HEC MONTRÉAL

Un cadre d'apprentissage automatique appliqué à des données oculométriques décontextualisées

Par
Jonathan Vallée

Sciences de la gestion
(Intelligence d'affaires)

Mémoire présenté en vue de l'obtention
du grade de maîtrise ès sciences
(M. Sc.)

Avril 2015

Résumé

Cet article propose un cadre d'apprentissage automatique basé sur des données oculométriques décontextualisées appliqué à la prédiction de caractéristiques d'utilisateurs d'un logiciel informatique, tel qu'un fureteur internet. Nos recherches montrent que ce cadre nous permet de faire des prédictions à propos de ces utilisateurs. Par exemple, nous sommes en mesure d'inférer s'il est un nouvel usager ou s'il a déjà visité un site internet avec un taux de classification aussi élevé que 85.26%, ce qui couvre 47.75% de l'écart entre le taux de classification naïf et une classification parfaite. Nous proposons également un nouvel hyper-paramètre spécifique à la recherche oculométrique, le délai de fenêtre d'extraction, qui nous permet de contrôler pour des sites web qui chargent lentement. Finalement, nous proposons une nouvelle mesure, adaptée du ratio horizontal-vertical d'Holmqvist 2011, l'angle absolu transformé, qui nous permet, entre autres, d'inférer si l'utilisateur est principalement en train de lire ou de naviguer.

Mots clés : mouvement des yeux, extraction de caractéristiques, sélection de caractéristiques, régression logistique, modélisation des utilisateurs.

Abstract

This paper proposes a machine learning framework based on decontextualized eye tracking data to make predictions about user's characteristics. Our research shows that this machine learning framework works to make predictions about a user, namely if the user has previously visited a website with a classification rate as high as 85.26% which covers 47.75% of the possible improvement between the naïve rate and the perfect classification. We also propose a new eye tracking data hyperparameter, the extraction window lag that aims at controlling for slow loading websites and help mitigate the noise in the data. Finally, we propose a new eye tracking feature, adapted from Holmqvist 2011's HV ratio, the transformed absolute angle, that can, for example, help us infer if the user is mostly reading or browsing.

Keywords: eye movements, feature extraction, feature selection, logistic regression, user modeling.

Remerciements

Au terme de ce projet, j'aimerais remercier ceux qui ont rendu ce mémoire possible grâce à leurs expertises et à leurs encadrement même en période de recherche plus difficile, mon directeur de mémoire, M. Marc Fredette et mon co-directeur de mémoire, M. François Courtemanche. Je suis très reconnaissant pour le temps investi dans ma recherche et pour leurs précieux conseils.

J'aimerais également remercier les membres du jury qui ont accepté d'évaluer ce travail.

Table des matières

Chapitre I

Introduction

Les oculomètres

Principalement utilisés à des fins de recherche en expérience utilisateur, les oculomètres sont en voie de devenir des outils indispensables pour comprendre les individus utilisant des interfaces. En ce sens, Nielsen et Pernice ont conduit une expérience d'utilisation du web sur plus de 300 participants et ont produit un large éventail de recommandations pour les concepteurs web. Ils ont montré que l'utilisation de dispositifs d'oculométrie est un facteur clé de succès dans ce type de recherche.

Dans le contexte d'affaires actuels où la satisfaction des consommateurs commence par la personnalisation de leur expérience (Potey and Sinha 2015), il est primordial de comprendre quelles sont les différences dans l'utilisation d'une interface entre les différents groupes d'utilisateurs, segmentés selon des caractéristiques diverses telles que l'expérience d'utilisation de la dite interface, le genre ou l'âge des individus.

Les données oculométriques

Lorsqu'un utilisateur web est devant son écran, il est possible de capturer des données oculométriques en utilisant un oculomètre calibré aux yeux du dit utilisateur. Ces données se divisent en deux groupes principaux : les données décontextualisées et les données contextualisées :

a) Les données décontextualisées

Les données décontextualisées sont celles de base qui sont capturées par l'oculomètre. Ce sont les mesures des yeux telles que l'emplacement du regard, transformé en coordonnées x et y et la taille de la pupille. Ces données sont complètement indépendantes de l'interface qui est utilisée.

Dans le même ordre d'idées, l'oculomètre est en mesure de classifier si l'utilisateur est en période de saccade ou de fixation à partir, entre autres, de l'algorithme « I-VT fixation filter » (Salvucci et al. 2000) et de paramètres définis par le chercheur. Nous ferons une utilisation exhaustive de cette classification fournie par l'oculomètre.

Finalement, l'oculomètre attribue aussi une cote de validité à chacune des observations qui indique s'il était confiant, lors de l'observation, d'avoir bien identifié chacun des yeux et sa position. Cette information est cruciale pour éliminer des données qui seraient bruitées et nuisibles pour l'analyse subséquente.

b) Les données contextualisées

Les données contextualisées ajoutent aux données décontextualisées une couche d'information qui permet de mieux comprendre les utilisateurs et l'environnement dans lequel il évolue lors de la séance de navigation. Ces données sont générées à partir des coordonnées x et y du regard de l'utilisateur combinées à des zones d'intérêt, souvent configurées par le chercheur, mais qui peuvent être automatiquement identifiées comme l'ont montré Privitera and Stark 2000 et Grindinger et al. 2010. Ces zones d'intérêt sont définies à partir des coordonnées des points formant un prisme autour d'une zone sur une interface. Par exemple, un carré pourrait être formé autour du formulaire d'authentification sur une page web en y indiquant ses quatre sommets. Ensuite, si l'utilisateur pose son regard dans le carré, l'observation sera augmentée par l'étiquette du nom de cette zone d'intérêt.

Les zones d'intérêt, lorsqu'elles sont combinées en séquence, forment une trajectoire de balayage (scanpath) qui permet au chercheur de comprendre, entre autres, comment un utilisateur effectue une tâche à l'aide d'une interface.

c) Les données décontextualisées vs les données contextualisées

Les données contextualisées sont beaucoup plus riches en termes d'information que les données décontextualisées. Cependant, elles contraignent aussi le chercheur au contexte dans lesquelles elles ont été capturées.

Conséquemment, les données contextualisées sont excessivement utiles pour la recherche en expérience utilisateur puisqu'elles permettent de comprendre l'utilisation d'une interface précise. Cette compréhension peut ensuite se transformer en modification de la dite interface pour améliorer son ergonomie.

Les données décontextualisées, quant à elles, peuvent être utilisées pour un large éventail de tâches. Par exemples, elles ont été utilisées en biométrie par Bednarik et al. 2005 et en recherche psychophysiologie par Beatty et Lucero-Wagoner 2000. De surcroît, ces données peuvent permettre de découvrir l'état et les intentions de l'utilisateur pour améliorer son interaction avec l'ordinateur (Hyrskykari 2003). Ce type de données est au cœur de la recherche que nous avons effectuée.

L'apprentissage automatique

L'apprentissage automatique est la discipline du développement d'algorithmes qui permettent d'apprendre à partir d'exemples à identifier des structures (apprentissage non-supervisé) ou à prédire une ou des variables cibles (apprentissage supervisé). Ces algorithmes sont généralement en mesure de s'adapter automatiquement lorsque de nouveaux exemples se présentent et leur performance tend à augmenter quand le nombre d'observations croît. Dans notre travail de recherche, nous nous concentrerons sur l'apprentissage supervisé.

Dans le contexte d'apprentissage supervisé, un algorithme apprend à partir d'une combinaison de variables explicatives et d'une ou des variables cibles. Par l'utilisation d'une mesure de performance et de techniques d'optimisation, l'algorithme minimise son erreur de prédiction sur un jeu de données d'entraînement. Ensuite, afin d'être en mesure de mesurer sa performance hors entraînement et, par ce fait même, sa capacité à généraliser, des prédictions sont effectuées sur des données qui n'ont pas été utilisées pour entraîner le modèle. Ce nouveau jeu de données est appelé le jeu de test. Cette mesure de performance permet d'estimer la capacité de prédiction de l'algorithme en contexte réel.

Les tâches d'apprentissage supervisé se divisent en deux groupes, soit la régression et la classification. La régression est d'apprendre à estimer une variable continue telle que le salaire d'une personne à partir de variables explicatives telles que l'âge, le sexe et le lieu de résidence. En contrepartie, la classification est d'apprendre à classifier des données en un nombre défini de groupes comme par exemple, la discrimination d'utilisateurs entre femmes et hommes à partir de variables explicatives.

Plusieurs types de modèles d'apprentissage automatique existent pour arriver à ces fins. Pour notre recherche, nous nous concentrerons sur les modèles linéaires qui ont l'avantage d'être interprétable comparativement aux modèles non-linéaires qui sont souvent plus performants, mais moins transparents en ce qui a trait aux causes.

Plus particulièrement, nous nous intéresserons à la régression logistique. Le nom de ce modèle porte à confusion puisqu'il permet de segmenter une variable cible en deux groupes distincts, ce qui en fait un algorithme de classification. Généralement, la variable cible sera encodée par

0 ou 1. Le résultat du modèle pour une observation sera interprété comme étant la probabilité pour l'observation d'obtenir au groupe 1.

Par ailleurs, afin de limiter la puissance du modèle et d'ainsi éviter le sur-apprentissage, nous utiliserons la pénalité L1 qui pénalise l'utilisation des variables en ajoutant un terme à la fonction d'erreur du modèle qui est minimisée. Ce terme est la somme des coefficients du modèle mis au carré. Ainsi, seulement les variables suffisamment significatives seront sélectionnées pour expliquer la variable cible.

