UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Software Engineering Curriculum

Brandon Christopher Autrey

Customer Journey Analysis at Pipedrive: A Process Mining Approach

Master's Thesis (30 ECTS)

Supervisor(s): Marlon Gerardo Dumas Menjivar
Ambar Raj

Tartu 2021

Customer Journey Analysis at Pipedrive: A Process Mining Approach

Abstract: Pipedrive is a company that provides Customer Relationship Management (CRM) with a Software as a Service (SaaS) business model. One of the challenges at Pipedrive is to enhance customer experience in order to convert as many trialled users into payed users and to prevent churned of paid users. In this thesis, we analyse the customer journey of users of Pipedrive SaaS system from the moment they create the trial account to the moment they convert into paid users or they abandon. To analyse this problem, we use process mining approach to extract high fidelity event logs from different information systems and use techniques from the field of process mining in order to identify patterns that distinguish between trialled customers converting to paid product and those who do not convert.

Keywords:
process mining, event logs, customer journey map

CERCS: P170, Computer science, numerical analysis, systems, control

Kasutaja teekonna analüüs Pipedrive'is: protsessikaevet kasutades

Lühikokkuvõte: Pipedrive on ettevõte, mis pakub kliendihaldustarkvara, kasutades tarkvara kui teenus ärimudelit. Üks väljakutseid Pipedrive'is on tugevdada kasutajakogemust, eesmärgiga muuta testperioodi lõpus kasutajad maksavateks kasutajateks ning ennetada maksvate klientide lahkumist. Käesolevas lõputöös uurib autor kliendi teekonda Pipedrive'i tarkvara kui teenus süsteemis, alates hetkest, kui klient loob testperioodiga kasutaja, kuni neist saab maksev klient või kuni nende lahkumiseni. Probleemi analüüsimiseks kasutame protsessikaevet, et eraldada kõrge täpsusega sündmuse logisi erinevatest informatsiooni süsteemidest ning kasutame protsessikaeve tehnikaid, eesmärgiga tuvastada mustreid, mis eristavad testperioodi kasutajaid kes muutuvad maksvateks kasutajateks, nendest kes ei muutu.

Võtmesõnad:
protsessikaeve, sündmuse logi, kasutaja teekonna kaart

CERCS: P170, Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Acknowledgement

I wish to express my sincere appreciation to my supervisor, Professor Marlon Dumas, who has guided and encouraged me throughout the process and provided tremendous feedback and suggestions. Thank you for providing me with an Enterprise version of Apromore to use in this thesis. Without his help, the goal of this project would not have been realized.

I wish to thank my co-workers at Pipedrive, who have provided support throughout this project. Thank you, Ambar Raj for being the Product Manager who helped realize the idea for this thesis and for the constant support. I thank the Product Managers Madis and Evelin for providing feedback. I also thank Data Analysts Veronika and Tiina, who supported my many questions about how to work with the data and solve problems. I finally thank Kieren for proofreading this thesis and approving that this thesis meets the security requirements of Pipedrive to be published.

I wish to show my gratitude to my lovely wife, who has provided love and support at home. I thank my mother for honing my English skills and proofreading this thesis multiple times throughout the process.

Contents

List of Figures

List of Tables

Listings

List of Acronyms

2FA Two Factor Authentication.

AWS Amazon Web Services.

B2B Business-to-business.

BPM Business Process Model.

CJM Customer Journey Map.

CRM Customer Relationship Management.

CSV Comma-Separated Value.

DW Data Warehouse.

ID Identifier.

IEEE Institute of Electrical and Electronics Engineers.

LR Logistic Regression.

NPS Net Promoter Score.

PII Personally Identifiable Information.

PM Product Manager.

PM^2 Project Management for Process Mining.

RF Random Forest.

S3 AWS Simple Cloud Storage.

SaaS Software as a Service.

SHAP Shapley Additive exPlanations.

SQL Structured Query Language.

SSO Single Sign-On.

T2E Trial-to-Expire.

T2P Trial-to-Pay.

UI User Interface.

XES eXtensible Event Stream.

XGB XGBoost.

Glossary of Terms

Apache Spark - an open source parallel processing framework for running large-scale data analytics applications across clustered computers. The primary programming language Spark is written in Scala.

Application Performance Monitoring - a tool used by businesses to observe whether their apps are behaving normally, alerting and collecting data if something happens, analysing the data in context of the impact of the business, and adapting the application environment to fix similar problems before they impact the business [1].

Business Intelligence reports - broadly defined as the process of using a Business Intelligence tool to prepare and analyse data to find and share actionable insights. In this way, Business Intelligence reporting helps users to improve decisions and business performance.

Column - a set of data values of a particular type, one value for each row of the database.

Event Log - the byproduct of process participants working on tasks. They can reveal important insights into how a process works in reality.

Listing - a printed list of lines of computer code.

PostgreSQL - a free and open-source relational database management system emphasizing extensibility and SQL compliance.

Process - an explicitly defined series of steps to perform an overall goal.

PySpark - a Python API for Apache Spark that lets you harness the simplicity of Python and the power of Apache Spark in order to tame Big Data [2].

Python - an interpreted, object-oriented, high-level programming language with dynamic semantics [3].

Relational Database - a type of database that stores and provides access to data points that are related to one another.

Row - a structured data record within a table.

Schema - the organization of data as a blueprint of how the database is constructed.

Table - a collection of related data held in a table format within a database. It consists of columns and rows.

Timestamp - a digital record of the time of occurrence of a particular event.

[1] https://www.appdynamics.com/blog/product/application-performance-monitoring/
[2] https://spark.apache.org/docs/latest/api/python/index.html
[3] https://www.python.org/doc/essays/blurb/

1 Introduction

1.1 Pipedrive

Pipedrive [4] was founded in 2010 by seasoned sales professionals. They realized that the Customer Relationship Management (CRM) landscape was populated by software designed to please the management while ignoring the needs of the people doing the actual selling. They designed a simple but powerful tool to visualize your sales process and follow-up with the right activities, so your efforts become more effective and your close rate increases. Pipedrive has 10 offices in 8 countries, 700+ employees, over 95,000 customers representing over 150 countries. The product line is mainly one web app, which contains all of the business processes in a streamlined view. Pipedrive focuses your mind and efforts on the pipeline so you can see where you are and what you need to do next. No time is wasted on administration, your emails, and calls, and progress is tracked automatically, so you can focus on doing what you do best. Because the pipeline is your starting point, Pipedrive keeps your actions on-point, not off-target, and your goals realistic.

Figure 1 shows the Deals page, which consists of the Pipeline to manage the deals. This is the first page when logging in. The left bar is the navigation where common functionalities are located, such as Leads, Deals, Mail, Activities, Contacts, Insights, Products, and More. On the top is the Search bar to find objects or to create new objects easily. The Settings button leads to more advanced actions.

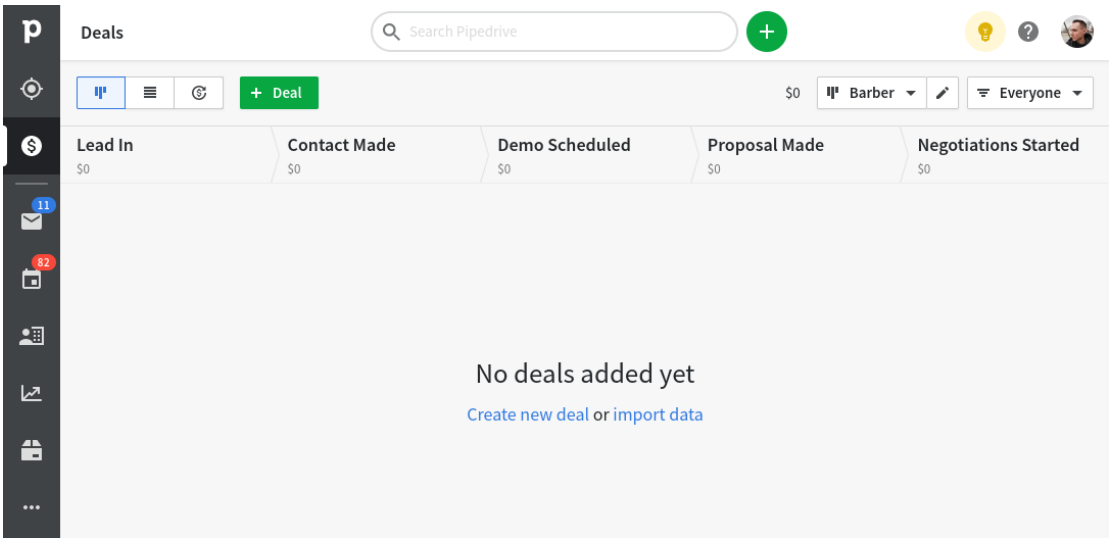


Figure 1. Pipedrive Deals Page

1.1.1 Subscription Plans

The key features in Pipedrive are manage leads and deals, track communications, automate and grow, along with insights and reports, privacy and security, and mobile apps and integration. There are four pricing plans with the first three being primarily

[4] www.pipedrive.com

used. Each plan contains a different set of features that are available. The plans from cheapest to most expensive are Essential, Advanced, Professional, and Enterprise. The plans are shown in Figure 2. Enterprise is rarely used, and has the same feature set as Professional, the only difference is the custom onboarding and support. As the plans get less expensive, the more advanced features are removed and the user only has access to the core functionality, which is still plenty to get the job done. One major feature missing on the Essential plan that the Advanced plan has is Email syncing. At the very top of the page is a button called "Manage subscription" that can be pressed to change the subscription plan.

Leadbooster [5] is a powerful and easy to use lead generation toolset. This is an add-on that is paid additionally to any Pipedrive plan. This can be used within the trial period as well. Since a subscription plan is required to use leadbooster, this thesis will not focus on increasing conversion rate for this add-on.

