Online Prediction of Aggregated Retailer Consumer Behaviour

Yorick Spenrath[0000−0003−0908−9163], Marwan Hassani[0000−0002−4027−4351], and Boudewijn F. van Dongen[0000−0002−3978−6464]

Eindhoven University of Technology, Eindhoven, Netherlands
{y.spenrath, m.hassani, b.f.v.dongen}@tue.nl

Abstract. Predicting the behaviour of consumers provides valuable information for retailers, such as the expected spend of a consumer or the total turnover of the retailer. The ability to make predictions on an individual level is useful, as it allows retailers to accurately perform targeted marketing. However, with the expected large number of consumers and their diverse behaviour, making accurate predictions on an individual consumer level is difficult. In this paper we present a framework that focuses on this trade-off in an online setting. By making predictions on a larger number of consumers at a time, we improve the predictive accuracy but at the cost of usefulness, as we can say less about the individual consumers. The framework is developed in an online setting, where we update the prediction model and make new predictions over time. We show the existence of the trade-off in an experimental evaluation on a real-world dataset consisting of 39 weeks of transaction data.

Keywords: Consumer Behaviour · Stream Analysis · Clustering

1 Introduction

Knowing the future behaviour of consumers is important to help retailers plan ahead [3]. One way to do so is by predicting how consumers will behave on an
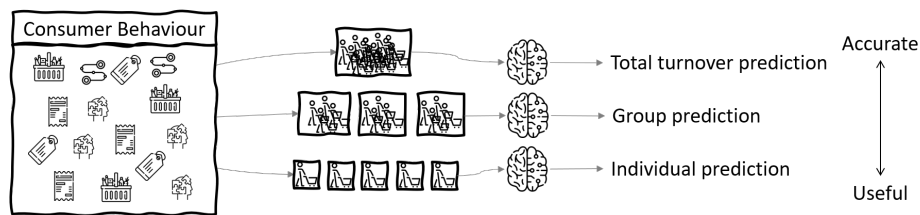
Fig. 1. Overview of the problem. Making predictions for individual consumers is more useful, but less accurate. Making predictions for all consumers (i.e. the entire retailer) is easier, but not useful on individual consumers. This paper balances the two by making predictions using groups of consumers.

individual level. Knowing which consumers are expected to increase or decrease their spending allows retailers to apply more targeted marketing strategies. Unfortunately, the human nature of consumers makes it difficult to accurately predict on an individual level. Two consumers can behave similar for some time, but then quite different in the next week. Another way to make predictions is by taking all consumers together, for example by predicting the turnover of the next week based on the turnover of the past weeks. This improves the accuracy of the predictions as we effectively remove outliers from individual consumers, but it also reduces the information we get about the individuals. This trade-off is schematically presented in Figure 1.

In this paper we aim to strike a balance between accuracy and usefulness. Instead of making predictions on individual consumers we make predictions on *groups* of consumers. The advantage is an increase in accuracy with respect to making predictions for individual consumers as we remove the effect of outliers on the prediction. At the same time, we increase the usefulness of the prediction with respect to the prediction on all consumers together. This is because the predictions are on a limited number of consumers at a time. We apply our framework in a streaming setting, at regular intervals we discover clusters of consumers to update the prediction model. We as such make the following contributions: 1) we propose a framework to overcome the loss in prediction accuracy for diverse consumers by making the predictions on carefully selected clusters of consumers in a streaming environment, 2) we show its effectiveness on a real-world dataset from the supermarket domain and 3) we show that over time, making predictions on clusters does not decrease the accuracy of a downstream prediction task, in contrast to making predictions on individual consumers.

The rest of this paper is organized as follows: we first present our framework in Section 2 and evaluate it in Section 3. We then discuss how this paper relates to existing work in Section 4. Finally, we conclude the paper in Section 5.

