# Behavioural User Identification from Clickstream Data for Business Improvement

Gaurav Misra, Matteo Migliavacca, and Fernando E. B. Otero

School of Computing, University of Kent, Canterbury, UK
{G.Misra, M.Migliavacca, F.E.B.Otero}@kent.ac.uk

**Abstract.** One of the key elements for businesses to succeed is to get to know their customers. Traditionally this task has been performed through user studies, however, over the last few years clickstream analysis has been proposed as a potential way of conducting automated behavioural studies at scale. In this paper, we explore the use of a recently-proposed unsupervised data-mining technique to identify common behavioural patterns from a clickstream and use them to automatically group users into clusters. In particular, our goal is to validate the potential of behavioural user identification with respect to a key business-level objective. We consider to which extent it is possible to link overall user in-application behaviour to the completion of a particular business-relevant action. Identifying behavior patterns resulting in such business-relevant actions can enable businesses to make changes to their interface, target relevant user groups or trigger actionable insights, all with the objective of maximizing the likelihood of preferable user actions. We analyzed a realworld dataset from a mobile application deployed on both the iOS and Android platforms for this experiment.

## 1 Introduction

The growing amount of data collected by online services, in particular mobile apps, presents a unique opportunity to understand how users interact with services. The analysis of these data could potentially help to identify frequent, unexpected and desired (or undesired) user behaviours, which in turn can be used to increase user experience and engagement. This insight may also be leveraged by businesses to personalize their product for certain behavior profiles, with the objective of steering users towards monetizable actions. Traditionally, user studies have been employed to understand how users interact with services. However, one significant drawback of employing user studies to analyse online services is the need to establish the questions in advance, potentially limiting the ability to characterise unexpected and undesired behaviour. Such studies also suffer from several biases (e.g., sampling, self-selection bias), which ends up leading to a situation where the business tends to capture data from only a subset of their users (or behavior profiles), limiting their reach and therefore making the findings less generalizable and actionable.

Recently, the analysis of actual user actions has emerged as an alternative to user studies, in order to learn about user behavior. The ability to collect and

analyze usage data from websites and apps has enabled researchers to create data mining techniques which can leverage this data to identify behavior patterns prevalent among users. Wang et al. [13, 12] proposed one such technique by using clickstream information to detect and model user behaviour. A *clickstream* is a sequence of timestamped actions performed by a user while interacting with an online system. Thus, the sequence of actions defines the user behaviour on the system, and the analysis of these sequences can lead to the identification of common patterns of user behaviour—i.e., common sequence of actions performed by users. To this end, they first create a graph to capture the similarity between user clickstreams, and then apply clustering to group users according to their behavioural patterns. Clustering is an unsupervised task, which consists of finding a finite set of categories (clusters) to describe the data [4, 14]. Clusters are created based on the similarity of features' values—in this case, the frequency of users' action patterns—and, as a result, similar users are grouped together. A clustering algorithm aims at grouping the users into clusters so that the similarity of users in a cluster is maximised and the similarity of users from different clusters is minimised.

The analysis of users' clickstreams addresses the limitation of user studies—in relation to both the design of questions and scale of users—since user behaviour is automatically detected based on how users interact with the system. Authors in [13, 12] employed a clickstream user behavioural model to detect malicious users. The rationale is to cluster users based on their clickstream to differentiate the behaviour of normal and malicious users. They further extended their technique to model user behaviours as hierarchical clickstream clusters, using an iterative feature pruning clustering (IFPC) algorithm. User behaviour is represented as a tree of clusters, where high-level clusters are formed based on the most important features and sub-clusters at lower levels of the hierarchy are formed based on less important features. The feature set that characterises each cluster can then be used to understand the behaviour of users of the cluster. They presented three case studies showing advantages of the iterative feature pruning algorithm in identifying *inactive*, *hostile* and *malicious* user behaviours in two real-world online social networks. Other researchers have successfully used clickstream data to detect malicious social bots [10, 6], model user engagement [5] and identify e-commerce item access patterns [15].