La prédiction à partir de données oculométriques

La modélisation des utilisateurs est un sujet critique dans le monde des affaires d'aujourd'hui alors que ceux-ci s'attendent à une personnalisation accrue de la part de leurs fournisseurs de services (Potey and Sinha 2015). Cependant, bien que plusieurs chercheurs aient montré la relation yeux-esprit (Tobii 2010), les données oculométriques sont généralement bruitées et ne sont pas facilement utilisables pour la prédiction (Courtemanche et al. 2014).

Conséquemment, il est important d'arriver à modéliser les comportements utilisateurs à partir de données oculométriques bruitées et décontextualisées. C'est pourquoi la création de caractéristique est au cœur de la présente recherche.

Une fois ces caractéristiques extraites des données de base, il est nécessaire d'appliquer un modèle de prédiction et d'interpréter les résultats. Afin d'obtenir des mesures de performance non biaisée, il est primordial d'effectuer la prédiction sur des jeux d'entraînement qui n'ont pas été utilisés pour entraîner le modèle. Lorsque les données sont limitées, la validation croisée

de type « leave-one-out » est un algorithme de choix puisqu'il permet d'extraire le maximum de puissance des données tout en s'assurant que la prédiction soit faite hors-échantillon.

L'expérience Neuroscript

L'expérience Neuroscript a été effectuée avant cette recherche. Nous avons choisi d'utiliser ces données puisqu'elles étaient appropriées pour ce genre de travail et permettaient de rentabiliser l'expérience qui avait été effectuée.

La conception de l'expérience faisait en sorte que deux groupes distincts d'utilisateurs étaient disponibles et se prêtaient donc bien à la classification binaire. De plus, il était facilement possible d'identifier si un utilisateur était récurrent sur un site web ou non, ce qui est généralement très difficile à faire avec certitude en utilisant des jeux de données capturés hors laboratoire.

L'article et son contexte

Cet article a comme objectif d'établir une base de recherche sur la prédiction de caractéristiques utilisateurs à partir de données oculométriques décontextualisées. Cette tâche n'est pas triviale, mais nous étions confiants qu'elle était réalisable compte tenu de la qualité des données qui avaient été récoltées par l'expérience Neuroscript.

a) Le développement requis

Le développement nécessaires à la préparation des données, à l'apprentissage automatique, ainsi qu'à l'analyse des résultats a été fait à partir du langage de programmation Python et représente plus de deux mille lignes de code.

Les librairies scikit-learn (Pedregosa 2011), pandas (McKinney 2012) et numpy (Van Der Walt et al. 2011) ont été d'une utilité capitale afin de réussir à obtenir les résultats présentés.

Les annexes 1 et 2 présentent le code de création des caractéristiques et d'une fenêtre d'extraction respectivement. Ce sont des extraits que nous jugeons des plus importants pour bien présenter le travail effectué.

Comme dans toutes tâches de forage de données, la préparation de celles-ci représente plus de 70% du travail et le code correspondant fait honneur à cette statistique.

J'ai écrit la totalité du code qui a été requis afin de compléter cette recherche.

b) Les résultats

Les résultats de cette recherche sont très encourageant. Étant pionnier dans ce domaine, nous n'avons pas de point de comparaison claire, mais sommes convaincus que nous stimulerons d'autres chercheurs à poursuivre dans cette voie.

Bien que majoritairement positifs, nos résultats restent volatils. Davantage d'efforts de recherche devront être mis sur cette tâche afin de stabiliser la performance des modèles de prévision sur toutes les fenêtres d'extraction puisque pour l'instant, nous observons des mesures en dents de scie, c'est à dire qu'il n'y a pas de tendance à la hausse ou à la baisse lorsque le nombre de secondes par fenêtre d'extraction augmente.

Par ailleurs, la caractéristique que nous avons proposée, l'angle absolu transformé, s'est avérée très intéressante puisque son importance relative est en moyenne de

14.22% sur les fenêtres d'extraction que nous avons utilisées. Afin de s'assurer que cette mesure n'était pas biaisée par la chance, nous avons également ajoutée une caractéristique aléatoire tirée d'une distribution Gaussienne de moyenne 0 et d'écart-type de 1, qui n'a jamais été sélectionnée par les algorithmes de sélection de variable. Ceci nous indique que l'importance relative de cette variable était effectivement significative.

Chapitre 2

A Machine Learning Framework Applied to Decontextualized Eye Tracking Data

A Machine Learning Framework Applied to Decontextualized Eye Tracking Data

Jonathan Vallée
HEC Montreal
3000 Ch. Cote-Ste-Catherine
Montreal, Quebec, Canada
jonathan.vallee@hec.ca

François Courtemanche
HEC Montreal
3000 Ch. Cote-Ste-Catherine
Montreal, Quebec, Canada
francois.courtemanche@hec.ca

Marc Fredette
HEC Montreal
3000 Ch. Cote-Ste-Catherine
Montreal, Quebec, Canada
marc.fredette@hec.ca

ABSTRACT

This paper proposes a machine learning framework based on decontextualized eye tracking data used for user modeling. Our research shows that this framework successfully makes predictions about a user, such as if it has previously visited a website with a classification rate as high as 85.26%. This represents an improvement over the naïve rate of 47.75%. We also propose a new eye tracking data hyperparameter, the extraction window lag that aims at controlling for slow loading websites and help mitigate the noise in the data, and a new feature, the transformed absolute angle, that models if the user is mostly reading or browsing.

Keywords

eye movements, feature extraction, feature selection, logistic regression.

1. INTRODUCTION

Eye trackers are becoming common in everyday life from user experience research tools, biometrics devices to gaze-based interface. It is a technology that provides a window to the cognitive processes while being unobtrusive. It can also empower users to make actions without using their hands (Eivazi and Bednarik 2011). For non-academic tasks, innovations such as Tobii Eye Experience (Tobii 2014) enable developers to use eye trackers embedded in devices. Thus, with commercial and scientific applications expanding, eye tracking technology is getting increasingly adopted.

The main research area of eye tracking devices is user experience. Nielsen and Pernice 2010 have conducted an in-depth web usability research on more than 300 participants and have extracted a wide array of recommendations for web designers. They have shown how the use of eye tracking devices is a key success factor in user experience research.

Subsequently, being able to make predictions about a software user is another key task in applied eye-tracking research. Successfully predicting a user's gender, if he is a new, a returning, a novice or an expert user or any otherwise unprocurable statistics can be leveraged by intelligent systems such as adaptable user interface (UI), targeted marketing software and websites.

To accomplish that, many researches have shown the existence of the eye-mind relationship. For example, it is possible to understand the user's behavioral, cognitive and emotional state from the variations in the pupil size (Bednarik et al. 2012). Pupils can also be used to tell if a user is retrieving information from its long-term memory, that result in higher pupil dilatation, called TERP - task-evoked pupillary response (Beatty and Lucero-Wagoner 2000). Additionally, the number of fixations during a period of time can tell if a user is confused while performing a task (Tobii 2010). Many other features, such as the sum, the mean and the standard deviation of fixation durations, saccade lengths and relative and absolute saccade angles, have been tested in the eye-tracking literature in order to describe what the user is feeling or thinking (Toker et al. 2013, Bednarik et al. 2012).

Similarly, advances in the field of adaptable UI have been made by Toker et al. 2013 who characterized users' perceptual speed, visual and verbal working memory and expertise with different types of graphics to determine if relations exist between those variables and the users' attention patterns. The paper suggests that some of these characteristics are detectable by eye tracking features and that they influence gaze behavior.

Other applications such as proactive computing, which tries to predict the user's action with gaze analysis and attempts to perform those actions on behalf of the user, shows tremendous promises to simplify the use of computers and mobile devices (Hyrskykari et al. 2003).

Those examples clearly show that there is a usable relationship between eye movements and users' characteristics and that this relation can be leveraged by intelligent systems. Eye tracking data can uncover the user's state and its intentions so it can be used to enhance human-computer interaction (Hyrskykari et al. 2003).

Building on this body of knowledge, this paper proposes a framework for using decontextualized eye tracking data for user's characteristics inference. With today's acquisition velocity, being able to leverage decontextualized eye tracking data is crucial. The benefits of having a model that can be trained on labeled data and improved with time as more labeled observations are gathered without a need for a context are enormous since users are and will continue to be demanding more personalized services (Potey and Sinha 2015). Being able to understand who they are without their intervention will enable quick personalization, adaptable user interface and consequently, user satisfaction. User centered and automatically adapting websites will become possible by using our framework.

To that end, this paper makes the following contributions:

- We propose a general framework to use decontextualized eye tracking data that can be applied to a wide array of prediction tasks
- We propose a new eye-tracking hyperparameter, the extraction window lag, used to better select time windows
- We propose set of manually engineered features and a new one, the transformed absolute angle, inspired by Holmqvist 2011's HV ratio measure, to model the user's reading or browsing behavior
- We propose a taxonomy of eye tracking features to better classify those features
- We perform an experimental validation on a dataset to show how to apply the framework to a prediction task

To present our work, we will first explain the key differences between contextualized and decontextualized data in section 2, review previous work on using eyes for classification tasks in section 3, present our framework and our manually engineered features in section 4, apply our framework on an experimental dataset in section 5 and conclude with a discussion and the conclusion in section 6.

2. CONTEXTUALIZED VERSUS DECONTEXTUALIZED EYE TRACKING DATA

Contextualized eye tracking data relies on knowledge of the user interface that is being looked at by the user. This information is transmitted to the eye tracking processor by the use of the interface's regions of interest (AOI) such as the login form, the menu bar and the product description. As shown in figure 1, it enables the eye tracking processor to infer what is being looked at from the x, y positions the eye tracker is capturing. For example, if the user is looking at the login form that the researcher has previously flagged as AOI1, the eye tracker will attach a AOI1 label to the observation. This helps researchers understand the user's path between multiple key areas of their interface. To do so, researchers refer to scanpaths that represent the sequence of the visited AOIs.

