Predicting Next Touch Point in a Customer Journey: a Use Case in Telecommunication

Marwan Hassani Stefan Habets
Department of Mathematics and Computer Science
Eindhoven University of Technology, The Netherlands
m.hassani@tue.nl s.habets@student.tue.nl

ABSTRACT

Customer journey analysis is rapidly increasing in popularity, as it is essential for companies to understand how their customers think and behave. Recent studies investigate how customers traverse their journeys and how they can be improved for the future. However, those researches only focus on improving the process for future customers by analyzing the historical data. This research focuses on helping the current customer immediately, by analyzing if it is possible to predict what the customer will do next and accordingly take proactive steps. We propose a model to predict the customer's next contact type (touch point). At first we will analyze the customer journey data by applying process mining techniques. We will use these insights then together with the historical data of accumulated customer journeys to train several classifiers. The winning of those classifiers, namely XGBoost, is used to perform a prediction on a customer's journey while the journey is still active. We show on three different real datasets coming from interactions between a telecommunication company and its customers that we always beat a baseline classifier thanks to our thorough pre-processing of the data.

I. Introduction

Nowadays, companies collect all sorts of data from their services and customers. Altogether, data-driven analysis has become more interesting for companies and the collected data is used by companies to analyze all of their products and services. In organizations, such as a telecommunication companies, the data is used to analyze the journey of a customer. Investigating and analyzing often reveals ways to potentially optimize services provided. This investigation is both in favor of the customer and the company, as the service is improved the customer will get a more tailored approach. For example, a customer that can install his newly received modem on his own or with the help of the website and does not have to call the service desk or worse, need a mechanic. Those customers save the company money because the service desk has less work or no mechanic has to be send, decreasing the overall expenses. Also the customer is happier since he does not have to wait in the phone queue of the service desk or wait at home to receive a mechanic, increasing the customer satisfaction.

One kind of such data is the customer journey, the customer journey represents the steps a customer takes with the company. Each step is called a touch point and is defined to be an interaction from the customer with the companies' products or services [3]. These customer journeys are mapped into a customer journey map (CJM) to perform analysis on. The customer journeys are very interesting to analyze since the company can see how customers actually behave. The company has an idea of how a certain process should work, but in practice this might not be the case at all. Reviewing the customer journeys gives insight in how customers follow certain flows. When an important step is missing from the process flow in a majority of the customer journeys then the company can investigate this matter and see why the customers do not perform those steps.

Before applying process mining to it, customer journey analysis was performed to improve processes in hindsight [10]. Historical data is checked and used to iteratively improve the customer experience when interacting with a certain process of the company. Therefore the company is always late in providing immediate support to the customer. When something goes "wrong", the mistake can later be found in the data to help improve the error for the future.

Knowing what a customer will do next, gives the company the ability to proactively provide support to the customer. Helping the customer proactively saves time and therefore also costs, as well as increases the customer satisfaction as he/she is helped faster. This leads to the following research question defined for this paper: Is it possible to predict a customer's next touch point by starting from the historical data from customer journeys in our use case?

This paper is structured as follows: Preliminaries and some existing solutions are discussed in Section II. In Sections III and IV an understanding of respectively the business and the data is shared. The data pre-processing and the applied models are discussed in Section V. The experimental results are discussed in Section VI. Finally, Section VII concludes this paper.

II. Preliminaries and existing solution

The data consists of customer journeys in the form of event logs. An event log $L = (Tr_1, Tr_2, \ldots, Tr_n)$ consists of a collection of n traces. A trace is a sequence of events $Tr_i \in E^*$, where E is a collection of events. An event $e_i = (c_j, a_k, t_i)$ needs to include an uniquely identifiable customer journey c_j, an activity a_k from

Communications of the ECMS, Volume 35, Issue 1,
Proceedings, ©ECMS Khalid Al-Begain, Mauro Iacono,
Lelio Campanile, Andrzej Bargiela (Editors)
ISBN: 978-3-937436-72-2 / 78-3-937436-73-9(CD) ISSN 2522-2414

