Data Aware Transition Systems for Prediction in Business Processes

Report submitted in fulfilment of the requirements for the Exploratory Project of

Third Year B.Tech.

by

Bhavna Singh (18134501003)
Ritik Pal (18134501025)

Under the guidance of

Mr. Vijay Bijalwan

Department of Computer Science

SCHOOL OF ENGINEERING AND TECHNOLOGY
HEMVATI NANDAN BAHUGUNA GARHWAL UNIVERSITY (A CENTRAL UNIVERSITY)

Srinagar, Uttarakhand 249174, India

September 2021

Declaration

I certify that

1. The work contained in this report is original and has been done by myself and the general supervision of my supervisor.
2. The work has not been submitted for any project.
3. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.
4. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Place: SOET, HNBGU, India
Date:

Bhavna Singh, Ritik Pal
B.Tech.
Department of Computer Science ,
School of Engineering and Technology
Hemvati Nandan Bahuguna Garhwal University,
Srinagar, Uttarakhand, India 246174

Certificate

This is to certify that the work contained in this report entitled "Data Aware Transition Systems for Prediction in Business Processes" being submitted by Bhavna Singh (18134501003), Ritik Pal (18134501025), carried out in the Department of Computer Science, School of Engineering and Technology, Hemvati Nandan Bahuguna Garhwal University (A Central University) is a bona fide work of our supervision.

Place: SOET, HNBGU
Date:

Mr. Vijay Bijalwan,
Department of Computer Science,
School of Engineering and Technology,
Hemvati Nandan Bahuguna Garhwal University
(A Central University) INDIA. 249174

Abstract

This work presents methodologies for predicting the remaining time of a running business process and it's future path (that is, future activity sequence of the process). We have used machine learning algorithms like Support Vector Regression(SVR), Naive Bayes(NB) Classifier. We present four approaches for remaining time prediction for a process. The first one is the specialization technique developed by Van der Aalst popularly known as VDA. In VDA, we construct an annotated transition system(TS) where each state is annotated with the average remaining time till completion from that state. In the second approach, we use support vector regression. We first one hot encode our data because SVR takes vectors as input. We then train our model by using these vectors generated. In the third approach, we use regression with contextual information. For this purpose, we construct a transition system. We then make use of the control-flow information to add contextual information to our model. In the fourth approach, we use data aware transition system. Here, each state is annotated with remaining time measurements and the Naive Bayes classifier probabilities of transition from this state to any other state. SVR is applied on each transition which gives the remaining time values if that transition occurs during the process. The remaining time is predicted as the weighted mean of all the possible transitions from that state, that is, sum of probability of transition to any state multiplied by the remaining time predicted by SVR for that transition.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 What is Process Mining?

Process Mining is a process analysis method that aims to discover, monitor and improve real processes by extracting knowledge easily from available event logs in the systems of current information of an organization[1]. It goes beyond the pure presentation of the key data of the process, recognizing the contextual relationships of the processes, presenting them in the form of graphical analysis in order to diagnose problems and suggest improvements in the quality of the process models. With process mining it is possible to detect or diagnose problems based on facts and not on conjectures or intuitions. Process mining seeks the confrontation between event data (observed behavior) and process models (hand-made or automatically discovered). Through the pairing of event data and process models, it is possible to check compliance, detect deviations, predict delays, support decision making and recommend process redesigns.

The application of process mining in an organization offers the following capabilities:

- Automated discovery of process models, exceptions and instances of processes

(cases) together with basic frequencies and statistics.

- Understanding of different perspectives on operations, not just a process perspective.
- Predictive analysis, prescriptive analysis, scenario testing and simulation with contextual data.
- Improvement of existing or previous process models using additional data from saved records.
- Data preparation and data cleansing support.
- Combination of different process models that interact with each other in a single process mining panel.
- Support for the visualization of how processes contribute to business value (such as business operating models) contextualization of processes.
- Improvement of operational excellence by optimizing processes.

Process mining exploits the information recorded in event logs to perform an analysis of the real process afterwards. There are three main types of process mining:

1. Discovery, which takes an event log and produces a process model without using any prior information, with the help of Process Mining algorithms[1].
2. Conformance, where the event records (real processes) and the corresponding process models (ideal and predefined processes in BPMN) are compared, and the resulting coincidences or differences are identified, in order to diagnose the

deviations or inefficiencies between the process model derivative business and ideal processes[1].