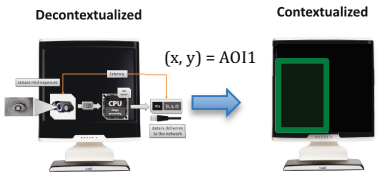

Figure 1: The Eye Tracking Process. This figure shows the contextualization process from the raw eye tracking data to the processed information. This image has been modified from a version that can be found in Tobii's user manual.

In comparison, decontextualized eye tracking data does not tell anything about the interface that is being looked at by the user. It strictly comprises the raw eye tracking measures that are being captured while the user interacts with the software. As a result, decontextualized data is completely interface agnostic.

Most of the research has been using contextualized data since it carries a lot of information. For example, knowing that the user is looking at the login form might tell us that he is a returning user trying to get in a secure area of the website.

The issue with contextualized data is that it requires the researcher to process the context of the user's browsing session. It is then tied to this context.

3. THE EYE AS A CLASSIFIER

Classifying users with contextualized eye tracking data has often been done in eye tracking research but continues to be difficult mainly due to the relatively high level of noise in the data (Courtemanche et al. 2011). For example, Liu et al. 2009 have attempted to discriminate between expert and novice users by using scanpath data. Novices and experts have been found to be different in their eye movement profiles.

String editing has often been used in scanpath comparison. It has been pioneered by Privitera and Stark (Privitera and Stark 2000). Letters are assigned to AOIs and the resulting scanpaths are compared. The researchers proposed a list of algorithms to identify AOIs and then used clustering methods to group related AOIs together.

Unfortunately, to be able to compare scanpaths, they must come from the same source such as the same website for web

browser users. In addition, this technique does not take into account the time dimension of a scanpath which means that two very similar eye trajectories can in fact be very different when you include time in the analysis. Finally, the method is very sensible to the distance metric used to compare scanpaths.

Grindinger et al. 2010 have extended Privitera and Stark's work by using the Levenshtein distance to perform a pairwise scanpath comparison. The Levenshtein distance computes the cost of transforming one string to another one by simple operations - deletions, insertions and substitutions. It is similar to the method used by Privitera and Stark but does not require the researchers to assign costs to the operations. They also used a different clustering algorithm to identify AOI: the mean-shift algorithm instead of the k-means that previous research was using**.**

Regrettably, scanpath based methods are not decontextualized. In the next section, we will provide a framework to classify users out of context, with their eye measures only, without using AOIs. This approach enables an array of real-world applications. As previously stated, this is becoming critical with today's speed of data acquisition and the shortage of data science, statistical and machine learning talent in the industry (Manyika 2011).

4. FRAMEWORK

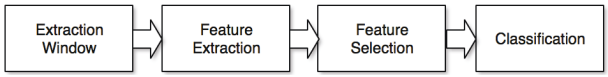

Figure 2: The Machine Learning Framework Applied To Decontextualized Eye Tracking Data.

Figure 2 shows the proposed machine learning framework that is composed of four steps. Firstly, given a user browsing session's data, we need to identify the time windows we are going to use to create the features, choose its length represented by the number of seconds of data we are going to consider and choose the number of seconds of lag before we start capturing data in order to control for slow loading websites. Secondly, we need to describe the user's gaze behavior by a set of features extracted from its related dataset. Then, in order to reduce noise and prevent overfitting, we need to select the most informative features. Finally, once the features are selected, we need to fit a prediction model to the data and assess its classification performance by using cross-validation.

Extraction Window

Selecting the extraction window used to create features is the first step of the framework. There are two business imperatives that govern this selection process:

- Predictions need to be made quickly since users may spend a limited number of seconds on a website before leaving it.
- Because of to the small number of observations we can gather in a limited number of seconds, the data needs to be as clean as possible.

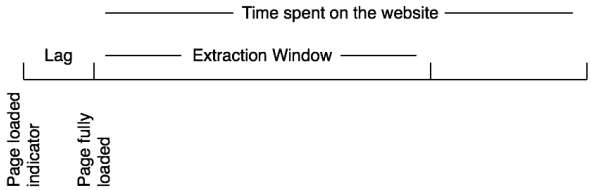

Figure 3: Extraction Window Selection.

To be able to react quickly once a prediction is made about a user, we aim at using the shortest extraction window possible while preserving prediction accuracy. As shown on figure 3, the extraction window lag plays an important part to lower noise in the dataset. Indeed, even though the page seems loaded to the eye tracker, media and heavier content is often not. The lag should be selected based on the dataset at hand, depending on the content that is being loaded on the page. For media heavy websites, the lag should be longer than for text-based ones.

The extraction window's length is an important hyperparameter of this framework. It needs to be short enough for fast prediction while long enough to be able to extract meaningful information about the user's gaze behavior. We suggest testing a range of acceptable lengths by using cross-validation and selecting the one with the best performance.

Feature Extraction

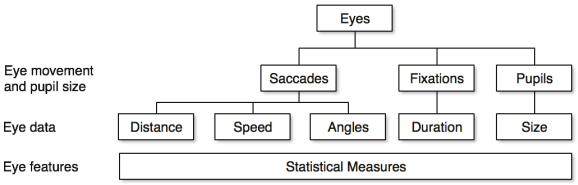

Figure 4: Hierarchical view of the eye measures.

In order to deduce good information about a user's cognitive process, a large number of features must be created by calculating descriptive statistics from the eye fixations, saccades, pupil size and gaze pattern during the selected time window (Salojarvi et al. 2005).

Features can either be learned from data or created manually. In order to automatically learn good features, a lot of data is required. If available, neural network type algorithms such as the Restricted Boltzmann Machines (Smolensky 1986) or

Auto-Encoders (Lecun 1987 ; Bourlard et Kamp 1988 ; Hinton et Zemel 1994) can be used. Those algorithms are mainly used for image recognition tasks because the structure of the data yields a good representation of the learnt features. Structure comes from the fact that pixels in the same neighborhood are correlated. Without much structure, as it's the case with eye tracking data, learnt features may be harder to interpret.

In the case where there is not a lot of data available and that interpretability is required, features should be manually engineered.

As shown on figure 4, every eye tracking dataset row includes the gaze's (x, y) coordinates, the type of the observation [fixation, saccade, unknown], the pupil sizes and the fixation duration. Then, from this data, we can compute distance speed, angles between saccades and a number of statistical measures to describe the preceding data points. In addition, in order to minimize the number of correlated features, we suggest to average the left and right pupil sizes to only one variable.

To organize our set of features, we propose a taxonomy that separates them in four groups. The first group includes the features that are descriptive and that can be calculated in one step, from the raw data. We refer to this group as "Descriptive Features". The second group of features includes features that are calculated on specific time series measures and that result in only one statistic - such as a ratio that describes the time window. We refer to this group as "Specific Time Series Features". The third group of features includes descriptive statistics of all measures that have first been extracted has time series. We refer to this group as "Time Series Features". The last group of features includes descriptive statistics that have been calculated on time series measures that have themselves been generated by time series of measures. We refer to this group as "Second Order Time Series Features". Our proposed feature set is calculated as follows:

Let *n* be the number of rows of a time window's dataset that is composed of a number of valid observations obtained by filtering out the invalid ones. Let *t* be the type of observations. Observations can either be a valid fixation *f*, at location *(x, y)* or a saccade *s*. (Invalid observations occur when the eye tracker is not confident that it has been able to identify the eyes. In addition, the length (*n*) of the time window dataset depends on the eye tracker's sampling frequency. For example, using an eye tracker that samples at 60Hz, a perfectly valid five seconds time window would have 300 rows.)

Descriptive Features

- Fixation and saccades features help us understand if the user is spending more time actively moving his eyes on the screen, which might be a result of confusion, or concentrating his gaze on specific areas of interest:
    a. **Number of fixations:**
    $$\sum_{i=1}^{n} \mathrm{I}(t_i == f) \quad (1)$$
    b. **Number of saccades:**
    $$\sum_{i=1}^{n} \mathrm{I}(t_i == s) \quad (2)$$
    c. **Proportion of fixations:**
    $$\frac{Number\ of\ fixations}{n} \quad (3)$$
    d. **Proportion of saccades:**
    $$\frac{Number\ of\ saccades}{n} \quad (4)$$
- The direction counts will tell us if the main directions the user is going. If the eyes are still during the whole time window, the following four features will have values of 0:
    a. **Number of times eyes went up:**
    $$\sum_{i=2}^{n} \mathrm{I}\,(y_i > y_{i-1}) \quad (5)$$
    b. **Number of times eyes went down**
    $$\sum_{i=2}^{n} \mathrm{I}\,(y_i < y_{i-1}) \quad (6)$$
    c. **Number of times eyes went right**
    $$\sum_{i=2}^{n} \mathrm{I}\,(x_i > x_{i-1}) \quad (7)$$
    d. **Number of times eyes went left**
    $$\sum_{i=2}^{n} \mathrm{I}\,(x_i < x_{i-1}) \quad (8)$$

Time Series Measures

- **Mean pupil size:** As previously stated, this measure is the mean of the left and right pupil size. Pupil sizes are in millimeters and are estimated by the eye tracker's algorithm.

$$\frac{(left\ pupil\ size_i + right\ pupil\ size_i)}{2} \forall\ i \in n \quad (9)$$

- **Saccade length:** The Euclidean distance between two fixations tells us if the user's gaze is more focused or spread all around the web page.

$$\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \forall\ i \in \{2{:}n\} \quad (10)$$

- **Absolute angles:** Figure 5 shows the absolute angles that will help us describe the direction of the eye between saccades. Absolute angles are calculated as follows:

$$\theta = \tan^{-1}\frac{b}{a}\ \forall\ i \in \{2{:}n\} \quad (11)$$

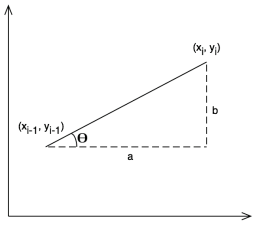

Figure 5: Absolute angle

- **Transformed absolute angles:** Since we are mostly interested by the behavior of the user, we chose to transform the absolute angles to the first Euclidean quadrant as another feature. This will tell us if the angles are mostly flat (around 0°) or are vertical (around 90°) and help us understand if the user is mostly reading or browsing. This feature is inspired by Holmqvist 2011's HV ratio since it's trying to describe the same kind of behavior but is better adapted to our framework by giving us the ability to describe it statistically with time series features.