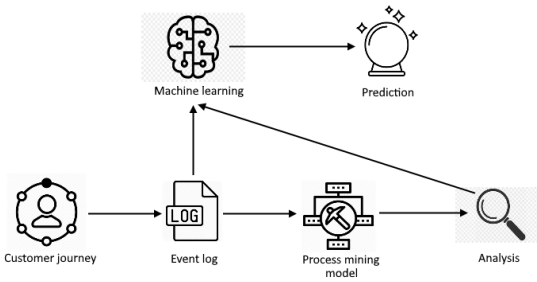

Fig. 1. Methodology overview

the set of l possible touch points $A = \{a_1, a_2, \ldots, a_l\}$ and lastly a timestamp t_i when the event took place [18]. Thus an event log has a collection of traces, which contain events that denote the activity that happened on what time and referring to what customer journey.

We want to predict the activity a of event e_{i+1} given that we are at event e_i belonging to the same customer journey c_j and having the timestamp $t_i < t_{i+1}$. We also have the information of previous events which will help determine the next touch point. Meaning that e_{i+1} is the dependent variable on our independent variables of the previous steps. We do not only use touch points, but also the additional *static* information in the customer journey.

In Figure 1 we can see the steps that can be used to perform predictions on the customer journey.

The main goal is to try and predict what touch point a customer will use next. This prediction is required if the company wants, for instance, to try preventing this next step from happening. This can be done by helping a customer proactively or even better by making sure that the next step is never needed. This does not only save the company resources as the customer needs less attention, the customer will also be more satisfied as the goal of the journey is reached faster.

One example of solutions from the literature addressing relatively similar problems is OARA [6] which this is based on customer journey prediction but also includes a recommendation afterwards, which deviates from the goal in our research. A similar solution is proposed by Terragni & Hassani [17] in an article on analyzing customer journeys for recommendations [16]. This research is useful to investigate as the basis is similar. However we want to predict what a customer will actually do and not what a customer will like (also called the final outcome). Therefore we can investigate their sources and analyze how they tackled the problem and learn from that.

III. Business understanding

The business owner *BO* which owns the scenario and the data preferred not to reveal their identity and is a market leader for telecommunications in the Netherlands. We will refer to them as *BO* in the remainder of this paper. They provide many different types of services for people living in the Netherlands, as well as different services for companies in the business market. The main services provided for customers are mobile telephony, internet for households and interactive television. The different types of services can be combined to get extra benefits on top of the packages. *BO* provides various ways to get in contact with their customers.

A customer can get into contact with *BO* for different reasons such as having a question about a service, an invoice, the installation of a new piece of hardware, reporting malfunctions, acquiring a new subscription with *BO* and a dozen more possible reasons. To facilitate these contacting customers, *BO* has a number of channels to reach them. Customers can contact the service helpdesk of *BO* over the phone, the website, a list of FAQ, a community forum, social media platforms or offline by physically showing up in one of *BO* stores.

A. Customer journey

The customer journey at *BO* is defined to end after a customer has not been in contact with *BO* for at least seven days. Thus, a customer journey can be of arbitrary time duration, the only restriction is that the duration between two consecutive touch points should not be longer than 7 days, otherwise it is assumed to be a new customer journey (new case).

The customer journey is a customer-driven process that starts with a touch point initiated by the customer to interact with *BO*. Most of the customer journeys are quite short, meaning of length one, two and three. The short length is actually a good sign for *BO* because it means that the customer is helped within a few steps.

B. Business Problem

Regarding customer support, there are three high cost factors within *BO*. As such those three are interesting to investigate and in context of this research, they are interesting to predict. Ideally, a customer would never have to call *BO*. All changes, sales, terminations and troubleshooting problems are being resolved using the online portal of *BO*. *BO* offers customer support through their telephone service helpdesk. The service helpdesk can be used in case of a question which could not be answered by the *BO* website and there are always people who prefer to call instead of using self-service options. However keeping the call centre operational is quite expensive. *BO* provides around the clock support for malfunctions, technical support and questions about the theft or lost of a mobile phone. Every call made costs *BO* money, this has already led to the creation of self-service portals online and robotic chat services.