In this paper, we apply the IFPC algorithm to analyse clickstream data from a real-world mobile application available for both iOS and Android platforms. We use this technique as it provides a view of the most dominant behavior patterns within clusters, which in our case, is demonstrated by the sequence of events that are invoked by user actions. Another advantage of using IFPC is that it has been proven to be able to analyze previously unknown behavior patterns, due to its non-reliance on prior knowledge of labels [13]. Our aim is to evaluate its performance in identifying patterns in the user behaviour that are linked to a particular action on the system. Understanding such behavior is key for the business to gain a better understanding of their users, and what makes them perform actions of interest for the business. This action could be

one which leads to direct monetary benefit for the business (e.g., user making an in-app purchase) or fulfills other key business outcomes (e.g., engagement with a newly released feature). Such insight about favourable user behavior can then be used to improve user experience by optimising the application interface, or create additional triggers, to ultimately identify and encourage similar user behavior in order to improve engagement/completion of this desired action.

The rest of the paper is organised as follows. Section 2 presents the details of the iterative feature pruning clustering algorithm. Section 3 provides details of the case study presented in this paper. The results and discussion are presented in Section 4, while conclusions and future research directions are presented in Section 5.

## 2 Method

**Overview.** We now describe the IFPC technique proposed in [13] in more detail, and in particular its open-source implementation[1], which we adopted to perform our experiments. At a high level, the technique obtains clusters by partitioning a similarity graph where nodes are users and weighted edges represent the similarity in activity between users. The partitioning is recursive: after the graph is partitioned into clusters, a new similarity graph is constructed for each cluster, and partitioned again until the quality of the resulting clusters drops below a minimum threshold. At each step, features that lead to the creation of a cluster are removed when creating the new similarity graph for users in that cluster, allowing the clustering to highlight finer differences in behaviour. Next, we explain how the user activity is captured by a set of features, which are used to compute the similarity graph, and how this is partitioned to obtain clusters.

**Clickstream model.** The starting point is the clickstream of each user, composed by all click events $e_i$ from that user, ordered by timestamp $t_i$. For each event, the model captures both the event type $T_i \in \mathcal{T}$ and a time gap until the next event $G_i = b(t_{i+1} - t_i) \in \mathcal{G} = \{1, 2, \ldots, 5\}$; $b$ is a bucketing function used to make gaps discrete using *less than 1s, 1s - 1min, 1min - 1hour, 1hour - 1day* and *more than 1day* as thresholds for the five buckets. The user clickstream is then converted into a sequence $S = (s_1, s_2, \ldots, s_{2n-1})$ where $s$ alternates event types $T$ and time gaps $G$. A simplified model where $S$ is only composed of event types is also considered in our study, as proposed in [12].

**Features.** Sequences are turned into features by counting the frequency of each sub-sequence of length $k \leq k_{\max}$ (k-grams). The value for feature $f = (s_j, s_{j+1}, \ldots, s_{j+k-1})$ is then $c_f$, the count of occurrences of $f$ in $S$. Conceptually a sequence, and thus a user's activity, is characterised by a feature vector $\boldsymbol{c}$ counting occurrences of each possible k-gram in $\bigcup_{k=1}^{k_{\max}} (\mathcal{T} \cup \mathcal{G})^k$, however in practice the feature vectors are sparse enough that is convenient to store features as a map between occurring k-grams and their count. We fixed $k_{\max}$ at

[1] https://github.com/xychang/RecursiveHierarchicalClustering

5 as in [13], with longer sub-sequences becoming unlikely to be repeated across users and thus not contributing substantially to the clustering quality.

**Similarity Graph.** The similarity between two clickstreams is then computed as the normalised polar distance between their feature vectors:

$$d(S_i, S_j) = \frac{2}{\pi} \arccos \frac{\boldsymbol{c_i} \cdot \boldsymbol{c_j}}{\|\boldsymbol{c_i}\| \|\boldsymbol{c_j}\|} \quad .$$

The distance $d$ is small when the two clickstreams are similar (e.g. $S_i$ is a repetition of $S_j$) and maximum $d = 1$ in the extreme case of all k-grams of $S_i$ and $S_j$ being disjoint.

