UHASSELT

KNOWLEDGE IN ACTION

Faculteit Bedrijfseconomische Wetenschappen

master handelsingenieur in de beleidsinformatica

Masterthesis

The effect of different time encodings on behavioral prediction

Jessica Van Suetendael

Scriptie ingediend tot het behalen van de graad van master handelsingenieur in de beleidsinformatica

PROMOTOR :

Prof. dr. Benoit DEPAIRE

UHASSELT

KNOWLEDGE IN ACTION

www.uhasselt.be
Universiteit Hasselt
Campus Hasselt:
Martelarenlaan 42 | 3500 Hasselt
Campus Diepenbeek:
Agoralaan Gebouw D | 3590 Diepenbeek

2022
2023

UHASSELT

KNOWLEDGE IN ACTION

Faculteit Bedrijfseconomische Wetenschappen

master handelsingenieur in de beleidsinformatica

Masterthesis

The effect of different time encodings on behavioral prediction

Jessica Van Suetendael

Scriptie ingediend tot het behalen van de graad van master handelsingenieur in de beleidsinformatica

PROMOTOR :

Prof. dr. Benoit DEPAIRE

The effect of different time encodings on behavioral prediction

Jessica Van Suetendael

University Hasselt

Abstract. Behavior can be described as an action or response from an entity, which occurs in a particular context and at a specific moment in time. This moment in time, or the time dimension, can contain valuable information, especially for prediction models that engineer their own features, such as neural networks. But how do we use the time dimension for behavioral prediction? Furthermore, does encoding the time dimension in different ways, bear different accuracy results? To answer those two questions, this research provides a first overview of possible ways to encode time. Furthermore, the first steps are taken in discovering the effect different time encodings have on behavioral prediction. More specifically, this paper aims to discover if different kinds of time encodings can influence behavioral prediction accuracy. To discover this, six different time encodings were applied to a 100-times bootstrapped transactional dataset. The dataset represents customers' purchasing behavior of an online retail shop in the United Kingdom. To make the predictions, a Bidirectional Long-Short-Term Model (LSTM) Recurrent Neural Network (RNN) was used in combination with grid search and cross-validation to find the optimal models for each time encoding and bootstrap sample combination. The accuracy scores were statistically compared utilizing two non-parametric tests, namely a Friedman and paired Wilcoxon test. Both tests showed no significant difference in accuracy scores for the different time encodings. Furthermore, after further analysis it appeared that the information from the time dimension was not used by the prediction model. The time dimension did not influence the accuracy of behavioral prediction.

Keywords: Behavior · Prediction · Time encoding · LSTM · Transactional data

1 Introduction

Behavior refers to an action or a response triggered by certain stimuli or a specific situation (Cao, 2010). It can be conducted by humans (Yassine, Singh, & Alamri, 2017), animals (Giese, Melzheimer, Bockmühl, Wasiolka, & Rast, 2021), or entities (Cao, 2010). Over the years, numerous studies have focused on examining behavior and its applications, including behavioral modeling (Stavinova, Bochenina, & Chunaev, 2021), behavior analysis (Jain, Sinha, Gupta, Sheoran, & Khosla, 2018), pattern mining (Yassine et al., 2017), and behavior prediction (Tax, Verenich, La Rosa, & Dumas, 2017).

With the advancements in understanding and analyzing behavior, researchers have been able to accurately predict future behavior, which has significant implications for decision-making in various domains. For instance, healthcare (Peltier et al., 2022), finance (Maree & Omlin, 2021), and marketing (Jain et al., 2018) are just some examples where accurate behavior prediction can have a significant impact. Also, in the field of Process mining, predicting future behavior has been widely researched due to its importance. This

could range from predicting the next step in a process (Tax et al., 2017) to predicting the outcome of a process (Wang, Yu, Liu, & Sun, 2019).

Behavior are actions that happen at a certain point in time. Time is, therefore, a feature of behavioral data. Nevertheless, how can this time dimension be utilized when making predictions? Do we use it as-is? Or does it need to be transformed or even discarded? Which form of time encoding has the most significant impact on the prediction accuracy of neural networks? There are various methods for expressing time in data, but to our knowledge, no overview of these encodings exists together with recommendations for which encodings are best for certain applications. Furthermore, to date, little research has explicitly explored the impact of time encodings on the accuracy of behavioral prediction both in the field of Process mining and in the field of Behavior Informatics. Different researchers have, however, used different kinds of time encodings, but little explicit attention was given to the effect different time encodings would have on the accuracy of their prediction models. Researchers may have tested different forms of time encodings, but their research rarely mentions this. Because of this, this research aims to answer the following question: "What is the effect of different time encodings on the accuracy of behavioral prediction?". Besides determining the effect of different time encodings, a first overview of different ways of encoding time will be presented in this research.

To answer our research question, a Bidirectional Long Short Term Memory (LSTM) classification model will use six different time encodings. Due to the fact that the chosen dataset is sequential in nature, LSTM was chosen as the prediction method. Furthermore, LSTM is a form of neural network that can engineer its features. Therefore, it can extract valuable information from the time dimension in a dataset. The results from each time encoding will be statistically analyzed, employing Friedman's and Wilcoxon's tests, to determine if there is a difference in prediction accuracy. The exact methodology can be found in section 4. Before the methodology, an overview of both related work and possible time encodings are given in sections 2 and 3, respectively. The results are described in detail in section 5. Finally, in sections 6 and 7, in addition to a conclusion, there will also be a discussion where the results will be discussed.

2 Related Work

2.1 Behavior and Process mining

Behavior is a response or action from a subject to an object or subject in a certain context at a particular moment in time (Cao, 2010). The subject can be a human (Yassine et al., 2017), animal (Giese et al., 2021) or entity (Cao, 2010). To scrutinize this behavior, several applications have been provided: behavior analysis (Jain et al., 2018), behavioral modeling (Stavinova et al., 2021), pattern mining (Yassine et al., 2017), and behavioral prediction (Tax et al., 2017). When the goal is to understand behavior deeply, it can be classified to the field of Behavioral Informatics. Various methods, techniques, and tools are developed in this field to better understand behavior and its hidden elements. Thereby utilizing behavior to its fullest potential (Cao, 2010). The techniques of Behavior Informatics can be used in various fields that portray behavior in some way. For instance, the field of Process mining portrays behavior executed in the form of a process by a process instance (Van Der Aalst, 2011).