3. Enhancement, where the process models are adapted and improved according to the data of the real process[1].

1.2 Transition System

A transition system consists of a set of configurations and a collection of transitions. Transition systems are used to describe dynamic processes with configurations representing states and transitions saying how to go from state to state. Transition systems can be represented as directed graphs. They differ from finite state automata in several ways:

1. The set of states is not necessarily finite, or even countable.

2. The set of transitions is not finite, or even countable.

Formally, a *transition system* is a pair (S,T) where S is the set of states and T is the set of transitions(a subset of S x S).

A *labelled transition system* is a tuple(S,E,T) where 'S' is the set of states, 'E' is the set of labels/events and 'T' is the set of transitions (a subset of S x E x S).

An *Annotated transition system* is a labelled TS having some set of measurements (also known as annotations) associated with each and every state of the TS. So, to define an annotated TS, we need event and state representation functions along with a measurement function[2].

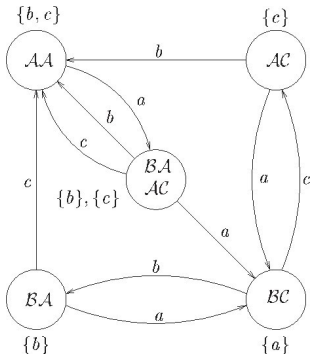

Figure 1.1 Example of a transition system[2]

Chapter 2

Project Work

2.1 About the Dataset

The data set concerns with the execution of process instances of the information system for the management of road-traffic fines by local police of an Italian municipality. The management of road-traffic fines have to comply with Italian laws, which detail the precise work-flow. Usually, when a driver commits a violation, a policeman opens a new fine management and leaves a ticket on the car glass. The actual amount of fine depends on the violation performed. The fine notification must be sent to the offender within 180 days. The payment can occur at any moment, i.e., before or after that the fine notification is sent by post. If the offender does not pay within 60 days since the reception of the fine notification, the fine doubles. If the offender never pays, eventually the fine is sent to a specialized agency for credit collection. In several circumstances, the fine can be dismissed. For instance, the policeman could make a typo when the fine form is filled in (e.g., the license plate number does not match the type of car/truck/motorbike). More importantly, the offender may think to have come under injustice and, hence, can appeal to a judge, the local prefecture, or both. If the appeal is in favor of the offender, the fine is dismissed. Otherwise, the management will carry on as if the offender had never appealed. We extracted

only that log from the dataset which ends with "sending for credit collection", i.e., the offenders have not paid the fine in full. We used 400 traces in total from the dataset. Every trace includes five events on an average. It consists of 15 features with the timestamp, case-number, and trace-number.

2.2 Data Preprocessing

The data is in XES format. So, we have converted it into a python readable format that is in the .csv file. For that, we made a python program(xestocsv.py) that converts the .xes file to the .csv file.

2.2.1 XES Format

Given below is an example of a file in xes format.

< trace >

< stringkey = "concept : name"value = "A10041"/ >

< event >

< stringkey = "concept : name"value = "CreateFine"/ >

< stringkey = "vehicleClass"value = "A"/ > < floatkey =

"totalPaymentAmount"value = "0.0"/ >

...

...

< /event >

< event >

< stringkey = "concept : name"value = "SendFine"/ > < stringkey

= "lifecycle : transition"value = "complete"/ >

...

...

< /event >

< /trace >

In this each trace start with <trace> and end with </trace>. Each trace contains some events each of which also start with <event> and with </event>. Also each event contain it's information, i.e., activity name, timestamp, etc.

2.2.2 CSV Format

We have to covert this into csv format which is:

$A10041$*,CreateFine,A,*0.0*,.....*

$A10041$*,SendFine,complete,.....*

In this, each event is separated by lines and features of every event is separated by a comma.

2.3 One Hot Encoding

We use SVR in most of our approaches. To train SVR regressor, we have to give input in numerical values, but we have data in string form. So, to convert it into the numerical value, we use one hot encoding. What one hot encoder does is, it takes the number of different types for a feature and assigns a value to each kind, now one hot coded vector for that type of feature consist of 0s with 1 at the position of its value[12]. Example, concept-name(features)

-Send Fine

-Collect Fine -Add Penalty let value

for all these features are:

Send Fine-1

2.3. One Hot Encoding

Collect Fine-2

Add Penalty-3

So, one hot encoded vector for each of them are-

Send Fine-[1,0,0]

Collect Fine-[0,1,0]

