Clustering-based Aggregations for Prediction in Event Streams

Author One[1*], Author Two[1] and Author Three[1]

[1]Department, Institution, City, Country.

*Corresponding author(s). E-mail(s): a.one@institution.edu;
Contributing authors: a.two@institution.edu; a.three@institution.edu;

Abstract

Predicting the behaviour of shoppers provides valuable information for retailers, such as the expected spend of a shopper or the total turnover of a supermarket. The ability to make predictions on an individual level is useful, as it allows supermarkets to accurately perform targeted marketing. However, given the expected number of shoppers and their diverse behaviours, making accurate predictions on an individual level is difficult. This problem does not only arise in shopper behaviour, but also in various business processes, such as predicting when an invoice will be paid. In this paper we present CAPiES, a framework that focuses on this trade-off in an online setting. By making predictions on a larger number of entities at a time, we improve the predictive accuracy but at the potential cost of usefulness since we can say less about the individual entities. CAPiES is developed in an online setting, where we continuously update the prediction model and make new predictions over time. We show the existence of the trade-off in an experimental evaluation in two real-world scenarios: a supermarket with over 160 000 shoppers and a paint factory with over 171 000 invoices.

Keywords: Predictive Process Mining, Stream Analysis, Clustering, Shopper Behaviour

1 Introduction

Knowing the future behaviour of shoppers is important to help retailers plan ahead [1]. One way to do so is by predicting how shoppers will behave on an individual level. Knowing which shoppers are expected to increase or decrease their spending allows retailers to apply more targeted marketing strategies. Unfortunately, the human nature of shoppers makes it difficult to accurately predict on an individual level. Two shoppers can behave similar for some time, but then quite different in the next week. This problem becomes extra difficult using product level data, which is often sparse [2]. Another way to make predictions is by taking all shoppers together. This can then be used to predict the total store turnover 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. Data analyzed at this high level of aggregation can also be used to find colloquial regions that explain purchase differences across different geographical locations [3]. This trade-off is schematically presented in Figure 1.

Similar considerations exist in different settings, such as the ordering process of a paint factory, where we want to predict the time between receiving and paying an invoice. Other work regarding prediction of human-based behaviour is discussed in [4].

In this paper we aim to strike a balance between accuracy and usefulness. Instead of making predictions on *individual* entities we make

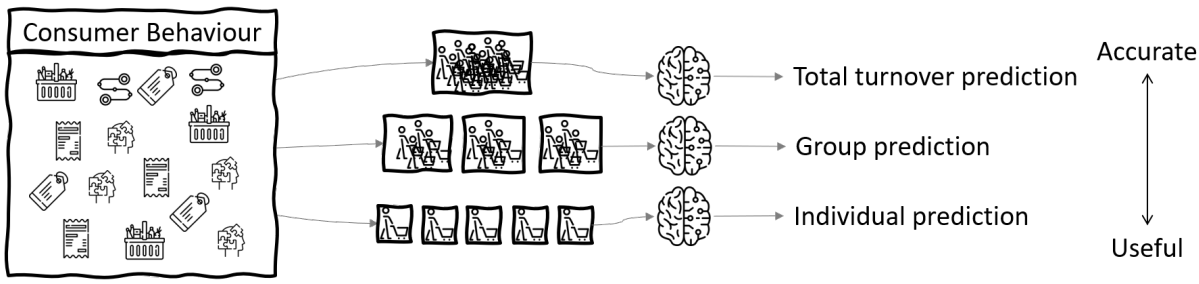

Fig. 1: Overview of the problem. Making predictions for individual consumers is more useful, but less accurate. Making predictions for all consumers (i.e. the entire retailer) is easier, but not useful on individual consumers. This paper balances the two by making predictions using groups of consumers.

predictions on *groups* of entities. The advantage is an increase in accuracy with respect to making predictions for individual entities 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 entities together. This is because the predictions are on a limited number of entities at a time. We call this framework 'Clustering-based Aggregations for Predictions in Event Streams', or CAPiES for short.

The outline of our method is shown in Figures 2 and 3, showing the training and prediction phase respectively. Given a set of events, we want to make predictions about entities that relate to these events. Without CAPiES, the training phase consists of three steps: extracting entities from the set of events relevant for training, encoding entities and training the prediction model. In the first step, we analyze the events and select which entities are suitable to update the prediction model. In the second step, this collection of entities is encoded into numerical values, assigning feature values and outcomes for each entity. In the training step, these values are used create a prediction model that learns the relation between the input (features) and output (outcomes). In CAPiES we add two intermediate steps: clustering and averaging. Instead of using the encoded entities to directly update the model, we first partition the set of entities into smaller groups. The idea is that we combine entities that are similar, in other words we apply clustering based on the encoded entities. Next, we compute a *proxy* entity that represents each cluster. The encoding of this proxy entity is obtained by averaging the encoding of the individual entities in the respective cluster. We then use these proxy entities in the training step to learn the prediction model. Without our framework, the prediction phase consists again of three steps: extracting, encoding and predicting. The first step is similar to the training phase, and selects entities for which a prediction can be made. Note that these can be different entities. The second step is the same as the training phase and encodes each entity of these entities, assigning feature values. In the predicting step the trained model is used to determine a predicted outcome using the input (feature values). In CAPiES we again first encode, cluster and average the entities (in the same way as we did during the training phase). This results in a proxy entity for each cluster, its encoding again computed by averaging the individual entities of each cluster. For each of these proxy entities we make a prediction of the outcome. We next assign this prediction to each individual entity in the respective cluster.

In our framework, a single parameter controls how many clusters are computed in the clustering step of the training and prediction phase. This parameter, ρ, is defined as the average number of entities per cluster. This means that on average for every ρ entities in the data, a single proxy entity is computed which is used for training or prediction. This introduces a trade-off. For lower values of ρ, fewer entities are clustered together, meaning that each proxy entity is closer to the original entities, the predictions are closer to an individual entity level, as such making them more useful. However, increasing ρ removes the effect outliers have on the predictions and training. Apart from this, an increase in ρ also decreases the number of proxy entities, which on the one hand potentially

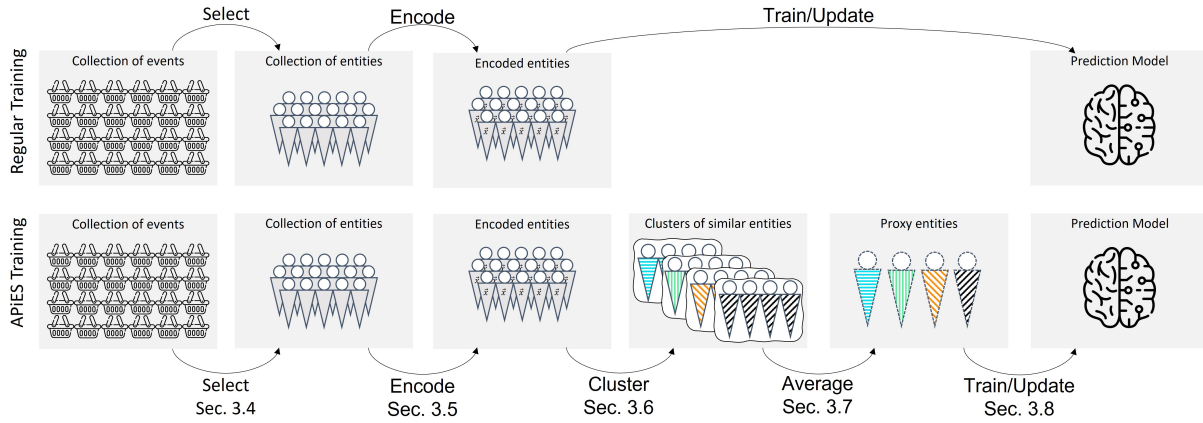

Fig. 2: Overview of CAPiES in the training step

reduces number of training points and hence the quality of the model.

Note that the framework does not depend on the specification of the clustering (algorithm, distance metric, other parameters) and the model or its training (model type, classification/regression, train/validation splits, hyper-parameter optimization). Furthermore, while the above description assumes a single training of the model, the same idea can be applied to multiple training phases of a model, for example in a streaming setting where the model updates and entity predictions occur at regular intervals.

In this paper, we define CAPiES in a streaming setting: discovering clusters of entities, updating the prediction model, and making predictions at regular intervals. We as such make the following contributions: 1) we propose a framework to overcome the loss in prediction accuracy for diverse entities by making the predictions on carefully selected clusters of entities, 2) we do so in a streaming environment, and 3) we show its effectiveness in two different real-world datasets, one from a supermarket (where entities are shoppers) and one from a paint factory ordering process (where entities are invoices).

The content in this paper is an extension to earlier work[1]. With respect to that paper we make the following extensions: 1) instead of a framework tailored towards supermarket shoppers specifically, we propose a generic framework that works on a variety of datasets. This generic framework, CAPiES, is expressed in a more formal manner. 2) Furthermore we change the parameter k (indicating the number of clusters searched for) to ρ (the average number of entities in a cluster), which generalizes better. 3) we apply CAPiES to the dataset used in the original work, as well as to a completely new dataset. This new dataset comes from a different domain, describing the ordering process in a paint factory. In the current paper, Section 3 is fundamentally different from the corresponding section in the original version. Next to this, Section 4.1 contains updated results and discussions of the existing dataset, Section 4.2 describes the results of experiments on the new dataset and Section 4.3 provides an extended discussion on the results of both experiments together. All other sections have been updated to reflect the new framework and additional experiments.

The rest of this paper is organized as follows: we first compare the above overview to existing literature in Section 2. Next, in Section 3 we explain the details of CAPiES, each step consisting of a generic formalization, how it applies to a working example of supermarket data, and how that specific step relates to existing literature. Following this, we evaluate CAPiES in experimental settings in Section 4. We conclude the paper in Section 5

2 Related Work

In this section we discuss related work, but we limit this to comparisons with CAPiES in general and making predictions in event based use cases.

[1] An anonymized version of this earlier work can be found at https://anonymous.4open.science/r/ITEM-CAPiES/Original%20Version.pdf

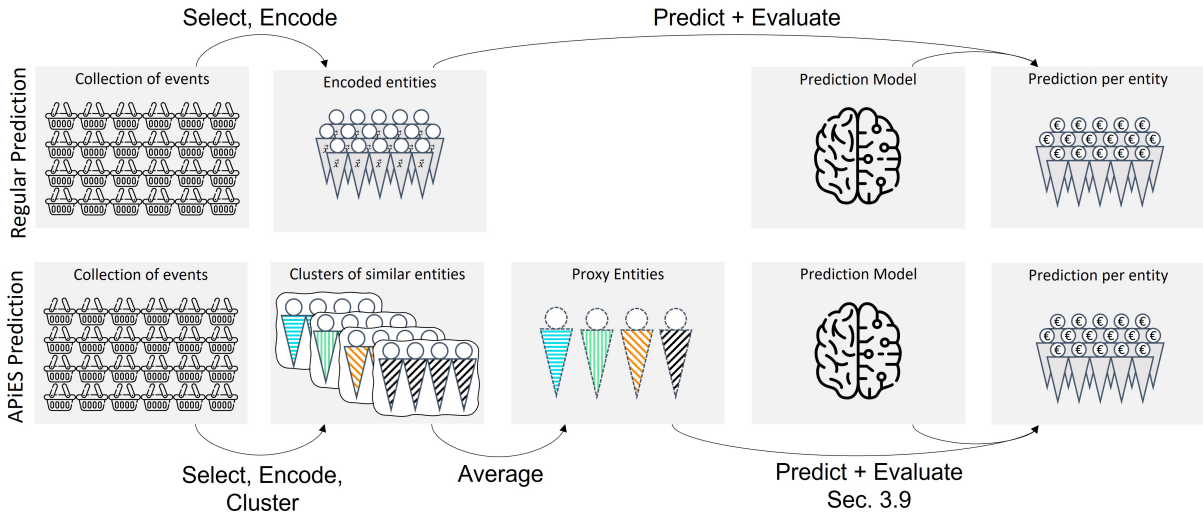

Fig. 3: Overview of the framework in the prediction step

A detailed comparison between specific steps in CAPiES and existing literature is incorporated in Section 3.