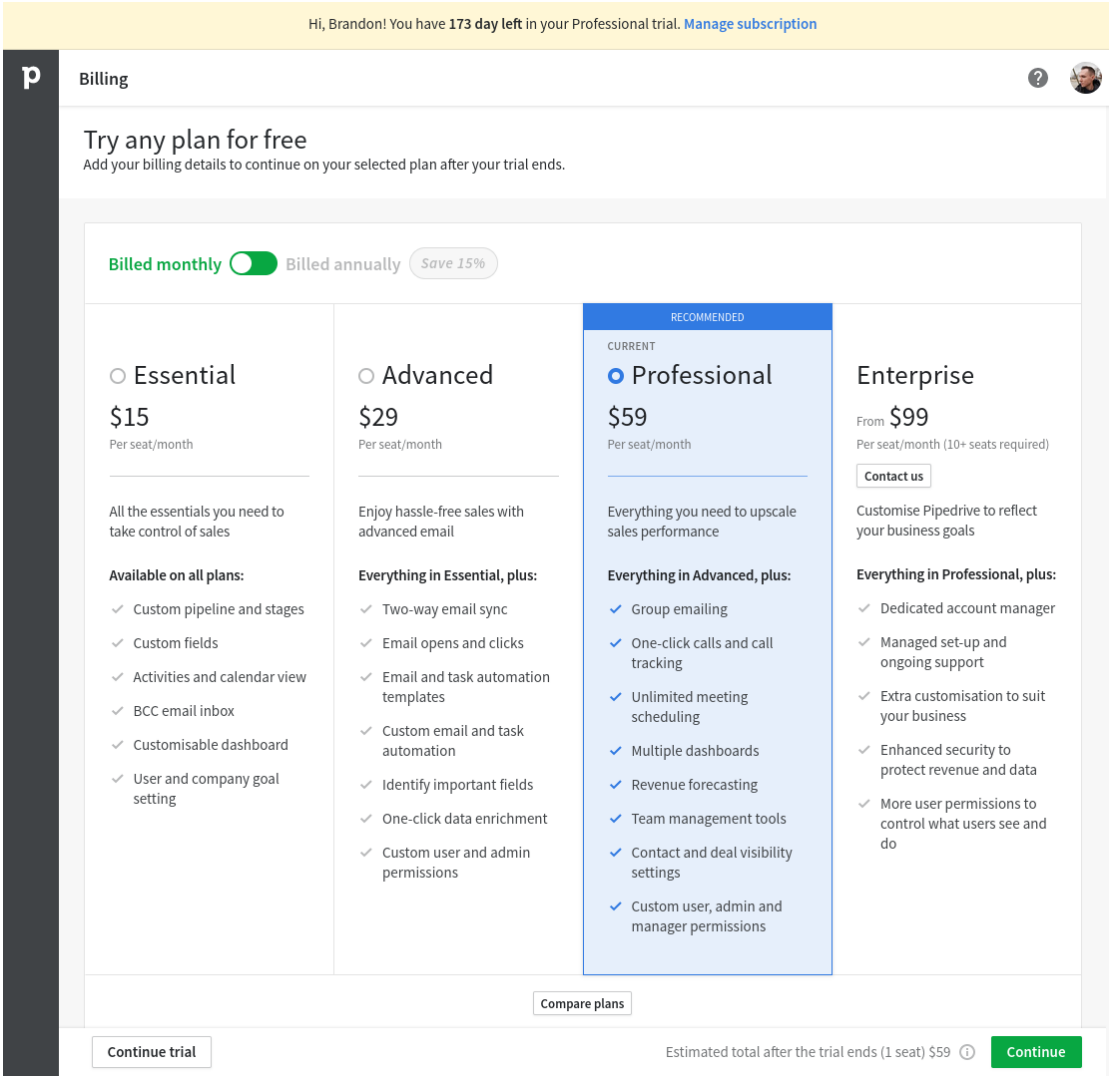


Figure 2. Pipedrive Subscription Plans

[5] https://www.pipedrive.com/en/features/leadbooster-add-on

1.2 Motivation

Pipedrive has no fine grained approach to discover how the user interacts with the system. The current method is to have a Product Manager (PM) know the system and conduct user surveys, interviews, analyse support data, issue reports, and analyse usage reports from data analytics. Nothing shows where the user gets stuck in the process and has to navigate somewhere else to get the information to complete the process. The Net Promoter Score (NPS) on the product scale shows customer's feedback on how well they like the service, but nothing on a customer journey level. For example, in the trial-to-pay process, the customer can do anything within a trial. We have no idea why they choose to pay or not. Knowing the details will help Pipedrive improve the trial to paid conversion rate, translating into more revenue for the company.

1.3 Pipedrive's Trial-to-Conversion Journey

Pipedrive provides trial accounts to potential customers of the product. These trial plans last for 14 days normally, but could be extended with promotions. During these 14 days, customers are given access to all product core functionality. Naturally, the objective of this trial is that the customer converts into a paid account. Different stakeholders at Pipedrive are interested in improving the customer journey from creation of a trial account to conversion into a paid account.

To shed insights into the potential frictions and bottlenecks in current customer journey from trial to paid account, we extract a data set from the Application Performance Monitoring system used by Pipedrive capturing high fidelity logs of trialled customers. Specifically these logs capture how customers interact with different components in Pipedrive during the trial.

1.4 Research Questions

RQ1: What distinguishes customer journeys between trialled customers that convert to a paid account versus those who abandon?

RQ2: Are there any particular roadblocks or bottlenecks that prevent trialled customers from converting?

1.5 Process Mining

The IEEE Task Force on Process Mining has created a Process Mining Manifesto [1] that says "The idea of process mining is to discover, monitor and improve real processes by extracting knowledge from event logs readily available in today's information systems... Process mining provides an important bridge between data mining and business process modeling and analysis". The three types of process mining are discovery, conformance, and enhancement [2].

- *Discovery*: Process discovery techniques take an event log as input and produces a process model.

- *Conformance*: Compares an existing process model with an observed behavior in the event logs. Conformance techniques check if reality, as recording in the logs, conforms with a predefined model.

- *Enhancement*: Improves and extends an existing process model based on insight generated by process mining.

This thesis will primarily focus on the *Enhancement* aspect of process mining to answer research questions in 1.4 and 1.4.

Traditionally, process mining is used to analyse back-end processes in an organization. It has also found some application in the field of customer journey analysis. In this context, the tasks recorded in the event log are effective steps that a customer takes while visiting a certain online product. These steps can be used to discover down-the line processes, identify deviations with expected pathways, and to identify potential performance issues.

From a methodology point of view, we adopt a structured process mining approach, mainly the Project Management for Process Mining (PM^2) which decomposes the process mining analysis into stages. This thesis is structured according to these stages.

1.6 Project Management for Process Mining

Project Management for Process Mining (PM^2) [3] is a methodology to guide the execution of process mining projects. PM^2 is inspired by the broader method for process mining, called CRISP-DM. CRISP-DM [4] is very high-level and provides little guidance for process mining specific activities as it has to be applicable for any data mining project. PM^2 is much simpler than CRISP-DM because it focuses on what is typically required in process mining projects. PM^2 is designed to support projects aiming to improve process performance or compliance to rules and regulations. It covers a wide range of process mining and other analysis techniques, and is suitable for the analysis of both structured and unstructured processes. There are six stages to PM^2 and each stage consists of inputs, output, and activities, as seen in Figure 3.

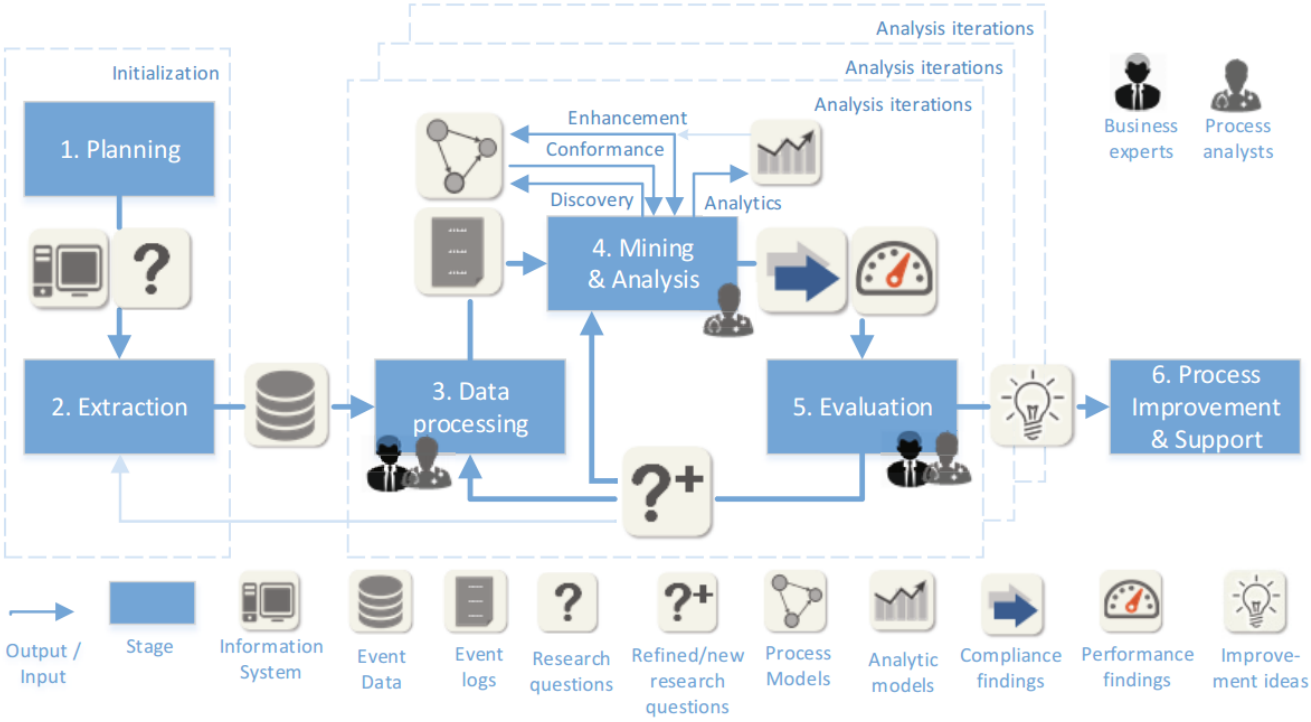


Figure 3. PM^2 Overview
Source: Adapted from [3]

These stages are briefly explained below:

Stage 1: Planning
The planning stage is meant to set up the project and determine the research questions. A business process is first selected to analyse and improve. The quality of the event data is taken into account to determine if the project is feasible given the common problems with data [5]. A research team is put together consisting of experts with different roles such as business owners and process analysts. The goals are identified and then translated into research questions to understand what is the output of this project.

Stage 2: Extraction
Before the data is extracted, the scope needs to be determined, such as which granularity, period, data attributes, and data correlations. Then the event data can be extracted from the relevant information systems and joined into a single collection of events. Tacit knowledge is exchanged between business experts and process analysts so that they can do their job properly.