Algorithm to transform absolute angles

Output: *transformed_angle*

```
if 90 < angle <= 180 then
    transformed_angle ← 180 – angle;
else if 180 < angle <= 270 then
    transformed_angle ← angle – 180;
else if angle > 270 then
    transformed_angle ← angle – 270;
else
    transformed_angle ← angle;
```

- **Relative Angles:** figure 6 shows the relative angles that will help us describe if the saccades are mostly linear, which would results in angles close to 180°, and are calculated as follows:

$$\theta = \cos^{-1}\frac{a^2 + b^2 - c^2}{2ab} \quad (12)$$

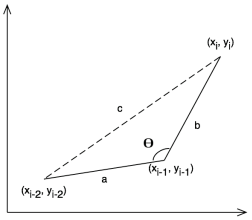

Figure 6: Relative angle

Specific Time Series Features

- **Horizontal-Vertical Ratio:** Holmqvist et al. suggest that a normalized version of this measure can be used to discriminate between users with different reading directions (Holmqvist et al. 2011). In our case, this measure tells us if the user spends more time reading (horizontal angles) than browsing the website (vertical angles). To discriminate between vertical and horizontal angles, we used this rule (shown on figure 7): an absolute angle is considered horizontal (H) if it's between 135° and 225° or 315° and 45°. Otherwise, it is considered vertical (V).

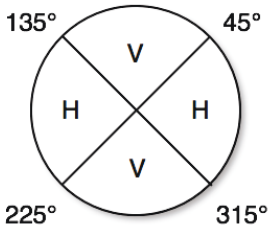

Figure 7: Horizontal and vertical angles

$$\frac{\sum_{i=1}^{m} \mathrm{I}(H_i)}{\sum_{i=1}^{m} \mathrm{I}(V_i)} \quad (13)$$

where m is the length of the absolute angle time series.

This feature calculation could be invalid if the absolute angle time series contains no vertical angle. In that case, the H/V ratio is set to 1.

- **Total saccades length:** By calculating the total Euclidean distance between points, this measure will tell us if the user is more focused on one website area or looking all around.

$$\sum_{i=1}^{m} saccade\ length_i \quad (14)$$

where m is the length of the saccade length time series

- **Latency to the maximum value of the pupil size:** as proposed by Beatty and Lucero Wagoner in the calculation of TERP (Beatty and Lucero-Wagoner 2000).

$$argmax(\text{mean pupil size}) \quad (15)$$

Time Series Features

Let $\boldsymbol{X}$ be a collection of time series comprising *mean pupil sizes*, *fixation durations*, *saccade lengths*, *relative angles*, *absolute angles, transformed absolute angles.* Let x_{ij} be the i^{th} element of the j^{th} time series of $\boldsymbol{X}$, $\boldsymbol{L}$ be the vector of the lengths of the time series, l_j be the length of the time series j and $\boldsymbol{X}_j$ be the j^{th} time series. Time series don't necessarily have the same length.

Each time series features will yield a descriptive measure for every time series.

- Minimum of time series

$$min(\mathbf{X}_j) \ \forall j \quad (16)$$

- Maximum of time series

$$max(\mathbf{X}_j) \ \forall j \quad (17)$$

- Mean of time series

$$\mu_j = \frac{1}{l_j}\sum_{i=1}^{l_j} x_{ij} \ \forall i \ \mathcal{E}\ \{1: l_j\}, j \in \mathbf{V} \quad (18)$$

- Standard deviation of time series

$$\frac{1}{l_j - 1}\sum_{i=1}^{l_j}(x_{ij} - \mu_j)^{\frac{1}{2}} \ \forall i \ \mathcal{E}\ \{1: l_j\}, j \in \mathbf{V} \quad (19)$$

- Skewness of time series

$$\frac{\sum_{i=1}^{n}\frac{(x_{ij} - \bar{x_j})^3}{n}}{{\sigma_j}^3} \ \forall j \quad (20)$$

where σ in the standard deviation of the time series j.

- Kurthosis of time series

$$\frac{\sum_{i=1}^{n}\frac{(x_{ij} - \bar{x_j})^4}{n}}{{\sigma_j}^4} - 3 \ \forall j \quad (21)$$

where σ in the standard deviation of the time series j.

Second Order Time Series Measures

- First variation of time series

$$\frac{x_{ij} - x_{i-1,j}}{x_{i-1,j}} \ \forall i \ \mathcal{E}\ \{2: l_j\}, j \in \mathbf{V} \quad (22)$$

- Second variation of time series

$$\frac{x_{ij} - x_{i-2,j}}{x_{i-2,j}} \ \forall i \ \mathcal{E}\ \{3: l_j\}, j \in \mathbf{V} \quad (23)$$

Second Order Time Series Features

The minimum, the maximum, the mean and the standard deviation are computed on the first and second order time series.

Resulting Feature Set

Table 1 summarizes the set of 95 features that describe the user's eyes measures and behavior. The features are then standardized by the *z-score*: $x'_{ij} = (x_{ij} - \mu_j)/\sigma_j$ where μ_j *and* σ_j are the j^{th} time series' sample mean and standard deviation. This statistical transformation makes the features in the same range of numbers so our prediction algorithm should converge more quickly.

Once we have all our standardized features, the next step is to compute interaction variables. We propose to calculate all second order interaction variables by multiplying all pairs of features selected by the univariate p-value filter that we will present in the subsequent section. Thus, the total number of features will be the 95 manually engineered features plus the corresponding interaction variables.

Feature and measure Group	Features	Number of features
Descriptive Feature	Number of fixations, number of saccades, proportion of fixation, proportion of saccades, number of time the eyes went up, number of time the eyes went down, number of time the eyes went left, number of time the eyes went right	8
Time series measures	Mean pupil sizes, fixation durations, saccade lengths, relative angles, absolute angles, transformed absolute angles	

Specific time series features	Horizontal-vertical ratio, total saccades length, latency to the maximum value of the pupil size	3
Time series features	Minimum, maximum, mean, standard deviation, skewness, kurthosis of time series measures	36
Second order time series measures	First variation of time series, second variation of time series of the time series measures	
Second order time series features	Minimum, maximum, mean, standard deviation of second order time series measures	48
Total		**95**

Table 1: Features table

Feature Selection

Even though adding all calculated features to the dataset is the intuitive thing to do, it increases the data dimensionality and adds noise to the relevant information. Furthermore, as described in the previous section, a lot of those features are modeling the same user behavior. Hence, it's crucial that we select only the most useful ones to train our algorithm and make our predictions (Guyon and Elisseeff 2006).

Feature selection algorithms are separated in three classes: wrappers, filters and embedded. Wrappers utilize the learning algorithm as a black box to select a subset of features according to their predictive power. Filters select subsets of features as a pre-processing step, independently of the chosen learning algorithm. Embedded methods perform feature selection while training and are usually specific to given learning algorithm (Guyon and Elisseeff 2003).

Ranking and selecting features based on their individual relevance is considered fast and effective, particularly when the number of features is large and the number of available training examples comparatively small (Guyon and Elisseeff 2006).

Thus, in our framework, we propose to use a univariate p-value filter followed by a L1 penalized classification algorithm in order to deal with the great number of features.

There are two reasons to select the features in two separate steps. First, when the number features is large considering the number of training examples, a univariate p-value filter can be used as a pre-processing step to reduce the number of features to include in the learning algorithm. To do so, it selects the features with p-values below a certain threshold. Second, using a model agnostic feature selection algorithm enables us to evaluate if any of our proposed features are significant.

Classification

The next step is to choose the model to be used for classification. Many models should be tested and compared on out-of-sample prediction performance to select the most suited one for the prediction task at hand. Models have different properties and will yield different results depending on the context and on the quality of the data that is used. For example, when it's important to understand which feature plays an important role in the prediction, it is more appropriate to select a linear model since regression coefficients can be interpreted. When prediction performance is more important, it can be interesting to test if non-linear models such as artificial neural networks perform better.

Similarly, in order to preserve prediction performance on out –of-sample data, we suggest to use cross-validation when training the models. This method separates the dataset in n folds, iterates on all of them using all but one to train the model and measures its performance on the remaining fold. This technique helps to extracts the most information possible from small datasets while measuring performance on out-of-sample data.

Furthermore, it is important to select a model performance metric that represents what the researcher expects from the model. For example, in a classification task, if one of the classes is rare, good classification might not represent a good model. Therefore, a measure like the sum of specificity and sensitivity is interesting.

$$Sensitivity = \frac{TP}{(TP + FN)} \quad (24)$$

$$Specificity = \frac{TN}{(FP + TN)} \quad (25)$$

where TP, FP, TN and FN are true and false positives and true and false negatives taken in the resulting confusion matrix.