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 [7] 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 terms of predictive process mining, this paper is part of a class of outcome prediction solutions. [8] adopts LSTMs to predict the remainder (suffix) of a case by repeated next activity predictions. The same target is predicted in [9] but then with the use of deep adversarial models. In [10], the authors predict whether an active case will be compliant or not according to the business process owner. The same prediction task is executed by [5], which makes use of the bucketing described above. Using techniques from text mining, [11] 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 [12].

Next to this, literature contains work specific to shopper behaviour prediction. Examples of these include the next interaction [13], losing a shopper (churning) [1, 14, 15], and life-time value [1]. Of these, [13] also uses a process mining oriented approach, and [15] also uses neural networks for their predictions.

3 Framework

In this section, we present CAPiES. We start with some notation preliminaries in Section 3.1. We then present a running example in predicting shopper behaviour in a supermarket as working example in Section 3.2. We next formalize CAPiES in Sections 3.3 to 3.9, each section formalizing part of Figures 2 and 3, relating it to the supermarket use case, and making a comparison to existing literature. Next we describe another use case: paying invoices in a paint factory, in Section 3.10. Both use cases are from real-world datasets, which are used in Section 4 to evaluate

CAPiES. All symbols used in the explanation are summarized in Table 1.

3.1 Preliminaries

In remainder of this paper, we use $\mathbb{N}$ to denote all natural numbers, starting from 0, and $\mathbb{N}_+$ for $\mathbb{N} \setminus \{0\}$. The positive real numbers are $\mathbb{R}_+$, and $\mathbb{R}_{\geq 0} = \mathbb{R}_+ \cup \{0\}$. Finally, the empty set is denoted as $\emptyset$. Vectors and matrices have indices that start with 1.

In this paper we consider systems that are described by a collection of *events* involving *entities*. Let $\mathcal{E}$ be the universe of all events and let $\mathcal{C}$ be the universe of all entities. An *event* e describes something happening, labelled by activity label $e.act \in \mathcal{A}$ (what happened?) relating to some entity $c = e.ent \in \mathcal{C}$ (to what did it happen?) at some time $e.time \in \mathbb{R}_{\geq 0}$ (when did it happen?). Each event has additional information $e.attributes \in \mathcal{F}_{event}$ called *event attributes* and each entity c has additional information $c.attributes \in \mathcal{F}_{entity}$ called *entity attributes*. Let $a, b \in \mathbb{R}_{\geq 0}$ with $a < b$. We define $\mathcal{E} \downarrow^{[a,b)}$ as the subset of $\mathcal{E}$ with events that occur on or after a and before b, formally:

$$\mathcal{E} \downarrow^{[a,b)} = \{e \in \mathcal{E} | e.time \in [a, b)\}$$

Furthermore, for an entity c, we define $\mathcal{E} \downarrow_c^{[a,b)}$ as the subset of $\mathcal{E} \downarrow^{[a,b)}$ with events that belong to entity c, formally:

$$\mathcal{E} \downarrow_c^{[a,b)} = \{e \in \mathcal{E} | e.time \in [a, b) \wedge e.ent = c\}$$

Let $\mathcal{E}'$ be a subset of $\mathcal{E}$. We define the function $\vec{f}(\mathcal{E}') \in [0, 1]^{|\mathcal{A}|}$ as a vector that lists the relative frequency of each activity in $\mathcal{A}$, i.e.

$$\vec{f}(\mathcal{E}') = (|\{e \in \mathcal{E}' | e.act = a\}| \,/\, |\mathcal{E}'|)_{a \in \mathcal{A}}$$

In some literature, $\vec{f}(\mathcal{E}')$ is known as the *Parikh* vector of the activities in $\mathcal{E}'$, normalized to sum to 1.

3.2 Running Example

In this section we first describe the supermarket use case. Shoppers visit this supermarket to purchase their groceries. They may do so several times a week, each visit recording what was purchased, at what time the purchase was made, and by which shopper. The latter is known because shoppers have an incentive to use loyalty cards, as they can save up for rewards by purchasing groceries. These loyalty cards allow shoppers to be identified from one visit to the next. Each of the visits can be described by a list of products purchased, including the quantity and price paid. The collection of information in these visits over a specific time period for a single shopper constitutes the 'shopper journey'.

Based on [16], we assign a label to each visit. The idea of this label is that it provides some information regarding the products purchased, such that visits with similar products, categories of products and quantities have the same label. These labels are learned using an unsupervised method. For full details of this labelling, please refer to [16]. Note that this method is applied to *single visits*; the shopper journey themselves are not categorized by this method. This can as such be seen as part of the feature extraction that describes the shopper journey, analyzing the labels given to visits in the shopper journey. Next to this, we can also compute aggregate values for each visit, presented in Table 2.

The goal is to make a prediction about one or more future visits of a shopper. For our application this is the total spend over a given period of time, the sum of the spend in the separate visits. At the end of every week we know the total spend of the completed week, which we use to update a prediction model, and we predict the total spend of the next week, given new information about the shopper journey. To make a prediction and train the model we extract features from several weeks of transaction data, together forming the shopper journey. In the running example we consider this to be a constant number of three weeks, but in the experimental evaluation we also look at different lengths of shopper journeys for input.

Like most other real-life applications, journeys can vary wildly between different shoppers. This makes it difficult to make predictions about individual shoppers. In CAPiES we partition shoppers into clusters of similar shoppers. We update the prediction model using proxy shoppers averaged from the clusters instead of using individual shoppers. We further make predictions for proxy shoppers instead of for each individual shopper. In

Table 1: Symbols used in CAPiES. Note that these are not parameters of the framework, just symbols indicating parts of it.

Symbol	Meaning	Symbol	Meaning
$\mathcal{C}$	Set of entities	$y(c,t)$	Outcome of entity c at time t
$\mathcal{C}_t^T$ / $\mathcal{C}_t^P$	Entities at t for training / predicting	$X(c,t)$	Feature values of entity c at time t
$\mathcal{E}$	Set of events	G_t^T/G_t^P	The clustering of $\mathcal{C}_t^T$ / $\mathcal{C}_t^P$
$\mathcal{E}\downarrow^{[a,b)}$	Events in the time frame $[a,b)$	$\tilde{y}(g)$	The average outcome in cluster g
$\mathcal{E}\downarrow_c^{[a,b)}$	Events in $\mathcal{E}\downarrow^{[a,b)}$, belonging to entity c	$\tilde{X}(g)$	Average feature values in cluster g
$\vec{f}(\mathcal{E}')$	Relative frequencies of activities in $\mathcal{E}'$	$\hat{\tilde{y}}(g)$	Prediction for proxy entity of cluster g
$\mathcal{A}$	Set of activity names	$\hat{y}(c,t)$	The prediction for entity c at time t

other words, we learn the behaviour of a *group of shoppers* and make a prediction on their behaviour *as a group*. We formalize this idea in the next sections, referring back to this use case as running example.

3.3 Streaming setting

In this paper we apply machine learning in a streaming setting. In such a setting, events e occur in the order defined by their time of occurrence, $e.time$. We apply batch learning to this stream: after every set amount of time we use events we have collected to update a prediction model and make predictions about entities. Without loss of generality, let $t = 0$ be the time of the first event, and let the time between two batches be 1 time unit, in other words: we update our prediction model at $t = 1, 2, 3, \ldots$. After updating the model at time $t-1$, we keep collecting events up to time t, this collection is indicated by $\mathcal{E}\downarrow^{[t-1,t)}$. From the collection we extract information about two sets of entities: those that are suitable for updating the model (those for which the outcome is known) and those used for prediction (those for which we want to predict the expected outcome). Depending on the use case, these two sets may overlap or be distinct.

In the supermarket use case, we collect information about transactions (events) of shoppers (entities). Each transaction describes the type of purchase ($e.act$, $|\mathcal{A}| = 8$), when the purchase was made ($e.time$), which shopper purchased it ($e.entity$) and additional attributes described in Table 2 ($\mathcal{F}_{event} = [0,1] \times \mathbb{R}_+ \times \mathbb{R}_+ \times \mathbb{R}_{\geq 0} \times \mathbb{N}$). While shoppers may provide demographic information when applying for the loyalty card, this information is not part of the scope of this research ($\mathcal{F}_{entity} = \emptyset$). We update and use our model at the end of every week (one time unit is a week).

In related work, different use cases are evaluated as well. In [17] the authors discover process models from an events stream describing a loan application use case. In [18] the authors analyze an event stream relating to a maintenance service company, proposing a way to overcome gradual and recurrent concept drift in supervised learning. Another example from the process mining field is discussed in [19], where a permit application process is discussed. Use cases from supermarkets are also discussed in literature. Examples include the inter-arrival time prediction of shoppers [20], basket content predictions [21] and adoption of native shopping habits by immigrants [22]

3.4 Training Collection

The first step of CAPiES, which also applies outside of the framework, is to determine which entities, and as such which events, are used in the training phase. This is a subset of $\mathcal{C}$, which we denote as $\mathcal{C}_t^T$, the superscript T indicating that we use the set in the training phase and the subscript t indicating this set is used at time t. The requirement for an entity to be part of $\mathcal{C}_t^T$ is that we have data needed to build the features (the input for the prediction model) as well as the outcome (the output for the prediction model). This requirement depends on the use case.

For the supermarket use case, we require that shoppers have made transactions at the store for at least four weeks: the first three weeks are used for feature engineering to define the shopper journey, the fourth week is used to extract the

Table 2: Values derived from a single visit.

Value	Description	Domain
e.attributes.freshness	Fraction of perishable items	$[0, 1]$
e.attributes.item_value	Average value of the items bought	$\mathbb{R}_+$
e.attributes.product_density	Average frequency of each product	$\mathbb{R}_+$
e.attributes.total_value	Total price paid for the the products	$\mathbb{R}_+$
e.attributes.total_item_count	Total number of items purchased	$\mathbb{N}_+$

outcome: the total spend of each shopper. Next to this we also require shoppers to have at least one transaction in the first three weeks to construct a shopper journey. This is visually depicted in Figure 4. Let $start(c)$ be the time of the first visit of shopper c. We formally have that

$$\mathcal{C}_t^T = \{c \in \mathcal{C} | start(c) < t - 3 \wedge \mathcal{E}\downarrow_c^{[t-4,t-1)} \neq \emptyset\}$$

In the setting of CAPiES we limit ourselves to use cases where it is clear to which entity an event belongs. Put differently, each entity can be considered as a 'generating process' that emits events related to it, and we can distinguish different sources. This is a common assumption in many real-world datasets, specifically to process mining [23] (events have a 'case identifier'), retailer data [20, 21, 24] (visits belong to some sort of loyalty card) and online customer journeys [25] (journeys of web clicks that belong to a specific user). As such we argue that this is a valid assumption. In related work, this assumption is sometimes dropped, and the focus lies on identifying the sources of events. In [26], the authors do not assume to know these sources and try to link incoming information to entities by finding resemblance between new and existing data. In [27], the authors also deal with events for which the entities (case identifiers) are not known, solving the problem by estimating a process model from incoming events and computing the probability that it belongs to an existing entity. A similar problem is solved in [28].

