77.78%	48.96%
0.41	21	1174	6	1229	77.78%	48.86%
0.42	27	1168	9	1226	75.00%	48.79%
0.43	42	1153	22	1213	65.63%	48.73%
0.44	66	1129	46	1189	58.93%	48.71%
0.45	92	1103	61	1174	60.13%	48.44%
0.46	323	872	247	988	56.67%	46.88%
0.47	412	783	330	905	55.53%	46.39%
0.48	592	603	479	756	55.28%	44.37%
0.49	779	416	679	556	53.43%	42.80%
0.5	877	318	800	435	52.30%	42.23%
0.51	964	231	921	314	51.14%	42.39%
0.52	1036	159	1006	229	50.73%	40.98%
0.53	1119	76	1083	152	50.82%	33.33%
0.54	1142	53	1121	114	50.46%	31.74%
0.55	1157	38	1150	85	50.15%	30.89%
0.56	1166	29	1170	65	49.91%	30.85%
0.57	1176	19	1182	53	49.87%	26.39%
0.58	1181	14	1195	40	49.71%	25.93%
0.59	1182	13	1200	35	49.62%	27.08%
0.6	1190	5	1214	21	49.50%	19.23%
0.61	1191	4	1220	15	49.40%	21.05%
0.62	1191	4	1227	8	49.26%	33.33%
0.63	1192	3	1228	7	49.26%	30.00%
0.64	1192	3	1230	5	49.22%	37.50%
0.65	1193	2	1232	3	49.20%	40.00%
0.66	1193	2	1233	2	49.18%	50.00%
0.67	1194	1	1235	0	49.16%	100.00%
0.68	1195	0	1235	0	49.18%	
1	1195	0	1235	0	49.18%	

Table G.12: Different classification results for different Thresholds for the Male Random Forest classifier trained on non-normalized input features from the airline Training dataset, applied to the Health Insurer set

Threshold	true_female	false_male	false_female	true_male	Precision	Recall
0.00	0	466	0	460	49.68%	100%
0.49	0	466	0	460	49.68%	100%
0.5	466	0	456	4	100.00%	1%
0.51	466	0	456	4	100.00%	1%
0.52	466	0	456	4	100.00%	1%
0.53	466	0	456	4	100.00%	1%
0.54	466	0	456	4	100.00%	1%
0.55	466	0	460	0		0%
1.00	466	0	460	0		0%

Table G.13: Regression Result

Polynomial Regression	
Model	reg_polynomial_airline
Train on	airline
Applied to	airline
RMSE	16.953
Prediction Average	43.279 +/- 16.322

Model	reg_polynomial_airline
Train on	airline
Applied to	insurer
RMSE	18339.16
Prediction Average	33.886 +/- 17.225
Model	reg_polynomial_insurer
Train on	insurer
Applied to	insurer
RMSE	4566.036
Prediction Average	33.886 +/- 17.225

Model	reg_polynomial_insurer
Train on	insurer
Applied to	airline
RMSE	241.592
Prediction Average	43.279 +/- 16.322

Table G.14: Linear Regression Model trained and tested on airline input data

Regression Age	
Test	airline
Train	airline
Performance metric	RMSE
RMSE	15.874
Prediction Average	43.279 +/- 16.322

Table G.15: The result of applying the regression model to the training set and afterwards discretizing the predictions and labels to evaluate performance. We can see that class precision is roughly equal to guessing, but recall is only good for classes around the prediction average

	25-	25 - 30	30 - 35	35 - 40	40 - 45	45 - 50	50 - 55	55 - 60	60+	class precision
pred. 25-	0	0	0	0	0	0	0	0	0	0.00%
pred. 25 - 30	1	0	0	0	0	0	0	0	0	0.00%
pred. 30 - 35	9	6	6	2	3	2	2	1	0	19.35%
pred. 35 - 40	111	83	64	54	49	50	21	25	34	11.00%
pred. 40 - 45	224	170	160	141	136	127	128	83	229	9.73%
pred. 45 - 50	110	82	87	74	91	112	122	83	209	11.55%
pred. 50 - 55	1	3	6	4	1	7	8	3	13	17.39%
pred. 55 - 60	0	0	0	0	0	0	0	0	0	0.00%
pred. 60+	0	0	0	0	0	0	0	0	1	100.00%
class recall	0.00%	0.00%	1.86%	19.64%	48.57%	37.58%	2.85%	0.00%	0.21%	