Stage 3: Data Processing
This stage is meant to create event logs that are optimal for the mining and analysis stage. The first step is to create specific views on the event data by defining the case notions and event classes. Case notions relate to events such that they form a process instance, while event classes distinguish different activities within a process instance. Events can be aggregated to reduce complexity and improve

the structure for mining results. This can be done in two different ways: *is-a* aggregation considers events belonging to a more general event class while keeping the number of events the same, while *part-of* aggregation merges multiple events into larger events. Next, the events logs are enriched with additional attributes by either deriving or computing additional events or adding external data. Finally, the event log can be filtered by *slice and dice* to remove events based on the values, *variance based filtering* to group similar traces through clustering, and *compliance based filtering* to remove events that do not comply with a given rule or fit a given process model.

Stage 4: Mining and Analysis
This stage applies the process mining techniques to event logs and aims to answer the research questions defined in the first stage. The analysis covers four methods: process discovery, conformance checking, enhancement, and process analytics. The first three techniques are described in Section 2.2. Process analytics are a complementary analysis technique where data mining and visual analytics can be applied in the context of business processes.

Stage 5: Evaluation
After the findings from the analysis are complete, they can be diagnosed, which includes correctly interpreting the results, distinguishing interesting result from the expected, and refining research questions for further investigations. The correctness of the findings is investigated through Verify and Validate. Verification compares them to the original data and system implementations, while validation compares the findings to the claims of process stakeholders. This will translate the findings to improvement ideas that achieve the project's goals. Including the process experts in this stage is very useful to guide the process analysts, as they may not be domain experts.

Stage 6: Process Improvement and Support
The final stage of PM^2 is to use the gained insights to implement changes in the process. Achieving process improvements is often the main motivation for a process mining project, however, the actual implementation is generally a separate project with different areas of expertise. Therefore, the results of a process mining project form the fast based input of process improvement efforts. Process mining can also provide operational support by detecting problematic running cases, predicting their future or suggesting recommended actions, but this is a challenging form of process mining.

This thesis will utilize an adapted version of PM^2 which is more suitable for single person projects. This methodology is inspired by Marlon Dumas as shown in Figure 4. The changes are:

- The first stage is reclassified from *planning* to *problem framing* because as a single person project, there is no team to assemble. Instead, key stakeholders are identified which we communicate with to establish the project goals.

- Stages two and three of the original PM^2 are combined into stage two of the modified version. As a single person project, data extraction and event log creation are done at once instead of handing off between team members.
- The process mining and analysis stage provides feedback to change how the data is extracted and compiled into an event log. For example, process analytics and process mining are used to verify the data integrity of the extracted data. When a problem is found, the data extraction and event log compilation must be changed.

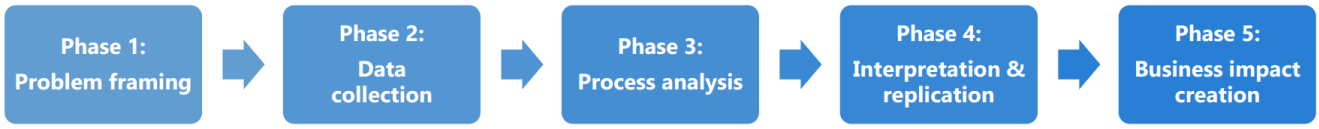


Figure 4. PM^2 Overview Modified
Source: Adapted from [6]

In this project, we decided to adopt a more exploratory approach where we did not have a previous hypothesis. This is because a customer journey analysis and optimization is a topic that has only started being explored at Pipedrive. Therefore, the goal of this project is not to validate, instead to generate hypothesis that can be explored more in-depth using other tools, such as Business Intelligence reports.

1.7 Thesis Outline

This thesis is organized according to the modified PM^2, as follows:

Chapter 2 introduces the background.

Chapter 3 discusses the problem scoping and data preparation stages of the PM^2 life-cycle.

Chapter 4 presents the analysis and interpretation of the results.

Chapter 5 validates the results from process mining with machine learning.

Chapter 6 concludes the thesis, provides feedback back to Pipedrive, shows limitations, and discusses future work.

[6] https://courses.cs.ut.ee/LTAT.05.025/2021_spring/uploads/Main/Lecture8.pdf

2 Background

This section defines concepts relating to customer journey analysis and process mining to address the problems in our thesis.

2.1 Customer Journey

A **customer journey** is the complete cycle of experiences a customer has while interacting with the system [6]. The purpose is to identify the customer's positive experiences so that they come back again and pay for the system. This can also uncover what the customers don't like and where they get stuck along with, the obstacle, and complexities of the system.

The **Customer Journey Map (CJM)** is used to understand customers behavior and better serve them. This can be done by using a visual, process-oriented method for conceptualizing and analysing people's behavior. The components of a CJM are described by Bernard [6].

Customer - the stakeholder experiences a service. In our thesis, the customers are the user's that sign up for a trial in Pipedrive and start using the software.

Journey - typical path followed by a customer. In our thesis, the journey starts at sign-up when the customer begins a trial and ends by either paying or stopping usage of Pipedrive.

Mapping - process consisting of tracking and describing customers' responses and experiences when using a service. In traditional customer journey mapping, this takes form as a sequence of steps followed by the customer. In our thesis, we are going to build the CJM from interactions of the front-end clicking actions the customers do while using Pipedrive.

Goal - customer journey should be mapped with a goal in mind. This thesis focuses on the conversion of customers into paid users.

Touchpoint - the interaction between customers and the system. In general, a touchpoint is an interaction via email, phone call, online interaction, etc.

Channel - the method chosen by the customer to interact with the touchpoint. In this thesis, we will focus only on online interactions as the channel because these are recorded into the monitoring system. We will not consider interactions with sales team via other channels like email, phone calls, or chatbot.

Stage - encompasses multiple touchpoints. For example, the initial stage where the customer creates a trial account, using the trial account, and then upgrades to a paid account.

Experience - encompasses customers' feedback and emotions. We will not be using informal or structured feedback; for example: surveys, reviews, interviews, or customer support dialogues. These data sources in interaction are not in the

scope of this thesis; i.e., to analyse the customers' emotional experience. Pipedrive is a business-to-business service, while business-to-customer services are more important in emotional experiences.

2.2 Process Mining

Process mining is a family of techniques to analyse data extracted from various information systems to analyse the performance of a business process and to identify improvement opportunities [1]. Process mining is a commonly used technique in the context of Business Process Management, as it allows us to analyse the current state of business process performance, identify areas of improvement, and assess the results of process improvements [7].

The starting point for process mining is an event log. An Event Log is a Table (or a dataframe in the data science sense) where each Row captures the execution of one step in a business process (or a customer journey in the context of this thesis). As a minimum requirement, every row in an event log must have:

- *Case Identifier* is an identifier of an execution of a business process. In the context of a customer journey, a case identifier could be either the customer identifier, or an identifier of a session that the customer had with a product.
- *Task Label* is the name of a step in a process. In our context customer journey analysis, a step is an interaction that the user had with the product.
- *Timestamp* tells when was this task (or step) performed.

For example, Table 1 and Table 2 show two different types of event logs used in this thesis. These event logs have two task labels, *activity_feature_name* and *activity_feature_action*, which are two different granularity levels of the events (*activity_feature_action* is the descriptive action of *activity_feature_name*). Column *datetime* is the timestamp each row occurs. An event log can also contain start and end timestamps for each row, which would show how long each event takes, instead of the instantaneous time at which they occur. The case identifiers in this log are *caseId_company* and *caseId_user* for the first table and *caseId_session* for the second.

	datetime	caseId_company	caseId_user	activity_feature_name	activity_feature_action
0	2021-01-14 15:28:10	7822867	–	Professional Plan	Professional Plan
1	2021-01-29 12:56:11	7822867	–	Essential Plan	Essential Plan
2	2021-01-22 02:49:51	7822867	11957260	Activities	Activity Added
3	2021-01-15 15:39:26	7822867	11957260	Activities	Activity Added
4	2021-01-27 19:35:04	7822867	11957260	Filters	Filter Used
...	...	...	...	...	...
500	2021-01-14 16:15:53	7814218	11950285	Contacts	Organization List Opened
501	2021-01-19 00:34:49	7814218	11967859	Contacts	Person List Opened
502	2021-01-09 05:20:35	7814218	11941093	Deals	Deal List Opened
503	2021-01-15 17:28:04	7814218	11950285	Activities	Activity Added
504	2021-01-12 14:41:28	7814218	11941093	Settings	Settings Page Opened

Table 1. Event Log with Company ID

	datetime	caseId_session	activity_feature_name	activity_feature_action
517079	2021-01-07 16:39:01	—	Insights	Dashboard Opened
571994	2021-02-03 14:01:37	—	Deals	Deal Won
23265	2021-01-16 12:57:58	—	Contacts	Contact List Opened
530526	2021-01-11 18:40:27	—	Deals	Deal Stage Changed
430752	2021-01-26 15:44:29	—	Deals	Deal Added

Table 2. Sessionized Event Log

Given an event log, process mining techniques provide the following capabilities:

- **Process discovery**: The goal is to construct a process model that captures the behavior of an event log.
- **Performance analysis**: Process mining allows us to analyse a business process with respect to several dimensions, such as time, cost, or quality. Event logs provide enough detailed information so that it is possible to spot a process that does not perform well in one of these dimensions.
- **Conformance checking**: Determines if the process follows predefined rules or constraints. When a constraint does not hold, we can call it a violation, and then appropriate corrective action can be applied.

There are a number of process mining tools available in the market such as Celonis [7], ProM [8], Apromore [9], etc. In our thesis, we use Apromore because we are experienced in using it and Apromore has all the features we need. Apromore has a free community version that enables basic process mining. We have access to the Enterprise version, enabling many more advanced features. The software is able to take events and map the flow of the tasks that occur within the process. Very powerful filtering tools get rid of noise or divide users into different groups based on a range of criteria, including temporal criteria (slow cases vs. fast cases), repetition or rework, paths taken, etc. Figure 5 shows an example of what this software looks like.