Process mining aims to extract valuable insights from event logs originating from business information systems. Event logs are datasets that record a change in state utilizing

events. Those events are linked to activities, acts of behavior that triggered the change in state. Besides the link with an activity, each event also has descriptive features such as a timestamp or a resource. Event logs can be used for various applications, such as discovering the underlying process from which the event log was created, finding discrepancies in the behavior portrayed in the event log (Van Der Aalst, 2011), or even predicting future executions of the behavior based on previous behavior from the event log (Rivera Lazo & Ñanculef, 2022). Besides the more traditional uses of event logs, other approaches exist. For example, the behavior from the event log can be extracted for more profound analysis or prediction. The extraction is done by transforming the event log into a behavioral vector. Whereafter, analysis is done using the techniques introduced by Behavior Informatics. Through this extraction or transformation, the full potential of the behavior in the event log is used (Cao, 2008).

2.2 Prediction

With the advancements in the field of Behavioral informatics, researchers have been able to accurately predict future behavior based on past behavior, which has significant implications for decision-making in various domains. For instance, in healthcare, behavioral disorders of patients can be determined by means of classification based on behavioral data of the patient (Peltier et al., 2022). Alternatively, in finance, customers can be segmented based on their spending behavior that is extracted from their financial transactions (Maree & Omlin, 2021). On the other hand, Jain et al. (2018) show the importance of behavioral prediction in marketing. Consumer conversion can be determined by predicting the purchase probability of consumers based on their user experience on online platforms (Jain et al., 2018). Also, in the field of process mining, predicting future behavior has been widely researched due to its importance. This could range from predicting the next step in a process (Tax et al., 2017) to predicting the outcome of a process (Wang et al., 2019).

For the prediction of behavior, several prediction methods can be used, like clustering (Peltier et al., 2022; Maree & Omlin, 2021) and recurrent neural networks (Jain et al., 2018; Tax et al., 2017). Besides the more commonly used methods, new methods are being developed to predict behavior specifically. For instance, a Network-Based Behavioral Similarity Score was introduced to accurately cluster customers based on their spending behavior (Martens, Provost, Clark, & de Fortuny, 2016).

One prediction method that we are particularly interested in is the Long Short Term Memory (LSTM) Recurrent Neural Network (RNN). This kind of neural network is beneficial for predicting sequential data since it is able to not only remember the given information but also forget information that is no longer relevant for the task it is performing (Gers, Schraudolph, & Schmidhuber, 2003; Greff, Srivastava, Koutnık, Steunebrink, & Schmidhuber, 2017). There have been multiple types of research where LSTM RNN was used to predict based on event data as its input. For example, Jain et al. (2018) used several sequences of actions as their input data, to measure the users' experience of online platforms (Jain et al., 2018). On the other hand, Maree et al. (2021) used aggregated event data as input for an LSTM RNN to cluster customers based on their spending behavior (Maree & Omlin, 2021). Stavinova et al. (2021) used transactional data as input for the LSTM RNN to predict future spending behavior. The data was transformed so that each client is represented by a subsequence of vectors representing the client's behavior for an entire month (Stavinova et al., 2021).

3 Time encodings

Time is an important feature both in the field of Behavioral Informatics and Process mining. In Process mining, each event in an event log is recorded with either a Start Timestamp and/or a Complete Timestamp (Van Der Aalst, 2011). In Behavioral Informatics, time can also represented by a Start Timestamp and/or a Complete Timestamp, yet sometimes only the duration of time is recorded (Peltier et al., 2022). These are all examples of different time encodings. A time encoding is a way time can be represented in data (Eder, Franceschetti, & Lubas, 2021). There are various methods for expressing time in data, but to our knowledge, no overview of these encodings exists together with recommendations for which encodings are best for certain applications. There is however, research that takes the first steps in categorizing the different ways to represent time in data. Eder et al. (2021) introduce three significant ways of representing the time dimension in data models: a point in time, an interval, and relative time. In his research, he describes the general concept of each category, but a limited amount of examples are given for each category. Furthermore, no overview of possible time encodings is provided. In this section, we fill this gap by providing a first overview of possible time encodings.

3.1 Category 1: Point in time

The first category defined by Eder et al. (2021) is a point in time. A point in time is a moment in time that can only happen once. For example the 8th of November in the year 2000, happened that exact day and cannot occur anew. In contrast to 10:45:00, which is not a unique moment in time because it occurs every single day. When we add a date to it, like 8/11/2000 10:45:00, the moment in time becomes unique. Based on this description of the time encoding category, two conditions can be constructed. Firstly, the time encoding represents a moment in time. Second, the moment in time that is represented must be unique. The time encoding 10:45:00 represents a moment in time, but is not unique. While 8/11/2000 represents a moment in time and the moment in time is unique.

The date is always necessary to fulfill the second condition, the time alone on the other hand is not sufficient. However the time can be added to enrich the time encoding with more precise information. This is what we call the precision of the time encoding. The time encoding can be presented at different levels of precision as longs as the two conditions are satisfied. The time encoding can be very precise, for example 8/11/2000 10:45:05, but can also be very general like 8/11/2000 or even just the year 2000. The year 2000 still satisfies the two conditions since it is a point in time and it is unique. The year 2000 only occurred once. The precision can be even more broader, like the 21st century. However, be careful, when the time is added to for example the year 2000, the two conditions do not hold anymore. 8 am in the year 2000 is not unique since it happened 366 times that year (keep in mind 2000 was a leap year). How precise the time encoding can be made, depends on the time dimension given in the dataset. When only the date is given, you cannot be more precise than the date.

Besides changing the precision, the time encoding can also be enriched by adding derived features or properties of the time encoding. Examples are the day of the week, seasons, holidays et cetera. These derived features or properties are extra pieces of information that are not included in the time encoding, but can be derived with the help of extra information. For example to know which days are holidays, a list of holidays is needed.

3.2 Category 2: Relative time

Our second category of time encodings is called "Relative time" and is a combination of the second, an interval, and third, relative time, category defined by Eder et al. (2021). An interval in time, refers to the time in between two points in time. Eder's relative time is defined as the time of a particular event that has to be calculated based on the time of another event. The two categories were combined due to their similarity. Both categories require at least two moments in time. This is also the condition that needs to be satisfied to belong the "Relative time" category.

The first and most simple form of this category is the order of events. The first event of a particular case is represented as a number like zero or one. The next event is then marked as one or two and so on. In this example the ordering is case-based, but it does not have to be. The events can also get an ordering without considering the case they belong to.