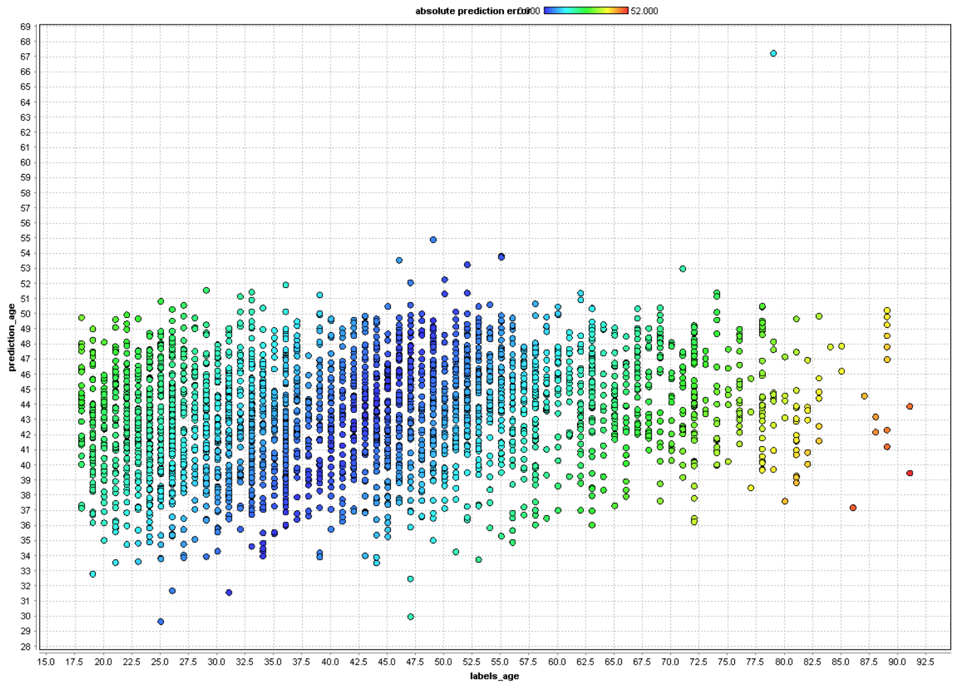


Figure G.1: A scatter plot of the linear regression result showing predicted age on the y-axis and actual age on the x-axis. Colour of the points encodes for the absolute error

Table G.16: Results from applying k-NN model on mouse click input data discretized on age group

Train	airline			
Test	airline			
Overall Accuracy	38.92%			
	true 0 - 30	true 30 - 60	true 60 - 90	*precision*
pred 0 -30	170	151	142	*36.72%*
pred 30 - 60	133	153	125	*37.23%*
pred 60 - 90	183	225	288	*41.38%*
recall	*34.98%*	*28.92%*	*51.89%*	

Table G.17: Results from applying SVM model on mouse click input data discretized on age group

Train	airline			
Test	airline			
Overall Accuracy	38.15%			
	true 0 - 30	true 30 - 60	true 60 - 90	*precision*
pred 0 -30	199	168	154	*38.20%*
pred 30 - 60	130	161	162	*35.54%*
pred 60 - 90	157	200	239	*40.10%*
recall	*40.95%*	*30.43%*	*43.06%*	

Table G.18: Results from applying RF model on mouse click input data discretized on age group (not normalized)

Train	airline			
Test	airline			
Overall Accuracy	41.27%			
	true 0 - 30	true 30 - 60	true 60 - 90	*precision*
pred 0 -30	267	244	174	*38.98%*
pred 30 - 60	0	0	0	*0.00%*
pred 60 - 90	219	285	381	*43.05%*
recall	*54.94%*	*0.00%*	*68.65%*	

Table G.19: Results from applying k-NN model on mouse click input data discretized on age group, trained on airline and applied to Health Insurance dataset

Train	airline			
Test	insurer			
Overall Accuracy	27.57%			
	true 0 - 30	true 30 - 60	true 60 - 90	*precision*
pred 0 -30	15	40	40	*15.79%*
pred 30 - 60	10	16	11	*43.24%*
pred 60 - 90	6	27	20	*37.74%*
recall	*48.39%*	*19.28%*	*28.17%*	

Table G.20: Results from applying SVM model on mouse click input data discretized on age group, trained on airline and applied to Health Insurance dataset

Train	airline			
Test	insurer			
Overall Accuracy	35.14%			
	true 0 - 30	true 30 - 60	true 60 - 90	*precision*
pred 0 -30	5	24	18	*10.64%*
pred 30 - 60	17	42	35	*44.68%*
pred 60 - 90	9	17	18	*40.91%*
recall	*16.13%*	*50.60%*	*25.35%*	

Table G.21: Results from applying RF model on mouse click input data discretized on age group (not normalized), trained on airline and applied to Health Insurance dataset

Train	airline			
Test	insurer			
Overall Accuracy	39.18%			
	true 0 - 30	true 30 - 60	true 60 - 90	*precision*
pred 0 -30	32	30	35	*32.99%*
pred 30 - 60	0	0	0	*0.00%*
pred 60 - 90	17	36	44	*45.36%*
recall	*65.31%*	*0.00%*	*55.70%*	