[7] https://www.celonis.com/
[8] https://www.promtools.org/doku.php
[9] https://apromore.org/

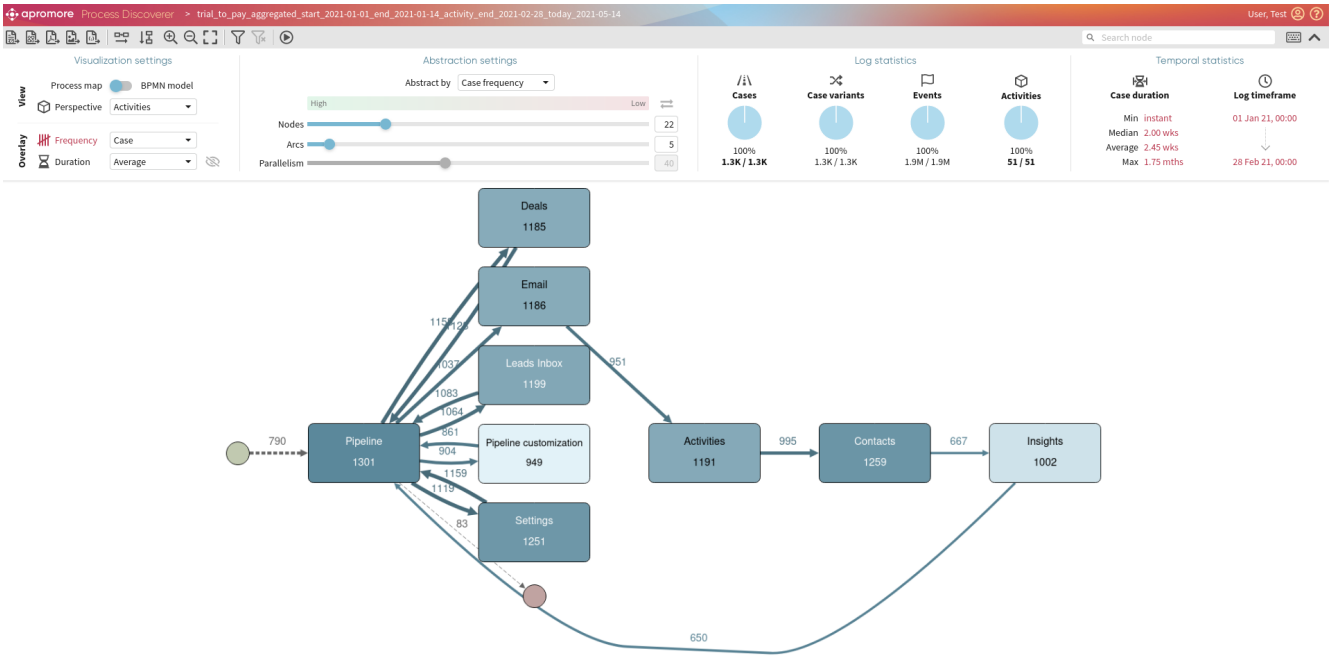


Figure 5. Example of Apromore

2.3 Event Log Extraction

Usually event logs are extracted from the database(s) of the system(s) on top of which the process is executed, either directly via SQL queries or indirectly by means of report-generation interfaces. In the case of a customer journey, an event log may be extracted from an Application Performance Monitoring tool, such as Matomo [10] or Amplitude.

There are five common challenges of extracting event logs from a system that is not designed to produce event logs [8].

1. **Correlation challenge** of identifying the case an event belongs to.
2. **Timestamp challenge** of obtaining accurate timestamp for logging, especially since logging is not a priority for some systems. This means that logging can be delayed until the system has idle time or little load, or logging of events occurs at different time-zones.
3. **Snapshot challenge** of having log data available for a certain period of time. Excluding incomplete cases with missing head or tail data can be good, but it may introduce a bias.
4. **Scoping challenge** of when the system does not directly produce event logs, which then requires understanding the data semantics to convert it.
5. **Granularity challenge** of difficulty defining a precise mapping between fine event logging and general process abstraction.

[10] https://matomo.org/

Starting the analysis of how to create the event log in Pipedrive, a *scoping challenge* to overcome is clear, since our current monitoring tools support creating aggregate data that is not directly mapped to a specific case, user, or session. This means that we will have to query the data from the database directly in order to create the event log. As seen in Table 1, there will be a *correlation challenge* because the case ID of the company or user is considered a quasi case ID, which are expanded upon in Section 3.6.1. The best case identifiers are the session identifier, as this shows exactly the period of time when the user is using the software as seen in Table 2. While analysing the event logs, it became clear that many events were missing the time from the timestamp. This is a *granularity challenge* and is expanded upon in Section 3.6.4.

At the physical level, event logs are usually represented as Comma-Separated Value (CSV) files. But they can also be represented using other formats, like the eXtensible Event Stream (XES), which is promoted by the IEEE Task Force on Process Mining. In this thesis, we use the CSV approach to represent event logs since this representation can be directly produced by SQL queries.

2.4 Process Mining for Customer Journey Analysis

Bernard and Andritsos [6] were among the first to apply process mining to customer journey mapping. In their next paper [9], they described a possible approach of using process discovery techniques to output the journeys onto a CJM instead of a Business Process Model (BPM). Visualizing event logs on CJMs give some advantages such as personal customer activities like emotions and also incorporate exceptional behaviors, rather than removing them to improve model readability.

Alessandro and Marwan [10] used customer journey analysis with process mining in order to discover the process and apply a recommendation system on top. A user can request a recommendation for a specific case by sending a partial case to the recommendation service, that applies process mining-on-the-fly to predict the future partial case. This is sent back to the user with a list of recommendations to assist the user in the next steps.

3 Problem Framing and Data Collection

This chapter will deal with the first two stages of the modified PM^2, Problem Framing and Data Collection.

3.1 Problem

The stakeholders are the project manager, data analysts, and process owner. Based on an initial meeting with the main stakeholders, we establish the goals of the project. The problem was formulated as, "How can we increase conversion rate of users?". We selected the process from the moment an account is created in a trial to the moment the account pays or becomes expired.

Normally, we would formulate hypotheses to validate on business experts so they can share their initial domain knowledge and direction. The goal of this project is to generate hypotheses that can be explored more in-depth using other tools which can validate the research questions defined in Section 1.4.

3.2 Data Collection Methodology

The Data Collection stage of the modified PM^2 methodology combines the Extraction and Data Processing stages of the original PM^2. The first step in data collection is to identify the data sources, process related entities, their identifiers, and relationships. Once these are understood a key instance object should become clear as the one central object, which can be the Case ID. Next is to identify the process milestones, also known as the process start and end event. For each pair of consecutive milestones, all activities you want in the event log must be enumerated. Given the objects, milestones, and activities, identify key table and relationships and where to get records of each activity's completion and, if available, the start time. From the columns in the identified tables, select possible case attributes, which are attributes of the object that don't change during the process. Also event attributes can be identified and selected as they are attributes that change during the case, which these can be useful for filtering purposes. Finally, the data can be extracted by writing SQL queries and written to a CSV file.

After the event log is created, process mining techniques and process analytics can be ran to verify whether the data in the event log are correct. If something is wrong, the data collection methodology will need to be revisited and modified in order to achieve the most accurate results.

3.3 About Data

In Pipedrive, the company is the core object. Users have their account and can be a member of one or more companies. Each user occupies one seat per company. In terms of billing for a company, each seat costs money, whether the seat has an active user or not. The price per seat is determined by the subscription plan level the company chooses, with all employees belonging to the same plan level. The user interacts with the different features, yet the company determines if they become paying or not. This division creates a complication in extracting the data as there are two primary IDs to work with.

3.4 Data Tools

The central location of the Data Warehouse (DW) is in AWS Simple Cloud Storage (S3). The data is served through Apache Spark [11] or Athena [12], with some replicating into Redshift database. Redshift [13] is a relational Relational Database loosely based on PostgreSQL database engine. It's optimized for analytical workflows and can scale in AWS to handle petabytes of data. To access this data, Data Steward is used to explore the data and Apache Zeppelin is used to write SQL and PySpark to extract the data.

These are great to verify if our manual SQL statements return accurate data. These work by aggregating events together, therefore the fine granularity is lost because the second level individual events are grouped into different buckets, such as in a day.

3.4.1 Amplitude

Amplitude [14] is the data analytical tool used to analyse individual behavior inside Pipedrive. This contains live data of what users are doing, graphs from event segmentations, dashboards, notebooks, and data alerts. Most of the information contains front-end tracking data, so users with ad-blocking, won't have any data. The tool has its own UI to create the event segmentation, grouping, and selecting so SQL statements are not used.

3.4.2 Tableau

Tableau [15] is mainly used to show aggregated data on business and product data. The tool can analyse and explore data visually, create interactive dashboards, combine data sources, share analysis with others, schedule alerts, and embed dashboards into other applications. The tool is great at showing PMs and analysts what is going on. The data comes from SQL statements that connect to the databases, which makes is easier to convert Zeppelin notebooks into Tableau notebooks.

3.4.3 Data Steward

Data Steward is a simple web app for browsing, searching, and editing our Data Warehouse's metadata - the data about databases, schemas, tables, columns, and comments. This web app is actually developed in-house by Pipedrive and is not available on the Internet. The tables are divided up into schemas and prefixes that have specific purposes. The Schemat are described as:

- **dw** - Data warehouse main schema
- **dwmodel** - Subject area-specific data models

The prefixes are described as:

[11] https://spark.apache.org/
[12] https://aws.amazon.com/athena/
[13] https://aws.amazon.com/redshift/
[14] https://amplitude.com/
[15] https://www.tableau.com/

- **d_** - dimension table
- **f_** - fact table

3.4.4 Apache Zeppelin

Apache Zeppelin [16] is an open source web-based notebook that enables data ingestion, transformation, ad-hoc analysis, reporting, presentation, and interactive collaboration. This enables documentation, code, and reports to exist on the same notebook. This is convenient as it allows fast prototyping of queries. The languages of Structured Query Language (SQL) and PySpark are available to be used. This enables the programmatic abstraction of large SQL queries using the Spark SQL query engine. Therefore, instead of writing a huge SQL query with many joins and nested queries, one can break up the query into smaller parts so that they are easier to maintain and understand how it works. Another useful feature is being able to pass parameters into a query to customize the output. Our Zeppelin notebook contains global parameters and feature flags that propagate throughout to customize the date ranges on the queries, and which data pre-processing feature is applied.

3.4.5 Jupyter Notebook

The Jupyter Notebook [17] is an open-source web application that allows creating and sharing documents that contain live code, equations, visualizations, and narrative text. Apache Zeppelin is also a notebook based tool, but there are limitations with the types of visualizations available. This notebook provides unlimited flexibility by enabling Python, Pandas [18], and Seaborn [19] to be used to create complex visualizations.

3.5 Key Database Tables

To create a data table diagram, we first looked into using dbDiagram.io [20], which was easy to use with visually appealing tables. However this tool was discarded, because of not being web-based and the diagram by default was public without paying a subscription fee. We decided on using DbSchema [21] to create the table schema figure as the program was able to be downloaded to our local computer and produced visually pleasing figures. Figure 6 shows an overview of the key tables and their relationship to one another used to construct the event log. The names of schemas and prefixes are simplified to make them easier to read [22].