Another big impact factor regarding cost is the mechanic. Sending a mechanic to a customer is costly for both *BO* as for the customer. Mechanics need proper training and planning the mechanics in a way that travel time is minimized is hard and thus costly. The mechanic is unfortunately partly unavoidable, as he needs to fix issues on the customer side. Though there is also a part which is unnecessary, it is hard to distinguish the latter from the former. Improving the quality of manuals might reduce the required number

of mechanics but they will always be needed. Even though, predicting that a mechanic will be needed can certainly help. As preemptive steps can be taken to offer a customer to receive a mechanic. Doing so will save the cost of a customer having to make a repeated call to *BO*.

Last costly defect is swapping of hardware for the customer. When a modem or tv-box has a defect and needs to be replaced, it creates a lot of administration and logistics. Especially keeping track of all aspects of the swap in the logistics is not a trivial task. Even though a large part of the modems returned are not actually broken. The customer will send the malfunctioning modem back to *BO* and *BO* has to send the same model modem from their warehouse to the customer.

These three costs are part of the customer journey and loads of different kinds of research is being done to better understand and improve them. The main costs for these three touch points is the overhead, they are often unnecessarily repeated. The costs of sending two mechanics is substantially higher than one mechanic who spends a bit more time at a customer. Not only those three items are investigated, the whole customer journey is under the loop and is being streamlined more and more.

The objective for *BO* is to resolve the issue of the customer within one contact, as all repeating touch points are considered excessive. To achieve this goal and besides the answer to the first sub-question is to get a prediction on what type of contact (i.e. touch point) a customer will use next and if possible also the subject for which the customer comes into contact with *BO*. As it can make quite a difference if the customer calls for explanation on his bill or to cancel their subscription.

IV. Data understanding

For this research we have two datasets with eight weeks of data, which both already include over a million rows of data and almost half a million customer journeys. The column called *contact_type* is the type of contact a customer has with *BO*, this attribute is what we have defined as a touch point in this paper. Meaning this is the attribute on which we want to do the prediction, we want to predict which type of contact the customer will use next. Therefore all the distinct touch points will be looked at and briefly explained in the list below.

A. Touch Points

- **call**: This touch point means that a customer has called with the service helpdesk of *BO*. The reason behind the phone call can vary a lot, from a malfunction to the theft of a mobile phone.
- **call - dvb**: This is when a call has to be forwarded to another department of the service helpdesk where employees are trained in other skills. For example if a customer wants to buy a product or service, he will be put through to the sales department of the helpdesk.
- **chat**: The chat occurs when a customer is on the *BO* website and uses the online portal to ask a question to the chatbox. First a bot will respond but later a real life employee may continue the conversation if necessary.
- **conversational**: Conversational is when a customer calls the service helpdesk of *BO* but before a real employee is on the phone. A bot will ask the customer to state his question, the bot then tries to classify the question. If the question is general, the bot will send a link to the webpage on which an answer for the question can be found. Conversational helps to reduce the number of calls that have to go through to actual employees.
- **logistiek**: The logistics part is for the swap and distribution of hardware. This is a more static step in the process, as a type of hardware is requested and the warehouse has to perform the logistics to get it to the customer.
- **monteur - levering**: This means a mechanic for delivery. The delivery to a customer who has gotten a new subscription or an upgrade and wants a mechanic to perform the installation of the new modem or box.
- **monteur- ondergrond**: This stands for mechanic - underground. Meaning that there has to be done actual digging or crawling in the crawlspace of the house. The mechanic will then check and replace the cable(s) if necessary. A mechanic - underground is not often needed and mostly only after a regular mechanic - service has come by the house already.
- **monteur - service**: This is the regular mechanic for service. The mechanic is send when a customer calls with a malfunction which cannot be solved by himself or over the phone with the service helpdesk employee.
- **online**: Online is when the customer checks the website of *BO*. This can be for everything on there, even if it is just browsing. Online can be hard to link to the actual customer as most people are not always logged in into their account.
- **order**: In this dataset an order has two categories, namely move and termination. Termination is when a customer cancels his subscription, so the provided services have to be stopped. Move is when a customer changes address and therefore the services, like internet and TV, have to be changed to the new address as well.
- **service ticket**: The service ticket is also like the logistics an intermediate, more static step. The ticket is created by a service helpdesk employee who has a customer with a problem on the phone. The service ticket states all the information needed for later reference if the customer calls again or for the mechanic to be informed about the problem.
- **winkel**: Winkel is dutch for store. So the store is when a customer walks into the store and speaks with an employee. The reason can also vary, it can be to buy a new product or to ask for information or even to report a malfunction.