Add Penalty-[0,0,1]

Chapter 3

Background

3.1 Prerequisites

- Set:A set is a well-defined collection of distinct objects. The objects that make up the set (also known as the set's elements or members) can be anything: letters, numbers, alphabets, people, etc.[3] There are 2 ways to describe or specify the members of a set:
  - Intensional definition, i.e., using a rule or semantic description.
  - Extensional definition, i.e., listing each member of the set.
- Multiset:A multiset (bag or mset) is the modification of the concept of set that, unlike a set, allows for multiple instances of the same element. The positive integer number of instances, given for each element is called the multiplicity of this element in the multiset. Order does not matter in multisets, for example, a,a,b and a,b,a are the same multisets. The cardinality of a multiset is defined as the sum of the multiplicity of all the elements of the multiset[4].
- Sequence:A sequence is an enumerated collection of objects in which repetitions are allowed. Like set it contains members (also called elements or terms).

The number of elements (possibly infinite) is called the length of the sequence. Unlike a set, the same elements can appear multiple times at different positions in a sequence, and order of elements matters[5].

- Event: An event is a tuple which contains a case id, process activity, timestamp and some other attributes. Formally, it is a tuple t = (activity, case id, timestamp, att1, att2, ...,attk)[6].
- Trace: A trace is a finite sequence of events σ c = [e1, e2,....,elength(trace)]. We define a partial trace of length k as the trace constituted by the first k events of the trace σc[6].
- Event Log: An event log 'L' the set of traces such that each event occurs at most once in the entire log[6].
- Event Representation Function: It is a function which produces some representation for any event 'e'[6].
- State Representation Function: It is a function which produces a state representation for any partial/complete trace[6].
- Similarity Function: It is a function which returns a similarity value between 2 states which lies in the range [0,1]. It is used to handle the cases of non-fitting traces[6].
- Measurement: It is a function that, given a trace σ and an event index i $\in [1,2,\ldots\ldots,|\sigma|]$ produces some measurement[6].

3.2 Machine Learning Library

Python3 libraries which we used during our projects are:

3.2. Machine Learning Library

3.2.1 Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python[7].

3.2.2 Scikit-Learn

Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy[8].

sklearn.svm.SVR	Epsilon-Support Vector Regression
sklearn.model_selection.GridSearchCV	Used for auto tuning

Table 3.1 Scikit-Learn Library Functions

3.2.3 CSV

The csv module implements classes to read and write tabular data in CSV format. It allows programmers to say, ’write this data in the format preferred by Excel,’ or ’read data from this file which was generated by Excel,’ without knowing the precise details of the CSV format used by Excel. Programmers can also describe the CSV formats understood by other applications or define their own special-purpose CSV formats[9].

3.2.4 Datetime

The datetime module supplies classes for manipulating dates and times in both simple and complex ways. While date and time arithmetic is supported, the focus of

the implementation is on efficient attribute extraction for output formatting and manipulation. For related functionality, see also the time and calendar modules[10].

datetime.strptime	Return a datetime corresponding to date_string.

Table 3.2 datetime Library Functions

Chapter 4

Methodology

We have used four different approaches to predict the remaining time of a running business process. Each approach has its pros and cons and when to use any particular approach depends on the situation. Given an event log containing historical traces about the execution of a business process, we want to predict, for an ongoing process, the time remaining till the completion of the process. Each approach uses some kind of model which will be described later. Each model takes a partial trace as its input, which represents a running process instance, and forecasts the remaining time of the process.

4.1 VDA

The comparison of our approaches is made with respect to the specialization of the technique reported in van der Aalst et al(VDA)[11]. In this, we implement an annotated transition system which keeps information about the meantime remaining from the current state and returns that time. We didn't use SVR in this approach, so this approach is much faster than other approaches. To train our model, we iterate through all the traces in the event log. For each event in a trace, we store the remaining time, till the process is completed, in the corresponding state. Now for each

state, we take the average of the remaining time measurements.

4.2 Simple Regression

In this approach we have used -SVR.

4.2.1 Support Vector Regression

SVR is a regression algorithm[13]. In simple regression we try to minimize the error rate, while in SVR we try to fit the error within a certain threshold. Some terms used in SVR are:

- Kernel: The function used to map a lower dimensional data into a higher dimensional data.
- Hyperplane: In SVM this is basically the separation line between the data classes. Although in SVR we define it as the line that will help us predict the continuous value or target value.
- Boundary line: In SVM there are two lines other than hyperplane which create a margin. The support vectors can be on the boundary lines or outside it. This boundary line separates the two classes. In SVR the concept is same.
- Support vectors: These are the data points which are closest to the boundary. The distance of the points is minimum or least.