2   Framework

In this section we start by giving an overview of our framework, we present the details and formalization of each step in the subsections. The data used can be considered analogous to concepts in process mining: events are represented by
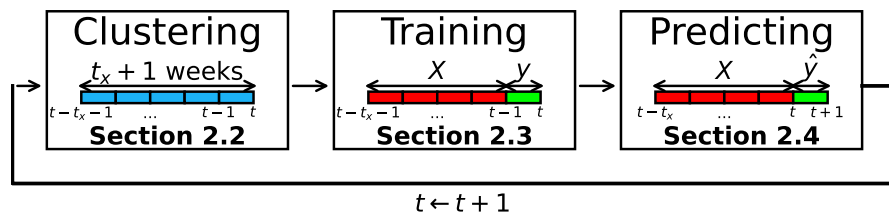

Fig. 2. Overview of the framework

transactions, cases by consumers, and timestamps by purchase dates. Multiple events involving the same consumer constitute to (part of) the consumer journey. These terms are interchangeable. For the description of the general framework we stick to 'events' and 'consumers', but use 'transactions' in the application details of this paper.

Like most other real-life applications, these journeys can vary wildly between different consumers. The goal is to make a prediction about one or more future events of a consumer. For our application this is the total spend over a given period of time, the sum of the spend in the separate events.

The framework starts with this collection of events containing information on a detailed level. This first step is to abstract from these single events. We do so by aggregating events per week and per consumer. This results in a vector of descriptive values that describes all visits of a consumer in a single week. We refer to this as *features*. The details of this are discussed in Section 2.1.

The framework consists of a loop of roughly three steps as presented in Algorithm 1 and schematically shown in Figure 2. At time t, we use the past $\tau \in \mathbb{N}_+$ weeks of data, $[t - \tau, t)$, to make a prediction about the next week of data, $[t, t+1)$. We do not do so on an individual consumer basis, but on a cluster of consumers. In other words, we learn the behaviour of a *group of consumers* and make a prediction on their behaviour *as a group*. To this end we first cluster the consumers based on the their features from $[t - \tau - 1, t)$ into $k \in \mathbb{N}_+$ separate clusters. The details of this clustering are discussed in Section 2.2. Next, we construct a training dataset to train a Recurrent Neural Network (RNN) regression model. Each datapoint in this dataset represents one cluster. The predictor space consists of the features in each week from $[t - \tau - 1, t - 1)$, averaged over the consumers in the cluster. The target value is the

---

Algorithm 1: Overview of the framework

    input : Stream of events, in batches of one week at time t
    output: Clusters g_j^t of consumers and predictions $\hat{y}_j^t$ for the average turnover
             per consumer in g_j^t after every week t

1 **while** *True* **do**
2   Cluster consumers based on $[t - \tau - 1, t)$
3   Extract descriptive features for consumers ▷ Section 2.1
4   Create clusters of consumers ▷ Section 2.2
5   Update LSTM
6   One sequence per cluster over $[t - \tau - 1, t - 1)$ as predictor
7   Turnover per cluster for $[t - 1, t)$ as target
8   ▷ Section 2.3
9   Make predictions
10   One Sequence per cluster over $[t - \tau, t)$ as predictor
11   Predict turnover per cluster for $[t, t+1)$
12   ▷ Section 2.4
13 $t \leftarrow t + 1$

Table 1. Values derived from a single transaction.

Index	Value	Description
v_1	Freshness	Fraction of perishable items
v_2	Item value	Average value of the items bought
v_3	Product density	Average frequency of each product
v_4	Total value	Total price paid for the transaction
v_5	Total item count	Total number of items purchased

turnover in $[t-1, t)$, averaged over all consumers in the cluster. The details of this training and the RNN architecture are described in Section 2.3. Finally, we make a prediction for every cluster using data $[t - \tau, t)$, predicting the average consumer spend of a cluster for $[t, t+1)$. The details of the prediction part are discussed in Section 2.4.