**Iterative clustering with feature pruning.** After a similarity graph is constructed, divisive hierarchical clustering is used to partition the similarity graph into a set of clusters $C = \{C_1, \ldots, C_{n_0}\}$. The number of clusters obtained is controlled by the *modularity* metric which compares the density of intra-cluster edges with the density of cross-cluster edges: clustering stops when the modularity reach the maximum.

As the top-level set of clusters $C$ is obtained, the most relevant features that led to the formation of each $C_i$ are determined by using a simple elbow method based on the $\chi^2$ score to compare users in $C_i$ with users outside it. These features explain the salient characteristics of $C_i$ in $C$, and thus can be used to interpret each cluster. These feature are then removed from the set of feature for $C_i$ and a new similarity graph is computed from the remaining features. The process is then repeated, performing divisive clustering again on the new similarity graph for $C_i$ resulting in a new cluster set $C_i = \{C_{i1}, C_{i2}, \ldots, C_{in_i}\}$. The process stops when every cluster can not be split further by the divisive clustering procedure.

## 3 Case Study

The IFPC technique has been previously applied to event streams generated by users of social media to identify malicious, unexpected or low engagement [13, 12]. In this paper, we wanted to analyze whether this technique is useful in analyzing and interpreting user behaviour with respect to a specific business key performance indicator (KPI). With this objective in mind, we obtained a labeled dataset of user generated events from Company X[2] which provides a financial management mobile app to its users on both iOS and Android platforms. The company appropriately anonymized the dataset, in accordance with their terms of use and privacy policies, before providing access to the researchers for the purpose of this experiment.

The main objective of this research was to explore how the IFPC technique applied to user clickstreams [13] may benefit the company in understanding how users interact with their mobile application, and identifying certain behaviour patterns which lead to specific actions of interest.

[2] Name withheld due to confidentiality

### 3.1 Data

For this research, we obtained a dataset of event streams generated by 1000 users who interacted with the mobile application during the period of $15^{th}$ January 2020 to $28^{th}$ July 2020. After pre-processing, where we removed any users who had just logged-in and logged-out of the application and had not performed any other tasks, we were left with a dataset composed by **956** unique users. In addition to providing the data, the company also helped us in categorizing the various event types into the following categories:

- **Signup** - This category represents all the events triggered when a user creates an account on their first use of the application.

- **Profile** - Events that are triggered when users go into the profile section of the application to edit information they provided during the signing up process.

- **Connect Bank** - The users can connect their bank accounts through the application. This enables them to benefit from the company's transaction categorization and get better insight into their spending patterns.

- **View Transactions** - Events in this category are triggered when users view their bank transactions, visible after connecting a bank account via the application.

- **Categorize Transactions** - Users may also choose to manually categorize a particular transaction into a certain category, if it wasn't automatically categorized by the application, or if the user wants to modify the category a transaction belongs to.

- **Insights** - The application provides timely and topical insights about users' financial behaviour.

- **Notifications** - These are generated as a "call-to-action" for the users. An example scenario is when a user needs to re-authenticate with the application if an access token has expired.

- **Financial Actions** - This category consists of events which are generated when the user takes a particular action in the application (e.g. saving money by selecting personalized product offerings).

The dataset provided to us by the company also contained labels assigned to all the users. Any user who performed a "Financial Action" at least once during the period for which the data was collected, was labeled as class '1' (considered as *converted user*) and others were labeled as class '0'. Out of the **956** total users in our dataset, **345 (36.1%)** of the users were converted users.

**Table 1.** Breakdown of events by category.

| Event Category | No. of Events |
|---|---|
| Sign Up (SU) | 3005 |
| Profile (P) | 1845 |
| Connect Bank (CB) | 6317 |
| View Transactions (VT) | 7928 |
| Categorize Transactions (CT) | 258 |
| Insights (I) | 1979 |
| Notifications (N) | 796 |
| **Total Events** | **22128** |

Table 1 shows the distribution of events for each category in the dataset used for the experiment. It is important to note here that when preparing the data, we truncated the clickstreams of the users who performed a financial action (and hence were assigned class label of '1') up to the point they performed the action for the first time. As a result, we excluded a large number of events triggered by them in their subsequent interactions with the application. This was done deliberately as we wanted to cluster based on user actions in other areas of the application and see whether it is a good predictor of a financial action.