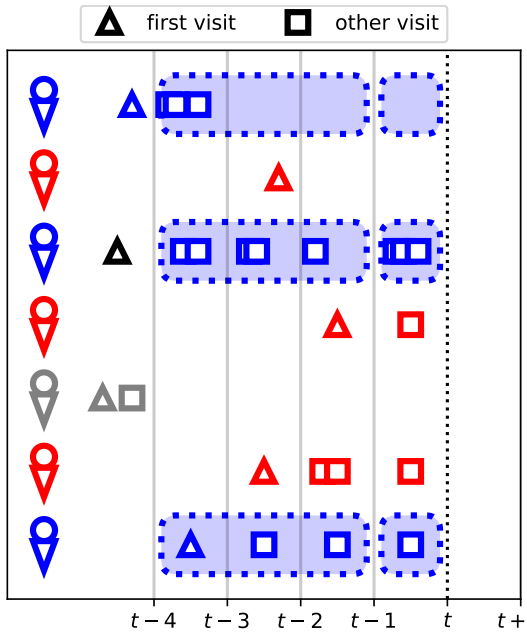

Fig. 4: Shoppers selected for training. The grey shoppers did not have a visits in the three weeks from which the journey is extracted. The red shoppers do not have a long enough journey since their start to be used in training. The blue shoppers have enough data to be used in training the model.

3.5 Encoding

For encoding an entity we combine information from past events of that entity. Specifically, we create values $X(c, t)$ and $y(c, t)$. The former describes the input data for the prediction model, the latter describes the outcome for that entity.

In the supermarket example, at time t we use the past week ($[t-1, t)$) for the outcome (the total spend of the shopper). The outcome is simply the sum of the value of all transactions:

$$y(c, t) = \sum_{e \in \mathcal{E}\downarrow_c^{[t-1,t)}} e.attributes.total\_value$$

With the help of domain knowledge[2] we convert events of a single shopper to an encoding of the shopper journey. To this end, we compute descriptive statistics of the transactions a shopper does in a single week, as such creating three sets of descriptive values to define the journey of a shopper over the period of three weeks. We take the average of the first three values of Table 2 and the sum for the final two values. We use the sum for e.attributes.total_value and e.attributes.total_item_count as this helps distinguishing shoppers with a large total spend from shoppers with a smaller total spend. Next to this we also use the relative frequency of each label and the number of visits. In total this creates a matrix of 14 rows (5 aggregates, 1 for the number of visits, and 8 activity label frequencies) and 3 columns (1 for each week).

In related work, there are alternatives to encoding, which is otherwise referred to as 'embedding'. In other works in the process mining field (where entities are often referred to as *cases*), the most frequently used method is to limit the number of events considered (prefix) per entity to create evenly sized embedding tensors, each row describing a single event, and each column describing a dimension of the event, such as a one-hot-encoded activity label or any of the event or entity attributes. Examples of such uses include [5, 6, 10]. The disadvantage of this approach in the supermarket use case is that the number of events in a time frame is highly relevant. The solution where all events in a single week are aggregated together allows the use of all events (including information about the number of events), while having evenly-shaped vectors for each entity (shopper) to accommodate machine learning model inputs. The frequency-based feature values we use are also used in for example [29]. Further alternatives are Hidden Markov Models [10], and frequencies of common subsequences [7, 30]. The disadvantage of these is that finding such models and relevant subsequences can be computationally expensive. Finally, the work of [1] further suggests the use of automatically learned features over handcrafted ones for the prediction.

[2]The authors gratefully thank the company [*redacted*] for making their data available for this project and their useful feedback on CAPiES.

3.6 Clustering

Traditionally, we would update a prediction model using the predictor-outcome values $(X(c,t), y(c,t))$ from Section 3.5 for the entities c in $\mathcal{C}_t^T$. In this paper we propose an intermediate step. Instead of using *individual* entities to train the model, we first create *clusters* of entities, and then update the model with proxy entities of each cluster. The number of clusters we use depends on the size of $\mathcal{C}_t^T$. Let $\rho \in \mathbb{N}_+$. The number of clusters $\mathcal{C}_t^T$ is partitioned into is:

$$k = \left\lceil \frac{|\mathcal{C}_t^T|}{\rho} \right\rceil \tag{1}$$

Put differently, we create a number of clusters such that the average cluster size is roughly equal to ρ. This leaves us with a group of k clusters, which is denoted as G_t^T, the super- and subscript defined similar as in $\mathcal{C}_t^T$. The advantage of choosing the average cluster size as parameter instead of the number of clusters is that the number of entities from one time step to the next may differ. As a result, the number of clusters learned will also differ. As we update the prediction model with a proxy entity from each cluster, this means that if a time step has more entities suitable for updating prediction model, the prediction model is updated with more proxy entities. Conversely, for a smaller $\mathcal{C}_t^T$ the prediction model is updated with fewer proxy entities. As we see in Section 3.10, the number of entities suitable for prediction varies greatly over time in the paint factory use case. In the original work, we used the number of clusters as parameter instead, which influenced the stability of the results for the paint factory dataset.

Of special note are two values: $\rho = 1$ and $\rho = |\mathcal{C}|$. For $\rho = 1$, CAPiES is effectively equal to not using it: proxy entities are the same as actual entities and individual entities are used for training (and, later on, predicting). As such, the value $\rho = 1$ can be considered as a comparison method, where CAPiES is not used. For $\rho = |\mathcal{C}|$, all entities are added to a single cluster; only a single proxy entity is used in prediction and training. These extreme values match the trade-off descriptions given earlier.

CAPiES does not depend on the choice of the clustering algorithm or the distance metric used for the clustering itself, just that there is *a* way of separating the entities in a given number

of subsets. This is mostly because the availability of clustering algorithms and distance metrics also depends on the use case; as we shall see the supermarket dataset requires a different approach than the paint factory dataset. CAPiES focuses on improving the predictive quality of existing models by grouping together similar entities, independent of the way in which these groups are created.

For the supermarket, we measure the distance between two shoppers by the difference in the progression of their journey over the three weeks. More specifically, for each of the 14 features discussed in Section 3.5 we fit a linear function for each shopper on the values of that feature over time (in the journey), estimating feature f_v as $a_v \cdot t + b_v$ with residual r_v. Shoppers with similar values for these coefficients have a similar average (b_v), progression over time (a_v) and linear fitness (r_v). Using these $3 \cdot 14 = 42$ coefficients, we cluster the shoppers using the Euclidean distance and Lloyd's algorithm for k-medoids [31]. Because we are clustering a large number of shoppers at every time step, we use an efficient approximation of the Euclidean distance, which also requires the use of k-medoids over k-means. Details of this approximation are discussed in [32].

Specifically for the supermarket scenario, we have chosen the measure the similarity of the linear fits of the individual shoppers as (inverse) distance metric for the clustering. An alternative to this approach is discussed in [33]. The authors propose a iterative learning approach that learns clusters of entities and a distance metric at the same, also with event-based features. The technique incorporates domain knowledge. This technique can be used in a preprocessing phase of CAPiES to determine a relevant distance metric over the entities. [7] uses the Euclidean distance on their defined encoding, together with k-means. In [5], DBSCAN [34] is used for the clustering of sequences of events. DBSCAN is not suitable for CAPiES as we define the number of cluster we aim for by design, where DBSCAN is based on different parameters that controls the density of the clusters instead of the number of clusters.

3.7 Averaging

Having found the partition G_t^T of entities, we next compute the proxy entities. For cluster $g \in G_t^T$, this is done by taking the average of $X(c,t)$ for each entity in g, creating proxy entity $\tilde{g} = (\tilde{X}(g), \tilde{y}(g))$ with proxy feature values $\tilde{X}(g)$ and proxy outcome $\tilde{y}(g)$. As such, we create a total of $k = \left\lceil \frac{|\mathcal{C}_t^T|}{\rho} \right\rceil$ proxy entities. Note that this assumes that each dimension in X can be averaged.

In the supermarket use case, each proxy entity $\tilde{g}$ is again represented as a 14 by 3 matrix, the value at row a and column b containing the average feature value at row a and column b over all shoppers in g. This creates a total of $k = \left\lceil \frac{|\mathcal{C}_t^T|}{\rho} \right\rceil$ proxy shoppers. Each of these features is a numerical feature, with a meaningful average if aggregated over multiple shoppers.

In CAPiES we take an average over a cluster of entities. Another options is to use one of the entities as representative. This can either be the medoid, which is a side-product of k-medoids, or the entity closest to the mean; though these may be the same. The latter is used in [35] to sample the majority class of a unbalanced dataset in supervised machine learning. Depending on the set of entities, the mean entity may be very close to the medoid entity. As shown in Figure 5, the distance between these two decreases when the number of entities over which they are taken increases. As a result, the medoid entity is less representative of the mean entity for a lower number of entities than for a higher number of entities. This may influence the results between different values of ρ, which have precisely this difference. As such, we use the mean entity as proxy entity.

3.8 Updating

In the final part of the training phase we train/update the model. In the first time step of the streaming setting a model is created from scratch (cold start), and in subsequent time steps the model is updated. As such the only requirement that we have for the model is that it is incremental: after initially training, updates to it with new training data must be possible.

CAPiES does not depend on the choice of the prediction model, its outcome type (classification or regression) or any of its hyper-parameters, just that there is *a* prediction model that can be updated after initial training. This is mostly because the most suitable model depends on the

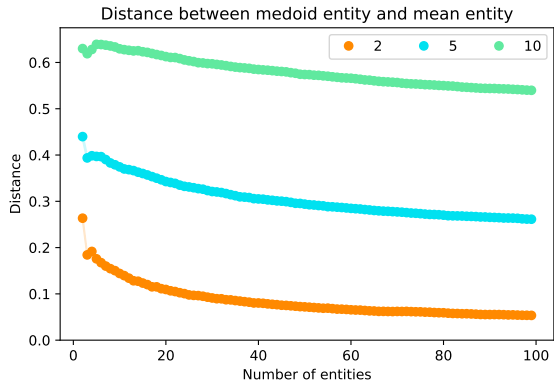

Fig. 5: Average Euclidean distance between mean and medoid entity, taken over 1000 random samples as function of the number of entities, for 2, 5 and 10 dimensions.

use case; as we shall see the supermarket dataset requires a different approach than the paint factory dataset. CAPiES focuses on improving the predictive quality of existing models by grouping together similar entities, independent of the model.

In the supermarket use case, we use a long-short term memory RNN (LSTM). LSTMs are trained on *sequences* of data and allow for incremental training. This makes them ideal for this use case. In an online training setting, we train the model from scratch in the first time step, and update it at every next time step. LSTMs are trained on sequences of data. Each element in the sequence is one week of shopper journey, so a single column of $\tilde{X}$. Columns are then added to the LSTM one at a time, which is trained to predict $\tilde{y}$.

In related work on predicting based on sequences of data, LSTMs are more often used. [8] uses it to predict the remainder (suffix) of a case by repeated next activity predictions. [9] uses deep adversarial models. Next to deep learning models, a decision tree or random forest is often used for predictions based on events, such as in [5, 10, 11]. The encoding in these works does not result in sequences, making the use of a decision tree/random forest more applicable than in the supermarket scenario. Next to this, decision trees and random forests are not incremental by themselves. Incremental variants do exist as Options Trees and their extensions, discussed in [36, 37] and implemented in [38, 39]. These have also been used in event stream predictions such as [40, 41]. We use one of these incremental variants in the paint factory use case of Section 3.10.