Confusion Matrix		Predicted Class	
		0	1
Actual class	0	TN	FP
	1	FN	TP

Table 2: Confusion Matrix

Similarly, Senecal et al. 2014 proposed a metric to evaluate model performance with imbalanced dataset. The improvement over naïve rate percentage is computed as such:

$$\frac{\varrho - \rho}{1 - \rho} \qquad (26)$$

where

- ϱ is the model accuracy
- ρ *is the proportion of the majority class*

This measure is very useful because it illustrates the proportion of the available prediction accuracy that the model is able to capture. For example, if the majority class has a proportion of 75%, the maximum improvement available is 25%. If the model accuracy is 87.5%, it grabbed 50% of the available accuracy.

$$\frac{87\,5\% - 75\%}{1 - 75\%} = 50\% \qquad (27)$$

Depending on the context, it is sometimes necessary to assign different costs to the different misclassifications. For example, it might be worse to misclassify a user as new when it's a returning than the contrary so we would assign a higher cost to this misclassification in the cost function. Since everything in machine learning revolves around the cost function that is being minimized, it's essential to design it to reflect the context in which the framework is being applied.

5. EXPERIMENTAL VALIDATION

THE EXPERIMENT

To test our framework, we performed a lab experiment on 76 participants. Subjects were provided with a prepaid credit card and were asked to buy songs on a number of music websites. Two subject groups were created. The first one, labeled *intrasite* were asked to always shop on the same website while the second one, named *intersite* were visiting a number of different music websites. To control for fatigue, each session lasted for at most an hour. Participants did not know the length of their experience.

Websites were selected based on three criteria:

- They had to allow users to buy and download single songs
- They had to be available to Canadian consumers
- They had to score differently on the electronic service quality scale (Bressolles and Nantel 2008)

To assign the electronic service quality score, three experts were asked to buy a song on every selected website and assess its ease-of-use, aesthetics, information quality, and interactivity dimensions. The results of this assessment show that the selected websites cover a wide array of scores on the Bressolles and Nantel 2008's scale.

Finally, an eye-tracker was used to capture the subject's raw eye tracking measures.

THE APARATUS

A Tobii X-60 (Tobii Technology AB) eye tracker was used to record users' eye movement and pupil diameter at 60Hz during the experiment. A nine points calibration has been performed for all participants and was repeated until sufficient accuracy was achieved. As indicated by Dimigen et al. 2011, current eye trackers have a spatial resolution of up to 0.01o/ 2 kHz. While it is not required to have such resolution for most information systems research contexts, an accuracy level of approximately 1 cm around the calibration points can be achieved with adequate calibration. The Tobii implementation of the I-VT fixation filter algorithm (Salvucci et al. 2000) was used to extract fixations from the eye tracking data (minimum fixation duration = 60 ms). The following parameters were used for fixation merging: maximum angle between fixations: 0.5 degrees, maximum time between fixations: 75 ms.

THE APPLIED FRAMEWORK

Extraction Window

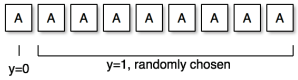

Figure 8: The experimental design and the labeled time windows for the intrasite group.

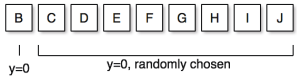

Figure 9: The experimental design and the labeled time windows for the intersite group.

Figure 8 and 9 show how we labeled our examples. Our target can take two values: new and returning visitor. Based on our experimental design, we selected all participants' first time window and assigned it the value of y=0, meaning a new visitor. Then, if the user was from the *intrasite* group, we randomly selected one of the returning visit to the same website and labeled this time window as y=1, meaning returning visitor. Finally, if the user was from the *intersite* group, we randomly selected a visited website that was not the first and labeled it as y=0.

The output from this phase is a set of 152 time windows each labeled with a target value that represents if a user is new to the website (y=0) or if a user is returning to the website (y=1). About 75% of those time windows are new visitors and the remaining are returning visitors. A user is therefore

two times in the dataset but at most one time as a returning individual.

We have tested intervals spanning 2 to 15 seconds with a step of 1 second. All time windows start 4 seconds after the URLStart marker that Tobii Studio places when the website is loaded to control for slow loading websites and capture only relevant eye tracking data. We chose a lag of 4 seconds based on our observation of the web page loading time. Different studies may benefit from shorter or longer lags.

Results per time window length are presented in table 3.

Feature Extraction

The features were created in accordance with the methodology we proposed except that we added a noise feature to measure the transformed absolute angles true impact in comparison to randomness.

The noise feature was generated by randomly drawing observations from a Gaussian distribution with a mean of 0 and a standard deviation of 1.

Feature Selection

As a first step, a univariate p-value filter was applied in order to drop all features that were not significant to classify between returning and new visitors.

Then, all pairs of significant variables were combined to create interaction terms that were added to the feature set.

Subsequently, the L1 penalty based classification model only assigned regression coefficients to the features that were the most informative.

The number of selected features per time window per algorithm is presented in table 3.

Classification

To be able to interpret the results and since our target can only take two values, we chose to use L1 penalized logistic regression as our prediction model.

As suggested in the framework definition, in order to prevent overfitting, we used a leave-one-out cross-validation algorithm to train and test our algorithm. Leave-one-out is cross-validation with the number of folds set to the length of the dataset. This extreme variation of cross-validation can only be used with small datasets because of the computing power required to train a model for every fold.

Finally, we measured performance on all predicted values that were out of every training iteration. The best results we obtained are shown in table 3. To present our results we chose the cutpoints that generated the best classification on the training sets.

Results

Table 3 shows the out-of-sample results of the leave-one-out logistic regression algorithm trained on the different time window lengths. The leave-one-out cross-validation algorithm has been run separately on every time window's length.

To assess the model performances, we report the improvement over naïve rate, the classification, the AUC, the sensitivity, the specificity and the sum of sensitivity and specificity.

Our results show that it is possible to use decontextualized data to predict characteristics of a user. The best improvement over naïve rate, classification, AUC, specificity and sum of sensitivity and specificity results in this test case have been achieved on a twelve seconds extraction window.

Also presented in table 3 are the numbers of features selected at least once by the univariate filter and by the L1 penalized logistic regression algorithm. To count the number of features selected by the L1 penalized logistic regressions, compute the sum of the absolute values of the betas of all leave-one-out trained logistic regression per time window length. Then, if the sum of the absolute values of the betas is greater than 0, add 1 to the number of features selected by L1 penalty.

Then, the importance of feature groups and types by the two feature selection algorithms is also presented in table 3. Feature importance is calculated as follows. First, if the feature is an interaction feature, get the group of both features. Once the groups and types are known, calculate the number of times each group and type is being selected for every time windows. Then, compute the weighted importance by dividing the count of feature groups or types by the proportion of that group or types (for example, the proportion of the "Descriptive Features" group is 8/95). Finally, normalize the importance by dividing it by the total of the importance for that time window.

A clear pattern emerges from the importance analysis. "Specific Time Series Features" have the most importance based on the L1 penalized logistic regression 11 of the 14 times. The total distance travelled by the eyes feature is the most important of those. This is consistent with the hypothesis that the more the eyes move, the more confused, hence less expert, the user is.

Finally, we report the relative importance of the transformed absolute angle variable. We compared it with a noise variable to evaluate if it could have been selected strictly by chance. Our results show that a noise variable has no relative importance while our transformed absolute angle has relative importance between 6.63% and 14.84% depending on the time window's length.

6. DISCUSSION AND CONCLUSION

We proposed a machine learning framework to use decontextualized eye tracking data to make user characteristic predictions. Being able to use out of context data is key to the future of eye tracking. As the technologies get increasingly adopted, the amount of data gathered will be exponential. Being able to use predictive algorithms quickly and without the need for data analysis will be a key success factor for those who try to leverage eye tracking data to model and classify users. Our experiment showed promising results with improvements over the naïve rate as high as 47.75%.

To achieve those results, we proposed a hyperparameter specific to eye tracking data, the extraction window lag that enables the researcher to exclude a number of non-relevant seconds where the web page is still loading.

Adapted from Holmqvist 2011, we also proposed a feature that illustrates if the user is reading or browsing, the transformed absolute angles. The relative importance of the transformed absolute angle features was 14.22% on average, which proves to be very significant.

While promising, performance results vary between time windows' lengths and no trend could easily be identified. More research is needed to assess if other features could stabilize and yield better performance.

As a potential solution to those volatile results, applying the framework on larger datasets might produce better and more stable prediction results. However, using leave-one-out cross validation on a larger dataset will be computationally intensive. Therefore, as the datasets get larger, reducing the cross-validation number of folds will be necessary. A tenfold cross-validation is often used in the machine learning literature.

Also, by using larger datasets, automatic learning of representations should be studied to gauge if learnt features have good prediction power even though the high level of noise and lack of structure in the eye tracking datasets.

Another area of improvement is to include multi-resolution windows to create the features optimally (Courtemanche et al. 2014). To do so, one has to select the best performance time window length per feature.

Finally, in order to apply this framework to other datasets, the extraction window lag should be calibrated based on the dataset at hand. In our experiment, we used a four seconds lag since we dealt with media intensive websites. Depending on the content, discarding data with a lag could hurt the classification results.