### 3.2 Feature Extraction

Once we had the event streams for the users, the next step was to extract the features to be used for clustering as explained earlier in Section 2. Similar to [13], we created k-grams up to $k_{\max} = 5$ and computed their occurrence count for each stream. We tried two variations of modeling the event data into k-grams, namely:

- **Excluding time gaps**: We created k-grams from the sequence of events, ignoring the time intervals between them. As an example the stream `ABAC` is encoded by the following features: $\{A \to 2, B \to 1, C \to 1, AB \to 1, BA \to 1, AC \to 1, ABA \to 1, BAC \to 1, ABAC \to 1\}$
- **Including time gaps**: In this variation, we considered time gaps between consecutive events, following the same discretisation of the intervals as [13], i.e., *less than 1s, 1s–1min, 1min–1hour, 1hour–1day* and *more than 1day*.

### 3.3 Evaluation Metrics

We explained earlier in Section 2 that the clustering algorithm optimizes for modularity when finding the optimum clustering configuration [1]. Each of the

clusters produced by the algorithm is associated with a certain behavioural pattern that characterise users in their interaction with the application. Our aim is to identify to which extent the identified patterns are related to certain business outcomes (e.g. user conversion) which are represented by class labels. Therefore, we used the following well established *external criteria* to evaluate the clustering [9],

**Purity**
*Purity* is a simple and transparent external criterion to measure cluster quality when the clustered data is labeled [11]. In our case, the two classes—i.e., whether a user performed a financial action (converted) or not—are the ground truth labels. Given a set of clusters where each data point (user) is labeled, purity is computed by labelling each cluster with the class (i.e., '0' or '1') which is the most frequent in that cluster, and then the accuracy of this assignment is measured by counting the number of correctly assigned points and dividing by the total number of points [7]. More generally, purity ($p$) is calculated as:

$$p = \frac{1}{N} \sum_{k} \max_{j} |c_k \cap y_j| \tag{1}$$

where,
$C = \{c_1, c_2, ..., c_k\}$ is the set of clusters,
$Y = \{y_1, y_2, ..., y_j\}$ is the set of class labels, and
$N =$ the total number of points.

To interpret equation 1, consider $c_k$ as the set of users in cluster $k$ and $y_j$ as the set of users belonging to class $j$. Therefore, for any given values of $k$ and $j$, $|c_k \cap y_j|$ represents those users in cluster $k$ who belong to class $j$. The value of purity lies between 0 and 1, where a value of 0 indicates bad clustering and a value of 1 indicates perfectly pure clusters. It is important to note that high purity, as well as the other metrics, is easier to achieve with a large number of clusters. For example, in the most trivial case, if each point is assigned to its own cluster, so we have $N$ clusters, the purity score will be 1. As mentioned earlier the number of clusters is determined according to an internal quality metric (modularity).

**Homogeneity**
A clustering is defined as being perfectly *homogeneous* (denoted by homogeneity value equal to '1') if all its clusters contain only data points belonging to a single class [8]. In other words, the class distribution within each cluster should be skewed to a single class (*zero entropy*). The homogeneity score of a particular clustering can be calculated based on the conditional entropy of the class distribution given the proposed clustering [8]. In the perfectly homogeneous case, this value, $H(Y|C)$ is 0, where $Y$ is the set of classes and $C$ is the set of clusters (refer to equation 1). In the degenerate case, when we have a single class across

all points in the dataset, i.e. $H(Y) = 0$, we have perfect homogeneity. For other cases, where there are more than one class, homogeneity ($h$) can be calculated as follows:

$$h = 1 - \frac{H(Y|C)}{H(Y)} \tag{2}$$

where, $C$ and $Y$ represent the set of clusters and class labels respectively (see equation 1). Conditional entropy $H(Y|C)$ is calculated as follows:

$$H(Y|C) = -\sum_{k=1}^{K}\sum_{j=1}^{J}\frac{n_{kj}}{N}\log_2\frac{n_{kj}}{\sum_{j=1}^{J} n_{kj}} \tag{3}$$

where, $n_{kj}$ is the number of points in cluster $c_k$ which belong to class $y_j$.