The next column after *contact_type* differs in the two datasets. The key feature of the first dataset is an important column named *bucket_name*. In this column the bucket in which the touch point is categorized is shown. Not all touch points use the same buckets to

Baseline of touch points	
eind	48%
call	21%
conversational	13%
online	7%
logistiek	3%
service ticket	2%
monteur - service	2%
order	1%
winkel	1%
chat	1%
monteur - levering	0%
monteur - ondergrond	0%

TABLE I: Baseline touch points in the data

be categorized into, which is interesting. We will check which touch points are connected to which bucket.

Touch point occurrence In Table I, the frequency of each touch point in the dataset is shown. All touch points were counted and then normalized, which is the result visible in Table I.

Process overview The journeys are very different for each customer but it could be the case that multiple customers have the same journey as there is only a limited number of touch points. In Figure 2 the process overview, made by the heuristic miner . In this figure we do see some connections. We observe on the left of the figure that a **call** often leads to an **order**, **service ticket** or **logistiek** (logistics). These are interesting observations and when thought about make sense. When a customer calls *BO* it can be about a malfunction or other question, this leads to a **service ticket**. The call can also be about acquiring a service, thus **order**. Moreover it could be a defect piece of hardware, leading to **logistiek** (logistics).

From the **service ticket** a followup is the **monteur - service** (mechanic - service), which is expected. The same holds for an **order** inducing the **monteur - levering** (mechanic - delivery). When a **monteur - service** is not enough to fix the problem a **monteur - ondergrond** (mechanic - underground) is sent, as already stated in the explanation of **monteur - ondergrond** in Section IV-A. So this dependency is also no surprise to see.

The link between **monteur - ondergrond** → **logistiek** and **monteur - ondergrond** → **conversational** is not immediately clear. Possibly for **monteur - ondergrond** → **conversational** the customer calls to ask for an update while the **monteur - ondergrond** is still working on the problem.

B. Distribution of journey types

In the second dataset, we looked at the distribution of the journeys regarding how many types one journey includes, which can be seen in Table II . Meaning in this dataset each customer journey has a category which is one out of the ten types of journeys. These tables show how many different journey types are included in one customer journey. We see that when ignoring customer journeys of length one, the biggest value is when

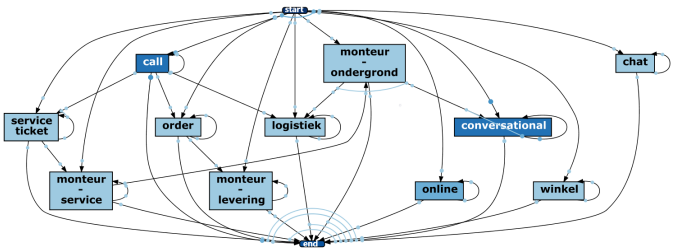

Fig. 2. Process overview using heuristic miner

Number of types	Percentage	Absolute
1	39%	55914
2	40%	57532
3	14%	20731
4	5%	6752
5	1%	2078
6	0%	539
7	0%	91
8	0%	15
9	0%	1

TABLE II: Journey distribution with $length > 1$.

there are two journey types in one customer journey. This is closely followed by just one type per journey. More than two types per journey has a much smaller percentage. Another interesting observation is the fact that there are a lot of customer journeys of length one, there are almost a 100k of them.