The boundry lines are the lines which are at a distance e (e denotes the parameter). Let us consider the case of linear hyperplane, we can say that the equation of hyperplane is $wx + b = 0$. So the equations of 2 boundry lines are

$$wx + b = +e \quad (4.1)$$

4.2. Simple Regression

$$wx + b = -e \quad (4.2)$$

So, the equation that our SVR satisfies can be given as

$$-e \leq -wx - b \leq e \quad (4.3)$$

For non-linear SVR, the kernel functions transform the data into a higher dimensional

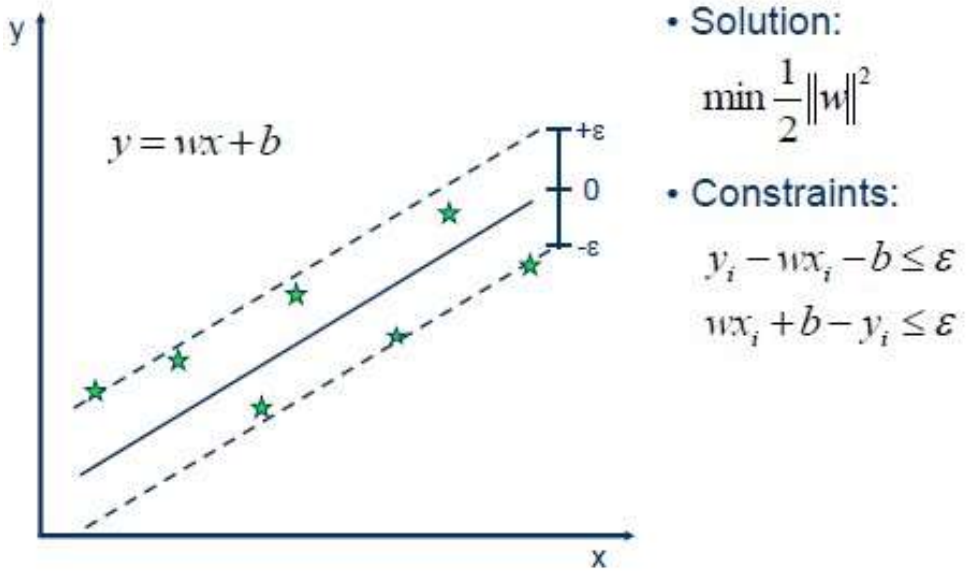

Figure 4.1 Example of a hyperplane [13]

feature space to make it possible to perform the linear separation.

The input to the model for training consists of traces, in particular the attributes corresponding to the events. For additional attributes we consider the last values chronologically observed. Since SVR takes vectors as input so we use the one hot encoded vectors that we got while preprocessing the dataset. To construct the training dataset for the model, we concatenate all the one hot encoded vectors for each and every trace and put this concatenated vector in our training set. Now we train our model with this training set and we use grid search for autotuning, i.e., to learn the parameters that give us

optimal results. For prediction, we first convert the partial trace into a vector, as above, suitable for input to our model and then our model

4.2. Simple Regression

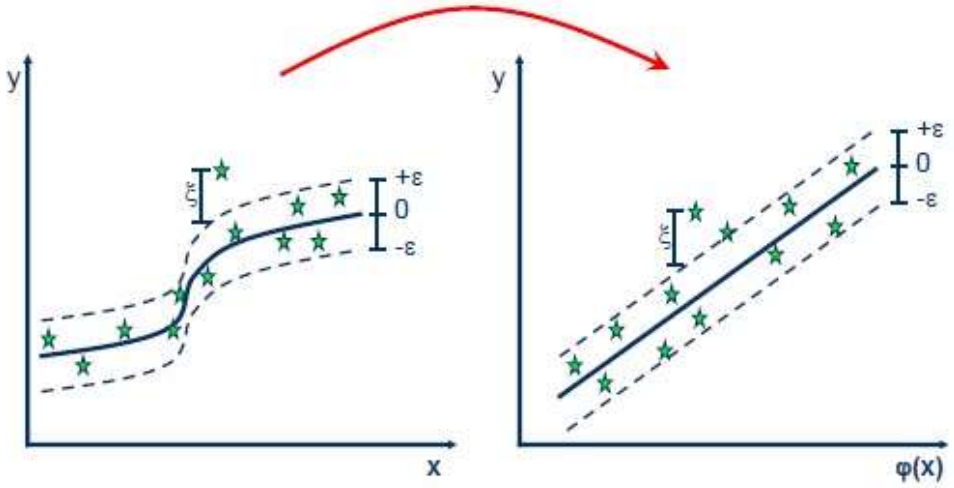

Figure 4.2 Feature set conversion from non-linear to linear [13]