Another example of a time encoding in this category is the time in between events. To calculate the time in between events, a previous and current event is necessary. The current event's time encoding is the time between the previous event and the current event. When applied to a dataset, one specific event can be chosen as a fixed previous event and time is always calculated based on that event. This is called time in between cumulative, since the time is represented cumulatively. The standard version of time in between is that the previous and current event shifts over the dataset. For example, to calculate the time encoding of event B, event A is the previous event and event B is the current event. Then to calculate the time encoding of event C (the event after event B), the previous event shift to event B and the current event shift to event C. A complication arises for the first event in the dataset since there is no previous event. Therefore, the time encoding can simply be the number zero. Furthermore, different kinds of timestamps can be used to calculate the time in between events. For example the start times or the end times can be used. But the time in between events can also be calculated by using the end time of event A and the start time of event B. This way you capture the inactive time between the events. Lastly, just as the order of events, this time encoding can be case bound or unrelated to cases.

A third kind of "Relative time" encoding is duration. Duration indicates how long a certain activity or a group of activities took to complete. It can be calculated on activity level or on a higher level, like on case level. When calculating the duration of a case, this can be the duration of the entire case or the duration of a case until a certain point. For example, when predicting the next activity of a case, the duration of the case is the duration until the last performed activity, thus not till the end of the case since the case has not ended yet. Furthermore, the duration of activities within a case can also be represented in a cumulative manner. When the duration of a case is calculated one has to decide whether breaks between events will be included in the duration or not. When both the start and complete timestamps are given, a distinction can be made between total time spend on case and total time exactly worked on the case. In the first instance, once an activity in a case is started the case duration keep adding up until the last activity has ended. For the second instance the duration of a case is the sum of the durations of the activities of the case. The breaks between the activities are not included in the second instance.

A last kind of time encoding is an interval, which is defined by two points in time, noted as the start and end. Individually those start and end times are part of the "Point in time" category, but together they are part of the "Relative time" category.

Just as the "Point in time" category, this category can be represented at different precision levels. Duration can be defined by the amount of weeks, days, hours, minutes et cetera. Intervals can be defined by the date, or the date and the time.

Examples of the time encodings belonging to the "Relative time" category are listed in Table 1 and 2. A distinction between examples with and without cases are made. Table 1 shows the time encodings when no cases are defined, while in Table 2 the time encodings do consider cases. In Table 1 the following time encodings are shown: order, time in between (calculated with start times), time in between (calculated with end and start times), duration, and interval. Table 2 shows the following time encodings: order, time in between (calculated with start times), time in between (cumulative), duration with breaks (1), duration without breaks (2), and duration (2) (cumulative).

Table 1. Overview of relative time encodings (without cases)

Event	Start	End	Order	TiB (Start-Start)	TiB (End-Start)	Duration	Interval
A	10:00	10:20	1	0'	0'	20'	10:00 - 10:20
B	10:30	11:10	2	30'	10'	40'	10:30 - 11:10
C	11:15	11:20	3	45'	5'	5'	11:15 - 11:20
D	12:00	12:15	4	45'	40'	15'	12:00 - 12:15
E	12:20	12:25	5	20'	5'	5'	12:20 - 12:25

Table 2. Overview of relative time encodings (with cases)

Case	Event	Start	End	Order	TiB	TiB (cum)	Duration (1)	Duration (2)	Duration (2) (cum)
1	A	10:00	10:20	1	0'	0'	80'	65'	20'
1	B	10:30	11:10	2	10'	10'	80'	65'	60'
1	C	11:15	11:20	3	5'	15'	80'	65'	65'
2	A	12:00	12:15	1	0'	0'	25'	20'	15'
2	C	12:20	12:25	2	20'	20'	25'	20'	20'

3.3 Overview

Figure 1 gives an overview of the two time encoding categories. The first category, "Point in time", is defined by its two conditions: moment in time and unique. The second category, "Relative time", is defined by one condition: uses at least two timestamps. Furthermore, this category consists of four kinds of time encodings: order, time in between, duration and interval. Order, time in between and duration can be defined on case level or activity level. Both time in between and duration can be represented cumulative. Time in between can be calculated in multiple ways by using different timestamps (start, end or both). Lastly, a distinction can be made between the incorporation of inactive time when calculating the duration of a case.

Different kinds of time encodings can be used together and one is not obliged to stay within one of the categories. Time encodings from different categories can be used in unison.

Fig. 1. Overview of time encodings

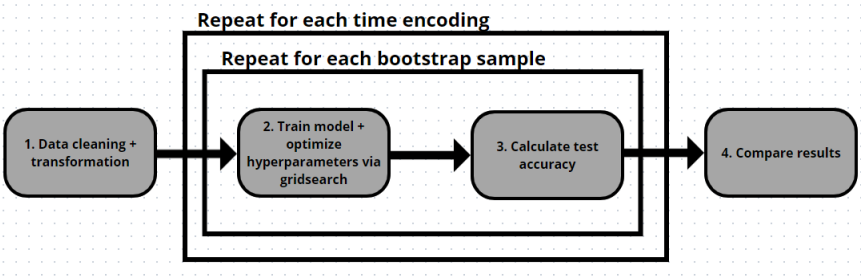
Point in time:
- Happens at a specific moment in time
- The moment in time is unique

Relative time (uses at least two timestamps)
- Order
  - case related or not case related
- Time in between
  - start and/or end times
  - case related or not case related
    - cumulative or not
- Duration
  - activity
  - case
    - with breaks or without breaks between the activities
    - cumulative or not
- Interval

4 Methodology

This paper aims to determine whether the use of different time encodings has an effect on the accuracy of behavioral prediction. To achieve this, six different time encodings will be compared. In this section, the experimental setup of this paper is described. A visual representation of this setup is shown in Figure 2.

Fig. 2. Experimental setup



The first step of the experimental setup is to clean our chosen dataset and transform it into the right shape for our prediction method, namely a Bidirectional LSTM model.

This step also encompasses bootstrapping the dataset to compare the effect of different time encodings over multiple bootstrap samples instead of just one.

The second step is determining the best prediction model for each combination of bootstrap sample and time encoding. In this step, a 5-fold cross-validation grid search is applied. Once the best model is determined, the model is applied to a test set to calculate the performance of the model (step 3). Those two steps are performed for each bootstrap sample and time encoding combination. For each time encoding variant, 100 accuracy scores are calculated, one for each bootstrap sample. In total, 600 accuracy scores will be compared, which is also the last step of the experimental setup. The accuracy scores of the different time encodings variants will be statistically compared with a Friedman and paired Wilcoxon test.