[16] https://zeppelin.apache.org/
[17] https://jupyter.org/
[18] https://pandas.pydata.org/
[19] https://seaborn.pydata.org/
[20] https://dbdiagram.io
[21] https://dbschema.com/
[22] https://dbschema.com/documentation/schema.html#fk-type

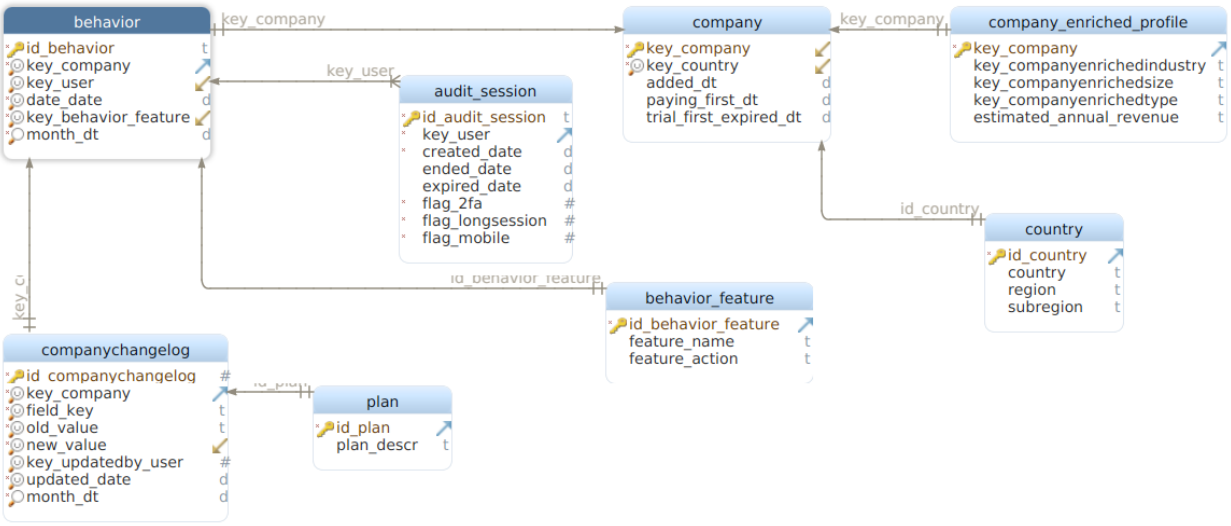


Figure 6. Key Database Tables

3.5.1 dwmodel.d_company

Table 3 lists the companies and their attributes. The companies in the table exclude test and fake companies. Aggregated attributes from the company and other models include *first_paying_dt* and *trial_first_expire_dt* which come from the *dw.f_companychangelog* table described in Section 3.5.7. These attributes are used to segment the event log into two parts. One log when the trials first become paying, and another when the trials first become expired. No Personally Identifiable Information (PII) attributes are in the table, as this is separated out to a separate schema with special permissions.

key_company	added_dt	paying_first_dt	trial_first_expired_dt
7759940	2021-01-26	1000-01-01	2021-02-10
7796493	2021-01-02	1000-01-01	2021-01-17
7797021	2021-01-03	1000-01-01	2021-02-03
7797306	2021-01-03	1000-01-01	2021-01-18
7797423	2021-01-04	1000-01-01	2021-01-19
7797534	2021-01-04	2021-01-18	2021-01-18

Table 3. Company Table

3.5.2 dwmodel.d_company_enrichedprofile

Table 4 contains additional attributes for the company. Many of the attributes are populated by Clearbit [23], which is a Business-to-business (B2B) marketing data engine to be able to better understand customers. Some of the data that we use are the company

[23] https://clearbit.com/

annual revenue, type, size, and industry. These enable segmentation of the customers into different groups to identify any significant differences.

key_company	key_companyenrichedindustry	key_companyenrichedsize	key_companyenrichedtype	estimated_annual_revenue
788	paper & forest products	251-1k	private	–
1348	education services	1-10	private	$0-$1M
1434	internet software & services	51-250	private	–
1616	–	–	private	–
3511	professional services	1-10	private	$0-$1M
40692	diversified consumer services	–	private	–

Table 4. Company Enriched Data Table

3.5.3 dw.d_country

Table 5 provides the country description. The countries are split up into regions and sub-regions, which enables grouping of companies to see differences between regions.

id_country	country	region	subregion
AD	Andorra	Europe	Southern Europe
AE	United Arab Emirates	Asia	Western Asia
AF	Afganistan	Asia	Southern Asia
AG	Antigua and Barbuda	Americas	Caribbean
AI	Anguilla	Americas	Caribbean
AL	Albania	Europe	Southern Europe

Table 5. Country Table

3.5.4 dwmodel.f_behavior

This core table enables process mining that contains all the user event-level interaction features in Pipedrive, with a millisecond timestamp granularity. Table 6 links the company to the user that performs these behavioral events, using these three keys: *key_company*, *key_user*, and *key_behaviorfeature*. The column *month_dt* is a partition key used to enable faster querying. This table is massive, containing many millions of rows. Our data is selected over the first two months of 2021 to reduce the amount of data extracted. In order to filter the data, we need to write the SQL to use the partition key in the table, as this reduces the amount of data the SQL has to traverse. Otherwise, a SQL query on the full table can take 35 minutes or more.

key_company	key_user	date_date	key_behaviorfeature	month_dt
7540274	11929758	2021-01-14 12:46:26.633	typ_100	2021-01-01
578037	4816912	2021-01-25 12:57:51.133	typ_100	2021-01-01
7632169	11638455	2021-01-11 19:36:53.818	typ_100	2021-01-01
7134715	1557868	2021-01-29 14:56:20.513	typ_100	2021-01-01
1281589	9165749	2021-01-12 13:05:47.17	typ_100	2021-01-01
445699	11560197	2021-01-27 06:51:33.277	typ_100	2021-01-01
7430150	11454728	2021-01-27 12:19:46.642	typ_100	2021-01-01
645633	11825477	2021-01-26 15:42:42.183	typ_100	2021-01-01
52963	5988375	2021-01-18 08:30:36.289	typ_100	2021-01-01
7535362	11554712	2021-01-09 13:26:52.045	typ_100	2021-01-01

Table 6. Behavior Events Table

3.5.5 dw.d_behaviorfeature

Table 7 describes the feature data with multiple levels of granularity. The column *feature_type* is either "Interaction" or "Behind the Scenes" and shows what causes the behavior to be logged. The column *feature_name* is the name of the feature the user has just used. The column *feature_action* is the action performed on the feature. All the features have a plan availability in column *feature_availability* where this is the plan the feature becomes active. Subscription plan information is described in Section 1.1.1.

id	feature_type	feature_name	feature_action	feature_availability
typ_107	Interaction	Insights	Public Dashboard Opened	Essential
typ_11	Interaction	Caller	Call Ended	Professional
typ_125	Interaction	Deals	Deal Details Opened	Essential
typ_126	Interaction	Deals	Deal Details Opened	Essential
typ_130	Interaction	Pipeline	Pipeline View Opened	Essential
typ_142	Interaction	Email	Email Sent	Advanced
typ_162	Interaction	Contacts	Organization Added	Essential
typ_172	Interaction	Web to Mobile Calling	Call Ended	Advanced
typ_189	Behind the Scenes	Web Forms	Web Form v3 Displayed	Leadbooster Add-on
typ_190	Interaction	Web Forms	Web Form v3 Added	Leadbooster Add-on

Table 7. Behavior Feature Dimension Table

3.5.6 dw.f_auditsession

Table 8 stores all of the session data. For displaying the information, the *id_auditsession* is represented as "...". A session is started when the user logs in and ends when the user logs out manually or the session expires after a set period of time. Sessions are quite complicated in Pipedrive because a user can log in through many different ways. A user can log in by e-mail/password, Two Factor Authentication (2FA), Google, Linkedin, Single Sign-On (SSO) and through two different platforms, Web and Mobile. Users can choose to remember their login, then the user will be logged in for a maximum of 30

days. Users are not logged out in the middle of their work. The four different scenarios in how sessions are used at Pipedrive are summarized below:

1. **Short** - User Logs in and logs out.
2. **Short Expire** - User logs in without clicking "Remember Me", session expires after 12 hours.
3. **Long Expire** - User logs in with clicking "Remember Me", session expires after 30 days.
4. **Mobile** - User logs in, standard session expires after 30 days. If the user continues to use the app after every 7 days, session is extended.

id_auditsession	key_user	created_date	expired_date	ended_date	flag_2fa	flag_longsession	flag_mobile
...	554517	2021-02-05 10:26:19	2021-02-05 22:26:19	2021-02-05 15:06:33	0	0	0
...	7454456	2021-01-26 09:30:39	2021-01-26 21:30:39	1000-01-01 00:00:00	0	0	0
...	8226280	2021-02-08 19:16:39	2021-02-09 07:16:39	1000-01-01 00:00:00	0	0	0
...	11939712	2021-01-29 07:24:31	2021-01-29 19:24:31	1000-01-01 00:00:00	0	0	0
...	10263682	2021-01-28 12:38:12	2021-01-29 00:38:12	1000-01-01 00:00:00	0	0	0
...	11664773	2021-01-20 02:07:39	2021-01-20 14:07:39	1000-01-01 00:00:00	0	0	0

Table 8. Audit Session Table

3.5.7 dw.f_companychangelog and dw.d_plan

These two tables combined together are able to track the plan change information for all users. The *dw.f_companychangelog* table has the data for tracking changes to the company, such as "status" and "plan level". The *updated_date* column is a timestamp but has granularity down to the second level, not as good as the other timestamp granularity down to the millisecond. The *key_updatedby_user* column is supposed to have the key of the user that made the change, but unfortunately all the values are blank with value "-9000". This is the case for all plans greater than 2021, suggesting this is a problem with data logging. The *field_key* column is the key to describe which event is being changed; i.e., the value changes from *old_value* to *new_value*. Then joining with the *dw.d_plan* table, shows the plan levels as described in Figure 2. Pipedrive's three plan levels determine which features are available to the user. In the plan table, the plans are divided into monthly, annual, and all historical plans at the many different pricing levels.