Polynomial

$$k(\mathrm{x}_i, \mathrm{x}_j) = (\mathrm{x}_i . \mathrm{x}_j)^d$$

Gaussian Radial Basis function

$$k(\mathrm{x}_i, \mathrm{x}_j) = \exp\left(-\frac{\left\|\mathrm{x}_i - \mathrm{x}_j\right\|^2}{2\sigma^2}\right)$$

Figure 4.3 Kernel Functions [13]

predicts the remaining time.

4.3 Regression with Contextual Information

This approach makes use of control-flow-information to add contextual information. We add a limited set of attributes/features able to encapsulate the control-flowinformation. We use a transition system for this purpose because it generally represents a good trade-off between expressivity and compactness.

4.3.1 Transition System

A TS consists of a set of states, set of transitions, set of events[2]. Therefore, we need an event representation function and a state representation function to construct a TS. We have used the process activity name to represent any event. To define a state, we first take a starting state Sstart (the state when no event has occurred yet) and then add the events (process activity name) as they occur to the current state to get a new state. We have used three different abstractions as state representation functions namely: 1. set 2. multiset 3. Sequence. The transitions are represented by a tuple t = (initial state, event, final state). In this approach, we might face the problem of non-fitting traces (a trace that produces a state which is not in our transition system) and to handle this problem we use a similarity function. The idea is to find a state which is similar to the state produced by this non-fitting trace. We use different similarity functions for different abstractions.

- For set,

$$f^{sim\_set}(x1, x2) = \frac{|x1 \cap x2|}{|x1 \cup x2|} \tag{4.4}$$

• For multiset/bag,

$$f_{sim\_bag}(x1,x2) = \frac{\#|x1 \cap x2|}{\#|x1 \cup x2|} \quad (4.5)$$

• For sequence,

$$f^{sim\_seq}(x1, x2) = 1 - \frac{f^{dis}(x1, x2)}{max(x1, x2)} \quad (4.6)$$

If we encounter a non-fitting trace (suppose it produces a state s') then we calculate the similarity values of every state in our TS with the state s' and the vector for this state (s') contains the normalized similarity values for every state in the TS. Now our dataset is ready for training. For training, we need TS as input along with the state representation function fstate and the similarity function fsim. We calculate the state associated with each partial trace and we encode it into a one-hot vector or into the normalized similarity vector if it is a non-fitting trace. For prediction, as for the Simple Regression approach, the -SVR model created in the previous step is used to forecast the remaining time of running process instances.

4.4 Data Aware Transition System

In this approach instead of a TS we use an annotated TS[6]. In annotated TS each state of the transition system is decorated with predictive information, called measurements. The set of measurements we have used is the remaining time starting from the state itself. The TS is constructed in the same manner as in the previous approach, the only difference is that with each state we add annotations/ multiset of measurements. Now to make use of these annotations effectively, we enrich each state with a Naive Bayes classifier and each transition with a support vector regressor.

4.4.1 Naive Bayes Classifier

Naive Bayes classifiers are a collection of classification algorithms based on Bayes Theorem[14]. It is a family of algorithms where all of them share a common principle, i.e., every pair of features being classified is independent of each other. The name naive is used because it assumes the features that go into the model are independent

of each other. That is changing the value of one feature, does not directly influence or change the value of any of the other features used in the algorithm. The Bayes Rule is a way of going from P(X|Y), known from the training dataset, to find P(Y|X).