3.9 Prediction phase

In the prediction phase (Figure 3) we apply similar steps as in the training phase, with some minor differences.

Prediction Entity Collection: We first determine which entities are suitable for making a prediction for. We denote this subset of $\mathcal{C}$ as $\mathcal{C}_t^P$, the superscript now indicating that we use the subset in the predictions at time t. The requirement for an entity to be part of $\mathcal{C}_t^P$ is slightly different from $\mathcal{C}_t^T$: we do not require the outcome of an entity, just the data needed to build the features used by the prediction model. The exact requirement again depends on the use case.

In the supermarket setting, we require that shoppers have made transactions at the store for at least three weeks ($[t-3, t)$); these together form a shopper journey that can be encoded to be used in the clustering (to find similar shoppers) and later in prediction (after aggregating shopper clusters to a proxy shopper). This is in contrast with the training collection, where four weeks were needed to also have the outcome value. The outcome value for this data is from the week following t ($[t, t+1)$), the value we want to predict. Next to this we also require shoppers to have at least one transaction in those weeks (so shoppers that had their last purchase more than three weeks ago are not used in the prediction phase). This is visually depicted in Figure 6. As such we have:

$$\mathcal{C}_t^P = \{c \in \mathcal{C} | start(c) < t - 2 \wedge \mathcal{E} \downarrow_c^{[t-3,t)} \neq \emptyset\}$$

Encoding, Clustering, Averaging: For the encoding we apply the same procedure as for the training phase, the difference being the entities on which the steps are applied: $\mathcal{C}_t^T$ instead of $\mathcal{C}_t^P$. This creates partition G_t^P, in which each cluster g is averaged into a proxy entity with feature values $\tilde{X(g)}$.

For the supermarket, all three of the encoding, clustering and averaging steps are the same, but now applied to $\mathcal{C}_t^P$ instead of $\mathcal{C}_t^T$.

Note that because of the requirements for $\mathcal{C}_t^P$ and $\mathcal{C}_t^T$, we have that if a shopper is in the shopper collection for predictions at time t ($\mathcal{C}_t^P$) it will also

be in shopper collection for training at time $t+1$ ($\mathcal{C}^T_{t+1}$), by which time we have the actual outcome as well. In other words, $\mathcal{C}^P_t = \mathcal{C}^T_{t+1}$ for all t. As such, the encoding and clustering only needs to happen once every time step, we can reuse the results of the prediction phase of the last time step for the prediction phase in this time step.

Prediction: In the prediction step, we use the proxy entity feature data, $\tilde{X}(g)$ to predict the outcome $\hat{\tilde{y}}(g)$ for the proxy entity of cluster g. We then also assign this same prediction to all entities in g. In the next time step we have the ground truth outcome for each of the entities; we can then assess the quality of the model at time t by comparison of the ground truth with the predicted outcome. The metrics used for this depend on the use case.

For the supermarket use case we compute four metrics to assess CAPiES. These metrics are either focused on accuracy or on usefulness of the prediction, to asses the trade-off show in Figure 1. The metrics are summarized in Table 3.

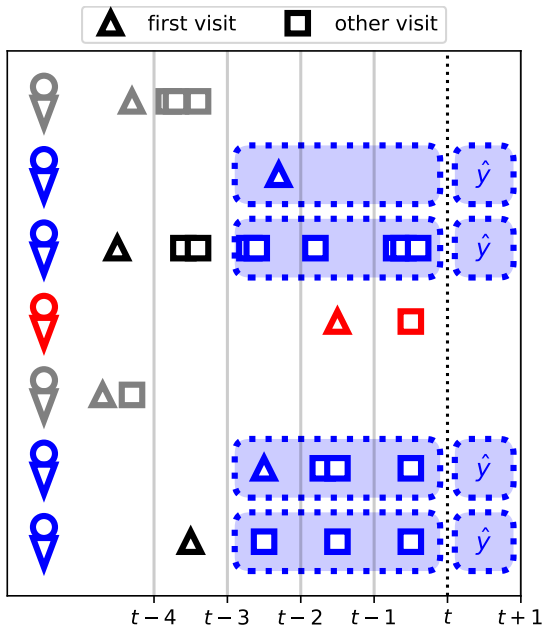

Fig. 6: Shoppers selected for prediction. The grey shoppers did not have visits to construct a journey in the past three weeks. The red shoppers did have visits, but only started in the past two weeks and are therefore also not used for prediction. The blue shoppers have transactions in the last three weeks and started their visits in time, they are used for prediction.

The first is measuring the root mean square error **RMSE** between the predicted turnover of a cluster ($\hat{\tilde{y}}(g)$) and the actual turnover of the cluster ($\tilde{y}(g)$), taken over all clusters in G^P_t. This says something about the accuracy of the prediction. The second measure is similar, but now comparing the predicted value of a shopper ($\tilde{y}(c,t)$) and the ground truth of the shopper ($y(c,t+1)$), taken over all shoppers in $\mathcal{C}^P_t$. This says something about the accuracy and usefulness of the prediction. We distinguish between these two metrics as cluster-**RMSE** and shopper-**RMSE** respectively. The third use is a downstream prediction task, predicting which shoppers have the highest loss in spend from one week to the next. We compare the spend of this week ($y(c,t)$) with the spend of last week ($y(c,t-1)$) for all shoppers, and label the decile with the highest decrease in spend with `TRUE`, and all others with `FALSE`. This can be used by the retailer to apply targeted marketing: only reaching the shoppers that are the most interesting as they decrease their spend by the largest amount. We also determine predicted labels based on the predicted spend for the subsequent week and use these with the ground truth labels to compute the $\mathbf{F_1}$ score. This metric focuses on usefulness. The fourth use is in the prediction of the total turnover of the next week, computed by summing the individual (predicted) spend values over all shoppers. This results in a single value for the ground truth (T) and a single value for the prediction $\hat{T}$, from which we compute the absolute percentage error **APE**, by dividing the difference by the ground truth ($|T-\hat{T}|/T \cdot 100\%$). This metric also focuses on usefulness.

3.10 Paint Factory

Having discussed CAPiES in general, as well as the application within the supermarket use case, we discuss how it can be applied in a different use case. This use case concerns the purchasing process of a paint factory. Each purchase comes with an invoice that is at some point in time received and at some later point in time paid. We are interested in predicting how much time there will be between receiving and paying an invoice. An overview of the purchasing process is shown in

Table 3: Metric used to asses CAPiES in the supermarket use case

Metric	Explanation	Focus
Cluster **RMSE**	Comparison of proxy prediction $\hat{\tilde{y}}(g)$ and ground truth $\tilde{y}(g)$	Accuracy
Shopper **RMSE**	Comparison of shopper prediction $\hat{y}(c,t)$ and ground truth $y(c,t)$	Both
$\mathbf{F_1}$	Ability to find shoppers with the highest turnover drop	Usefulness
APE	Ability to predict the total turnover	Usefulness

Figure 7. To this end, we collect information on every step in the process.

Streaming Setting: In this use case we deal with invoices as the entities. Whenever one of the process steps of Figure 7 occurs, and event e is saved. This event details when it occurred ($e.time$), to what invoice it belongs ($e.entity$), and what happened ($e.act$). Two activity labels are of particular interest: 'Vendor Creates Invoice' (`VCI`) and 'Record Invoice Receipt' (`RIR`) which respectively denote the creation and payment of the invoice that belongs to an order. Other than the time, entity and activity, no additional information is recorded on events ($\mathcal{F}_{event} = \emptyset$). Each invoice does contain additional information, presented in Table 4. For more details on this dataset see [43]. We update and use the prediction model at the end of every day (one time unit is a day).

Training Collection: For the set of invoices that can be used for training at step t we require that the invoice has been paid. This is noted by the occurrence of an event with activity label `RIR` for that invoice. Let $rir(c)$ be the timestamp of that event. We use the invoices for which this event occurred since $t-1$ for updating the prediction model:

$$\mathcal{C}_t^T = \{c \in \mathcal{C} | rir(c) \in [t-1, t)\}$$

This is visualized in Figure 8.

Encoding: the outcome of an invoice is defined as the time difference between the payment of the invoice (when `RIR` occurs) and the sending of the invoice (when `VCI`) occurs. Let $vci(c)$ be the time of the latter. We have that $\hat{y}(c,t) = rir(c) - vci(c)$. For the prediction model, it is important to realize that we can only use information available at the moment of prediction. That is, for the construction of prediction features of invoice c we should only use data available by $vci(c)$, even if we also have information from the period $[vci(c), rir(c))$ when we update our model. The predictor values come from two sources: the relative frequency of each activity and the invoice attributes. The latter of these are the ones described in Table 4. For the relative frequencies we only consider events that occur up to the prediction point, i.e. $\vec{f}(\mathcal{E} \downarrow_c^{[0,vci(c))})$. As such, $X(c,t) \in [0,1]^{|\mathcal{A}|} \times \mathcal{F}_{entity}$.

Clustering: Some of the features of $X(c,t)$ are categorical and some are numerical. For the clustering we apply Gower's distance [44] to deal with this, again applying k-medoids, as k-means does not work for categorical values.

Averaging: We apply one-hot-encoding to the categorical features to allow for averaging them for the proxy invoice encoding.

Updating: As we are not dealing with sequences of data, we do not require the use of models that can handle them. To further validate CAPiES we therefore use a different model, the Adaptive Hoeffding Option Tree [37]. This model is an incremental model: we can update after initial training.

Predicting: For the set of invoices that can be used for prediction at step t we require that the invoice is received. This is noted by the occurrence of an event with activity label `VCI` for that invoice:

$$\mathcal{C}_t^P = \{c \in \mathcal{C} | vci(c) \in [t-1, t)\}$$

This is visualized in Figure 9.