V. Data preparation and models

A. Predicted and independent variables

In this research the predicted variable is the data in the column *contact_type_next*. For the independent variables X_i the columns with information have to be selected. For the customer journey this will always be the column containing the touch points, as the touch points are the most important feature in a customer journey. Besides the touch points some additional information regarding the reason of the contact can be added, like a callreason or category. If it would be helpful then also customer information can be added to provide more personal information to the model.

In our research the most important feature is the touch point, i.e. the column *contact_type*. This column contains the current touch point in the journey and has the same values as the outcome variable minus **eind**. Furthermore we also want to include previous touch points belonging to the current journey.

To provide meaning to the touch points, the column with *bucket_name* is used. The bucket provides a sort of category for the current touch point. As we also include previous touch point, we will likewise include the previous buckets in our data.

Another data feature which supports the information surrounding the touch point is the *callreason*. As described previously the *callreason* consist of a written or selected reason made by a *BO* employee. As such there occur many different variations in *callreasons* of

which some are very similar to each other. They can even be as similar as using different punctuation or capital letters. This makes them less suited for the use in prediction, however they contain valuable information. This is why we will perform a basic cleaning performance on the *callreason* data, so it is usable for our models. Because there are still many options available for *callreason*, we will only use the directly previous *callreason* and the current *callreason*.

B. Data cleaning

To reduce the noise in our data we have removed the touch point **call - dvb**. We simply do this by removing the entire row in which **call - dvb** is the *contact_type*, because we will run a script later that reshapes the data in the right way. We made the decision to limit the *callreason* to a maximum of ten characters. We chose ten because it cuts off all too specific parts of the reason while still providing enough room within the first ten characters to be different. Lastly we had the buckets *undefined*, *nog niet toebedeeld* and *unknown*. These are all non-informative buckets and therefore we aggregated them into one bucket instead of three separate buckets. We looped over the data and whenever we encountered one of the three buckets, we replaced the entry with *missing*.

We will use a logistic regression, random forest, boosted trees and a LSTM neural network. Then we will measure the performance of the models with a metric and assess them.

C. Applied models

Logistic regression: The logistic regression is the most basic technique we use. We only have to set a few parameters for this model. As our problem consists of predicting one of multiple outcomes, we need a multi-class classification model. Therefore we have to set the *multi_class* parameter in our logistic regression to *multinomial*, then it will use cross-entropy loss to find the best model. The other parameter we set is the solver, we cannot use liblinear for our multi-class problem as this is only suited for binary problems. We choose for the SAGA solver [5]. This solver performs well in practice and is faster on large datasets than other solvers. We will use these settings to train our logistic regression model.

Random forest: Thousand trees are enough to get an average and reduce overfitting. To further control overfitting we set the *max_depth* parameter to 25.

XGBoost: XGBoost [4] is also based on decision trees. For XGBoost we have to choose the objective function to perform the gradient boosting on, in our case we choose for the *multi:softprob* option. This indicates that we are dealing with a multi-class output and the softprob refers to the softmax function. Instead of returning the label, it returns the probability for each output. Similar to the random forest, here we also choose a *n_estimators* of a thousand and a *max_depth* of 25 to help with memory management and overfitting.

LSTM [9]: Hyperparameter tuning is very important for neural networks. There are a few ways to do the hyperparameter tuning. We can manually tune the hyperparameters but this is very time consuming and you need an expert or else the tuning will not be much of an improvement. The most standard option in parameter tuning is gridsearch. Gridsearch evaluates all the different possible combinations of parameters in a grid-like manner, therefore it is called a gridsearch. However testing all different possible combinations of parameters and finding the best combination takes a lot of computational power and time. This is why a randomized gridsearch was introduced, the randomized gridsearch does not compute all possibly combinations but it randomly chooses a subset of them. By Bergstra and Bengio [1] it is empirically and theoretically shown that randomly chosen trials are more efficient for hyperparameter tuning than trials on a grid.