Number of seconds	2	4	6	8	10	12	14
General performance metrics							
Improvement over naïve rate	0.00%	5.13%	30.96%	20.93%	0.00%	**47.75%**	13.93%
Classification	72.32%	74.31%	80.67%	77.78%	71.61%	**85.26%**	76.28%
y=0 proportion	72.32%	72.92%	72.00%	71.90%	71.61%	71.79%	72.44%
AUC	81.16%	78.00%	86.90%	86.96%	79.98%	**90.48%**	84.52%
Sensitivity	58.06%	61.54%	76.19%	76.74%	70.45%	79.55%	55.81%
Specificity	77.78%	79.05%	82.41%	78.18%	72.07%	**87.50%**	84.07%
Sensitivity + Specificity	1.36	1.41	1.59	1.55	1.43	**1.67**	1.40
Number of significant features selected with univariate filter	120	182	170	226	142	102	199
Number of features selected by L1 penalized logistic regression	46	62	44	47	51	58	62
Importance of groups of features selected by the univariate filter							
Descriptive Feature	7.91%	17.09%	22.66%	26.73%	21.54%	7.18%	18.29%
Time Series Features	20.22%	20.39%	**32.90%**	20.79%	**33.75%**	25.52%	**28.98%**
Specific Time Series Feature	**38.68%**	28.79%	32.23%	**30.99%**	25.85%	28.71%	27.56%
Second Order Time Series Feature	33.19%	**33.73%**	12.21%	21.50%	18.85%	**38.58%**	25.18%
Importance of types of features selected by the univariate filter							
Feature	**54.65%**	**61.57%**	21.76%	38.74%	**76.08%**	31.76%	**66.31%**
Interaction Feature	45.35%	38.43%	**78.24%**	**61.26%**	23.92%	**68.24%**	33.69%
Importance of groups of features selected by the L1 penalized logistic regression							
Descriptive Feature	8.85%	9.40%	5.16%	18.20%	15.20%	12.41%	11.36%
Time Series Features	18.36%	17.75%	26.40%	16.18%	29.05%	25.52%	22.73%
Specific Time Series Feature	**47.21%**	**43.86%**	**55.09%**	**48.55%**	**32.43%**	24.83%	**42.42%**
Second Order Time Series Feature	25.57%	28.98%	13.34%	17.07%	23.31%	**37.24%**	23.48%
Importance of types of features selected by the L1 penalized logistic regression							
Feature	**76.63%**	**80.48%**	**52.22%**	**50.54%**	**80.00%**	0.00%	**80.48%**
Interaction Feature	23.37%	19.52%	47.78%	49.46%	20.00%	**100.00%**	19.52%
Relative importance of the transformed absolute angles							
Transformed absolute angles	14.83%	14.84%	12.17%	6.63%	11.38%	10.01%	14.77%
Noise (Gaussian, μ=0, σ=1)	0%	0%	0%	0%	0%	0%	0%

Table 3: Results table. We tested our framework on time window's lengths ranging from two to fifteen seconds but presented only the positive number of seconds for clarity purposes.

REFERENCES

Beatty, J. and Lucero-Wagoner, B. The pupillary system. Handbook of psychophysiology, 2:142–162, 2000.

Bednarik, R., Vrzakova, H., and Hradis, M. What do you want to do next: A novel approach for intent prediction in gaze-based interaction. In Proceedings of the Symposium on Eye Tracking Research and Applications, ETRA '12, pages 83–90, New York, NY, USA, 2012. ACM.

Bishop, C.M. et al. Pattern recognition and machine learning, volume 1. Springer New York, 2006.

Bourlard, Hervé, and Yves Kamp. "Auto-association by multilayer perceptrons and singular value decomposition." Biological cybernetics 59.4-5 (1988): 291- 294.

Breiman, L. Random forests. Machine learning, 45(1): 532, 2001.

Bressolles, G., and Nantel, J. 2008. "The Measurement of Electronic Service Quality: Improvements and Application," International Journal of E-Business Research, 4(3), pp. 1-19.

Courtemanche, F., et al. "Activity recognition using eye-gaze movements and traditional interactions." Interacting with Computers 23.3 (2011): 202-213.

Courtemanche, F., Dufresne, A., & LeMoyne, É. (2014). Multiresolution Feature Extraction During Psychophysiological Inference: Addressing Signals Asynchronicity. In H. P. da Silva, A. Holzinger, S. Fairclough & D. Majoe (Eds.), Physiological Computing Systems (pp. 42-55): Springer Berlin Heidelberg.

Dimigen, O., Sommer, W., Hohlfeld, A., Jacobs, A., and Kliegl, R. Coregistration of eye movements and eeg in natural reading: analyses and review. Journal of Experimental Psychology: General, 140(4):552, 2011.

Eivazi, S. and Bednarik, R. Predicting Problem Solving behavior and performance levels from visual attention data. In In the proceedings of 2nd workshop On Eye Gaze in Intelligent Human Machine Interaction at IUI, pages 9–16, 2011.

Grindinger, T., Duchowski, A., and Sawyer, M. "Group wise similarity and classification of aggregate scanpaths." Proceedings of the 2010 Symposium on Eye Tracking Research & Applications. ACM, 2010.

Guyon, I. and Elisseeff, A. An introduction to variable and feature selection. The Journal of Machine Learning Research, 3:1157–1182, 2003.

Guyon, I. and Elisseeff, A. An introduction to feature extraction. In Feature Extraction, pages 1–25. Springer, 2006.

Hinton, Geoffrey E., and Richard S. Zemel. "Autoencoders, minimum description length, and Helmholtz free energy." Advances in neural information processing systems (1994): 3-3.

Holmqvist, K., et al. Eye tracking: A comprehensive guide to methods and measures. Oxford University Press, 2011.

Hyrskykari, A., Majaranta, P., and Räihä, K.J. Proactive response to eye movements. In INTERACT, volume 3, pages 129–136, 2003.

LeCun, Y. (1987). Modèles connexionistes de l'apprentissage. Ph.D. Thesis, Université de Paris VI

Liu, Y., Hsueh, P.Y., Lai, J., Sangin, M., Nussli, M.A., and Dillenbourg, P. Who is the expert? analyzing gaze data to predict expertise level in collaborative applications. In ICME, pages 898–901. IEEE, 2009.

Manyika, J., et al. "Big data: The next frontier for innovation, competition, and productivity." (2011).

Nielsen, J., and Pernice, K. Eyetracking web usability. New Riders, 2010.

Potey, Madhuri, and Pradeep K. Sinha. "Review and analysis of machine learning and soft computing approaches for user modeling." (2015).

Privitera, C.M. and Stark, L.W. Algorithms for defining visual regions-of-interest: comparison with eye fixations. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 22(9):970–982, Sep 2000.

Salojärvi, J., Puolamäki, K., Simola, J., Kovanen, L., Kojo, I., and Kaski, S. Inferring relevance from eye movements: Feature extraction. 2005.

Salvucci D. and Goldberg, J. Identifying fixations and saccades in eye-tracking protocols. In Proceedings of the 2000 symposium on Eye tracking research & applications, pages 71–78. ACM, 2000.

Senecal, S., Kalczynski, P.J., and Fredette, M. "Dynamic identification of anonymous consumers' visit goals using clickstream." International Journal of Electronic Business 11.3 (2014): 220-233.

Smolensky, Paul. "Information processing in dynamical systems: Foundations of harmony theory." (1986): 194.

Tobii eye tracking - an introduction to eye tracking and tobii eye trackers. Technical report, Tobii Technology AB, 2010.

Tobii Technology. Tobii eye experience, 2014.

Toker, D, Conati, C., Steichen, B., and Carenini, G. Individual user characteristics and information visualization: connecting the dots through eye tracking. In proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 295–304. ACM, 2013.

Chapitre 3

Conclusion

Retour sur l'article

Cet article marque une importante étape dans la prédiction de caractéristiques d'utilisateurs en fournissant une liste exhaustive de caractéristiques de base calculées à partir de données oculométriques décontextualisées et en démontrant l'utilisation de celles-ci en accomplissant une tâche de prédiction avec succès.

Cependant, il est important de noter que nous avons fait des choix pratiques quant à la sélection de variables et à la sélection de modèles. Premièrement, le fait de sélectionner les variables en deux étapes fut utile afin de démontrer la quantité de variables significatives d'une part, sans égards au modèle de prédiction sélectionné, et ensuite avec la régression logistique. Il n'est pas garanti que ce soit la méthode la plus performante dans la pratique. Par exemple, les algorithmes non supervisés d'apprentissage de représentation comme les « Restricted Boltzmann Machines » (Smolensky 1986) ou les « Auto-Encoders » (Lecun 1987 ; Bourlard et Kamp 1988 ; Hinton et Zemel 1994) permettent d'apprendre automatiquement des caractéristiques de base à partir des données seulement.

Aussi, nous avons opté pour un modèle linéaire afin d'être en mesure de comprendre l'impact des caractéristiques sélectionnées. Par contre, compte tenu que nous avons créé des variables d'interaction, les avantages liés à cette famille de modèles ont été diminués. Il serait sage d'essayer la prédiction par modèle non-linéaire afin d'évaluer leur performance.

Ceci dit, les résultats présentés sont très encourageant et suggèrent qu'il y a matière à recherche éventuelle sur le sujet. Les données oculométriques décontextualisées semblent, comme la relation yeux-esprit l'indique, être liées aux caractéristiques des utilisateurs.

Retour sur les résultats

Nos résultats sont généralement positifs bien que très volatiles. Par exemple, l'amélioration par rapport aux taux de prédiction naïf est de 47.75%, à son plus haut niveau, avec une fenêtre d'extraction de 12 secondes, mais à 0% et à 13.93% pour des fenêtres d'extraction de 10 et 14 secondes respectivement. Ceci suggère que nous manquons de données pour pouvoir affirmer que nous sommes en mesure de généraliser nos résultats.