**Classification Metrics**
The two metrics discussed so far in this section do not discriminate between the cluster labels, and treat any "misclassification", i.e., assigning a point to a cluster where it is in the minority, equally. However, for our use cases, it is important to understand those incorrect classifications in terms of their types: "false positive" (a class '0' point is assigned to a majority class '1' cluster) or "false negative" (a class '1' point is assigned to a majority class '0' cluster). To achieve this, each cluster is assigned the class label to which the majority (more than 50%) of its points (users) belong. For clusters which have equal number of users belonging to each class, we assign it the positive class label (class '1'). It is worth noting that fully homogeneous clusters will not have any incorrect classifications. We define the following scenarios, in order to use classification metrics to evaluate our clustering:

- **True Positive (TP)**: user labeled as class '1' is assigned to a majority class '1' cluster.
- **True Negative (TN)**: user labeled as class '0' is assigned to a majority class '0' cluster.
- **False Positive (FP)**: user labeled as class '0' is assigned to a majority class '1' cluster.
- **False Negative (FN)**: user labeled as class '1' is assigned to a majority class '0' cluster.

From these values, we can calculate the Precision ($\frac{TP}{TP+FP}$), Recall ($\frac{TP}{TP+FN}$) and $F_1$ score ($2 \times \frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$).

## 4 Results

In this section, we evaluate the clusters produced by the IFPC technique (§2) using the external evaluation metrics (§3.3) that represents whether IFPC is effective in identifying user behaviour which leads to performing a financial action through the app.

**Table 2.** Events per user.

| Class | Users | Events | Events per user |
|---|---|---|---|
| 0 | 611 | 14478 | 23.7 |
| 1 | 345 | 7650 | 22.2 |
| **Total** | **956** | **22128** | **23.1** |

Our dataset for this experiment consisted of **956** users with a combined total of 22128 events (Table 2). As we can see from Table 2, we had 611 users who did not perform a financial action using the app whereas there were 345 users who did so. Interestingly, the average number of events triggered by both these sets of users is very similar. Therefore, it seems that users interact with the app to a similar extent before deciding whether they want to perform a financial actions (class '1') or not (class '0'). This indicates that the length or scale of user engagement rarely determines whether they will perform this action, and therefore the actual behaviour is important.

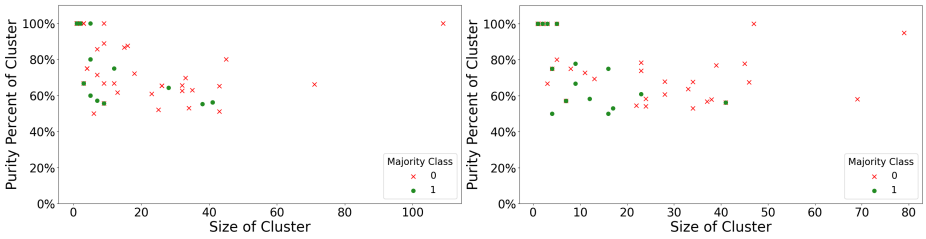
### 4.1 Purity

Table 3 shows the purity scores for the clustering for this experiment. We see that both clustering configurations produce reasonably high values of overall purity. In both cases, the purity value is well above 0.5, which means that most users were placed in clusters where the majority of users had the same class. We see that including time gaps produces a lower number of clusters (50) as compared to when excluding time gaps (63), and the overall purity score for this configuration (0.687) is also slightly lower (0.700 when excluding time gaps).