During the timeframe under analysis, the plans were further divided because of an ongoing geographic pricing test. This means that certain countries were given discounts to Pipedrive plans to determine if this affects total revenue positively. In return, the test was successful and more users in those countries did indeed convert. For our thesis, we will ignore the results of this test on our data and just assume they behave normally, because the users affected were minor compared to the total being analysed. These users were mainly from countries with already low conversion rates.

3.6 Challenges Encountered

The raw data extracted from the system contained a number of issues that had to be analysed and addressed before being able to create an event log. The subsections below discuss the major challenges we encountered.

3.6.1 Quasi Case ID

The company or user ID is not considered a real Case ID. This is because process analysis is supposed to analyse the time when the user is currently using the software. This is important to be able to figure out the duration in events and between events. When the Case ID is active for the entire duration of the process, the max waiting times are probably the time when the user stopped using the software and when they started using it again, which could be days later. This makes figuring out the longest waiting times impossible. A fix is to divide the Case ID up into user sessions. In Section 3.7.6, this is solved by using Sessionization.

3.6.2 Instant Event Duration

All of the behavior events contain a single timestamp, as seen in Table 6. Thus, it is impossible to see how long an event has taken place because there is no start and end timestamp. This problem can be fixed by coalescing rows together that relate to the same event, such that you can find the minimum and maximum timestamp within the time period. This coalescing is done in Section 3.7.5.

3.6.3 Missing Subscription Plan Level

The features in Pipedrive are available according to a companies' subscription plan. During a trial, it is possible to switch plans in the middle of a trial to try out the features. The button is available at the very top of the screen, such as in Figure 2. The event data does not include when the user switches between plans. This affects the features available to the user, which can affect the analysis of the usage of such features. To fix this, we would need to add our own events into the event log from the tables *dw.f_companychangelog* and *dw.d_plan* as previously described in Section 3.5.7.

3.6.4 Missing Time Granularity on Events

While analysing the event log, we see visually that many behavioral events occur at midnight, at the time 00:00:00. By default, the time is set to 00:00:00 when the time data is missing for events. This can mess up the event log as the events will not show in the correct order for process mining analysis. Data analysis on this issue is done in Section 3.8.5. These events missing the time granularity can only be used for usage statistics but cannot be used in process mining.

3.7 Event Log Creation

The first question we have to answer with creating an event log is, "How much data should we collect?" That is easy, as much as our personal computer and the process mining software can handle so that it does not slow down the analysis. From experimentation, it appears a couple million rows in the event log is a good value as there is plenty of data and does not cause our personal computer to come to a halt.

First, we need to gather the companies and then split into two groups. One where the companies eventually pay for a subscription plan in Pipedrive, and the other where they eventually have their trial become expired. Secondly, we need to gather all the behavior events for those companies until they reach an ending condition. A trial period normally lasts for two weeks, but can be extended through promotions. Finally, we apply event log pre-processing to fix the issues as previously discussed in Section 3.6. The parameters for our event log are:

- *signup_start_dt* (2021-01-01) - Beginning of company signup period
- *signup_end_dt* (2021-01-14) - End of company signup period
- *behavior_end_dt* (2021-02-28) - End of behavior events period

This means that we gather all the companies that signed up in the first two weeks of January 2021 and gather all the behavior events for them until the end of February 2021. If a company trial lasts longer than the behavior event period, the company is not included. By excluding companies that have very long trial periods, this may introduce a bias towards giving results based on shorter length trials. This is called the *snapshot challenge* as mentioned in Section 2.3. Although the number of companies that exceed a trial period of more than six weeks is very low, that should not affect the overall analysis of results.

3.7.1 Dividing Companies into Pay vs Not Pay

To divide the companies into pay vs not pay, we need to know which comes first, whether the trial becomes a paying plan or the trial become expired. The order is important as trials that are expired can also be converted to paying in the future. These fields are described in Section 3.5.1. Listing 1 shows the code for Trial-to-Pay (T2P) companies and Listing 2 shows the code for Trial-to-Expire (T2E) companies.

According to the analysis in Section 3.8.2, we have determined that when a trial expires, the company can pay for Pipedrive afterwards. A heavy emphasis of close to 75% of the users that pay after trial, do so on the same day. A significant yet minor portion of customers pay after five days. We make an assumption in Listing 1 that if a customer pays for Pipedrive after five days or less after the trial expires, then that conversion is counted as paid. We do not want to include periods longer than this because the effect of the trial is probably negligible after one week. This is common sense, as a user that one day finds his trial expired and has liked the software, should subsequently want to pay for the software shortly afterwards. The list of companies is stored in a temporary view called *df_companies*, which can be used in subsequent SQL without having to create a nested SQL statement.

Another problem we ran into was invalid dates for *paying_first_dt* and *trial_first_expired* that stated they were at year 1000. Even though the company sign-up had started between the sign-up dates, we had to do a second filter on the qualifying column to ensure the date was accurate as well. Listings [1, 2] are the code snippets to find which companies pay and which companies do not.

```
df_t2p = spark.sql("""
    select key_company,
        paying_first_dt as end_dt
    from dwmodel.d_company
    where added_dt between '%s' and '%s'
    and paying_first_dt between '%s' and '%s'
    and datediff(paying_first_dt, trial_first_expired_dt) < 5
""" % (signup_start_dt, signup_end_dt, signup_start_dt,
    behavior_end_dt))
df_t2p.createOrReplaceTempView('df_companies')
```

Listing 1. List of Companies that are Trial-to-Pay

```
df_t2e = spark.sql("""
    select key_company,
        trial_first_expired_dt as end_dt
    from dwmodel.d_company
    where added_dt between '%s' and '%s'
    and trial_first_expired_dt between '%s' and '%s'
    and (trial_first_expired_dt > paying_first_dt
        OR  datediff(paying_first_dt, trial_first_expired_dt) >= 5)
""" % (signup_start_dt, signup_end_dt, signup_start_dt,
    behavior_end_dt))
df_t2e.createOrReplaceTempView('df_companies')
```

Listing 2. List of Companies that are Trial-to-Expire

3.7.2 Link Companies to Behavior Data

Listing 3 is the code to gather the behavioral feature events for the companies in Section 3.7.1. The behavioral features are joined with the companies to obtain the user ID that performs these events. The *month_dt* partition column is being used on the where clause so that the SQL only selects data from the specified months provided, so as to not take half an hour to run. The results are stored in a view called *df_behavior* to enable easy access in another part of the Zeppelin notebook.

Flaws were noticed after beginning to apply process mining analysis. One problem was the duplicate rows. The only way this is possible is if the exact same event happened in under one second. The timestamp granularity does not go into millisecond, so this is technically possible. The rate of row duplication is quite significant. For T2P event log, 8.1% are duplicates. For the T2E event log, 16.6% are duplicates.

Another problem noticed during the analysis, is that the behavior events should be limited to the timeframe the customer is active on the trial. Once the customer pays for Pipedrive, they will continue to accumulate activity events. However, once a customer's trial expires, then they will no longer use Pipedrive and activity data will cease. If the

dates are not limited, then my analysis will include events when the customer has paid, negating the focus of my thesis on the trial.

```
df_beh = spark.sql("""
    select beh.date_date as datetime,
        beh.key_company as caseId_company,
        beh.key_user as caseId_user,
        fea.feature_name as activity_feature_name,
        fea.feature_action as activity_feature_action
    from dwmodel.f_behavior as beh
    inner join (
        select key_company, end_dt from df_companies
    ) as c
        on c.key_company = beh.key_company
    inner join dw.d_behaviorfeature as fea
        on fea.id_behaviorfeature = beh.key_behaviorfeature
    where beh.month_dt between date('%s') and date('%s')
    and beh.date_date <= end_dt
    order by beh.date_date
""" % (signup_start_dt, behavior_end_dt))

# drop distinct rows with same datetime and event
df_beh = df_beh.distinct()

df_beh.createOrReplaceTempView('df_behavior')
```

Listing 3. Link Companies to Behavior

3.7.3 Adding Case Attributes

In addition to events, having case attributes about the company enable segmentation about these attributes to see if any differences in conversion exist. Some attributes from the table of *dwmodel.d_company_enrichedprofile* and *dw.d_country* are added as columns in the event log. These are added to the event log stored in the view of *df_behavior*. Previously, these attributes were in the original event log query, but they had to be separated as it is impossible to join with the subscription plan events when the case attributes are in the event log because all columns must match.

```
df_attr = spark.sql("""
    select beh.*,
        cty.country as case_attr_country,
        cty.region as case_attr_region,
        cty.subregion as case_attr_subregion,
        c_enrich.key_companyenrichedsize as case_attr_company_size,
        c_enrich.key_companyenrichedindustry as
            case_attr_company_industry,
        c_enrich.estimated_annual_revenue as
            case_attr_company_est_annual_revenue,
        c_enrich.key_companyenrichedtype as case_attr_company_type
    from (
        select * from df_behavior
    ) as beh
```

```
    inner join dwmodel.d_company_enrichedprofile as c_enrich
        on beh.caseId_company = c_enrich.key_company
    inner join dw.d_country as cty
        on c_enrich.key_companyenriched_country = cty.id_country
""")

df_attr.createOrReplaceTempView('df_behavior')
```

Listing 4. Adding Case Attributes

3.7.4 Remove Missing Time Events

According to the analysis done in Section 3.8.5, some events need to be taken out because they do not make sense for the event log.

```
df = spark.sql("""
    select beh.*
    from  (
        select * from df_behavior
    ) as beh
    inner join dw.d_behaviorfeature as fea
        on fea.feature_action = beh.activity_feature_action
    where fea.id_behaviorfeature NOT IN ('typ_52', 'typ_35', 'typ_33
        ', 'typ_73', 'typ_74', 'typ_75', 'typ_77', 'typ_36', 'typ_72
        ', 'typ_53', 'typ_46', 'typ_54', 'typ_45')
""")

df.createOrReplaceTempView('df_behavior')
```

Listing 5. Removing Midnight Events

3.7.5 Resolve Instant Duration By Temporal Coalescing

This section solves the problem described in Section 3.6.2. The solution is to apply a temporal coalescing to the event log. Temporal Coalescing merges value-equivalent tuples that agree on explicit attribute values with overlapping timestamps [11]. For the event log, this means that we merge multiple feature events over a timeframe that has events repeated, so that we end up with one row for each event that contains a start and end timestamp. This squashes the detailed event data of *feature_action* in favor of a better representation of higher level *feature_name* event data. Table 9 shows the event log before temporal coalescing and Table 10 shows the result. Notice how the events of "Email" and "Settings" are squashed into one row. The start timestamp is the minimum timestamp in the timespan and the end timestamp is the maximum timestamp in the timespan.