Cependant, la caractéristique que nous avons proposée, l'angle absolu transformé, a démontrée une importance relative de 14.22% en moyenne, ce qui nous porte à croire qu'elle est d'importance capitale pour segmenter les utilisateurs récurrents de ceux qui sont nouveaux.

Aussi, le groupement que nous avons fait des caractéristiques devrait également être utilisé dans de futures recherches puisqu'il permet de bien présenter et comprendre celles-ci. Par exemple, nos résultats ont montré que les caractéristiques de séries chronologiques spécifiques ont été les plus significatives des quatre groupes.

Faiblesses et prochaines étapes

La principale faiblesse de cette recherche est le petit nombre d'exemples que nous avons utilisé pour arriver à nos résultats. Bien que ceux-ci soient très motivants, il est évident que davantage de recherche devra être effectuée afin de conclure sur la performance de la méthode. Cependant, nous sommes d'avis que d'utiliser un cadre d'apprentissage automatique rigoureux, comme celui que nous avons proposé, permettra à de futurs chercheurs d'obtenir des résultats prometteurs plus facilement.

Références (Non incluses dans l'article)

McKinney, Wes. "pandas: a python data analysis library." see http://pandas. pydata. org (2012).

Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." The Journal of Machine Learning Research 12 (2011): 2825-2830.

Van Der Walt, Stefan, S. Chris Colbert, and Gael Varoquaux. "The NumPy array: a structure for efficient numerical computation." Computing in Science & Engineering 13.2 (2011): 22-30.

Annexe 1

Code de création des caractéristiques

```
from __future__ import division, print_function
import numpy as np
import pandas as pd
import math
from scipy.stats import skew, kurtosis


def create_metrics_for_period(period_dataframe):
    metrics = {}

    # Keep only the rows where ValidityLeft == 0 and ValidityRight == 0 so both eyes are
valid
    period_dataframe = period_dataframe[period_dataframe['ValidityLeft'] == 0]
    period_dataframe = period_dataframe[period_dataframe['ValidityRight'] == 0]
    period_dataframe = period_dataframe.reset_index()

    # Fixations and saccades analysis

    # Get number of rows that are fixations and saccades + calculate proportions
    number_of_fixations_rows = sum(period_dataframe['GazeEventType'] == 'Fixation')
    number_of_saccades_rows = sum(period_dataframe['GazeEventType'] == 'Saccade')
    percentage_of_fixations = number_of_fixations_rows / (number_of_saccades_rows +
number_of_fixations_rows) if \
        number_of_saccades_rows > 0 else 1
    percentage_of_saccades = number_of_saccades_rows / (number_of_saccades_rows +
number_of_fixations_rows) if \
        number_of_fixations_rows > 0 else 1

    # Get number of fixations and saccades by grouping by id
    number_of_fixations = len(period_dataframe[period_dataframe['GazeEventType'] ==
'Fixation'].
                              groupby('FixationIndex').count())
    number_of_saccades = len(period_dataframe[period_dataframe['GazeEventType'] ==
'Saccade'].
                             groupby('SaccadeIndex').count())

    # Update dict
    metrics['number_of_fixations'] = number_of_fixations
    metrics['number_of_saccades'] = number_of_saccades
    metrics['percentage_of_fixations'] = percentage_of_fixations
    metrics['percentage_of_saccades'] = percentage_of_saccades

    # BEGINNING OF PUPILLARY MEASURES

    # Based on
    # Jackson Beatty and Brennis Lucero-Wagoner. The pupillary
    # system. Handbook of psychophysiology, 2:142162,
    # 2000.
```

```
    mean_pupils = (period_dataframe['PupilRight'] + period_dataframe['PupilLeft'])/2

    # We create a filter to remove anomalies (e.g. blinking)
    mean_pupils = [mean_pupil for mean_pupil in mean_pupils if np.isfinite(mean_pupil)]

    # If the time series was all NAN, len will be 0
    if len(mean_pupils) > 0:

        # Calculate this measure before removing the 5th and 95th percentile
        mean_pupils = pd.Series(mean_pupils)
        metrics['latency_to_pupil_max'] = mean_pupils.idxmax()
        mean_pupils = remove_outliers_from_serie(mean_pupils)
        # End of filter

        # Those will be the generic pupillary measures including the mean and peak dilatation
dilatation
        metrics.update(generate_metrics_for_time_series(mean_pupils, 'Pupils'))

        # Then as suggested in the Beatty and Lucero-Wagoner article, latency to peak should
be computer

    # END OF PUPILLARY MEASURES

    # Angles
    metrics.update(calculate_angles_directions_speed(period_dataframe))

    # Fixation
    period_dataframe_fixations =
period_dataframe[np.isfinite(period_dataframe['GazeEventDuration'])]
    metrics.update(generate_metrics_for_time_series(period_dataframe_fixations[
                                        period_dataframe_fixations['GazeEventType'] == 'Fixation'].
                                        groupby('FixationIndex').mean()['GazeEventDuration'],
                                        'Fixation_GazeDuration'))

    for metric in metrics.itervalues():
        if np.isnan(metric):
            break

    return metrics


def calculate_angles_directions_speed(period_dataframe):

    # Absolute angles: angle between horizontal and a saccade
    # http://www.mathsisfun.com/algebra/trig-finding-angle-right-triangle.html
    # Relative angles: angle between two saccades
    # http://mathcentral.uregina.ca/QQ/database/QQ.09.05/keith1.html
    # Distance formula:
    # http://www.purplemath.com/modules/distform.htm
```

```
    # From degrees to radians: radians * (180/pi)
    # http://www.teacherschoice.com.au/maths_library/angles/angles.htm

    metrics = {}
    # Confirm that period_dataframe is a pandas df
    period_dataframe = pd.DataFrame(period_dataframe)

    # Keep only the Fixations.
    dataframe_to_analyze = period_dataframe[period_dataframe['GazeEventType'] ==
'Fixation']

    # Remove NAN
    dataframe_to_analyze =
dataframe_to_analyze[np.isfinite(dataframe_to_analyze['GazePointX (MCSpx)'])]
    dataframe_to_analyze =
dataframe_to_analyze[np.isfinite(dataframe_to_analyze['GazePointY (MCSpx)'])]

    # Since we use pixels and that 0,0 is on the top left corner, we'll add a - for all y
    dataframe_to_analyze['GazePointY'] = - dataframe_to_analyze['GazePointY (MCSpx)']
    # Just for clarity
    dataframe_to_analyze['GazePointX'] = dataframe_to_analyze['GazePointX (MCSpx)']

    # Reset indexes
    dataframe_to_analyze = dataframe_to_analyze.reset_index()

    absolute_angles = []
    relative_angles = []

    direction_up = 0
    direction_down = 0
    direction_left = 0
    direction_right = 0
    # distance_left = 0
    # distance_right = 0

    horizontal = 0
    vertical = 0

    distance = 0
    dist_saccades = []

    # Calculate absolute and relative angles
    for line_number in range(2, len(dataframe_to_analyze)):
        y1 = dataframe_to_analyze.loc[line_number - 2, 'GazePointY']
        y2 = dataframe_to_analyze.loc[line_number - 1, 'GazePointY']
        y3 = dataframe_to_analyze.loc[line_number, 'GazePointY']
        x1 = dataframe_to_analyze.loc[line_number - 2, 'GazePointX']
        x2 = dataframe_to_analyze.loc[line_number - 1, 'GazePointX']
        x3 = dataframe_to_analyze.loc[line_number, 'GazePointX']
```

```
        # Absolute angles
        x21_delta = x2 - x1
        x31_delta = x3 - x1
        x32_delta = x3 - x2
        y21_delta = y2 - y1
        y31_delta = y3 - y1
        y32_delta = y3 - y2

        d21 = np.sqrt(x21_delta ** 2 + y21_delta ** 2)
        d31 = np.sqrt(x31_delta ** 2 + y31_delta ** 2)
        d32 = np.sqrt(x32_delta ** 2 + y32_delta ** 2)

        # add angles in degrees
        absolute_angles.append(np.arctan(y21_delta/x21_delta) * (180 / np.pi))
        absolute_angles.append(np.arctan(y32_delta/x32_delta) * (180 / np.pi))
        relative_angles.append(np.arccos((d21 ** 2 + d32 ** 2 - d31 ** 2) / (2 * d21 * d32)) *
(180 / np.pi))

        if x2 > x1:
            direction_right += 1
        else:
            direction_left += 1

        if y2 > y1:
            direction_up += 1
        else:
            direction_down += 1

        distance += d21 + d32
        dist_saccades.append(d21)
        dist_saccades.append(d32)

    # Compute the number of vertical and horizontal lines
    # Transform the absolute angles to always be in the first quadrant
    transformed_absolute_angles = []
    for angle in absolute_angles:
        # Vertical and horizontal
        if 0 < angle <= 45 or 135 < angle <= 225 or 315 < angle <= 360:
            horizontal += 1
        else:
            vertical += 1

        # Transform absolute angles to be in the first quadrant
        if angle <= 90:
            transformed_absolute_angles.append(angle)
        elif 90 < angle <= 180:
            transformed_absolute_angles.append(180 - angle)
        elif 180 < angle <= 270:
```

```
            transformed_absolute_angles.append(angle - 180)
        else:
            transformed_absolute_angles.append(angle - 270)

    metrics.update(generate_metrics_for_time_series(absolute_angles, 'absolute_angles'))
    metrics.update(generate_metrics_for_time_series(relative_angles, 'relative_angles'))
    metrics.update(generate_metrics_for_time_series(dist_saccades, 'saccade_length'))
    metrics.update(generate_metrics_for_time_series(transformed_absolute_angles,
'transformed_abs_angles'))

    metrics['direction_up'] = direction_up
    metrics['direction_down'] = direction_down
    metrics['direction_left'] = direction_left
    metrics['direction_right'] = direction_right
    metrics['total_distance'] = distance
    if vertical > 0:
        metrics['horizontalVerticalRatio'] = horizontal / vertical

    return metrics


def generate_metrics_for_time_series(time_series, label, measures=None):
    if not measures:
        measures = ['min', 'max', 'mean', 'std', 'variation',
                    'second_variation', 'skewness',
                    'kurtosis']

    # Remove NaN from timeseries
    time_series = [value for value in time_series if not math.isnan(value)]

    metrics = {}

    if len(time_series) > 0:
        if 'min' in measures:
            metrics[label + '_min'] = np.min(time_series)
        if 'max' in measures:
            metrics[label + '_max'] = np.max(time_series)
        if 'mean' in measures:
            metrics[label + '_mean'] = np.mean(time_series)
        if 'std' in measures:
            metrics[label + '_std'] = np.std(time_series)
        if 'skewness' in measures:
            metrics[label + '_skewness'] = skew(time_series)
        if 'kurtosis' in measures:
            metrics[label + '_kurtosis'] = kurtosis(time_series)

        if len(time_series) > 1:
            if 'variation' in measures:
                variations = []
```

```
            for i in range(1, len(time_series)):
                variations.append((time_series[i] - time_series[i - 1]) / time_series[i - 1])

            variations = [value for value in variations if np.isfinite(value)]

            if len(variations) > 1:
                metrics[label + '_min_variation'] = np.min(variations)
                metrics[label + '_max_variation'] = np.max(variations)
                metrics[label + '_mean_variation'] = np.mean(variations)
                metrics[label + '_std_variation'] = np.std(variations)

            if 'second_variation' in measures:
                second_variations = []
                variations = [value for value in variations if value != 0]
                if len(variations) > 1:
                    for i in range(1, len(variations)):
                        second_variations.append((variations[i] - variations[i - 1]) / variations[i -
1])

                    second_variations = [value for value in second_variations if np.isfinite(value)]

                    # Remove outliers
                    if len(second_variations) > 0:
                        second_variations = remove_outliers_from_serie(second_variations)

                    if len(second_variations) > 0:
                        metrics[label + '_min_second_variation'] = np.min(second_variations)
                        metrics[label + '_max_second_variation'] = np.max(second_variations)
                        metrics[label + '_mean_second_variation'] = np.mean(second_variations)
                        metrics[label + '_std_second_variation'] = np.std(second_variations)

    return metrics


def standardize_dataframe(dataframe):
    for col in dataframe:
        dataframe[col] = (dataframe[col] - dataframe[col].mean())/dataframe[col].std()

        # If there's not enough variation in the column, the col was transformed to NAN so place
0s instead
        if not all(np.isfinite(dataframe[col])):
            dataframe[col] = 0

    return dataframe


def remove_outliers_from_serie(serie, lower_perc=5, higher_perc=None):
    # Lower and higher perc must be given in integer (0.05 = 5)
    if higher_perc is None:
```

```
        higher_perc = 100 - lower_perc

    # Remove outliers
    if len(serie) > 0:
        lower_bound = np.percentile(serie, lower_perc)
        upper_bound = np.percentile(serie, higher_perc)
        serie = [value for value in serie if (lower_bound < value < upper_bound)]

    return serie


def shuffle_dataset_columns(dataset):
    import random as rnd
    new_order = range(len(dataset.columns))
    rnd.shuffle(new_order)
    new_order_colnames = dataset.columns[new_order]

    dataset = dataset.reindex(columns=new_order_colnames)

    return dataset
```