4.1 Step 1: Data cleaning and transformation

The chosen dataset represents transactions (purchases and returns) from 1/12/2009 until 9/12/2011 from a UK-based online retail store (Dua & Graff, 2019). Each instance in the data represents a bought or returned item. In other words, a customer purchase consisting of multiple items is spread across multiple instances. The data was aggregated to ensure that every act of behavior/purchase is represented by one instance. Furthermore, instances that did not have a Customer ID were removed and discounts were kept in the dataset.

The data had to be transformed into sequences to prepare for the LSTM model. Each customer is represented as one sequence, where each timestep in the sequence is a purchase of that customer. Each sequence consists of 24 timesteps/purchases. These are the first 24 purchases a customer has made in the time span of the original dataset. The goal is to predict the price of the customer's next purchase, the 25th purchase. To achieve this, only customers with at least 25 purchases are considered, which is a total of 315 customers. Of those 315 customers, 25 were from a different country than the UK. Besides the UK, 7 other countries were listed in the dataset. Further descriptive features of the used dataset can be found in Table 3. The value row represents each individual purchase value, while the total value represents the total value of the 25 purchases of a customer.

Table 3. Descriptive features dataset

	Min	Max	Mean	Median
Buys	11	25	20	21
Returns	0	14	5	4
Value	-8985,6	33167.8	413.87	270.1
Total value	924.7	120879.4	10346.69	7096.48

4.1.1 Features To choose the features of the data that will be used, the conceptual model of a behavioral vector from Cao (2008) will be used as inspiration (Cao, 2008). An adapted version of the simplified model is used as base to choose the features. The adapted version is constructed as follows:

$$\overrightarrow{\gamma} = \{s, a, f, e, t\}$$

Where $\overrightarrow{\gamma}$ represents a behavioral vector. The attributes, their meaning and the implementation of the dataset are listed below:

- Subject (s): The subject is the entity that conducts an activity. In other words, the customer that buys or returns one or more items. The customer is identified by their customer ID. In our case each sequence represents a customer, therefore the customer ID will not be explicit represented in the data, but it will be implicitly implied.
- Action (a): The action taken by the subject. For the used dataset this can take on two values: buy or return.
- Impact (f): The results created by the action. This is the total price of the purchase.
- Context (e): The context wherein the behavior is conducted. In our case this is the country where the customer is located. Note that this variable has been added to the adapted vector in comparison to the original vector.
- Time (t): At what time/date occurs the behavior? Different values of this variable will be tested to identify if there is a difference in the accuracy of the behavioral prediction.

4.2 Repeat for each time encoding

Six different ways of representing time in data will be used to determine if different time encodings affect behavioral prediction accuracy.

The first time encoding/variant represents the order of purchases. The order will be depicted as a number between 1 and 24 since the input consists of 24 transactions of a customer. This time encoding belongs to the second category, "Relative time", since determining the purchase order relates purchases to the other purchases. The order of a purchase can not be determined without considering other purchases.

The second variant depicts the exact moment the purchase was made using the timestamp given in the dataset (date and time). The timestamp will be represented as a float since this is a requirement for LSTM models. To this end, UNIX time will be used since this time-form can be represented by a number. UNIX time records the number of seconds that have passed since the first of January in 1970 at UTC (Hauser, 2018). Since the timestamp from the dataset is recorded to the minute, the UNIX time representation will also depict minutes instead of seconds. This variant belongs to the first time encoding category, "Point in time".

By transforming the timestamp into UNIX time, possible valuable information like the day of the week, the month, et cetera, are not explicitly defined. Due to this, the third variant was introduced: UNIX time and derived features. The derived features introduce to the prediction model the information that was hidden in the UNIX transformation. The derived features used are the different elements found in a timestamp: time within the day, weekday, month, and year. Just as the second time encoding, this time encoding belongs to the first category, namely "Point in time".

A fourth time encoding is the time in between events. How long does a customer wait before making their next purchase? This time encoding is an example of the "Relative time" category since calculating the time from transaction b requires transaction a's time. The first transaction of each customer will get a time equal to zero. The second transaction will get a time equal to the difference (in minutes) between the first and second transaction.

The fifth encoding includes all of the above-mentioned time encodings. It is, therefore, part of both the "Point in time" and "Relative time" category.

Besides the five mentioned time encodings, a variant of the first time encoding, order, will be added. The sixth variant is the same as the first one, namely the order of

transactions. The only difference is that the transactions were randomly shuffled within each customer. This variant can be seen as one where time is not included, since the time is randomly assigned. This variant will be used to determine which effect the time dimension, in general, has on the behavioral prediction accuracy.

An overview of the time encodings is given in Table 4. From the chosen time encodings, two are examples of "Point in time", two (+ one) of "Relative time", and one that combines both categories.

Table 4. Used time encodings/variants

Variant	**Time encoding**	**Category**
Variant 1	Order	Relative time
Variant 2	UNIX time	Point in time
Variant 3	UNIX time + derived features	Point in time
Variant 4	Time in between	Relative time
Variant 5	All	Relative time and Point in time
Variant 6	Random order	Relative time

4.3 Repeat for each bootstrap sample

Besides repeating training the model (step 2) and calculating the accuracy of the test set (step 3), for each time encoding variant, both steps will also be repeated for each bootstrap sample. The chosen dataset will be bootstrapped 100 times, creating 100 different samples. Bootstrapping randomly selects sequences of the data and creates thereby a new dataset. Once a sequence is selected, it can be selected again since bootstrapping works with replacement. The selection process stops once the new dataset has the size of the original dataset. To achieve 100 different datasets, the selection process is simply repeated 100 times. By applying bootstrapping, not just one accuracy score is calculated for each variant, but 100 different accuracy scores. One accuracy score for each bootstrap sample.

Bootstrapping is possible for the chosen dataset since the purchases of customers are not interrelated. The purchasing behavior of customer 1 does not influence the purchasing behavior of customer 2 and vice versa. Randomly selecting purchase sequences does not change the structure of the dataset.

4.4 Step 2: Train model and optimize hyperparameters via gridsearch

To predict the 25th purchase of a customer, a Bidirectional LSTM model will be used. The Bidirectional LSTM model is build and run with the Keras API in Python. To determine the best set of hyperparameters for each bootstrap sample and time encoding a grid search is applied.