2.1   Features

Our framework starts with a set of events $\mathcal{L}$. Each such an event e has a timestamp $e.time$, a unique consumer identifier $e.cid \in \mathcal{C}$, a label $e.label$ in some activity space $\mathcal{A}$, and additional values $e.v_1$ through $e.v_m$ that contain further information on the event. The first step in our framework consists of summarizing multiple of these events over a period of time. For a time period $[t_1, t_2)$ and a consumer identified by c, we combine all events e that satisfy the predicate $t_1 \leq e.time < t_2 \wedge e.cid = c$, we indicate this set of events as $\mathcal{L}'$. Over these events we compute the fraction of each label $a \in \mathcal{A}$ as $freq(a) = |\{e \in \mathcal{L}' | e.label = a\}|/|\mathcal{L}'|$

For the additional values $e.v_1$ through $e.v_m$ we define functions h_1 through h_m, with $h_i : \mathcal{P}(\mathcal{L}) \rightarrow \mathbb{R}$, which aggregates the values $e.v_i$ in $\mathcal{L}'$. We also record the number of events $|\mathcal{L}'|$. In total, we therefore have $|\mathcal{A}| + m + 1$ descriptions that together summarize the events in $\mathcal{L}'$. We refer to these as *features* in the remainder of the paper, and indicate them as $F = f_1, f_2, \ldots, f_{|F|}$, where $|F| = |\mathcal{A}| + m + 1$.

For the purpose of this paper, each event is a transaction made by a consumer at a retailer. The consumer identifier $e.cid$ is shared between transactions of the same consumer, which is known because consumers hold loyalty cards that uniquely identify them at each purchase. The timestamp $e.time$ is the date (and time) of the transaction. The label $e.label$ provides a description of the contents of the purchase. This labelling is based on an extension from earlier work [12]. It roughly consists of learning *eight* separate clusters over a large collection of transactions, based on the categories and quantities of the products in a transaction. Finally, the values $e.v_1$ through $e.v_5$ provide additional aggregate information on the transaction, as indicated by Table 1.

We therefore have $|\mathcal{A}| = 8$ (labels), $m = 5$ (values in Table 1), and as such a total of $|F| = 14$ features that describe the events over a period of time. With the

help of domain knowledge[1], we define h_1 through h_3 as the mean, and h_4 and h_5 as the sum of the values of the individual events. The latter two, aggregating the `Total value` and `Total item count`, are summed instead of averaged, as this helps distinguishing consumers with a large total spend from consumers with a smaller total spend.

The way in which our framework is constructed allows it to be generalized for the use in certain business processes as well. The event logs of such processes consist of labelled events that belong to a case with some case identifier, and possibly contain additional values per event [1]. This more generic application is beyond the scope of this paper.

2.2   Clustering

In the clustering step, we group consumers with similar behaviour over $[t - \tau - 1, t)$, conform the left box of Figure 2. We apply the aggregation of Section 2.1 to each separate week $[t - \tau - 1, t - \tau), [t - \tau, t - \tau + 1), \ldots, [t - 1, t)$. As such, each consumer is described by $|F|$ features at $\tau + 1$ points in time, over the period $[t - \tau - 1, t)$. While we learn how these values evolve over time during training, we still want consumers with similar evolution to be grouped together. For each feature f_i we extract a linear fit over the values of f_i in those weeks as $f_i(t') = a_i \cdot t' + b_i$ with residuals r_i. Consumers with similar a_i, b_i, and r_i will have a similar average value for f (b_i), a similar increase/decrease (a_i), and a similar fitness to a linear trend (r_i). We compute these a_i, b_i, and r_i for each feature f_i, and cluster based on the resulting $3 \cdot |F|$ coefficients.

As consumers can have different points in time where they first visit the store, some may not have started their visits in or before the first week of the clustering period. We exclude those consumers who have their first purchase after $t - \tau$ in the clustering step, i.e. they are not assigned a cluster and clusters are not based on these consumers.

For the clustering we apply a Lloyd's algorithm [10] with the Euclidean distance, to find a given number of k clusters. In our studies, we evaluate a dataset of tens of thousands of consumers at every clustering step. As such, we apply an approximation to the Euclidean distance. Instead of taking the real space for each dimension, we divide each dimension in a discrete number of bins. This effectively reduces the number of actual datapoints, since some consumers may be in the same bin in every dimension. This also allows a more efficient calculation of the Euclidean distance[2]. After the clustering step at time t we have a clustering $G^t = \{g_1^t, g_2^t, \ldots, g_k^t\}$, with $g_i^t \subseteq \mathcal{C}$ and $g_j^t \cap g_i^t = \emptyset$ for $i \neq j$.