Annexe 2

Code de création du jeu de données pour une fenêtre d'extraction

```
from __future__ import print_function, division

import os

import pandas as pd
import random as rnd

import DataTransformations as dt


class DataPreprocessor:

    files_list = []
    path_to_files = None
    n_wrong_files = 0

    def __init__(self, path_to_files):
        print('Reading tobii data from {}'.format(path_to_files))
        # Get all files to load
        self.path_to_files = path_to_files
        self.files_list = os.listdir(path_to_files)

    def load_and_preprocess_data(self, number_of_seconds_per_period, lag=0):

        wrong_files_list = []
        wrong_files_n_segments_list = []
        metrics_list = []
        y = []

        for file_num, this_file in enumerate(self.files_list):
            print('Reading and parsing file {} ({}/{})'.format(this_file, file_num + 1,
len(self.files_list)))

            # Get all indexes and websites of ScreenRecStarted and ScreenRecStopped
            studio_event_df = pd.read_csv(os.path.join(self.path_to_files, this_file), sep='\t',
usecols=[35, 36])
            screen_rec_started_ix = studio_event_df[studio_event_df['StudioEvent'] ==
'ScreenRecStarted'].index
            screen_rec_stopped_ix = studio_event_df[studio_event_df['StudioEvent'] ==
'ScreenRecStopped'].index

            real_started = []
            real_stopped = []
            urls = []
            for start, stop in zip(screen_rec_started_ix, screen_rec_stopped_ix):
                start_index, url = self.identify_index_and_url_of_segments_from_screenrecstarted(
                    studio_event_df, start, stop)
                if start_index > 0:
                    real_started.append(start_index)
```

```
                real_stopped.append(stop)
                urls.append(url)

            screen_rec_started_ix = real_started
            screen_rec_stopped_ix = real_stopped

            # Truncate the websites to their root
            truncated_urls = []

            for url in urls:
               truncated_urls.append(self.truncate_url_to_root(url))
            ##### End of getting the good index (meaning no unipark nor walmart)

            # This is getting the data from the file and creates a dataframe
            tobii_dataframe = pd.read_table(os.path.join(self.path_to_files, this_file))

            # Proceed only for the dataframes that have 6 or 8 segments
            #if len(screen_rec_started_ix) > 1 and (len(screen_rec_started_ix) == 6 or
len(screen_rec_started_ix) == 8):
            if len(screen_rec_stopped_ix) > 3:
               # Find the y of all chunks
               # y=1 if returning (meaning truncated_ulr - 1 == truncated_url
               # y=0 otherwise

               second_index = -1
               y.append(0)
               available_y1_second_indexes = []
               for i in range(1, len(truncated_urls)):
                  if truncated_urls[i] == truncated_urls[i - 1]:
                     available_y1_second_indexes.append(i)

               good_second_index = False
               number_of_iterations = 0
               while not good_second_index and number_of_iterations < len(truncated_urls):
                  if len(available_y1_second_indexes) > 0:
                     second_index = rnd.sample(available_y1_second_indexes, 1)[0]
                     start_ix = int(screen_rec_started_ix[second_index] + (lag * 60))
                     end_ix = int(screen_rec_stopped_ix[second_index])
                     # Make sure the visit is at least a minute
                     if end_ix - start_ix > (60 * 60):
                        good_second_index = True
                        y.append(1)
                     else:
                        second_index = -1
                  else:
                     second_index = rnd.randint(1, len(truncated_urls) - 1)
                     start_ix = int(screen_rec_started_ix[second_index] + (lag * 60))
                     end_ix = int(screen_rec_stopped_ix[second_index])
                     # Make sure the visit is at least a minute
```

```
                if end_ix - start_ix > (60 * 60):
                    good_second_index = True
                    y.append(0)
                else:
                    second_index = -1

                number_of_iterations += 1

            for i in [x for x in [0, second_index] if x > -1]:

                start_ix = int(screen_rec_started_ix[i] + (lag * 60))
                end_ix = int(screen_rec_started_ix[i] + (lag * 60) +
(number_of_seconds_per_period * 60))

                if end_ix > screen_rec_stopped_ix[i]:
                    end_ix = int(screen_rec_stopped_ix[i])

                metrics = dt.create_metrics_for_period(tobii_dataframe[start_ix:end_ix])
                metrics_list.append(metrics)

        else:
            print(
                '*** Discarded because of unauthorized number of
screen_rec_started/screen_rec_stopped: {} ***'.
                format(len(screen_rec_started_ix)))
            self.n_wrong_files += 1
            wrong_files_list.append(this_file)
            wrong_files_n_segments_list.append(len(screen_rec_started_ix))

    metrics_dataframe = pd.DataFrame(metrics_list)
    #metrics_dataframe[:].fillna(0, inplace=True)

    metrics_dataframe['y'] = y

    metrics_dataframe.to_csv('../../final_full_dataset_nsec_' +
                             str(number_of_seconds_per_period) + '.csv', index=False)

    return 1

  @staticmethod
  def identify_index_and_url_of_segments_from_screenrecstarted(dataset,
                                                               screen_rec_started_ix,
                                                               screen_rec_stopped_ix):
    """ Return the index and the website only if the website is note unipark of walmart

        The returned index is of the URLStart

        Otherwise return -1, None
    """
```

```
        for i in range(screen_rec_started_ix, screen_rec_stopped_ix):
            if dataset.ix[i]['StudioEvent'] == 'URLStart' and not 'walmart' in
dataset.ix[i]['StudioEventData'] and \
                    not 'unipark' in dataset.ix[i]['StudioEventData']:
                return i, dataset.ix[i]['StudioEventData']

        return -1, None

    @staticmethod
    def truncate_url_to_root(url):
        for i in range(1, len(url) - 1):
            if (url[i] == '/' and url[i + 1] != '/' and url[i - 1] != '/') or i == len(url):
                return url[:i + 1]
        return url
```