- Bayes Rule,

$$P(Y|X) = \frac{P(X|Y) * P(Y)}{P(X)} \tag{4.7}$$

The Bayes rule provides the formula for the probability of Y given X. But, in realworld problems, we typically have multiple X variables. When the features are independent, we can extend the Bayes Rule to what is called Naive Bayes. Since all the Xs are assumed to be independent of each other, we can just multiply the likelihood of all the X's and call it the "Probability of likelihood of evidence" . This is known from the training dataset by filtering records where Y=c (Y is of any particular class c). After applying Naive Bayes classifier on every state in the TS, each transition is trained with epsilon-SVR using historical data considering all attributes. In a state s Naive Bayes estimates the probability of transition from s0 to s1, while epsilon-SVR predicts the remaining time if the next state will be s1. Each state is enriched with Naive Bayes Annotation and SVR annotation. Suppose the current state is s and there are 'k' exiting states from s namely s1,s2,s3.....sk. Let Pi be the NB probability of transition from state s to si and ti be the remaining time predicted by our SVR model for this transition then our prediction value for remaining time is

$$t_i = (P1 * t1) + (P2 * t2) + \ldots + (Pk * tk). \tag{4.8}$$

That is, we have taken the weighted mean of remaining times for all possible continuations from the state s where the NB probability acts as the weight.

To make prediction for any partial trace, we first find the state using the state representation function and this state is given as input to our predictor TS which predicts the remaining time.

4.5 Future Path Prediction

The model we have used in the last approach is also exploited in order to predict, for a running case, which is the most likely sequence of activities until the end of the case. Finding the full sequence of states from the current state (path) with the highest probability is not a trivial task. We face this problem as a shortest path problem in which the goal is to find a path between two vertices of a graph, such that the sum of the weights of its constituent edges is minimized. Specifically, let us consider the transition system as a directed graph: we would like to find the shortest path between the current node and an accepting node. We use Dijkstra's algorithm
for this.

4.5.1 Dijkstra's Algorithm

In Dijkstra's algorithm, we generate a SPT (shortest path tree) with given source as root[15]. We maintain two sets, one set contains vertices included in shortest path tree, other set includes vertices not yet included in shortest path tree. At every step of the algorithm, we find a vertex which is in the other set (set of not yet included) and has a minimum distance from the source. It includes the following steps:

- Create a set sptSet (shortest path tree set) that keeps track of vertices included in shortest path tree, i.e., whose minimum distance from source is calculated and finalized. Initially, this set is empty.

- Assign a distance value to all vertices in the input graph. Initialize all distance values as infinite. Assign distance value as 0 for the source vertex so that it is
  picked first.

- While sptSet doesn't include all vertices

  – Pick a vertex u which is not there in sptSet and has minimum distance

4.5. Future Path Prediction

  value.

  – Include u to sptSet.

  – Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if sum of distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

To use Dijkstra's algorithm, we need suitable weights for every transition (which act as edges in the graph) but we have probabilities which have to be multiplied to get the likelihood of a sequence but we cannot use it directly as edge cost because we need to follow the definition of the shortest path problem (i.e., edge cost have to be summed). However, we exploit the properties of the logarithm function to transform probabilities into distance like values, in particular: $log(pq) = log(p) + log(q)$, and since p is a probability: $\forall\ 0 \leq p \leq 1 \in R, log(p) \leq 0$, where low values of log(p) mean that the transition probability is low, or, from a graph view point, the distance between the nodes is high. Using this idea, we can use as edge cost the opposite of the logarithm of the transition probability. Using this transformation, we construct a graph corresponding to the TS where the shortest path problem is found using
Dijkstra's algorithm.

Chapter 5

Results

To measure and compare the accuracy, we used two indicators: the Mean Absolute Percentage Error (MAPE) and the Root Mean Square Percentage Error (RMSPE). Let n be the number of samples and let Ai and Fi be respectively the actual value and the predicted value for the i-th example. MAPE and RMSPE usually expresses the accuracy as a percentage:

$$MAPE = \frac{100}{n} * \frac{\Sigma abs(Ai - Fi)}{Ai}\% \quad (5.1)$$

$$RMSPE = 100 * \sqrt{\frac{\Sigma(\frac{Ai-Fi}{Ai})^2}{n}}\% \quad (5.2)$$

MAPE and RMPSE values for all the approaches are:

Chapter 5. Results

Method	MAPE Value	RMSPE Value
VDA	6.524	10.059
SVR	2.478	3.153

SVR+TS(seq)	3.777	8.268
SVR+TS(set)	3.805	8.264
DATS(seq)	2.524	3.091
DATS(set)	1.724	2.338

We obtained the following plots comparing all the approaches implemented by us.