---

[2] More details on this approximation can be found at github.com/YorickSpenrath/ICPM2021/blob/main/BitBooster.pdf

2.3   Training

In the training step we extract features and ground truth from the same time frame we use in the clustering. We use the period $[t - \tau - 1, t - 1)$ for the predictor values, and $[t - 1, t)$ for the value of the total turnover that is to be predicted, depicted in the centre box of Figure 2. For the purpose of our framework, we require any model that uses these sequences of data to make predictions. For the scope of this paper we use a long-short term memory RNN (LSTM) as it learns from sequential data and has the advantage of allowing incremental training. Each training point is a matrix where each row is one element of the sequence, and each column is one feature. More specifically, for a cluster $g_i^t \in G^t$ we compute a matrix M_i^t such that $M_i^t[a, b]$ is the value of feature f_b over $[t - \tau - 2 + a, t - \tau - 1 + a)$, averaged over all consumers in g_i^t.

For the LSTM architecture we use the one defined by [14], staying as close to that work as possible, only changing the input and output layers to match our input and output representations. In an online trainig setting, we train the model from scratch in the first time step, and update it at every next time step.

2.4   Predicting

During the predicting we construct the predictor values in the same way as discussed in Section 2.3, using the clusters found in Section 2.2. In other words, during a time step we use the same clusters to construct training and testing points. We use $[t - \tau, t)$ to construct the sequence, i.e. we shift predictor period by 1 week, as depicted in the right box of Figure 2. We use these predictor sequences to predict the average turnover of a consumer in the cluster over the week $[t, t+1)$, indicated as $\hat{y}_j^t$ for cluster g_j^t. This results in k predictions, one for each cluster. From this we can compute the total expected turnover by summing the products of the cluster sizes and cluster predictions. We define this value as $\hat{T} = \sum_{j=1}^{k} |g_j^t| \cdot \hat{y}_j^t$.

3   Experimental Evaluation

In this section we discuss the experimental evaluation of the method described in Section 2. For the experiment, we use data from a real-life retailer. Because of privacy restrictions, we cannot disclose all details. The transaction data comes from 39 weeks and contains over 160000 consumers with at least one purchase in that time. The number of visits per consumer varies between 1 and 312 with a mean of 55.0 and a standard deviation of 21.8. We discuss the experimental setup in Section 3.1 and then present and discuss the results in Section 3.2.

3.1   Experimental Setup

For the experiments we vary the two parameters of our framework: the sequence length τ and the number of clusters k. For τ we take $2, 4, 6, 8, 10$. For k we take

values $250, 500, 750, 1000$ and 2000η for $\eta = 1\ldots5$. We also add two special cases. The first has $k = 1$: all consumers belong to the same cluster. The second does not use clusters, i.e. every consumer has its own cluster. This is indicated by $k = |\mathcal{C}|$. These latter two can be regarded as competitors: existing solutions that do not use clustering to improve predictions. In total, we conduct $5 \cdot 11 = 55$ experiments.

Each prediction estimates the average turnover per consumer in each cluster. The ground truth of this is the actual average turnover per consumer in the cluster. Based on this, we define three metrics to asses the quality of each parameter combination at each time step. The first is the root mean square error (**RMSE**) on the cluster predictions. This is a measure of the prediction accuracy. We further identify the 10% of the consumers that have the largest decrease in their turnover with respect to the previous week. Formally, we label each consumer with `True` if their decrease in turnover is among the 10% highest of all consumers (top decile), and `False` otherwise. We can do this using the actual turnover (as true label) and using the predicted turnover of the cluster they belong to (as predicted label). Using this classification, we then compute the $\mathbf{F_1}$ score at each time step as a second quality metric. Finally, we compute the predicted total turnover using $\hat{T}$ from Section 2.4 and compare this with the actual total turnover. We compute the absolute percentage error (**APE**), $100\% \cdot |T - \hat{T}|/T$, as a third metric. The latter two metrics are measures of usefulness.