As a consequence of the definition of our prediction, we make exactly one prediction for each invoice, and we use each invoice exactly once to update the model. Put differently, for $t \neq t'$ we have $\mathcal{C}_t^P \cap \mathcal{C}_{t'}^P = \emptyset$ and $\mathcal{C}_t^T \cap \mathcal{C}_{t'}^T = \emptyset$. This is in contrast to the supermarket use case where shoppers are used for training and prediction at the end of every week. This makes sense from a model perspective as well: the model is trained using feature data of invoices *up to* the `VCI` activity. Adding more data after that would change the state of the (proxy) invoice with data the model is not trained

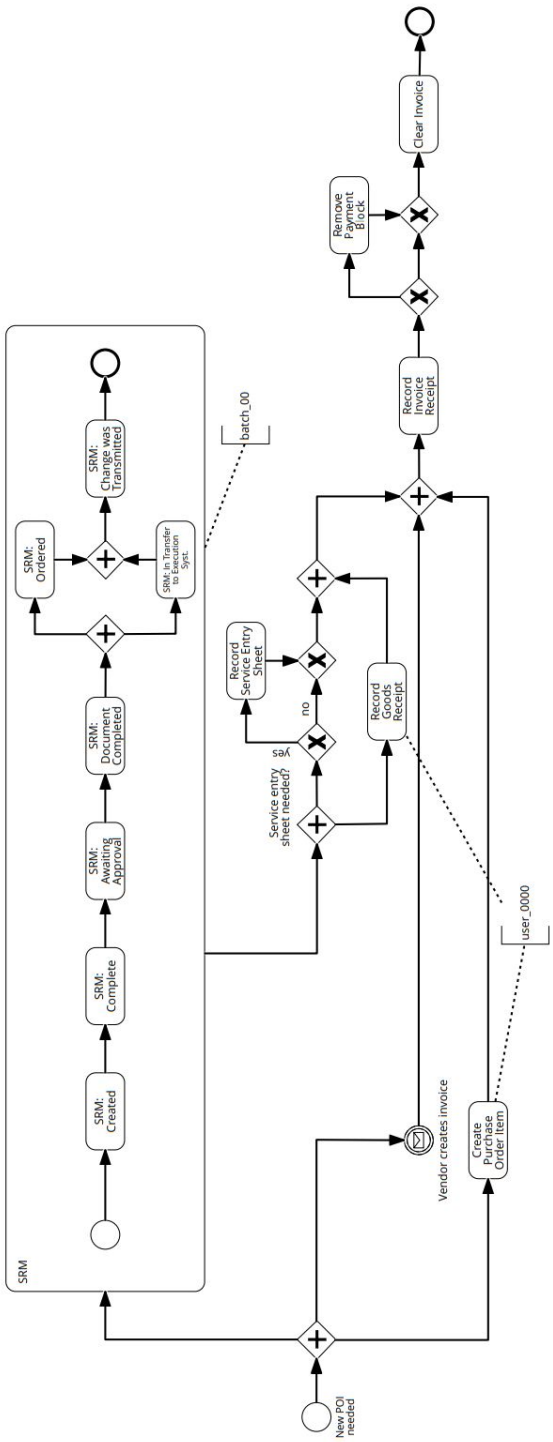

Fig. 7: Process overview as constructed by [42]

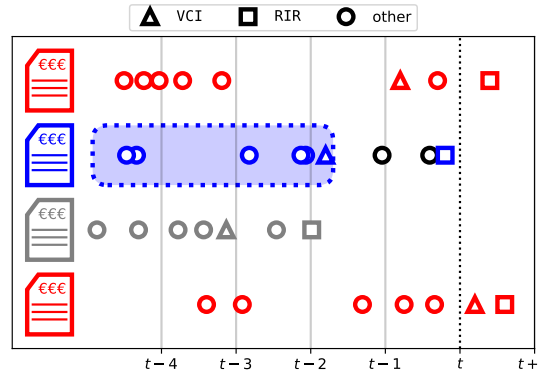

Fig. 8: Invoices selected for training. The invoices in grey have already been used for training. Invoices in red did not get `RIR`, so we do not know the duration yet and as such cannot use these invoices for training. Invoice in blue did get `RIR` since the last training time, so we can use them for training.

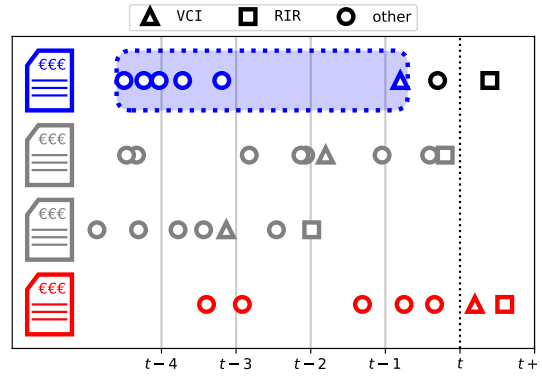

Fig. 9: Invoices selected for predicting. The invoices in grey have already had their prediction. Invoices in red did not get `VCI`, so we can not make a prediction yet. Invoices in blue did get `VCI` since the last prediction time, so we can make a prediction for them.

on. This is different from the supermarket use case where shoppers shop continuously instead of a process that progresses and eventually ends such as for the paint factory. For consistency, we do still use the notations for X and y that involve t.

As metric for evaluation we use the **RMSE** between the actual $(y(c,t))$ and predicted $(\hat{y}(c,t))$ duration of paying the invoice, in other words, the **RMSE** on the entities.

Table 4: Invoice features used in the paint factory use case.

Feature	Explanation	Values
Company	Vendor from which the product is ordered	Categorical (3)
Document Type	Type of order	Categorical (3)
GR-Based Inv. Verif.	Flag regarding requirements of the process	Boolean
Goods Receipt	Flag regarding requirements of the process	Boolean
Item Category	Further purchase/invoice regulations	Categorical (3)
Item Type	Type of product purchased	Categorical (5)
Spend area text	Department for which the order is created	Categorical (21)
Spend classification text	Additional information to Spend area text	Categorical (4)

Table 5: Summary of the use cases to which CAPiES is applied.

Definition	Supermarket	Paint Factory
Event	Visit	Process Step
Entity	Shopper	Invoice
Time unit	1 week	1 day
Model Input	Shopper journey	Process execution
Entity outcome	Weekly spend	Payment duration

3.11 Discussion

As discussed above, CAPiES can be used for different use cases. The two discussed are summarized in Table 5. In this section we review CAPiES, discussing important aspects in the design.

As mentioned, CAPiES is about the use of clustering to improve the effectiveness of prediction models. As such, the clustering method and the prediction model used in CAPiES are not a design choice, but depend on the use case CAPiES is applied to. In principle, any method that partitions a set of entities in a specified number of subsets can be used for this step. In the above we have discussed the use of the approximate Euclidean distance from [32] and the use of Gower's distance from [44] together with k-medoids. In the experimental evaluation we also compare using k-medoids with randomly assigning entities to clusters. The same holds for the prediction model: in principle, any learner that can be updated after initial training can be used in CAPiES. In the above we have discussed the use of an LSTM and the Adaptive Hoeffding Option Tree, but even static learners that are completely retrained at every time step are a valid model. CAPiES is about the combination of *a* clustering method and *an* incremental learning mechanism to improve the quality of the predictions. Existing literature focuses on which prediction models are used [12, 15, 38] and how entities are clustered instead [10, 33, 35]; in that sense, the contribution is orthogonal to these. This holds for prediction tasks in general, and also specific to recommendation systems and other related work on shopper behaviour prediction.

This paper and in particular the experimental evaluation discusses the effect CAPiES has on the quality and usefulness of the predictions. CAPiES may also have an influence on the computation time. On the one hand, the clustering increases the duration of the training/testing phase. On the other hand, the use of proxy entities means that the model is trained on fewer datapoints and makes predictions for fewer datapoints. These two changes in the training and prediction phase have opposing effects on the computation time. It is possible that the added time required for clustering is outweighed by the reduced time required for training/predicting. However, an analysis of this is beyond the scope of the current paper.

4 Experimental Evaluation

In this section we discuss the experimental evaluation of the method described in Section 3. We first discuss the supermarket use case from Section 3.2 in Section 4.1, followed by the results from the paint factory use case from Section 3.10 in Section 4.2. We then conclude with lessons learned from both datasets in Section 4.3. The implementation of CAPiES as well as the experiments can be found at https://anonymous.4open.science/r/ITEM-CAPiES.

4.1 Supermarket Shopper Experiments

For the experiment, we use data from a real-life supermarket. The visit data comes from 39 weeks and contains over 160000 shoppers with at least one purchase in that time. The number of visits per shopper varies between 1 and 312 with a mean of 55.0 and a standard deviation of 21.8. Because of privacy reasons, this dataset cannot be made publicly available. We discuss the experimental setup in Section 4.1.1 and discuss the results in Section 4.1.2 through 4.1.7.

4.1.1 Experimental Setup

For the experiment we vary the average number of shoppers per cluster ρ, taking values $2^1 = 2, 2^3 = 8, 2^5 = 32, \ldots, 2^{17}$ (i.e. in multiples of 4), the last of these being just smaller than the largest value of $\left|\mathcal{C}_t^P\right|$. We also add the special values of $\rho = 2^0 = 1$ and $\rho = |\mathcal{C}|$ as competitors that do not apply clustering. In the running example we used a shopper journey of three weeks to make predictions. We also vary this value, indicated by the parameter τ, exploring values $2, 3, 4, \ldots, 9$ weeks. Note that τ is a variable for the supermarket use case only; it is not part of the framework, nor does the paint factory use case use it. As such we test CAPiES on $11 \cdot 8$ combinations of ρ and τ.

For the LSTM architecture we use the one defined by [8], staying as close to that work as possible, only changing the input and output layers to match our input and output representations.

Each prediction estimates the turnover of a proxy shopper: the average turnover per shopper in the respective cluster, which is then also used as prediction for each individual shopper in that cluster. Using the metrics defined in Section 3.9 we asses the quality of CAPiES for each combination of τ and ρ. We compute the metrics at every time step, and average them over time. The results are shown in Figures 10 to 13.

The results of each metric are presented as a heatmap, where better values (lower **RMSE** and **APE**, higher $\mathbf{F_1}$) have a brighter colour, and worse values a darker one. Each column contains a single value of ρ (increasing from left to right) and each row contains a single value of τ (increasing from top to bottom).

We answer the following questions:

1. How does τ influence the **RMSE** (prediction accuracy)?
2. How does ρ influence the cluster-**RMSE** (prediction accuracy)?
3. How does ρ influence the shopper-**RMSE** (accuracy and usefulness)?
4. How does ρ influence the **APE** in $\hat{T}$ (usefulness)?
5. How does ρ influence the $\mathbf{F_1}$ on the top decile (usefulness)?
6. How does ρ influence the $\mathbf{F_1}$ *over time* (usefulness)?

In each of Sections 4.1.2 through 4.1.7 we answer each of the above questions.

4.1.2 The effect of τ on RMSE (prediction accuracy)

We first analyze the effects of τ on the cluster- and shopper-**RMSE** in Figures 10 and 11 respectively, as this will be relevant during the discussions on ρ. We distinguish between $\rho < 2^{15}$ and $\rho \geq 2^{15}$, where $\rho = |\mathcal{C}|$ is considered part of the latter.

$\rho < \mathbf{2^{15}}$: For smaller values of ρ, all experiments show a clear decrease of **RMSE** with increasing τ for both **RMSE** results. This is expected, as a higher τ means that the data sequences for each cluster are longer and the LSTM can learn from a longer period of time, which benefits its performance [45].

$\rho \geq \mathbf{2^{15}}$: For higher ρ, the **RMSE** is not as consistently decreasing with increasing τ. The reason for this is that for a higher value of ρ we have fewer clusters of shoppers 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 worse predictions than shorter sequences (lower τ).

4.1.3 The effect of ρ on cluster-RMSE (prediction accuracy)

We next analyze the influence of ρ (left to right) on the cluster-**RMSE**, presented in Figure 10. Up to $\rho = 2^{15}$ we see a clear patterns: increasing ρ decreases the clusters-**RMSE**. This makes sense, as we increase the number of shoppers in each cluster. Because we are comparing the prediction of the average shopper with the average

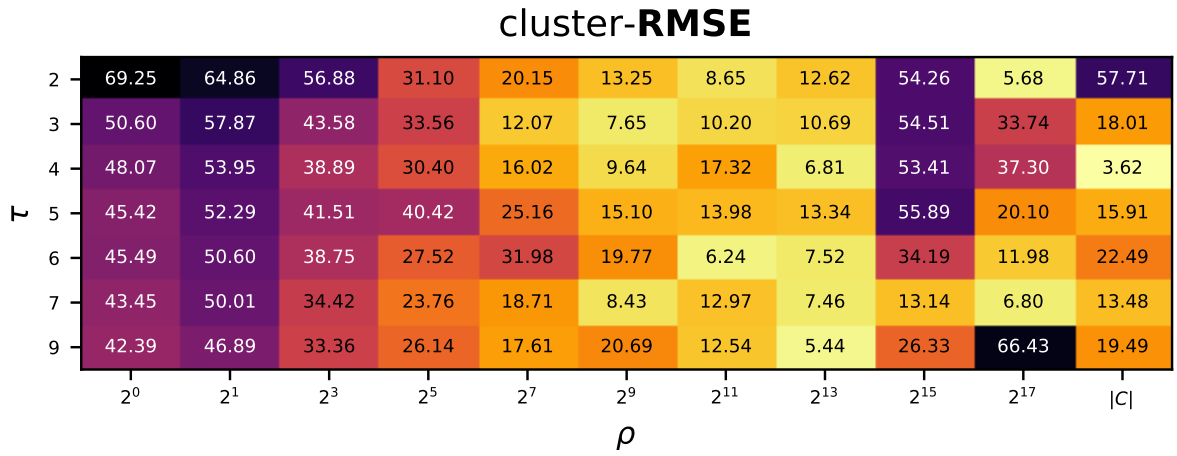

Fig. 10: Results for the cluster-**RMSE**. Lighter colours indicate a better (lower) value, the values are in monetary units.