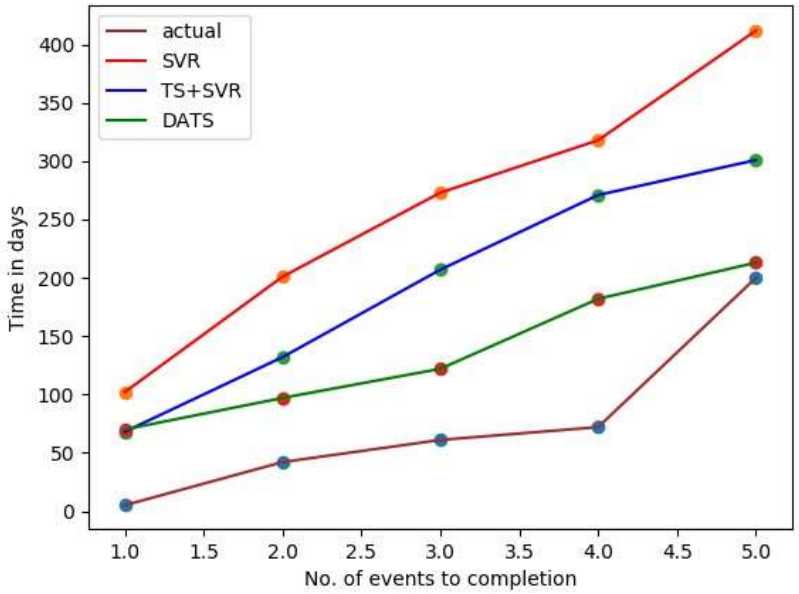

Figure 5.1 Comparison of predicted time by various approaches on a randomly selected trace

TS+SVR and DATS gave almost the same results on the multiset abstraction as they did for the sequence abstraction.