To apply temporal coalescing on our event log, Rakesh's code [24] was used as a similar example to our problem. Listing 6 is the code to achieve the result in Table 9. The code is described as:

[24] https://www.javaer101.com/en/article/959161.html

- Line 3: Read the event log saved in the temporary view *df_behavior* by Listing 3.
- Line 4: Partition [25] event log by company so that all operations occur per company and not across companies.
- Line 5 - 9: Create a new column to identify when the higher level feature of *feature_name* changes. The "lag" function retrieves the value from the previous row. Writes a "1" when feature changes.
- Line 10 - 13: Fill in N/A fields with default of "1". This occurs on the first row of the event log per company, because there is no previous event to look at.
- Line 14 - 17: Create a new group column to list all similar features together as the same number. This is done by summing over *feature_change* column.

```
1  from pyspark.sql import functions as F, Window
2
3  df = spark.sql("select * from df_behavior")
4  w1 = Window.partitionBy('company').orderBy('datetime')
5  df = df.withColumn(
6          'feature_change',
7          (F.col('feature_name') != F.lag('feature_name').over(w1))
8                  .cast('int')
9      ) \
10     .fillna(
11         1,
12         subset=['feature_change']
13     ) \
14     .withColumn(
15         "group",
16         F.sum(F.col('feature_change')).over(w1.rangeBetween(Window.
               unboundedPreceding, 0))
17     )
```

Listing 6. Creation of Aggregation Group Flag

	timestamp	company	feature_name	feature_action	feature_change	group
0	2021-01-07 00:00:00	7802988	Email Sync	Email Sync Enabled	1	1
1	2021-01-07 15:01:21	7802988	Pipeline	Pipeline View Opened	1	2
2	2021-01-07 15:01:28	7802988	Email	Email View Opened	1	3
3	2021-01-07 15:01:42	7802988	Email	Email View Opened	0	3
4	2021-01-07 15:08:56	7802988	Email	Email View Opened	0	3
5	2021-01-07 15:09:13	7802988	Contacts	Person List Opened	1	4
6	2021-01-07 15:09:41	7802988	Settings	Settings Page Opened	1	5
7	2021-01-07 15:09:46	7802988	Settings	Settings Page Opened	0	5
8	2021-01-07 15:09:48	7802988	Contacts Timeline	Person Timeline Opened	1	6
9	2021-01-07 15:09:58	7802988	Insights	Dashboard Opened	1	7

Table 9. Event Log Result after Adding Group Flag

[25] https://sparkbyexamples.com/pyspark/pyspark-partitionby-example/

The final step in Listing 7 is to coalesce the multiple consecutive steps into one interval step. The event log is grouped by company and group, then aggregated [26] to calculate the minimum and maximum timestamp. The "first" method on line 5 is an aggregated function to output the *feature_name*. Table 10 shows rows with one event have the same start and end timestamp. Rows such as "Email" and "Settings" are coalesced into one row so they contain the duration of the event.

```
1 from pyspark.sql import functions as F
2
3 df = df.groupBy('company', 'group') \
4     .agg(
5         F.min('datetime').alias('start_time'),
6         F.max('datetime').alias('end_time'),
7         F.first('feature_name').alias('feature_name')
8     ) \
9     .drop('group')
```

Listing 7. Create Start to End Timestamp for Event

	company	start_time	end_time	feature_name
0	7802988	2021-01-07 00:00:00	2021-01-07 00:00:00	Email Sync
1	7802988	2021-01-07 15:01:21	2021-01-07 15:01:21	Pipeline
2	7802988	2021-01-07 15:01:28	2021-01-07 15:08:56	Email
3	7802988	2021-01-07 15:09:13	2021-01-07 15:09:13	Contacts
4	7802988	2021-01-07 15:09:41	2021-01-07 15:09:46	Settings
5	7802988	2021-01-07 15:09:48	2021-01-07 15:09:48	Contacts Timeline
6	7802988	2021-01-07 15:09:58	2021-01-07 15:09:58	Insights

Table 10. Event Log Result after Group Aggregate Operation

3.7.6 Sessionization

Sessionization is the process of breaking down the user interaction with the software into sessions where the user is using the software. The process of sessionization is based off of session identification as stated by Like Zhuang [12]. From their paper, a session is a series of page views of a single user when accessing a certain domain. The purpose of session identification is to separate the web logs of one user into individual sessions for further analysis. They used a process that identified sessions by determining a threshold between page accessing; if the threshold is exceeded, the user has started a new session. This method is called time-based heuristics as determining the appropriate threshold is difficult and the result may not be optimal. The method we employ is called time-based delimiters, where we use the built-in session information from Pipedrive to construct a session that starts when the user logs in and ends when they log out. If no logout event occurs, the session will end at a maximum of 12 hours.

[26] https://sparkbyexamples.com/pyspark/pyspark-aggregate-functions/

Section 3.5.6 describes multiple types of sessions the user can be in at Pipedrive. Long sessions cannot be used as these cover a timeframe of when the user is not actively using Pipedrive. Only short sessions will be used in the sessionization process, as these have bounding times of user log in and user log out. A potential problem with this approach, is that we will have to throw away many sessions and this might introduce a bias to short session users. It could be the case that longer sessions are more likely to convert, but this will be lost. Also there may be large holes in user activity, as a user will likely start off as a short session and then transition into a longer session. After sessionization, the case duration will no longer be the length of the trial, but the length of the time the user is logged into Pipedrive.

Table 11 shows the counts of the number of sessions for each flag type. Four different flags are available in the *f_auditsession* table, but only two flags relate to session length. The mobile sessions are always considered as long sessions, so in our sessionization code, we only include sessions with *flag_longsession* equals zero.

flag_longsession	flag_mobile	t2p_count	t2e_count
0	0	66404	15550
1	0	3511	1628
1	1	2309	914

Table 11. Session Flag Count

Listing 8 shows the code to create the session. Lines 1 to 9 show extracting all the unique user IDs from original event log stored in *df_behavior* and filtering out empty users. Lines 12 to 28 uses the unique user IDs to retrieve all the short user sessions between the start and end timestamps. Lines 30 to 45 join the user sessions with the event log so that events are tied to the correct user if the user IDs match and the event is between the time of the session.

```
1  df_beh = spark.sql("""select * from df_behavior""")
2  df_company_users = df_beh.withColumnRenamed('caseId_company', '
       key_company').withColumnRenamed('caseId_user', 'key_user') \
3       .dropDuplicates(['key_company', 'key_user']) \
4       .select('key_company', 'key_user')
5
6  # Drop rows where key_user is -9000. Feature data is not applicable.
7  df_company_users = df_company_users.filter(df_company_users.key_user
       > 0)
8
9  df_company_users.createOrReplaceTempView('v_company_users')
10
11 # only consider short sessions
12 df_session = spark.sql("""
13     select sess.id_auditsession,
14         sess.key_user,
15         sess.created_date as start_date,
16         (case when sess.ended_dt = date('1000-01-01') then sess.
               expired_date else sess.ended_date end) as end_date
```

```
    from dw.f_auditsession sess
  inner join (
        select key_user from v_company_users
    ) as company_users
        on company_users.key_user = sess.key_user
    where sess.created_dt >= '%s'
        and (case when sess.ended_dt = date('1000-01-01') then sess.
            expired_dt else sess.ended_dt end) <= '%s'
        and sess.flag_longsession == 0
    order by sess.key_user, sess.created_date
""" % (signup_start_dt, behavior_end_dt))

df_session.createOrReplaceTempView('df_sessions')

df_sessionized = spark.sql("""
    select beh.datetime,
        sess.id_auditsession as caseId_session,
        beh.activity_feature_name,
        beh.activity_feature_action
    from (
        select * from df_behavior
    ) as beh
    inner join (
        select * from df_sessions
    ) as sess
        on beh.caseId_user = sess.key_user
    where beh.datetime between sess.start_date and sess.end_date
""")

df_sessionized.createOrReplaceTempView('v_export')
```

Listing 8. Sessionization of Event Log

3.8 Descriptive Data Analysis

Of the companies that signed up to use Pipedrive between the beginning of January to February 2021, 83.3% of company's trial expired and 16.7% became paying customers. In terms of distinct behavior events, the companies that paid generated 1.49M events, while the companies that expired generated 403K events. The companies that paid are 5 times fewer, yet generated 3.7 times the number of events. The paid companies generated 18.7 times the number of events than companies that expired.

3.8.1 Trial Lengths

Figure 7 shows the dates when a company signs up for a trial to when the trial becomes paying or expired, according to how we split them in Section 3.7.1. Zeppelin was used to gather the data for the three different SQL queries, but it was not sufficient to produce a figure combing all the data into one. The data was exported by CSV and then manipulated with a Python Pandas using a Jupyter Notebook as described in Section 3.4.5. The code to create this figure is described at Listing 10 in the Appendix.

The figure shows the dates of all sign-ups (green line) in the data being analysed, which is the first two weeks of 2021. When a company first signs up they start the trial, which is ended by either the trial expiring (orange line) after a normal period of two weeks, or the company starts paying (blue line). In all of these dates, a big dip forms in the middle, representing the weekend period where fewer sign-ups are normal. It makes sense that the expire and paying lines also follow this trend. For both paying and expire companies, there is a significant drop off after the normal two week trial period. This is expected as getting a promotion to extend the trial is supposed to not be available to everyone. The ratio of trial extensions for companies that pay is higher than the ones that expire. It also appears possible to have the trial expire under the normal two week period, which we cannot explain. The few cases this happens to makes this insignificant. After an extension of two weeks from the normal trial period, the number of trials drop off to almost zero. This means that the number of trial extensions lasting this long are also not significantly important.