In addition to calculating the purity of clusters, we were interested in analyzing the nature of clusters with a higher purity score. Figure 1 shows a plot of cluster purity (in %) *vs* the size of the clusters (number of users present in the cluster). Ideally, clusters in the top right corner of the plots—i.e., large clusters having very high purity—are beneficial, as that would enable us to definitively identify behaviour patterns which can be attributed most accurately to each class. We can see from the figure that most such clusters are majority class '0' clusters. For both configurations, the few clusters which are majority '1' class and also have a reasonably high purity score, are very small in size (below 20 users). On the other hand, there are one or two very large clusters which are almost entirely made up of class '0' users (i.e., purity score close to 100%). While such clusters do not contribute in identifying which user profiles are likely to perform a financial action through the app, they nevertheless inform the company as to what action patterns associated with these clusters are very unlikely to result in a financial action in the future.

**Table 3.** Purity and Homogeneity of clustering.

| | Clusters | Purity | Homogeneity |
|---|---|---|---|
| **No gaps** | 63 | 0.700 | 0.188 |
| **With gaps** | 50 | 0.687 | 0.150 |




(a) No gaps (63 clusters) (b) With gaps (50 clusters)

**Fig. 1.** Comparison between purity and cluster size

### 4.2 Homogeneity

Table 3 shows that the homogeneity scores for both configurations is quite low. This is expected by looking at purity values as well as Figure 1, which show that there are many clusters which contain a mix of both classes, and therefore are not homogeneous. Clustering user event streams excluding time gaps produces slightly higher values of homogeneity (0.183) when compared to including time gaps (0.142).

### 4.3 Classification

Table 4 shows the results of assigning the majority class label to each cluster. We can see that in both configurations, a higher number of clusters were assigned '0' as the majority class label, i.e., more than 50% of users in these clusters had never performed a financial action.

As explained in section 3.3, we use the classification metrics by the majority class label to each cluster generated by the algorithm. Then, we consider how many correct classifications would be made, if the majority label was assigned to each user in each cluster.

To better contextualize the results in this section, we also utilized two well-known classification algorithms, namely *Decision Tree (DT)* [2] and *Random Forests (RF)* [3] with the data—these were used with their default parameters. The feature vectors used by these algorithms are the same k-grams constructed as described in §2, however IFPC is not used to cluster the users according to the similarity graph, and the algorithms operate directly on the (labelled) k-gram frequency vectors. The comparison with baseline classification algorithms helps us better understand the performance of the IFPC clustering.

**Table 4.** Label assignments for clusters.

| | No. of Clusters (majority class) | | |
|---|---|---|---|
| | **0** | **1** | **Total** |
| **No gaps** | 40 | 23 | 63 |
| **With gaps** | 33 | 17 | 50 |

**Table 5.** Results for classification metrics.

| | | Metrics | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | **TP** | **FP** | **TN** | **FN** | **Accuracy** | **$F_1$** | **Precision** | **Recall** |
| **No gaps** | **IFPC** | 140 | 82 | 529 | 205 | **70.0%** | **49.4%** | **63.1%** | 40.6% |
| | **DT** | 112 | 113 | 498 | 233 | 63.8% | 39.3% | 49.8% | 32.5% |
| | **RF** | 151 | 122 | 489 | 194 | 66.9% | 48.9% | 55.3% | **43.8%** |
| **With gaps** | **IFPC** | 112 | 66 | 545 | 233 | **68.7%** | 42.8% | **62.9%** | 32.5% |
| | **DT** | 176 | 185 | 426 | 169 | 63.0% | **49.9%** | 48.8% | **51.0%** |
| | **RF** | 156 | 125 | 486 | 189 | 67.2% | 49.8% | 55.5% | 45.2% |

Table 5 summarises the classification performance for both configurations. The results show that IFPC produces higher accuracy than Decision Tree and Random Forests in both configurations. Comparing the two configurations, it seems that IFPC performs better without including time gaps in the clickstreams while both the other classification algorithms (in particular Random Forests) seem to perform slightly better when time gaps are included. It is also evident from looking at the results that IFPC has better precision than the other algorithms. This is due to the significantly lower number of false positives (FP) seen for IFPC in comparison to the other algorithms. On the other hand, Random Forests has a better recall than IFPC when excluding time gaps and both Decision Tree and Random Forests outperform IFPC in terms of recall when considering time gaps. Overall, looking at the results for IFPC, it indicates that using clickstream data without time gaps produces better performance, which is in line with results seen for purity and homogeneity as well (Table 3).