Fig. 11: Results for the shopper-**RMSE**. Lighter colours indicate a better (lower) value, the values are in monetary units.

ground truth, we make a more accurate prediction for the shoppers *as a group*. Put differently, some shoppers will have a lower ground truth than the predicted cluster average, others will have a higher ground truth than the cluster average. These two effects cancel each other to get a lower cluster-**RMSE**, provided that the model can make a decent prediction in the first place. As argued in the analysis of τ, this does not seem to apply for $\rho \geq 2^{15}$, where the above pattern is not guaranteed.

4.1.4 The effect of ρ on the shopper-RMSE

Contrary to the results for the cluster-**RMSE**, Figure 11 shows that lower values of ρ tend to produce better results for the shopper-**RMSE**. Increasing ρ from $2^0 = 1$ (no clustering) to $2^1 = 2$, we first see a small improvement when combining an average of two shoppers into a cluster, followed by a gradual increase of the shopper-**RMSE** as ρ increases. This different result is caused by two factors. The first is that a lower value of ρ

means that fewer shoppers are together in the cluster, which means that the individual shoppers are likely closer to each other and as such to the average shopper. This means the average prediction can be a better indicator for the individual shoppers, also resulting in an average prediction that is closer to the individual ground truths. The opposite happens for larger ρ where individual shoppers have more distance to the average shopper, and so are the individual ground truths to the predicted average. The second factor is that, as discussed previously, the cluster-**RMSE** benefits from the averaging effect, which allows over- and under-predictions to cancel each other. For the shopper-**RMSE**, over- and under-predictions are both penalized without cancelling each other, increasing the **RMSE**.

4.1.5 The effect of ρ on APE

As depicted in Figure 12, we see a similar relation between ρ and the **APE** on the turnover. What is notable though, is that for medium high values of ρ (between 2^5 and 2^{13}), some results are closer to the low values ($\leq 2^5$), indicated by the lighter colours. This is in contrast to the results of the shopper-**RMSE**, where the results for these ρ values are much worse (indicated by the darker colours for that range). Similar to the better results on the cluster-**RMSE**, predicting the total turnover allows under- and over-predictions to cancel. This allows some of the mid-range ρ to still reasonably predict the total turnover, despite the worse shopper-**RMSE**.

4.1.6 The effect of ρ on $\mathbf{F_1}$

We next discuss the effect of ρ on the $\mathbf{F_1}$ metric when predicting the top decile of shoppers that lower their spend from one week to the next. Similar to the results for the shopper-**RMSE** in Section 4.1.4, the quality of the results decreases (lower $\mathbf{F_1}$) with increasing ρ. While this is partly to be expected given the increase in shopper-**RMSE** for higher ρ, this metric punishes for incorrect under/over predictions. A shopper with the ground truth of 10 monetary units lower that the predicted value will contribute in the same way to the shopper-**RMSE** as a shopper with 10 units under-prediction. This is not true for the metric presented in Figure 13, it captures whether the error is, to a certain extend, consistent between shoppers as well, so over-predicting all shoppers by the same amount is less penalizing. As discussed, being able to predict which shoppers will decrease their spend most from one week to the next is a measure of the usefulness. We see that being closer to the individual prediction (lower ρ) is better, though there is a small advantage of clustering with two shoppers per cluster.

4.1.7 The effect of ρ on $\mathbf{F_1}$ over time

The results above have shown the effect of ρ on prediction accuracy and usefulness as average over all time steps. We now discuss some of the effects that can be seen as CAPiES progresses over time, presented in Figure 14. The numbers in the $\tau = 9$ row of Figure 13 are the averages of these plots. From the picture we see again that lower values of ρ prevail, but that the progression of $\mathbf{F_1}$ over time is similar. Note that each value of ρ, $\mathbf{F_1}$ of the first time step is rather low. This is because the first timestep has a cold-start: the prediction model needs data over more time to reach a steady level, with the higher values of ρ settling about 0.1 lower than the lower values of ρ. Next to this, we see the effect of external events at three points in time, indicated by vertical dotted lines. While we cannot disclose the details of the timeline, the third of these is very likely to temporarily change shopper behaviour. The other lines describe similar though less significant external effects. Because of these points of sudden changes in the shopper behaviour one expects a decrease in the predictive accuracy in the time around them. This results in a sudden drop in $\mathbf{F_1}$, especially for the first and third point in time, but these recover quickly. Of special interest is the ability of the model to handle concept drift does not seem to be influenced by the value of ρ. In other words, any effect concept drift is shown to have on the prediction without CAPiES ($\rho = 2^0$) is not worsened or improved by CAPiES. The progression of $\mathbf{F_1}$ over time is very similar over all values of ρ (in that major deviations from the stable $\mathbf{F_1}$ occur at the same time). This is partly due to the incremental nature of the LSTM model. At the same time this is not trivial, as a higher ρ means fewer datapoints to train on, and as such fewer datapoints to correct for concept drift. A similar argument can be made for the variety in seasons; the half a year of data does not show seasonal effects on the $\mathbf{F_1}$.

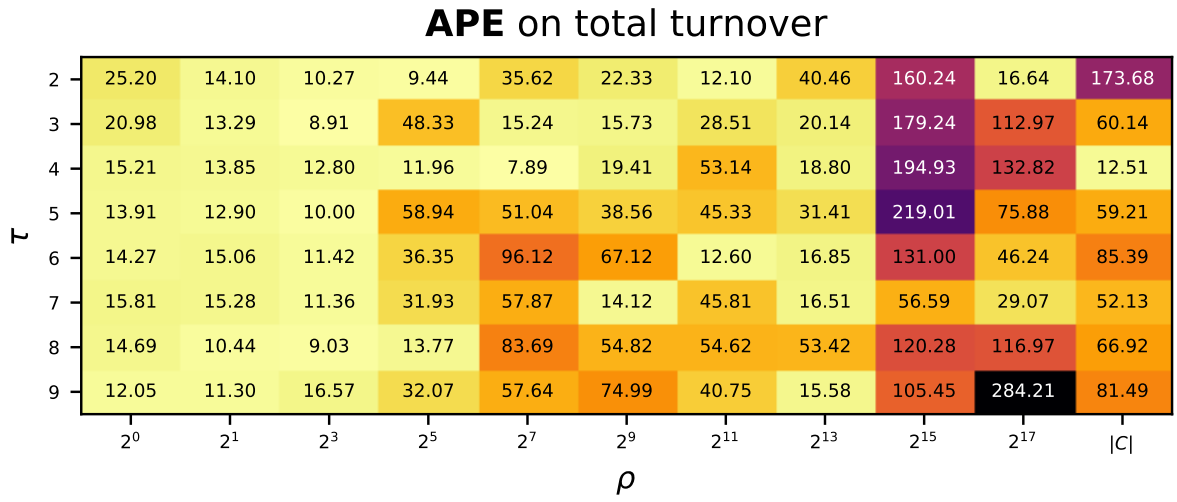

Fig. 12: Results for the turnover **APE**. Lighter colours indicate a better (lower) value, the values are error percentages with respect to the actual turnover.

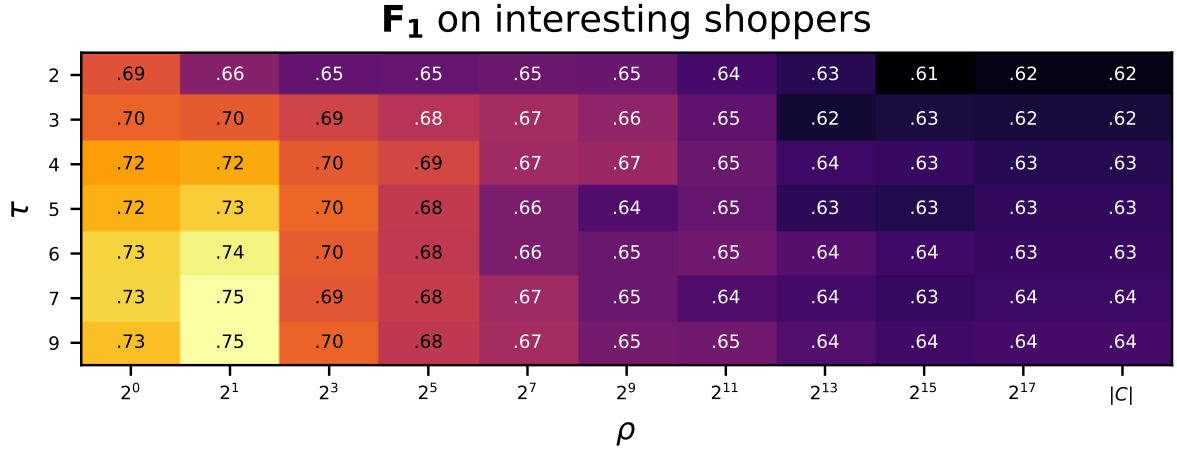

Fig. 13: Results for the $\mathbf{F_1}$ on the shopper decile with the highest drop in turnover. Lighter colours indicate a better (higher) value.

4.2 Paint Factory Experiments

For the second use case, we use the BPIC 2019 dataset [43] as depicted in Figure 7, for which the invoice features are summarized in Table 4. We first filter the dataset keeping only invoices with exactly one occurrence of 'Vendor creates invoice' and 'Record invoice receipt', in the correct order. We also remove cases that started before 2018 or ended after 2018. Furthermore, we only keep the invoice attributes described in Table 4. After this filtering, we have 171517 invoices containing a total of 1025949 events with $|\mathcal{A}| = 40$ different labels. We discuss the experimental setup in Section 4.2.1 and present and discuss the results in Sections 4.2.2 to 4.2.4.