However there is also a disadvantage in the randomized gridsearch. Randomized gridsearch does not adapt its behavior based on the previous outcomes. This means that a poorly chosen parameter can prevent the model from learning effectively. For example if the dropout rate should be between 0 and 0.5 but we test for values between 0 and 1 then 50% of the tests will return bad results. This is an unnecessary waste of time and therefore the range in which the hyperparameters lie, needs to be chosen carefully. The Bayesian optimization methods by Snoek et al. [13] are capable of learning from the previous trials. Bayesian optimization creates a surrogate objective function to approximate the best hyperparameters for the real model. A study by Bergstra et al. [2] shows that Bayesian optimization methods produce significantly better results whilst also limiting the computation time. Therefore we will use Bayesian optimization in this research. We will now shortly discuss the different hyperparameters we will tune.

- **Gradient descent optimization algorithm**: This optimization technique tries to minimize the loss function after each iteration by tweaking the weights. Some of these optimization algorithms are Momentum, Adagrad, RMSprop, Adam and Adamax [11]. Adam is in general the best performing optimizer [11].
- **Number of neurons in hidden layer**: The number of neurons in the hidden layer determines how well the model learns without underfitting or overfitting.
- **Dropout**: Dropout is a regularization method which drops out random nodes to reduce overfitting and improve overall model performance.
- **Batch size**: The batch size defines the number of samples that will be used every iteration. This hyperparameter is also a balance between not overfitting the model and unable to escape a local minimum. In general it is advised to use a power of two as batch size since this would increase efficiency.
- **Epochs**: The number of epochs is the number of times your model trains on the entire dataset. If this number is too high it will cause overfitting on the opposite side if it is too low then there will be underfitting.

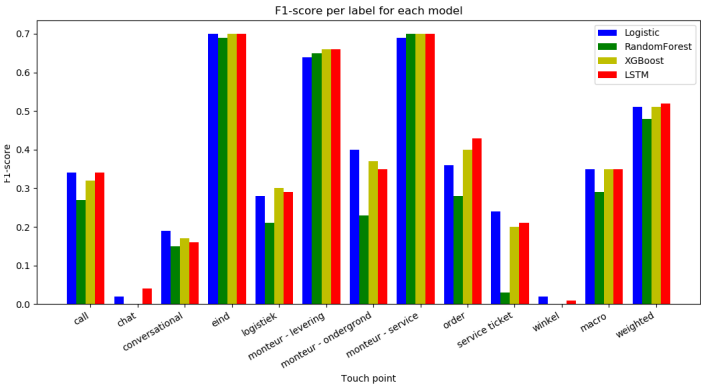

Fig. 3. F_1 of all models w.r.t. the ground truth touch point.

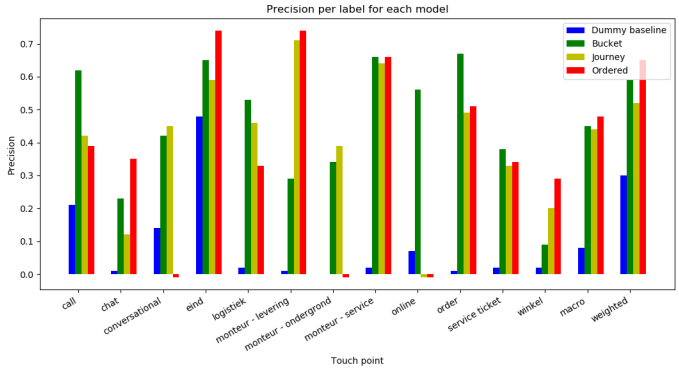

Fig. 4. Precision comparing dummy and three datasets

VI. Experimental Evaluation

To evaluate the prediction quality we used: Precision $= \frac{True\ Positives}{True\ Positives + False\ Positives}$, Recall $= \frac{True\ Positives}{True\ Positives + False\ Negatives}$ and $F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$.

We have trained the four different models on the two different datasets which were split into three different options. The datasets are split individually into an 80% training set and a 20% testing set .