For the business, the implication of the performance of IFPC seems to be that it is better than other classifiers at not making false positive errors, therefore, using this technique, the business is less likely to mistakenly identify unfavourable behavior as favourable (low false positives), with the risk of missing out on some favourable user behavior (higher false negatives).

### 4.4 Action Patterns

The IFPC technique produces action patterns (sequence of k-grams) for each identified cluster. These are the most relevant features that contributed to the

formation of a cluster at a specific iteration (see section 2 for details). Such action patterns are interesting for the business to understand and link to particular actions—i.e., for this experiment, a user performing a financial action through the app. To briefly illustrate this, we selected one majority '0' class and one majority '1' class cluster from each configuration.

Looking at Table 6, we see that cluster number 24, when clustering with time gaps, has 79 total users and has only 5.1% of its users labeled as class '1'. The characteristic action pattern shown in the last column illustrates that the users in this cluster did not go beyond the signup flow (SU) of the app. for cluster 46, which is majority class '1' cluster (56.1% users were labeled as class '1'), we find that the users were more engaged with the "view transaction" functionality (VT) of the app. As this is one of the core functionalities of the app, it can be concluded that these users spent more time exploring the app when compared to those in cluster 24.

Considering the clustering without time gaps, we present results from cluster 14 and cluster 49 in Table 6. We can see that cluster 14 is characterized by multiple action patterns and include engagement with "Insights" (I) and "Notifications" (N). However, considering that this is a majority '0' cluster, it indicates that these features of the app did not direct all users towards taking a financial action. Such information can be a useful method of evaluating the effectiveness of certain features (such as notifications and insights) if they are released with clear objectives. This depth of analysis is impossible by just looking at front-end analytics of event streams which generally analyze sequence of events but fail to capture behavior patterns such as these.

### 4.5 Discussion

The results presented show that iterative feature pruning clustering does a good job in identifying user behaviours which lead to a financial action in our experiment. We already observed that the difference in levels of engagement, measured as average events per user, between the two classes (i.e., users who converted *vs* those who didn't) was minimal (Table 2) and therefore being able to identify different behaviour patterns using the clustering technique can be beneficial. We find that precision, in particular, is better than recall when using classification metrics, which enables the company to be selective in nudging users who have favorable behaviour profiles with targeted insights and notifications which may lead to user conversion. When comparing the results against classification algorithms, we observed that both precision and accuracy are higher when using IFPC. On the other hand it would be possible to increase the recall performance at the expense of precision and accuracy, by lowering the labelling threshold for class '1' clusters (below the 50% threshold used in our experiment). This would result in identifying more users belonging to the positive class (who can potentially perform an important action), while still maintaining some "targeting" power. While this technique does indeed provide the company with a deeper understanding of user activity in the app, the overall results, including purity and

**Table 6.** Action patterns associated with clusters

| Configuration | Cluster ID | Size | Majority Class | Class '1' User Percentage | Action Patterns |
|---|---|---|---|---|---|
| With Gaps | 24 | 79 | 0 | 5.1% | SU 1H<br>SU 1H SU<br>SU 1M SU 1H |
| | 46 | 41 | 1 | 56.1% | VT 1H VT 1M<br>VT 1H<br>VT 1H VT |
| No Gaps | 14 | 35 | 0 | 37.1% | I<br>N<br>I I VT VT VT<br>I VT |
| | 49 | 38 | 1 | 55.3% | VT<br>CB VT |

Symbols for time gaps:
- 1S : Gap $< 1s$
- 1M : Gap $\in [1s, 1min)$
- 1H : Gap $\in [1min, 1hour)$

Symbols for events:
- SU $\longrightarrow$ Signup
- VT $\longrightarrow$ View Transactions
- I $\longrightarrow$ Insights
- N $\longrightarrow$ Notifications
- CB $\longrightarrow$ Connect Bank

homogeneity of the clusters, indicate that using iterative feature pruning clustering will still leave margins for improvement for predicting user activity on its own, and should rather be used to augment other intelligent systems which rely on other sources of user data (such as information exchanged by users through the platform).