Our aim is to answer the following questions. 1) How does τ influence the **RMSE** (prediction accuracy)? 2) How does k influence the **RMSE** (prediction accuracy)? 3) How does k influence the $\mathbf{F_1}$ on the top decile (usefulness)? 4) How does k influence the **APE** in $\hat{T}$ (usefulness)? 5) How does k influence the $\mathbf{F_1}$ *over time* (usefulness)? 6) What are the considerations for a 'good' value for k (balance)? For the first four questions we average all of the above metrics over time; starting at $t = 11$. This is because if $\tau = 10$, the first prediction made is at $t = 11$. In this way, we average the same number prediction metrics for every experiment. The implementation of the framework is open-source and can be found at www.github.com/YorickSpenrath/ICPM2021.

3.2   Results

In this section we present and discuss the experimental results. Each combination of k and τ delivers one value for **RMSE**, $\mathbf{F_1}$ and **APE**, averaged over time. The results are presented in Figure 3. Each row contains the results for one value of τ, increasing k from left to right, each column contains the results for one value of k, increasing τ from top to bottom.

The effect of τ on RMSE (prediction accuracy) We first analyze the effects of τ on the **RMSE** in Figure 3A as this will be relevant in the discussions on k. We distinguish between values $k \leq 1000$ and $k > 1000$.

$\mathbf{k > 1000}$ For larger k, all experiments show a clear decrease of **RMSE** with increasing τ. This is expected, as a higher τ means that the data sequences for

(A) Cluster *RMSE*										
5±2	24±8	21±2	26±6	33±9	35±4	46±4	49±3	52±3	54±3	51±2
13±2	16±1	16±3	25±7	24±6	30±6	37±5	40±4	42±3	44±3	47±2
70±3	14±2	16±1	18±4	19±2	26±6	32±4	37±4	40±3	41±3	43±2
68±3	25±19	23±13	24±11	22±5	29±6	33±4	35±4	37±3	39±3	41±2
13±4	27±21	24±14	26±12	28±7	27±7	31±5	34±4	35±3	37±3	43±2
(B) F_1 on top decile										
.64±.01	.68±.01	.67±.01	.67±.01	.67±.01	.67±.01	.65±.01	.65±.01	.65±.01	.65±.01	.56±.01
.63±.01	.66±.01	.67±.01	.67±.01	.67±.01	.69±.01	.69±.01	.69±.01	.69±.01	.69±.01	.51±.01
.61±.01	.66±.01	.66±.01	.67±.01	.67±.01	.67±.01	.68±.01	.68±.01	.69±.01	.69±.01	.49±.02
.60±.01	.65±.01	.66±.01	.66±.01	.67±.01	.68±.01	.68±.01	.69±.01	.69±.01	.69±.01	.49±.02
.64±.01	.65±.01	.66±.01	.67±.01	.66±.01	.68±.01	.68±.01	.69±.01	.70±.01	.69±.01	.51±.03
(C) $\hat{T}$ absolute percentage error										
19±6	27±3	35±3	16±3	28±3	17±3	7±3	7±2	12±2	10±2	47±2
59±6	16±3	26±3	43±3	21±3	22±3	27±3	22±3	21±3	22±3	37±2
270±13	21±3	16±3	12±3	16±3	12±3	19±3	19±3	19±3	29±3	36±2
239±18	17±4	17±4	54±3	25±4	22±4	23±3	25±3	28±3	31±3	38±3
41±18	41±5	38±4	25±5	62±2	17±5	20±5	23±5	28±4	32±4	42±4
1	250	500	750	1000	2000	4000	6000	8000	10000	$|c|$

(τ values for the rows, top to bottom: 2, 4, 6, 8, 10. The x-axis is k.)