4.2.1 Experimental Setup

In this experiment we compare the error, expressed by **RMSE**, between the predicted invoice duration $\hat{y}(c,t)$ and the actual invoice

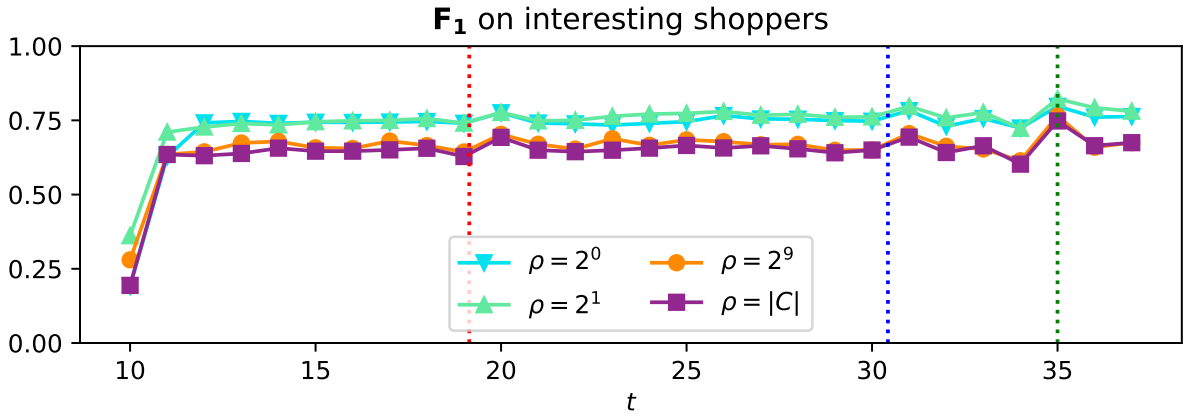

Fig. 14: Progression of $\mathbf{F_1}$ over time for $\tau = 9$. The vertical lines indicate significant external influences that can influence shopper behaviour.

duration $y(c,t)$, against the value of ρ. For the experiment we vary ρ, increasing from 1 to 178 in steps of 1, the final value being just below the maximum value of $\left|\mathcal{C}_t^P\right|$, and $\rho = |\mathcal{C}|$. We therefore have a total of 179 experiments. This higher resolution for ρ (compared to Section 4.1) is possible because the dataset has fewer events and each entity is only used once for prediction and training. Next to this, we repeat all experiments, but assign cases to *random* clusters. This is done to compare the effect 'averaging' in general with carefully selected clusters of similar entities. For the prediction model we use the Adaptive Hoeffding Option Tree, as implemented in [38]. This is different from the LSTM used for the shopper dataset, as we want to show that CAPiES works for different prediction models, as those are only a choice within CAPiES.

4.2.2 Results

The results for the BPIC'19 log are presented in Figure 15. The plot shows two colours: the blue indicates the results for clustering with Gower's distance and k-medoids, and in orange the results for randomly assigning the entities to one of k clusters. In both, k is based on ρ and $\left|\mathcal{C}_t^P\right|$ or $\left|\mathcal{C}_t^T\right|$ as per Equation (1). The blue square and orange upward triangle indicate the special value $\rho = 1$ (no clustering). The blue circle and orange downward triangle indicate the special value $\rho = |\mathcal{C}|$ (all entities in a single cluster).

4.2.3 Influence of ρ on RMSE

Looking at the blue line in Figure 15, we distinguish three sections for ρ: low values from 2 to about 50 (indicated by `I`), mid values from 50 to 100 (`II`) and high values above 100 (`III`). For low values we see an improvement of CAPiES over single invoice predictions (the blue square). This is in line with the results for the shopper dataset, where we find that a limited average cluster size has a positive influence on the prediction accuracy. For the mid range, we see a steady increase in the **RMSE**, also confirming our findings from Section 4.1. In this range the invoices in the clusters deviate too much from the cluster average, resulting in worse invoice-**RMSE** values. At some point, the number of clusters rarely exceeds 1 as per Equation (1), leading to results close to $\rho = |\mathcal{C}|$, decreasing performance further as we reach a limited number of training points in each time step.

4.2.4 The effect of clustering versus random clusters

Given CAPiES, one can wonder what the contribution is of creating clusters of *similar* entities, and to what extend *any* averaging plays a role in the results of CAPiES. To look into this, we compare two sets of experiments: applying the k-medoids clustering with Gower's distance against assigning entities to groups at random. The latter of these methods creates k clusters, and for each entity selects one of the clusters at random

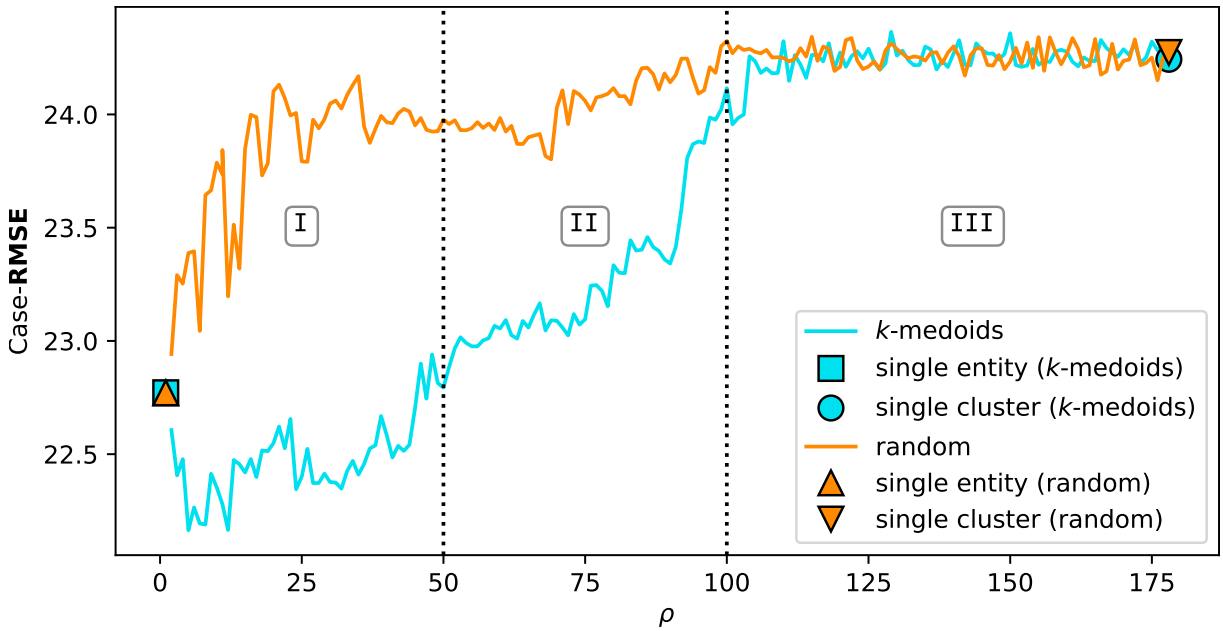

Fig. 15: Results for the BPIC'19 event log. The blue line/symbols indicate the results for assigning entities to clusters using k-medoids and Gower's distance. The orange line/symbols indicate the results for randomly assigned groups.

to be assigned to. This creates the results presented in orange in Figure 15. For low values of ρ (up to 100) we see large differences between the clusters created with k-medoids and the cluster created at random, while for high values (from 100) this difference vanishes. Similar to the discussion in Section 4.1.4, having fewer invoices in a cluster means that the invoices will be closer to the average invoice, but only if clusters are defined as such. For random clusters, the invoices may just as well be close to each other (and to the average) as far away from each other (and from the average), with most clusters expected to be somewhere in between. As we increase the number of invoices in a cluster (higher ρ) we also increase the distance from the invoices to the average invoice.

4.3 Considerations for ρ

In this section we look back at the results from the two rather different datasets and discuss conclusions that hold for both datasets. Based on the results there are three considerations for the optimal value of ρ: *Model stability*, *Cluster stability*, and *Cluster detail*. For Model stability, we require enough clusters to be found to create sufficient training data for a stable model. This effect is explicitly visible for $\rho > 2^{13}$ in the shopper dataset. For the Cluster stability, we find that increasing ρ improves the prediction accuracy on clusters, i.e. how well are we able to predict the behaviour of the entities in a cluster *as a group*. This is visible in Figure 10. Finally, for Cluster detail we do not want ρ to be too large, as we make our predictions more tailored to the individual, an effect clearly visible in Figures 11 through 15. These are three considerations that should be balanced and taken into account when deciding the best value of ρ for a given use case and dataset.

In the original version of this work, the parameter of CAPiES was the number of clusters k. In other words, the number of clusters that we find at each time step is predefined as a parameter and does not depend on $|\mathcal{C}_t^P|$ and $|\mathcal{C}_t^T|$ as per Equation (1). While this works fine for datasets where the number of entities is roughly similar over time (such as for the supermarket dataset), it poses difficulties when such a condition does not hold (such as for the paint factory dataset). The introduction of ρ solved this issue. As a result, CAPiES presented in this paper, next to being more generic, delivers a more stable performance for the paint factory dataset. The results on the

supermarket dataset are similar (except that an increase in ρ is a decrease in k and vice-versa).

An interesting question for future work is to see how these conclusions change for different datasets, especially with the number of entities in the datasets. The introduction of ρ in this paper to replace k makes reasoning about this easier. If we increase the size of the dataset, we also increase $\left|\mathcal{C}_t^P\right|$ and $\left|\mathcal{C}_t^T\right|$. This has two effects on the clustering. First, we increase the number of clusters we find at each time step (as per Equation (1)), and second, the entities are likely to be more similar. The first effect is expected to benefit Model stability, removing the negative effects that high ρ experience. For the second effect, consider the supermarket dataset. If we add transaction data from another 160000 shoppers, for the same value of ρ, each of the 160000 shoppers in the original dataset will be as close or closer to the shoppers in the cluster they are assigned. As a result, the average shopper in each cluster will be a better representative. For the same value of ρ, this will improve the Cluster stability while maintaining the same level of Cluster detail, resulting in an overall performance increase.

5 Conclusion and Future Work

In this paper we proposed CAPiES to make predictions about future events of entities by combining the training and prediction data of similar entities. This aims to overcome the problem that single entities may show very different behaviour, making accurate predictions unstable and inaccurate. Combining entities is done using clustering, where the number of clusters is determined by the desired average number of entities in a cluster ρ, a parameter of CAPiES. ρ introduces a trade-off: a higher ρ results in fewer training points and as such poorer trained model, but because we have fewer and as such larger clusters, the behaviour of entities as a group is better averaged, increasing prediction accuracy. Next to this, a lower ρ also means smaller clusters, and as such the predictions are closer to the individual entities. We applied CAPiES to two use cases: shopper behaviour in supermarkets and invoice throughput prediction in the ordering process of a paint factory. In the supermarket use case we showed the trade-off. Making small groups of shoppers increases the quality of down-stream prediction tasks over using individual shoppers, but moving to larger groups decreases overall prediction quality. In the paint factory ordering process, we see a similar trend. Clustering invoices in small groups improves the prediction performance initially, but the prediction quality deteriorates for larger clusters.

For future research, several directions can be identified. The most important one is to see how the size of the dataset affects the considerations for the optimal number of entities per cluster, as discussed in Section 4.3. This includes applying CAPiES to datasets from different retailers, longer periods of time or more shoppers, but also the application in different use cases. Apart from different datasets and use cases, the effect of CAPiES on other prediction tasks within the supermarket use case can be looked into as well. One such example is basket analysis. Where this paper focuses on the spend of shoppers as outcome (and the derived interesting-shopper outcome of Sections 4.1.6 and 4.1.7), other work looks into which products are likely to be bought by a shopper [21] or when a shopper next makes a visit [20]. CAPiES is not limited to spend as outcome per se, but an interesting future research direction is translating the basket content prediction to be used with CAPiES. Another direction to consider is using event-oriented clustering methods like the ones explored in the field of process mining. The applications of CAPiES in this paper do not make use of the individual events for clustering, but on the encoding learned from entities extracted from events. Alternatives are existing methods from the process mining field such as [10, 30] that apply their clustering on the sequences of events. In CAPiES, the found clusters are used as a *means to an end*. Put differently, the found clusters are only used to improve the predictions, but are not analyzed beyond this. Further analysis on these clusters can show potential improvement points for CAPiES. For example, is there a relation between the density of the cluster and the error in the prediction for the proxy-entity? In other words, if a cluster is made of entities that are not as similar as entities in a different cluster, does that influence the quality of the prediction on entities in that cluster? This can also lead to an additional consideration for the value of ρ: to what

extend does the quality of the data and the presence of homogeneous clusters in the data affect the effectiveness of CAPiES, and can knowledge on that be used to further improve CAPiES or estimate the optimal value for ρ? Next to this, the analysis of how clusters progress over time can be an interesting result as well, particularly in the supermarket use case. For example, do shoppers change between clusters from one week to another, and to what extend do external effects (concept drift, discussed in Section 4.1.7) influence this? Finally, at each time step the clusters are recomputed based on the most recent data of the entities. An extension lies in incorporating past information on clusters, such that longer-term similarities in entity behaviour can also be considered. However, this is specific to use cases like the supermarket, where entities are used for training and prediction on multiple time steps, unlike the paint factory.