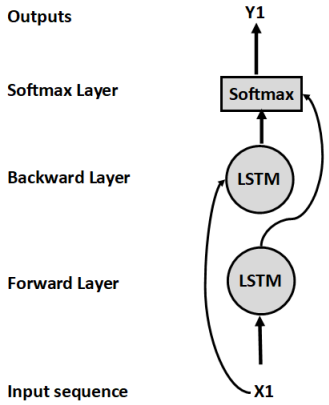
4.4.1 Bidirectional LSTM The original dataset for this research is a transactional event dataset, where each record represents a particular customer's transaction at a specific moment in time. This means that a customer can be represented as a sequence of transactions. Due to the sequential nature of the data, a Recurrent Neural Network is preferred over a traditional Neural Network for predicting since a Recurrent Neural Network (RNN) is more suitable for predicting sequential data. It is able to discover long-term dependencies and patterns in sequential data. Furthermore, it is able to not only remember

the given information but also forget information that is no longer relevant to the task it is performing (Gers et al., 2003; Greff et al., 2017).

A Bidirectional Long Short Term Memory (LSTM) RNN was chosen over a standard RNN since a Bidirectional LSTM RNN is more capable of handling long-term dependencies. It is better capable to remember and forget information based on how relevant the information is (Maree & Omlin, 2021). Furthermore, the input data are sequences with 24 time steps. To detect and use the long-term dependencies between the time steps to their fullest potential, a Bidirectional LSTM RNN is preferred over a standard RNN.

The architecture of our Bidirectional LSTM is shown in Figure 3. The input is the sequential data which is fed to two LSTM layers, a backward and a forward layer. By having a backward and forward layer, the input is processed in a forward and backward manner instead of just forward (Greff et al., 2017). This is also why a Bidirectional LSTM was chosen over a unidirectional LSTM. After going through both layers, the information is fed to the softmax layer. Softmax is an activation function that allows the model to predict the probability that an observation is a part of one of the defined classes. The class with the highest probability is then chosen as the output class (Zaccone & Karim, 2018).

Fig. 3. Bidirectional LSTM network

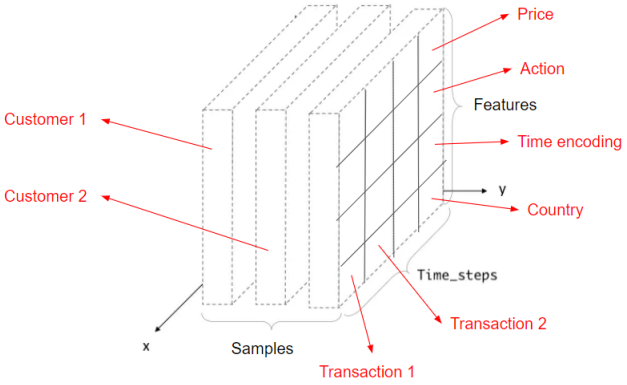


For our LSTM the output is the price class of the 25th purchase for each customer. The price of each transaction has been discretized into one of four categories. Since the 25th purchase is being predicted, the input of the LSTM model are the 24 previous transactions of the customers. Furthermore, an LSTM model's input must be a three-dimensional dataset. Therefore, our (2d) dataset has to be transformed. As seen in Figure 4, the three dimensions of an LSTM input dataset are samples, timesteps, and features. The customers represent the sample dimension. The timesteps are the transactions made by a customer and the features are the remaining parts of the vector from Cao's (2008) model: action, impact/price, context/country, and time/time encoding.

To test the performance of the models, the data was split into two sets: a training and a test set. The test set comprises 30 % of the data, which are 95 customers. The other 70 % is the training data which consists of 220 customers.

4.4.2 Hyperparameter optimization Different combinations of hyperparameters were tested to achieve the best model for each combination of bootstrap sample and

Fig. 4. Input dimensions



time encoding. To ensure the best combination of hyperparameters is discovered, a grid search with 5-fold cross-validation is applied. The grid-search algorithm tests every possible combination of hyperparameters within the predefined range of hyperparameters. Based on a mean accuracy score, which is calculated through 5-fold cross validation, the algorithm decides which model is the best model for the bootstrap sample it was given. The cross-validation is an add-on to grid search and is used to get a more reliable estimate of the model's performance (Zaccone & Karim, 2018).

As seen in Figure 3, the used LSTM architecture consists of five layers: input, forward, backward, softmax, and output. The input, softmax, and output layers were already discussed in subsection 4.4.1. In this subsection, the forward and backward layers are discussed. The forward layer of the LSTM model will process and analyze the input sequences in a forward manner, while the backward layer will do the same but in a backward motion. By processing and analyzing sequences in both directions, the relations between the different purchases of a client can be scrutinized (Greff et al., 2017). Each of those layers consists of different cells that contain a cell and hidden state. The cell state represents the long-term memory and the hidden state the short-term memory. The cell receives an input-sequence and the output from the previous cell state. After analyzing the input, they return an output to the next cell. The flow of information between the different cells are regulated by three gates (input, forget, and output). The inputgate determines which information from the input will be stored in that particular cell. The forgetgate decides which information from the previous cell can be forgotten. Lastly, the outputgate decides which information from the current cell is transferred to the next cell. The flow of information and the performance of each cell can be adjusted and optimized by the use of hyperparameters (Greff et al., 2017; Gers et al., 2003). The used hyperparameters and their values are listed in Table 5 and are further discussed below. The listed hyperparameters are chosen based on which hyperparameter settings were used most often in previous research. Some hyperparameters only have one possible value, this value is the standard value that is rarely changed.