Fig. 3. Results for the experiments. Lighter colours indicate a better value, the colour scale is with respect to the values in each separate table. The **RMSE** is in monetary units, the **APE** are percentages.

each cluster are longer and the LSTM can learn from a longer period of time, which benefits its performance [18].

k ≤ 1000 For smaller k, the **RMSE** is not as consistently decreasing with increasing τ. The reason for this is that for a lower value of k we have fewer clusters of consumers and hence fewer training points. This makes the model possibly less stable, as it has less data to improve its performance. As a result, some models may fail to perform as expected. This means that longer sequences (higher τ) may result in poorer predictions than shorter sequences (lower τ).

The effect of k on RMSE (prediction accuracy) We next analyze the influence of k (left to right) on the **RMSE**. With the exception of $k = 1$, from Figure 3A we clearly see that the **RMSE** increases (worse predictions) with k. This is to be expected, as an increase in the number of clusters means that each cluster is smaller in size. As a result of this, the values in the clusters that are used to construct the sequences fed to the LSTM are based on fewer consumers. This increases the effect of outliers on the mean feature value and hence decreases the quality of the sequences on which the LSTM is trained. With the exception of $\tau \in [6, 8]$, $k = 1$ follows a similar trend, outperforming all other values of k. The exception for $\tau \in [6, 8]$ is likely caused by the above-mentioned

instability: the model is only updated with a single value every training step. This may still result in a decent model (as evident from other τ values) though there is no guarantee. While the **RMSE** for the other τ values is consistently lower than those for other k, we do note that $k = 1$ makes a single predictions for all consumers together. This means that little to nothing can be said about the individual consumers making it less useful than many clusters, where the predictions are on fewer consumers at a time.

The effect of k on F_1 (usefulness) Figure 3B clearly shows the inverse effect of Figure 3A in terms of k. An increase in k shows a clear increase in the F_1, making the resulting models more useful. There does seem to be a limit to this though, once we make predictions on individual consumers ($k = |\mathcal{C}|$), the F_1 decreases again. We expect the cause of this to be the further increase in **RMSE** for this value, though future research should look into this.

The effect of k on APE (usefulness) As depicted in Figure 3C, the relation between k and **APE** is less obvious than for the other two metrics. For lower values of τ, a higher value of k is preferred. For mid range values of τ, lower values of k perform better. For higher values of τ, the mid range of k shows a better performance. It is difficult to find the exact reason for this. One explanation is that each cluster prediction can either be too high (over-predicting) or too low (under-predicting). Summing all these predictions to compute the total turnover then effectively self-corrects these errors. There are then two factors affecting the total turnover error: the **RMSE** of the individual predictions and the self-correction effect. As discussed, the former increases with k. It is reasonable that the latter effect is more present for higher k. The question is which of these effects is stronger. As discussed before, the **RMSE** decreases with τ. As such, for lower τ, the stronger self-correction effect for higher k is more important to get a decent total turnover prediction. For higher values of τ, the reduced **RMSE** appears to allow lower values of k (with $k > 1000$) to be preferred. For the mid-range of τ values, it is not exactly clear why even lower k values are better, though this can be caused by the same model instability effect discussed before.

The effect of k on F_1 over time (usefulness) The results above have shown the effect of k on prediction accuracy and usefulness as average over all time steps. We now discuss some of the effects that can be viewed as the model progresses over time, presented in Figure 4. The numbers in the bottom row of Figure 3B are the averages of these plots. From the figure we clearly see the importance of making predictions in groups of consumers. The F_1 for making individual predictions ($k = |\mathcal{C}|$) quickly diminishes over time, stabilizing up to 0.25 points lower than the other experiments. Next to this, we see the effect of external events at three points in time, indicated by vertical dotted lines. These external events are likely to cause a sudden change in the consumer behaviour,

decreasing the predictive accuracy in the time around them. This results in a sudden drop in $\mathbf{F_1}$, especially for the first and third event.

The optimal value of k Based on the above results there are three considerations for the optimal value of k: Model stability (lower k means fewer training points), Cluster stability (lower k means better averaged clusters), and Cluster detail (higher k makes predictions closer to individual consumers).