We also observed that modeling the data with or without time gaps has a negligible effect on the clustering. The results produced with these differing models of clickstreams results in very similar clustering which indicates that the algorithm is more sensitive towards the sequence of events rather than the time gaps between them.

## 5 Conclusions and Future Work

In this paper, we applied the iterative feature pruning clustering (IFPC) algorithm to analyze a real-world dataset of clickstreams from a mobile application. The aim of the research was to explore whether clustering technique can enable businesses to identify different behaviour profiles among their users or not, which may lead to specific actions of interest for the business. Our results show that the clustering is effective in identifying user behaviour that leads to a specific action (e.g., financial action)—the measurements of purity, homogeneity and accuracy

are better than a random assignment (50% mark). The method used for experimentation in this paper is agnostic to the target action of interest and may be used in the future to identify behavioural patterns leading to other actions by the company. Hence, the company can potentially use this technique to identify relevant user behaviours and encourage users to trigger any number of actions which may be critical to business outcomes.

A limitation of this research is the amount of data that was available to us for the experimentation. A natural next step would be to consider the impact of larger datasets on the performance of behavioural identification in terms of number of users, event granularity and length of the observation period.

## References

1. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment **2008**(10), P10008 (2008)
2. Breiman, L., Friedman, J., Stone, C., Olshen, R.: Classification and Regression Trees. Chapman and Hall (1984)
3. Breiman, L.: Random Forests. Machine Learning **45**, 5–32 (2001)
4. Larose, D.T.: Discovering Knowledge in Data. John Wiley & Sons (2005)
5. Liu, Y., Shi, X., Pierce, L., Ren, X.: Characterizing and forecasting user engagement with in-app action graph: A case study of snapchat. In: Proc. of ACM SIGKDD'19. pp. 2023–2031. ACM (2019)
6. Luceri, L., Giordano, S., Ferrara, E.: Detecting troll behavior via inverse reinforcement learning: A case study of russian trolls in the 2016 us election. Proc. of International AAAI Conference on Web and Social Media **14**(1), 417–427 (2020)
7. Manning, C.D., Raghavan, P., Schutze, H.: Flat Clustering, chap. 16, pp. 356–359. Cambridge University Press (2008)
8. Rosenberg, A., Hirschberg, J.: V-measure: A conditional entropy-based external cluster evaluation measure. In: Proc. of EMNLP-CoNLL. pp. 410–420 (2007)
9. Schütze, H., Manning, C.D., Raghavan, P.: Introduction to information retrieval, vol. 39. Cambridge University Press Cambridge (2008)
10. Shi, P., Zhang, Z., Choo, K.K.R.: Detecting malicious social bots based on clickstream sequences. IEEE Access **7**, 28855–28862 (2019)
11. Solomonoff, A., Mielke, A., Schmidt, M., Gish, H.: Clustering speakers by their voices. In: Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181). vol. 2, pp. 757–760. IEEE (1998)
12. Wang, G., Zhang, X., Tang, S., Wilson, C., Zheng, H., Zhao, B.Y.: Clickstream user behavior models. ACM Transactions on the Web (TWEB) **11**(4), 1–37 (2017)
13. Wang, G., Zhang, X., Tang, S., Zheng, H., Zhao, B.Y.: Unsupervised clickstream clustering for user behavior analysis. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems. pp. 225–236 (2016)
14. Witten, H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, 2nd edn. (2005)
15. Xylogiannopoulos, K., Karampelas, P., Alhajj, R.: Clickstream analytics: An experimental analysis of the amazon users' simulated monthly traffic. In: IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). pp. 841–848. IEEE (2018)