Table 5. Used hyperparameters in grid search

Units	5,10, 15
Drop out	0.2, 0.4, 0.6
Loss function	Sparse, Kullback, Hinge
Optimizers	Sigmoid, Adam, RMSprop
Activation function	Tanh

The first hyperparameter is called units. Units represents the dimension of the hidden states in the LSTM layers. The higher the dimensionality, the more predictive power an LSTM will have (Zaccone & Karim, 2018), but there is a risk. The higher the dimensionality, the larger the chance of overfitting the model to the training data. The LSTM model will work very well on the training data but will perform poorly on data it has not yet seen. This is a phenomenon that has to be avoided (Witten, Frank, Hall, & Pal, 2017). Due to the small dataset and the fact that the dataset is not very complicated (a small number of variables), the dimensionality was restricted to 5, 10, and 15. A second hyperparameter is a drop-out layer. This layer randomly chooses input units to drop by giving them a value of zero. The number of units that will be dropped is decided by the frequency given to the layer (Zaccone & Karim, 2018). In our experiment, the layer can have the following frequencies: 0.2, 0.4, and 0.6. The dropout assures that the prediction model does not have too much predictive power and does not overfit the training data. The third hyperparameter is the way that the difference between the predicted and actual output is calculated, namely the loss function (Witten et al., 2017). Since the goal of the LSTM is to classify transactions, a probabilistic loss function is needed. For each class, the probability that a transaction will belong to a class is calculated, and the class with the highest probability is chosen as the output class. The way the probability is calculated depends on the chosen loss function. This experiment uses three different loss functions: Sparse categorical cross entropy, Kullback-Leiber divergence, and Categorical Hinge. Optimizers, as the name suggests, optimize the attributes in the LSTM layers such that the loss is minimized. Each optimizer has a different set of parameters that are used to optimize the LSTM layer. Three different optimizers are used, namely Sigmoid, Adam, and RMSprop. Lastly, the activation function determines the output of the different cells. The function decides whether the cell should be activated or not. In LSTM models, the Tahn activation function is traditionally used. All the other hyperparameters defined in the LSTM class in Keras, are the standard values chosen by the developers (Zaccone & Karim, 2018).

4.5 Step 4: Compare results

The last step of the experimental setup is statistically comparing the obtained accuracy scores. According to Demsar et al. (2006), non-parametric tests are the best way to compare the results of classification models. Because of this, the Friedman and paired Wilcoxon test will be used to determine if there is a difference in accuracy when using different kinds of time encodings. The Friedman test will determine if there is a difference between the six models in general, while the paired Wilcoxon test will compare the time variants pairwise.

5 Results

In this section, the results from the applied methodology, described in section 4, will be discussed. First, the accuracy scores of the six time encodings are generally discussed by a descriptive data analysis. Afterward, the differences in accuracy results between the different kinds of time encodings are statistically compared by two non-parametric tests. First, the Friedman test will be applied to test if there is a general difference between the six time encodings. Afterward, the paired Wilcoxon test compares the different time encodings pairwise.

5.1 Description accuracy results

An optimal LSTM model was obtained using a grid search for each combination of bootstrap sample and time encoding. Afterward, this optimal LSTM predicted the test set from the bootstrap sample to calculate an accuracy score. A total of 600 accuracy scores were calculated (6 variants, 100 bootstrap samples).

We plotted the accuracy score per variant on a box plot to compare the different scores. As seen in Figure 5, the box plots are very similar, but minor differences exist. For example, variant 2, UNIX time, has a smaller spread in comparison to the other time encodings. Variants 1 and 6, order and random order, have the most extensive spread of accuracy scores. The median values for each time encoding lie just under 0.3. This can also be observed in Table 6. Table 6 lists the following statistics: minimum, quartile 1, median, mean, quartile 3, maximum, variance, and standard deviation. The highest accuracy score obtained in a model was for variant 1, with a score of 49,47 percent. The lowest accuracy score is 6,32 percent, obtained by variant 6. Furthermore, when the different statistics are observed among the variants, one can notice that all values are rather similar. There are no considerable differences between each of the statistics among the variants. This can be a first indication that there are no significant differences in the accuracy scores of the different time encodings. To confirm this speculation, the accuracy scores will be statistically compared in the following subsection.

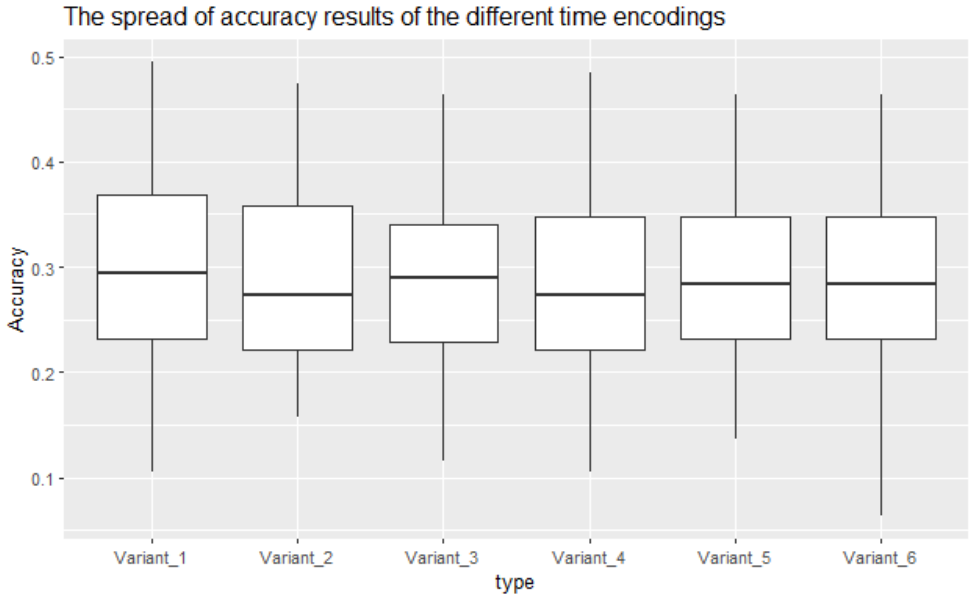


Fig. 5. Box plot of accuracy values for each time encoding

5.2 Statistical comparison

The first statistical test that will be used is the Friedman test. The Friedman test will determine whether the accuracy results differ depending on the time encoding used in a

Table 6. Summary of accuracy results for each time encoding

LSTM	Min	Q1	Median	Mean	Q3	Max	Var	SD
1	0.1053	0.2316	0.2947	0.2981	0.3684	0.4947	0.007248	0.085134
2	0.1579	0.2211	0.2737	0.2855	0.3579	0.4737	0.006593	0.081196
3	0.1158	0.2289	0.2895	0.2853	0.3395	0.4632	0.005996	0.077432
4	0.1053	0.2211	0.2737	0.2852	0.3474	0.4842	0.006391	0.079944
5	0.1368	0.2316	0.2842	0.2869	0.3474	0.4632	0.005985	0.077361
6	0.0632	0.2316	0.2842	0.2812	0.3474	0.4632	0.007018	0.052095

prediction model. The Friedman test was performed in the programming language R. For ease, the accuracy scores and, thereby, the performance of the model will be represented by the gamma sign (γ). The number under the gamma sign represents the time encoding. The null and alternative hypotheses can be described as follows:

$$H_0 : \gamma_1 = \gamma_2 = \gamma_3 = \gamma_4 = \gamma_5 = \gamma_6$$
$$H_1 : \gamma_i \neq \gamma_j \text{ for at least one pair (i,j)}$$