A. Comparison of applied models

First we will compare the four different kinds of models with each other. For this we show a bar plot depicting the F_1-score for the different models in Figure 3. On the x-axis we plotted the predicted labels, i.e. the touch points. Also a macro average and a weighted average is shown. To compare the different models, it is best to look at the macro average and weighted average. Overall we see that the macro average is equal for the logistic, XGBoost and LSTM models with only the random forest (RF) underperforming. The same observation holds for the weighted average. The models perform similar with only RF underperforming. The reason that RF is worse than the other three could be caused by the fact that we did not tune the parameters of the RF, while we did tune the LSTM and XGBoost boosts itself.

B. Comparison of datasets

In this section we will compare three different types of datasets using the winner model from the previous step: XGBoost. The first is the first dataset used in this research, which we call *buckets*. The second and the third datasets are inferred from the second dataset in this research and differ according to the journey types which are used in two ways. The first way is by just using the dataset trained on all data in a journey, so only sorting by journey id (we call it *journey*). In the other way, we group the journeys on journey id and also on journey type (we call it *ordered*). This creates more journeys and therefore also smaller journeys. However these journeys should all be related to the same subject as they share the same journey type. The precision, recall and F1-score are shown in the Figures 4, 5 and 6 respectively. In the figures the metric is on the y-axis and the different touch points on the x-axis. The bars are the three different datasets as well as a dummy baseline for reference. A dummy baseline randomly predicts one of the labels but with a probability weighted by its relative frequency in the ground truth. For some datasets certain touch points are not available and this is displayed by a small negative value, for example for the ordered dataset there is no **conversational** label. First we will inspect the overall score with the F1-score measure shown in Figure 6. Comparing the macro average and weighted average, we see that on all three datasets, XGBoost outperform the dummy baseline which is good. For the macro average, XGBoost performs the best on bucket dataset and worst on the ordered dataset. While on the ordered dataset it performs the best in the weighted average, on the bucket dataset it also performs well.

We observe two touch points that are very poorly predicted. These touch points are **chat** and **winkel** (store). This could be explained by the fact that these are two of the smaller labels and therefore less tuned on by the models. However thinking about these touch points, they both do not belong to clear processes. A customer can go to the store whenever he wants but he is never expected to go to a store. This makes the store a very unpredictable touch point. The chat has the same issue only when we see a customer online, we could predict that he is going to chat. However there is never a clear indication that the customer will chat with a *BO* employee.

Now we will look at the touch points most interesting to us, namely **call** and **monteur - service**. We observe that **monteur - service** is very good predictable in all datasets. This is likely caused by the fact that a mechanic for service is always sent in a reaction to something and it is not sent out of the blue. The indications are used by the model to predict when a mechanic will be sent. Looking at the **call** touch point only the bucket dataset performs well and the ordered dataset performs very poorly. The poor performance of the ordered dataset could be caused by the fact that there is no **conversational** data in this dataset and **conversational** is one of the biggest indicators that a **call** might follow.