4   Related Work

One class of supervised learning is called 'bucketing'. In bucketing, datapoints from a training set are first clustered using some clustering method, and a separate machine learning model is trained on each cluster. This approach is also extensively used in predictive process mining. Examples of such works are [5] (offline) and [6] (online). Our approach is different in the sense that we do not train one model per cluster, but train and update a single model using datapoints that are each extracted from a single cluster. While having a different target, the work in [4] applies clustering for the same reason as we do. The aim of that work is to discover process models that describe the sequences in an event log. A difficulty in discovering such process models is the variability in sequences. As a solution, the authors iteratively split the collection of sequences to create smaller event logs to create better models. The splitting is based on clustering to combine comparable sequences, much like our approach.

In Section 2.1 we described how we summarize a sequence of events over a period of time into a feature vector. This process is referred to as *encoding* or *embedding*. In the process mining field, different techniques of encoding exist. The most frequently used method to limit the number of events considered (prefix) to create evenly-sized vectors. These vectors then either list only the labels ($e.label$), or also each of the attributes ($e.v_1$ - $e.v_m$). Examples of the use of this encoding
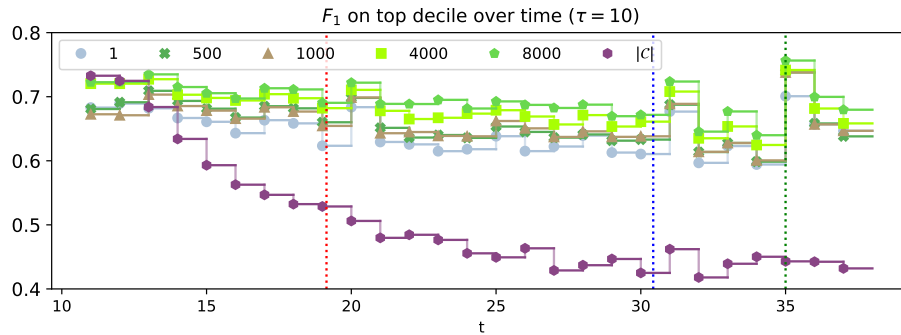
Fig. 4. Progression of $\mathbf{F_1}$ over time for $\tau = 10$. The vertical lines indicate significant external influences that can influence consumer behaviour.

are [5,6,9]. The disadvantage of this approach in our use case is that the number of events in a given time frame is highly relevant, limiting to a fixed-number of first events would lose this information. The frequency-based encoding we apply is also used in for example [11]. [9] further uses Hidden Markov Models (HMMs): a value of likeliness that a sequence belongs to the target class based on initially learned HMMs is added to the feature space. A more dedicated approach is to find relevant subsequences and count their frequencies, such as in [2,4]. While this can be highly relevant as an addition to our current encoding, it is computationally expensive to find which subsequences are relevant.

In terms of predictive process mining, this paper is part of a class of outcome prediction solutions. [14] adopts LSTMs to predict the remainder (suffix) of a case by repeated next activity predictions. The same target is predicted in [15] but then with the use of deep adversarial models. In [9], the authors predict whether an active case will be compliant or not according to the business process owner, leveraging complex encoding case as explained above. The same prediction task is executed by [5], which makes use of the bucketing described above. Using techniques from text mining, [16] aims to early signal whether a case will have a outcome that requires intervention using unstructured textual information from events. A more detailed summary of recent outcome-oriented tasks can be found in [17]. Next to this, literature contains specific to consumer behaviour prediction. Examples of these include the next interaction [8], losing a consumer (churning) [3,7,13], and life-time value [3]. Of these, [8] also uses a process mining oriented approach, and [13] also uses Neural Networks for their predictions. The work of [3] further suggests the use of automatically learned features over handcrafted ones for the prediction.

5   Conclusion and Future Work

In this paper we proposed a framework to make predictions about future events of consumer behaviour, aiming to strike a balance between accuracy and usefulness. Larger clusters lead to better predictions but say less about the individual consumers, and vice-verse for smaller clusters. Apart from this, a lower number of clusters likely causes the prediction model to be less stable as fewer training points are available. We also demonstrated the benefit of clustering consumers over time. When making predictions on an individual consumers, the usefulness ($\mathbf{F_1}$) rapidly decreases over time, this effect is not seen when consumers are grouped together for the prediction.