The null hypothesis states that the accuracy results from each time encoding are equal. The alternative hypothesis states that for at least one pair of time encodings, the accuracy results differ. As shown in Figure 6, the Friedman test has a p-value of 0.9535. Therefore, the null hypothesis cannot be rejected, and there are no significant differences between the accuracy scores of the six different time encodings.

```
        Friedman rank sum test

data:  df$Accuracy, df$Variant and df$Dataset
Friedman chi-squared = 1.1064, df = 5, p-value = 0.9535
```

Fig. 6. Friedman test

The second statistical test is a paired Wilcoxon test, which is also performed in R. This test compares each population/time encoding pairwise and determines whether there are differences between each combination. The null and alternative hypotheses can be described as follows:

$$H_0 : \gamma_i = \gamma_j \text{ where } i \neq j \text{ and i,j=}\{1,2,3,4,5,6\}$$
$$H_1 : \gamma_i \neq \gamma_j \text{ for at least one pair (i,j)}$$

The accuracy scores are represented by the gamma sign (γ). The number under the gamma sign represents the time encoding.

Figure 7 displays the results of the paired Wilcoxon test. The p-value for each combination of time variants was equal to one. The null hypothesis can, therefore, not be rejected for any of the combinations. There are no significant differences in accuracy results among this dataset's different time variants.

There are no significant differences in the accuracy scores of the different time encodings. The outcome of this experiment indicates that the way time is encoded does

```
        Pairwise comparisons using Wilcoxon rank sum test with continuity correction

data:  data2$Accuracy and df$Variant

  1 2 3 4 5
2 1 - - - -
3 1 1 - - -
4 1 1 1 - -
5 1 1 1 1 -
6 1 1 1 1 1

P value adjustment method: bonferroni
```

Fig. 7. Wilcoxon test

not influence the behavioral prediction accuracy for this particular dataset. However, this does not necessarily mean this also applies to other datasets. The used dataset is perhaps not very sensitive to time or even not sensitive to time at all. The results and its implications are further discussed in the Discussion section.

6 Discussion

The results of this experiment determined that there are no significant differences in the accuracy scores when different kinds of time encodings are applied. These results suggest that the way time is encoded does not influence the prediction power of a neural network. A neural network is capable of extracting the intrinsic information from the time dimension no matter which encoding is used. Researchers should therefore save their time in testing different time encodings and use the one that is most convenient for them, for example, the one recorded in their dataset. Like, Wang et al. (2019) did this when predicting the outcome of a process. They used the timestamp of the event in their LSTM model. However, a few connotations have to be made about these results. Firstly, this experiment was only applied to one specific dataset. The results of this experiment can be dependent on the dataset. Secondly, the dataset that was used only contained 315 customers that each made 25 transactions, which sums up to 7875 data points. This is a relatively small dataset for the application of deep learning. Due to the small dataset, the LSTM could possibly not discover the more intrinsic elements in the time dimension.

Due to the black box principle of neural networks (Witten et al., 2017), we do not know why exactly there are no differences in the different time encodings. Currently, there is no way of knowing what kinds of patterns the LSTM models discovered and how they are linked to how time is encoded. There have been experiments that tried to discover the black box behind deep learning prediction methods, but it remains, as the name suggests, a mystery for researchers (Carmona, Dwekat, & Mardawi, 2022; Yijun et al., 2021). Nevertheless, we are able to test if the time dimension influences the accuracy scores of our LSTM models. To do this, the sixth time encoding was added to this experiment. The sixth time encoding is the same as the first one, namely the order of transactions. The only difference is that the transactions were randomly shuffled within each customer. This time encoding can be seen as one where time is not included, since the time is randomly assigned. Normally, if the accuracy results are dependant on the time dimension of the dataset, the accuracy scores of the sixth variant should be lower than the accuracy scores of the first variant. Yet, there were no differences detected between the six variants. This suggests that the time dimension does not contain any information that the prediction

model uses to predict the 25th transaction. When the prediction model neglects the information from the time dimension, the way the time dimension is encoded will not affect the prediction accuracy. This can be an explanation for why no significant differences were found between the different kinds of time encodings. A possible explanation for the lack of use of the time dimension by the prediction method is the fact that no prediction about time was made. Perhaps when instead of predicting the price category of the 25th purchase, the time until the next purchase was predicted or both, time would have played a more dominant role in the prediction model. For example, in the research of Tax et al. (2017) they predicted the next event of a process in combination with the timestamp of that event. To do this, they used three different time encodings. However, the same considerations as mentioned above can be made, namely the restrictions of the use of one small dataset.

7 Conclusion

This paper aimed to determine if different kinds of time encodings affect behavioral prediction. Various methods for expressing time have been used in behavioral prediction, such as the exact timestamp, the duration of an event, the time between events, et cetera. However, the effect of the kind of time encoding that is used was not yet explicitly researched. Furthermore, there was no overview of the way time can be used in data. This paper addresses these gaps by giving an overview of possible time encodings and taking the first step in discovering the effect time encodings can have on prediction. More specifically, this paper aims to discover if different kinds of time encodings can influence behavioral prediction accuracy. To achieve this, six different time encodings were applied to a 100-times bootstrapped dataset representing customers' purchasing behavior of an online retail store in the United Kingdom. To make the predictions, a Bidirectional Long-Short-Term Memory (LSTM) model was used in combination with a grid search with cross-validation to find the optimal models for each combination of time encoding and bootstrap sample. The accuracy scores were statistically compared using two non-parametric tests, namely a Friedman and paired Wilcoxon test.

No significant differences were found between the six time encodings. The way time is encoded in the used dataset does not influence prediction accuracy. These results indicate that researchers do not have to consider how the time dimension is represented in their dataset when making behavioral predictions. Researchers can use the original form of the time dimension in their dataset. To further examine these results, the sixth time encoding was introduced. The sixth time encoding was the order of events, but the transactions within each customer were randomly shuffled. Since each of the six time encodings had no significant differences between their accuracy scores, the order in which the transaction occurred did not influence the prediction accuracy. The prediction model did not use the information from the time dimension in the used dataset. These results are quite remarkable and require further investigation. Therefore, we recommend expanding this experiment to multiple datasets instead of just one, preferably from different domains. The effect of time encodings can be both dataset and domain sensitive. Furthermore, the output of the prediction model can also be expanded. Instead of just predicting the value of the next transaction, the time of the next transaction can also be predicted. A last recommendation for future work is to use more kinds of time encodings.