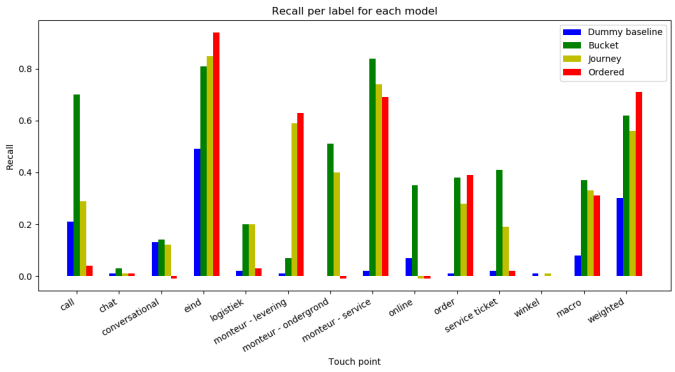

Fig. 5. Recall comparing dummy and three datasets

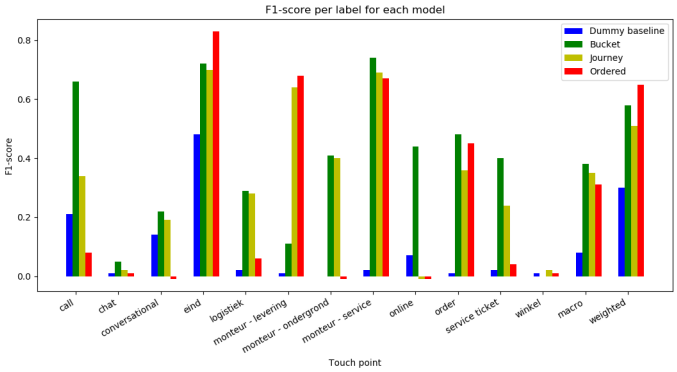

Fig. 6. F_1-score comparing dummy and three datasets

VII. Conclusions

In this paper we have discussed the customer journey and the predictability of its different touch points. First we want to get a better overview of the customer journey. Therefore we started this research by investigating the customer journey. We applied process mining techniques to discover the process model. Then we moved towards predicting the next step in the journey. We have shown the intuition behind each preprocessing step either from the business understanding perspective or from the data analysis perspective. We concluded that among four, carefully-tuned prediction models, XGBoost was the winner so we proceeded with testing it on three datasets. In the results we have shown that we are always able to beat a dummy baseline which predicts randomly one of the labels with a probability weighted by its existence in the ground truth. To have a structured approach to our investigation, we followed a framework similar CRISP-DM [12].

In the future, we would like to address the meaning of the used distance metric in categorizing the journeys into variants by inferring an accurate distance metric that decides the similarities between the journeys [15]. Additionally, we would like to address the predictability under a streaming setting of the customer journeys [8], with varying underlying distributions [7], [14].

References

[1] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *JMLR*, 13(Feb):281–305, 2012.

[2] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *NeurIPS*, pages 2546–2554, 2011.

[3] Gaël Bernard and Periklis Andritsos. A process mining based model for customer journey mapping. In *CAiSE 2017*, pages 49–56.

[4] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *KDD*, pages 785–794. ACM, 2016.

[5] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *NeurIPS*, pages 1646–1654, 2014.

[6] Joël Goossens, Tiblets Demewez, and Marwan Hassani. Effective steering of customer journey via order-aware recommendation. In *ICDMW'18*, pages 828–837.

[7] Marwan Hassani. Concept drift detection of event streams using an adaptive window. In *ECMS*, pages 230–239, 2019.

[8] Marwan Hassani, Daniel Töws, Alfredo Cuzzocrea, and Thomas Seidl. *BFSPMiner*: an effective and efficient batch-free algorithm for mining sequential patterns over data streams. *Int. J. Data Sci. Anal.*, 8(3):223–239, 2019.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[10] Katherine N Lemon and Peter C Verhoef. Understanding customer experience throughout the customer journey. *J. of market.*, 80(6):69–96, 2016.

[11] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[12] Colin Shearer. The CRISP-DM model: the new blueprint for data mining. *Journal of data warehousing*, 5(4):13–22, 2000.

[13] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *NeurIPS*, pages 2951–2959, 2012.

[14] Yorick Spenrath and Marwan Hassani. Predicting business process bottlenecks in online events streams under concept drifts. In *ECMS*, pages 190–196, 2020.

[15] Yorick Spenrath, Marwan Hassani, Boudewijn F. van Dongen, and Haseeb Tariq. Why did my consumer shop? learning an efficient distance metric for retailer transaction data. In *PKDD*, pages 323–338, 2020.

[16] Alessandro Terragni and Marwan Hassani. Analyzing customer journey with process mining: From discovery to recommendations. In *FiCloud '18*, pages 224–229.

[17] Alessandro Terragni and Marwan Hassani. Optimizing customer journey using process mining and sequence-aware recommendation. In *SAC'19*, pages 57–65.

[18] Wil M. P. van der Aalst. *Process Mining: Data Science in Action*. Springer, April 2016.