For future research, several directions can be identified. The most important one is how the size of the dataset affects the considerations for the optimal number of clusters. Another direction is to replace the linear fit clustering method. The framework operates on events that belong to consumers, and as such existing clustering methods from the process mining field, such as [2,9], are alternatives to this. Finally, at each time step the clusters are recomputed. An extension lies in incorporating past information on clusters, such that longer-term similarities in consumer behaviour can also be considered.

References

1. Van der Aalst, W.: Process Mining. Springer Berlin Heidelberg, Berlin, Heidelberg, 2nd edn. (2016)
2. Bose, J.C.R.P.: Process mining in the large. Ph.D. thesis, Eindhoven University of Technology (2012)
3. Chamberlain, B.P., Liu, B., Pagliari, R., Deisenroth, M.P.: Customer Lifetime Value Prediction Using Embeddings. KDD 2017 ADS **17** (2017)
4. De Medeiros, A.K.A., Guzzo, A., Greco, G., Van der Aalst, W.M.P., Weijters, A.J.M.M., Van Dongen, B.F., Saccà, D.: Process Mining Based on Clustering: A Quest for Precision. Lecture Notes in Computer Science **4928**, 17–29 (2007)
5. Di Francescomarino, C., Dumas, M., Maggi, F.M., Teinemaa, I.: Clustering-Based Predictive Process Monitoring. IEEE Transactions on Services Computing **12**(6), 896–909 (2019)
6. Di Francescomarino, C., Ghidini, C., Maggi, F.M., Rizzi, W., Persia, C.D.: Incremental predictive process monitoring: How to deal with the variability of real environments (2018), www.arxiv.org/abs/1804.03967
7. Günesen, S.N., Şen, N., Yıldırım, N., Kaya, T.: Customer Churn Prediction in FMCG Sector Using Machine Learning Applications. In: Artificial Intelligence for Knowledge Management. pp. 82–103 (2021)
8. Hassani, M., Habets, S.: Predicting next touch point in a customer journey: A use case in telecommunication. Proceedings ECMS **35**(1), 48–54 (2021)
9. Leontjeva, A., Conforti, R., Francescomarino, C.D., Dumas, M., Maggi, F.M.: Complex Symbolic Sequence Encodings for Predictive Monitoring of Business Processes. LNCS **9253**, 297–313 (2016)
10. Lloyd, S.P.: Least Squares Quantization in PCM. IEEE Transactions on Information Theory **28**(2), 129–137 (1982)
11. Song, M., Günther, C.W., van der Aalst, W.M.P.: Trace Clustering in Process Mining. In: Lecture Notes in Business Information Processing. vol. 17, pp. 109–120 (2009)
12. Spenrath, Y., Hassani, M., van Dongen, B.F., Tariq, H.: Why did my Consumer Shop? Learning an Efficient Distance Metric for Retailer Transaction Data. In: Proceedings of ECML PKDD 2020 LNCS Springer (2020)
13. Tariq, M.U., Babar, M., Poulin, M., Khattak, A.S.: Distributed model for customer churn prediction using convolutional neural network. Journal of Modelling in Management (2021)
14. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive Business Process Monitoring with LSTM Neural Networks. In: Advanced Information Systems Engineering. pp. 477–492. Springer International Publishing, Cham (2017)
15. Taymouri, F., La Rosa, M., Erfani, S.M.: A Deep Adversarial Model for Suffix and Remaining Time Prediction of Event Sequences. Proc. SDM pp. 522–530 (2021)
16. Teinemaa, I., Dumas, M., Maggi, F.M., Francescomarino, C.D.: Predictive Business Process Monitoring with Structured and Unstructured Data. LNCS **9850**, 401–417 (2016)
17. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-Oriented Predictive Process Monitoring: Review and Benchmark. ACM Trans. Knowl. Discov. Data **13**(2), 1–57 (2019)
18. Wen, Y., Zhang, W., Luo, R., Wang, J.: Learning text representation using recurrent convolutional neural network with highway layers. In: Neu-IR workshop preceedings, SIGIR'16 (2016)