References

Cao, L. (2008). Behavior Informatics and Analytics: Let Behavior Talk. In *2008 IEEE International Conference on Data Mining Workshops* (pp. 87–96). doi: https://doi.org/10.1109/ICDMW.2008.95

Cao, L. (2010). In-depth behavior understanding and use: The behavior informatics approach. *Information Sciences*, *180*(17), 3067–3085. doi: https://doi.org/10.1016/j.ins.2010.03.025

Carmona, P., Dwekat, A., & Mardawi, Z. (2022). No more black boxes! Explaining the predictions of a machine learning XGBoost classifier algorithm in business failure. *Research in International Business and Finance*, *61*, 101649. doi: https://doi.org/10.1016/j.ribaf.2022.101649

Dua, D., & Graff, C. (2019). {UCI} Machine Learning Repository. *University of California, Irvine, School of Information and Computer Sciences*. Retrieved from http://archive.ics.uci.edu/ml

Eder, J., Franceschetti, M., & Lubas, J. (2021). Time in Data Models. In T. K. Dang, J. Küng, T. M. Chung, & M. Takizawa (Eds.), *Future Data and Security Engineering* (pp. 23–35). doi: https://doi.org/10.1007/978-3-030-91387-8_2

Gers, F. A., Schraudolph, N. N., & Schmidhuber, J. (2003). Learning precise timing with lstm recurrent networks. *The Journal of Machine Learning Research*, *3*, 115–143. doi: https://doi.org/10.1162/153244303768966139

Giese, L., Melzheimer, J., Bockmühl, D., Wasiolka, B., & Rast, W. (2021). Using Machine Learning for Remote Behaviour Classification—Verifying Acceleration Data to Infer Feeding Events in Free-Ranging Cheetahs. *Sensors*, *21*(16), 5426. doi: https://doi.org/10.3390/s21165426

Greff, K., Srivastava, R. K., Koutnık, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A Search Space Odyssey. *IEEE Transaction on Neural Networks and Learning Systems*, *28*(10), 2222–2232. doi: https://doi.org/10.1109/TNNLS.2016.2582924

Hauser, E. (2018). UNIX time, UTC, and datetime: Jussivity, prolepsis, and incorrigibility in modern timekeeping. *Proceedings of the Association for Information Science and Technology*, *55*(1), 161–170. doi: https://doi.org/10.1002/pra2.2018.14505501018

Jain, D., Sinha, A. R., Gupta, D., Sheoran, N., & Khosla, S. (2018). Measurement of Users' Experience on Online Platforms from Their Behavior Logs. In D. Phung, V. S. Tseng, G. I. Webb, B. Ho, M. Ganji, & L. Rashidi (Eds.), *Advances in Knowledge Discovery and Data Mining* (pp. 475–487). doi: https://doi.org/10.1007/978-3-319-93034-$3_3$8

Maree, C., & Omlin, C. W. (2021, December). Clustering in Recurrent Neural Networks for Micro-Segmentation using Spending Personality. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1–5). doi: https://doi.org/10.1109/SSCI50451.2021.9659905

Martens, D., Provost, F., Clark, J., & de Fortuny, E. J. (2016). Mining Massive Fine-Grained Behavior Data to Improve Predictive Analytics. *MIS Quarterly*, *40*(4), 869–888. doi: https://doi.org/https://doi.org/10.25300/MISQ/2016/40.4.04

Peltier, C., Lejeune, F.-X., Jorgensen, L. G. T., Rametti-Lacroux, A., Tanguy, D., Godefroy, V., ... Batrancourt, B. (2022). A temporal classification method based on behavior time series data in patients with behavioral variant of frontotempo-

ral dementia and apathy. *Journal of Neuroscience Methods*, *376*, 109625. doi: https://doi.org/10.1016/j.jneumeth.2022.109625

Rivera Lazo, G., & Nanculef, R. (2022). Multi-attribute Transformers for Sequence Prediction in Business Process Management. In P. Pascal & D. Ienco (Eds.), *Discovery Science* (pp. 184–194). doi: https://doi.org/10.1007/978-3-031-18840-4$_1$4

Stavinova, E., Bochenina, K., & Chunaev, P. (2021). Predictability Classes for Forecasting Clients Behavior by Transactional Data. In M. Paszynski, D. Kranzlmüller, V. V. Krzhizhanovskaya, J. J. Dongarra, & P. M. Sloot (Eds.), *Computational Science – ICCS 2021* (pp. 187–199). doi: https://doi.org/10.1007/978-3-030-77967-2$_1$6

Tax, N., Verenich, I., La Rosa, M., & Dumas, M. (2017). Predictive Business Process Monitoring with LSTM Neural Networks. In E. Dubois & K. Pohl (Eds.), *Advanced Information Systems Engineering* (pp. 477–492). doi: https://doi.org/10.1007/978-3-319-59536-8$_3$0

Van Der Aalst, W. M. P. (2011). *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Berlin, Heidelberg: Springer. doi: https://doi.org/10.1007/978-3-642-19345-3

Wang, J., Yu, D., Liu, C., & Sun, X. (2019, July). Outcome-Oriented Predictive Process Monitoring with Attention-Based Bidirectional LSTM Neural Networks. In *2019 IEEE International Conference on Web Services (ICWS)* (pp. 360–367). doi: https://doi.org/10.1109/ICWS.2019.00065

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2017). *Data mining: practical machine learning tools and techniques* (Fourth ed.).

Yassine, A., Singh, S., & Alamri, A. (2017). Mining Human Activity Patterns From Smart Home Big Data for Health Care Applications. *IEEE Access*, *5*, 13131–13141. doi: https://doi.org/10.1109/ACCESS.2017.2719921

Yijun, S., Link to external site, t. l. w. o. i. a. n. w., Cheng, Y., Shah, R. U., Weir, C. R., Bray, B. E., & Qing, Z.-T. (2021). Shedding Light on the Black Box: Explaining Deep Neural Network Prediction of Clinical Outcomes. *Journal of Medical Systems*, *45*(1). doi: https://doi.org/10.1007/s10916-020-01701-8

Zaccone, G., & Karim, R. (2018). *Deep Learning with TensorFlow: Explore Neural Networks and Build Intelligent Systems with Python, 2nd Edition*. Packt Publishing, Limited.