References

[1] Chamberlain, B.P., Liu, B., Pagliari, R., Deisenroth, M.P.: Customer Lifetime Value Prediction Using Embeddings. KDD 2017 ADS **17** (2017). https://doi.org/10.1145/3097983.3098123

[2] Ishigaki, T., Terui, N., Sato, T., Allenby, G.M.: Personalized market response analysis for a wide variety of products from sparse transaction data. International Journal of Data Science and Analytics **5**(4), 233–248 (2018). https://doi.org/10.1007/S41060-018-0099-9

[3] Mishra, S., Leroy, V., Amer-Yahia, S.: Colloquial region discovery for retail products: discovery and application. International Journal of Data Science and Analytics **4**(1), 17–34 (2017). https://doi.org/10.1007/S41060-017-0048-Z

[4] De Cnudde, S., Martens, D., Evgeniou, T., Provost, F.: A benchmarking study of classification techniques for behavioral data. International Journal of Data Science and Analytics **9**(2), 131–173 (2020). https://doi.org/10.1007/S41060-019-00185-1

[5] Francescomarino, C.D., Dumas, M., Maggi, F.M., Teinemaa, I.: Clustering-Based Predictive Process Monitoring. IEEE Transactions on Services Computing **12**(6), 896–909 (2019). https://doi.org/10.1109/TSC.2016.2645153

[6] Francescomarino, C.D., Ghidini, C., Maggi, F.M., Rizzi, W., Persia, C.D., Apr, A.I.: Incremental Predictive Process Monitoring: How to Deal with the Variability of Real Environments. Technical report (2018). https://arxiv.org/abs/1804.03967

[7] 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 (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **4928 LNCS**, 17–29 (2007). https://doi.org/10.1007/978-3-540-78238-4_4

[8] Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive Business Process Monitoring with LSTM Neural Networks. In: Dubois, E., Pohl, K. (eds.) Advanced Information Systems Engineering, pp. 477–492. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59536-8_30

[9] Taymouri, F., La Rosa, M., Erfani, S.M.: A Deep Adversarial Model for Suffix and Remaining Time Prediction of Event Sequences. Proceedings SDM 2021, 522–530 (2021). https://doi.org/10.1137/1.9781611976700.59

[10] Leontjeva, A., Conforti, R., Francescomarino, C.D., Dumas, M., Maggi, F.M.: Complex Symbolic Sequence Encodings for Predictive Monitoring of Business Processes. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **9253**, 297–313 (2016). https://doi.org/10.1007/978-3-319-23063-4_21

[11] Teinemaa, I., Dumas, M., Maggi, F.M., Francescomarino, C.D.: Predictive Business Process Monitoring with Structured

and Unstructured Data. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **9850 LNCS**, 401–417 (2016). https://doi.org/10.1007/978-3-319-45348-4_23

[12] 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). https://doi.org/10.1145/3301300

[13] Hassani, M., Habets, S.: Predicting next touch point in a customer journey: A use case in telecommunication. Proceedings - European Council for Modelling and Simulation, ECMS **35**(1), 48–54 (2021). https://doi.org/10.7148/2021-0048

[14] Günesen, S.N., Şen, N., Yıldırım, N., Kaya, T.: Customer Churn Prediction in FMCG Sector Using Machine Learning Applications. In: Mercier-Laurent, E., Kayalica, M.Ö., Owoc, M.L. (eds.) Artificial Intelligence for Knowledge Management, pp. 82–103. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-80847-1_6

[15] 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). https://doi.org/10.1108/JM2-01-2021-0032

[16] Spenrath, Y., Hassani, M., van Dongen, B., Tariq, H.: Why Did My Consumer Shop? Learning an Efficient Distance Metric for Retailer Transaction Data. In: Proceedings of ECML PKDD 2020 Lecture Notes in Computer Science Springer, pp. 323–338 (2021). https://doi.org/10.1007/978-3-030-67670-4_20

[17] Burattin, A., Sperduti, A., van der Aalst, W.M.P.: Control-flow discovery from event streams. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 2420–2427 (2014). https://doi.org/10.1109/CEC.2014.6900341

[18] Spenrath, Y., Hassani, M.: Ensemble-Based Prediction of Business Process Bottlenecks With Recurrent Concept Drifts. Workshop proceedings of the EDBT/ICDT 2019 Joint Conference (2019)

[19] Bose, R.P.J.C., van der Aalst, W.M.P., Žliobaitė, I., Pechenizkiy, M.: Dealing With Concept Drifts in Process Mining. IEEE Transactions on Neural Networks and Learning Systems **25**(1), 154–171 (2014). https://doi.org/10.1109/TNNLS.2013.2278313

[20] Chen, T., Keng, B., Moreno, J.: Multivariate Arrival Times with Recurrent Neural Networks for Personalized Demand Forecasting. In: 2018 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 810–819 (2018). https://doi.org/10.1109/ICDMW.2018.00121

[21] Doan, T., Veira, N., Keng, B.: Generating Realistic Sequences of Customer-level Transactions for Retail Datasets. IEEE ICDM Workshop on Data Mining for Services 2018 (2019). https://doi.org/10.48550/arXiv.1901.05577

[22] Guidotti, R., Nanni, M., Giannotti, F., Pedreschi, D., Bertoli, S., Speciale, B., Rapoport, H.: Measuring Immigrants Adoption of Natives Shopping Consumption with Machine Learning. In: Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) vol. 12461 LNAI, pp. 369–385 (2021). https://doi.org/10.1007/978-3-030-67670-4_23

[23] van der Aalst, W.M.P.: Process Mining, 2nd edn. Springer, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49851-4

[24] Ariannezhad, M., Jullien, S., Nauts, P., Fang, M., Schelter, S., de Rijke, M.: Understanding Multi-Channel Customer Behavior in Retail. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 2867–2871. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3459637.3482208

[25] Bai, G.-J., Lien, C.-Y., Chen, H.-H.: Co-Learning Multiple Browsing Tendencies of a User by Matrix Factorization-Based Multi-task Learning. In: IEEE/WIC/ACM International Conference on Web Intelligence. WI '19, pp. 253–257. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3350546.3352526

[26] Unnikrishnan, V., Beyer, C., Matuszyk, P., Niemann, U., Pryss, R., Schlee, W., Ntoutsi, E., Spiliopoulou, M.: Entity-level stream classification: exploiting entity similarity to label the future observations referring to an entity. International Journal of Data Science and Analytics **9**(1), 1–15 (2020). https://doi.org/10.1007/S41060-019-00177-1

[27] Ferreira, D.R., Gillblad, D.: Discovering Process Models from Unlabelled Event Logs. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) Business Process Management, pp. 143–158. Springer, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03848-8_11

[28] Bayomie, D., Helal, I.M.A., Awad, A., Ezat, E., ElBastawissi, A.: Deducing Case IDs for Unlabeled Event Logs. Lecture Notes in Business Information Processing **256**, 242–254 (2016). https://doi.org/10.1007/978-3-319-42887-1_20

[29] 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). https://doi.org/10.1007/978-3-642-00328-8_11

[30] Jagadeesh Chandra Bose, R.P.: Process mining in the large. PhD thesis, Mathematics and Computer Science (2012). https://doi.org/10.6100/IR730954

[31] Lloyd, S.P.: Least Squares Quantization in PCM. IEEE Transactions on Information Theory **28**(2), 129–137 (1982). https://doi.org/10.1109/TIT.1982.1056489

[32] Spenrath, Y., Hassani, M., Van Dongen, B.F.: BitBooster: Effective Approximation of Distance Metrics via Binary Operations. In: Proceedings of COMPSAC 2022 (2022)

[33] Van den Berg, S., Hassani, M.: On Inferring a Meaningful Similarity Metric for Customer Behaviour. In: Proceedings of ECML PKDD 2021 Lecture Notes in Computer Science Springer, pp. 234–250 (2021). https://doi.org/10.1007/978-3-030-86517-7_15

[34] Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. KDD'96, pp. 226–231 (1996). https://dl.acm.org/doi/10.5555/3001460.3001507

[35] Lin, W.-C., Tsai, C.-F., Hu, Y.-H., Jhang, J.-S.: Clustering-based undersampling in class-imbalanced data. Information Sciences **409-410**, 17–26 (2017). https://doi.org/10.1016/j.ins.2017.05.008

[36] Baena-García, M., Campo-Ávila, J., Fidalgo-Merino, R., Bifet, A., Gavaldà, R., Morales-Bueno, R.: Early Drift Detection Method, pp. 77–86 (2006). https://www.cs.upc.edu/%7Eabifet/EDDM.pdf

[37] Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., Gavaldà, R.: New Ensemble Methods for Evolving Data Streams. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '09, pp. 139–148. ACM, New York, NY, USA (2009). https://doi.org/10.1145/1557019.1557041

[38] Montiel, J., Read, J., Bifet, A., Abdessalem, T.: Scikit-Multiflow: A Multi-output Streaming Framework. Journal of Machine Learning Research **19**(72), 1–5 (2018). https://doi.org/10.5555/3291125.3309634

[39] Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: massive online analysis, vol. 11 (2010). https://dl.acm.org/doi/10.5555/1756006.1859903

[40] Spenrath, Y., Hassani, M.: Predicting Business Process Bottlenecks In Online Events

Streams Under Concept Drifts. In: Steglich, M., Muller, C., Neumann, G., Walther, M. (eds.) Proceedings of European Council for Modelling and Simulation (ECMS) 2020. Proceedings European Council for Modelling and Simulation, pp. 190–196 (2020). https://doi.org/10.7148/2020-0190

[41] Li, Z., Xiong, Y., Huang, W.: Drift-detection Based Incremental Ensemble for Reacting to Different Kinds of Concept Drift (2019). https://doi.org/10.1109/BIGCOM.2019.00025

[42] Kiarash Diba, S.R., Pufahl, L.: BPI Challenge 2019: Performance and Compliance Analysis of Procurement Processes Using Process Mining. In: BPI Challenge (2019). https://icpmconference.org/2019/wp-content/uploads/sites/6/2019/07/BPI-Challenge-Submission-6.pdf

[43] Van Dongen, B.: BPI Challenge 2019. 4TU.ResearchData (2019). https://doi.org/10.4121/uuid:d06aff4b-79f0-45e6-8ec8-e19730c248f1

[44] Gower, J.C.: A General Coefficient of Similarity and Some of Its Properties. Biometrics **27**(4), 857–871 (1971). https://doi.org/10.2307/2528823

[45] 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). https://doi.org/10.48550/arXiv.1606.06905