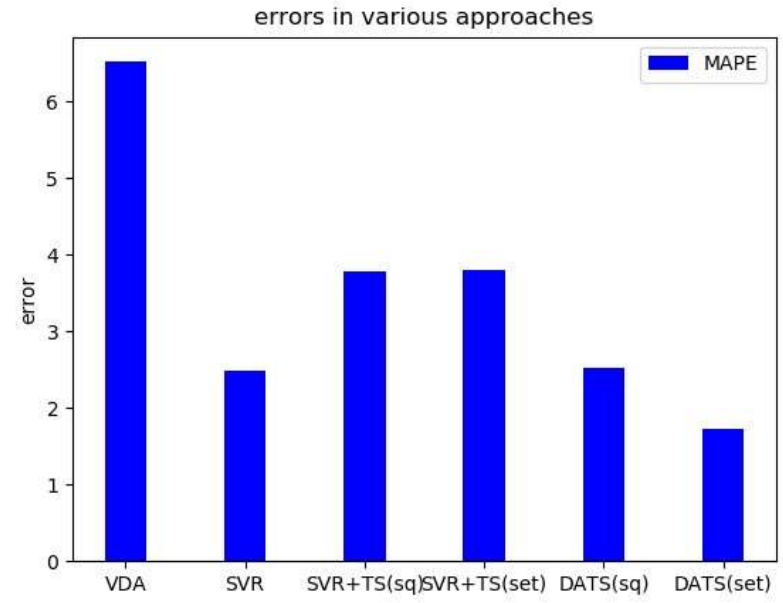

Figure 5.2 MAPE values for all the approaches

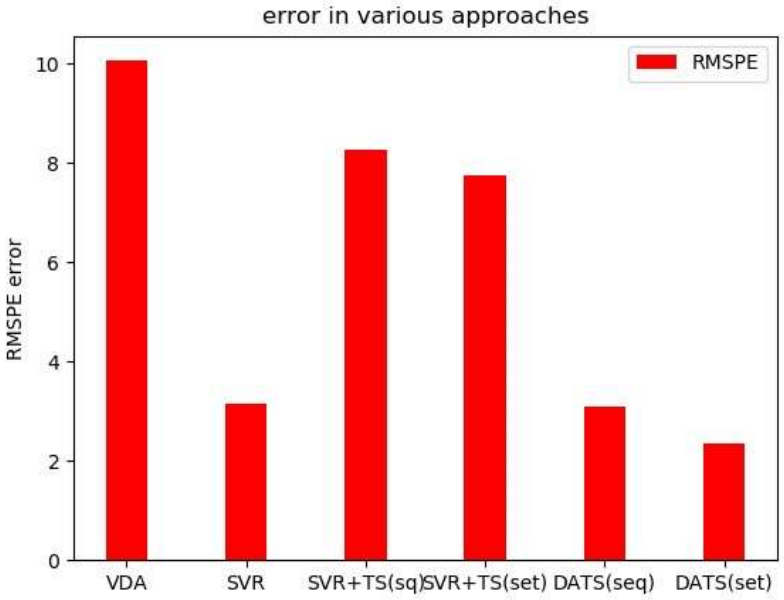

Figure 5.3 RMSPE values for all the approaches

Chapter 6

Conclusions and Discussion

6.1 Conclusion

In this project we presented 3 methods that can be employed to tackle the problem of predicting the time-to-completion of running business process instances. We presented three prediction methods, which take advantage of the additional data present in the event log. we leveraged on solid and well-studied machine learning techniques in order to build models able to manage the additional information. We constructed our approach in order to deal with unexpected behaviors or noisy data, by looking at the closeness between the new trace and the most similar process flows already observed.

By looking at the results and observations made in this project, we conclude that DATS is the best method for time prediction among the 4 approaches implemented here followed by the method (TS + SVR), SVR, VDA respectively. DATS outperforms all the other approaches because it makes use of both a classifier and a regressor, and it is able to utilize the control-flow information more efficiently.

6.2 Discussion

The proposed methods are designed to be used in different operative scenarios. Specifically, we can classify a business process on the basis of its complexity and its dynamicity over time. Anytime the process is stable and it is also not very complex (i.e., structured and with a reasonable number of activities) DATS should be preferred over the

other methods. Thanks to the TS and the information on the states/transitions, it is able to take advantage of the peculiarities of each single activity by keeping the learning time (and space) acceptable. In all other cases, i.e., complex and/or very dynamic processes, SVR based approaches should be considered. In particular, SVR+TS is a better choice in cases of dynamic but simple processes because it uses more information and, in general, should be more accurate. In the case of dynamic processes, SVR or SVR+TS can be used depending on how complex the process is. It is a matter of trade off between (expected) accuracy and training time. With particularly complex process SVR is of course the most reasonable choice.From a computational point of view, all the models once trained can be useful in an almost real time setting because prediction can be calculated in the order of milliseconds. However, the training time could take several hours.From a scalability point of view, the use of an explicit process model, such as a TS, can be an issue. In particular, by using a TS, the number of states can easily grow and this can cause the following problems: (i) for DATS, the number of prediction models that have to be trained could be too high and the training time could be not reasonable; (ii) for SVR+TS, the number of features are way greater than the number of training examples which can cause a drop in the performance. It is always possible to limit the number of states of a TS by using a horizon in the abstraction, but we have to be careful in order to not underfit the real process model. However, we think that in most of the real world business processes it is possible to have a reasonable TS without overgeneralizing the actual model.

Chapter 7

References

[1]Process Mining- https://medium.com/@pedrorobledobpm/process-mining-plays-anessential-role-in-digital-transformation-384839236bbe

[2]Transition System https://en.wikipedia.org/wiki/Transition_system

[3]Set- https://en.wikipedia.org/wiki/Set_(mathematics)

[4]Multiset- https://en.wikipedia.org/wiki/Multiset

[5]Sequence https://en.wikipedia.org/wiki/Sequence

[6] M. Polato, A. Sperduti, A. Burattin, M. de Leoni, Time and Activity Sequence Prediction of Business Process Instances(24-2-2016).

[7] numpy - https://www.numpy.org/

[8]Scikit-learn- https://en.wikipedia.org/wiki/Scikit-learn

[9]CSV- https://docs.python.org/3/library/csv.html

[10]Datetime- https://docs.python.org/3/library/datetime.html

[11] W. M. P. van der Aalst, H. Schonenberg, M. Song, Time prediction based on process mining, Information Systems 36 (2) (2011) 450âĂŞ475.

[12] Rakshith Vasudev, What is One Hot Encoding? Why And When do you have to use it?(3-08-2017), https://hackernoon.com/what-is-one-hot-encoding-why-and-whendo-you-have-to-use-it-e3c6186d008f.

[13] Indresh Bhattacharyya, Support Vector Regression Or SVR(29-01-2018), https://medium.com/coinmon vector-regression-or-svr-8eb3acf6d0ff.

[14] Naive Bayes, https://www.machinelearningplus.com/predictive-modeling/ hownaive-bayes-algorithm-works-with-example-and-full-code.

[15]Dijkstra' Algorithm, https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithmgreedy-algo-7/

Appendix

The following is the link to all the python3 code implemented by us in this project : Check