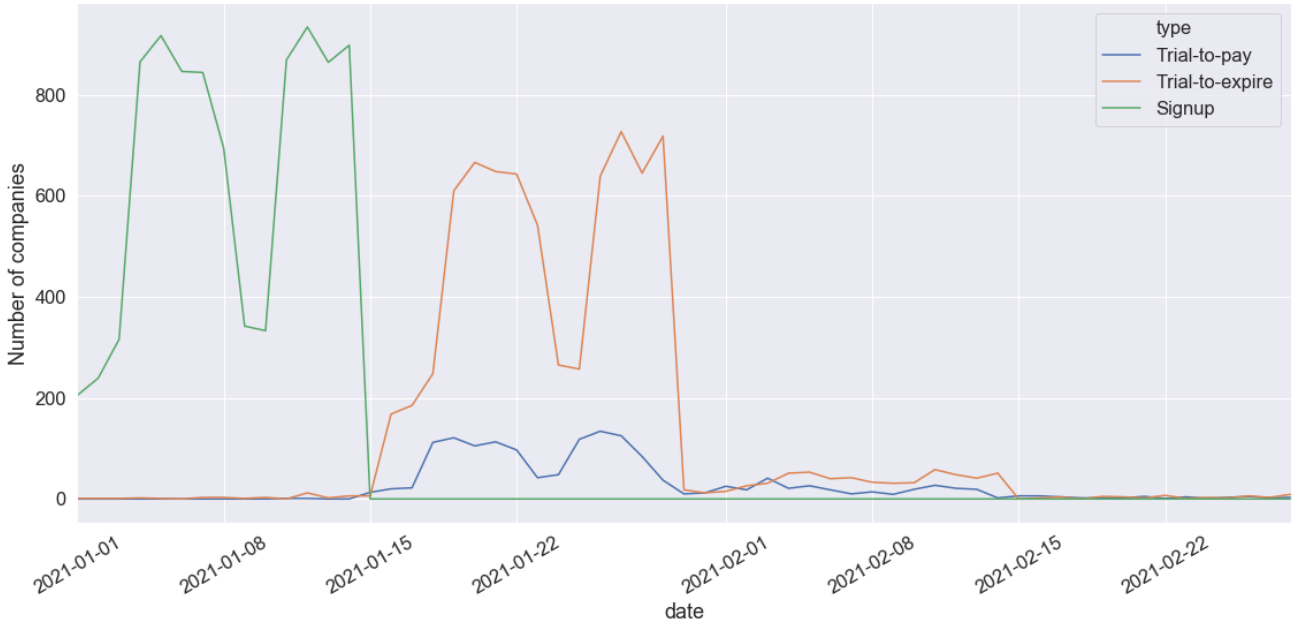


Figure 7. Trial Length Differences

3.8.2 Paying After Trial Expires

Figure 8 shows that when trials expire, 75% of the companies also pay within the same day. This also shows that a significant minority that will pay within 5 days. Only a few outliers pay for Pipedrive after one week. We used a grouping of 5 after 5 days, because of many small percentages, so the data was not visible. When splitting the event log into paying and expire, this difference must be taken into account. For our event log, if a company's trial expires before paying but subsequently pays within 5 days, we will consider that company as going from Trial-to-Pay instead of Trial-to-Expire. The code for this figure is located at Listing 9.

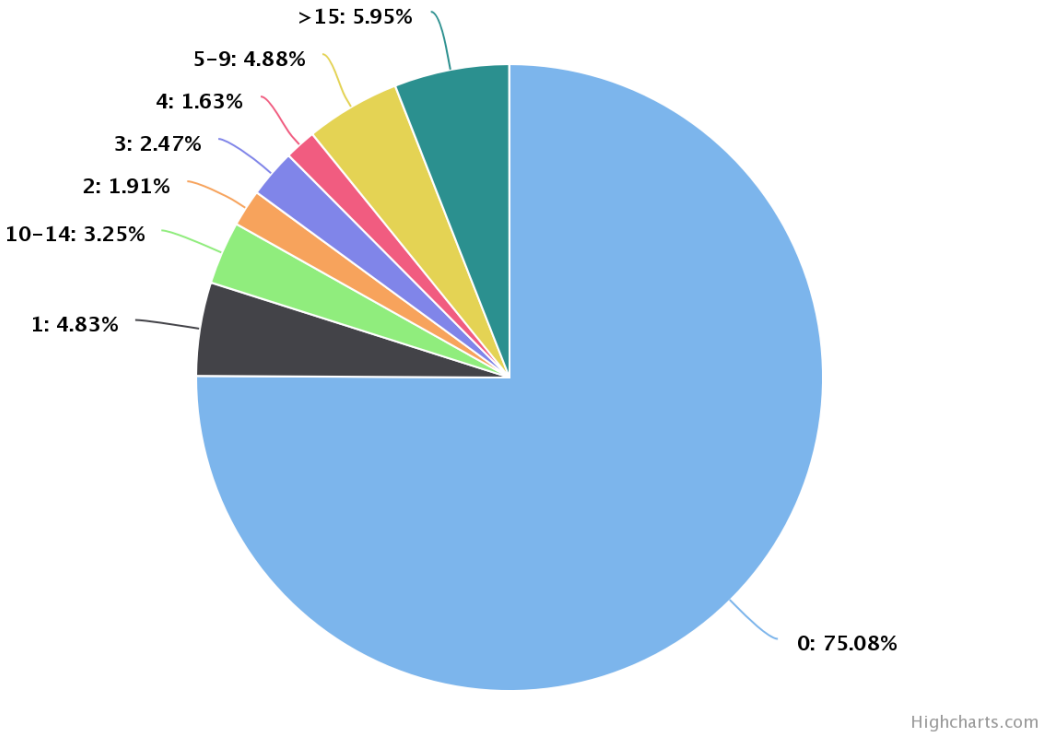


Figure 8. Days Between Paying and Trial Expire

3.8.3 Subscription Plan Usage

Table 12 shows how many times the subscription plans are changed between T2P and T2E. The Essential and Advanced plans have less features available to the user which affects feature usage. Listing 11 shows how this table is created. Each run was done on T2P and T2E and the results were joined together. This code assumes the *df_companies* temporary view has already been created from the event log.

It is interesting how the T2E has a much lower usage of the plans with reduced features than the T2P. One theory is that users thinking of paying for Pipedrive could be more likely to switch plans to find one that better fits their needs.

Plan Description	T2P Count	T2E Count
Professional Plan	1470	7295
Essential Plan	738	490
Advanced Plan	639	219
Total	2847	8004

Table 12. Subscription Plan Change Statistics

3.8.4 Session Lengths

Table 13 shows how the session lengths differentiate between the T2P and T2E data-sets. Short sessions are defined as less than 12 hours and they either end when a user logs out

or when the session expires in 12 hours. Most of the short sessions end at expiration instead of the user clicking log out. A long session occurs when the user clicks the "Remember Me" button on the log in screen. These sessions expire at 30 days or can be extended if the user is on the mobile app of Pipedrive. It is interesting how there are fours times more long sessions for trials that expire.

Log	less than 12h	equal to 12h	between 12h and 30d	equal to 30d	more than 30d	total sessions
T2P	24.1%	64.3%	4.6%	2.2%	0.33%	74025
T2E	16.1%	69.5%	5.3%	8.5%	0.57%	18086

Table 13. Session Length Difference

3.8.5 Events at Midnight

Figure 9 shows that 15.6% events in the months of January and February occur at the time 00:00:00 or midnight. This shows the granularity challenge of the event log as described in Section 2.3. Listing 12 shows how these figures are made, after obtaining code from posting on Stack Overflow [27]. The most likely case is that a couple of events are not logging the time.

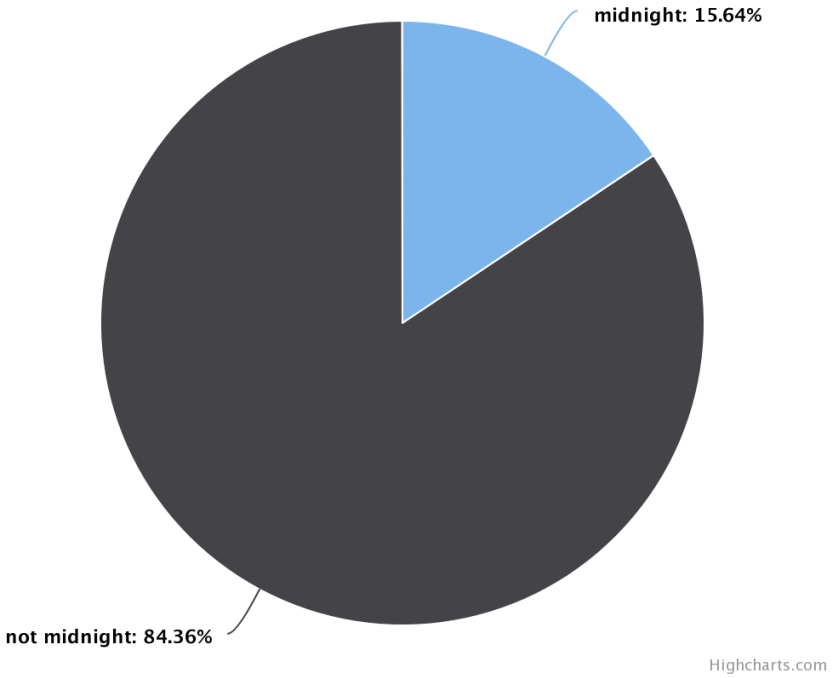


Figure 9. Percentage of Events Missing Time

Table 14 shows the events that are missing the time attribute in the timestamp. Listing 13 is the code to generate this list. To identify this list, we searched for events

[27] https://stackoverflow.com/questions/67577266/sql-match-timestamps-with-time-only-parameter-to-group-and-count-unique-times-a

that always had midnight timestamps. Events do occur exactly at midnight, though not very likely. What is interesting is these events are all user statuses, not actual behavioral events. They show that a user has this feature enabled. Each day, if this feature is enabled, then that event will show up in the *f_behavior* table at midnight. Talking to a data analyst at Pipedrive, this is indeed the case, and PMs want this behavior. But in the case of an event log where each event needs to be sequential so that process mining can occur, all of these events must be taken out. The only way these events can be used in our thesis, is in terms of usage statistics. Therefore, in the event log creation, a new option is added to keep or throw out these events depending on the analysis to be done.

	id_behaviorfeature	feature_name	feature_action
0	typ_52	2FA	2FA Enabled
1	typ_35	Calendar Sync	Calendar Sync Enabled
2	typ_33	Contact Sync	Contact Sync Enabled
3	typ_73	Custom fields	Deal Custom Field Enabled
4	typ_74	Custom fields	Organization Custom Field Enabled
5	typ_75	Custom fields	Person Custom Field Enabled
6	typ_77	Custom fields	Product Custom Field Enabled
7	typ_36	Email Sync	Email Sync Enabled
8	typ_72	Marketplace	Marketplace Adoption
9	typ_53	Permissions	Advanced Permission Sets Enabled
10	typ_46	SSO	SSO Enabled
11	typ_54	Security Alerts	Security Alerts Enabled
12	typ_45	Security Rules	Security Rules Enforced

Table 14. Events Missing Time Attribute

4 Analysis and Interpretation Stage

This chapter will deal with stages three and four of the modified PM^2 methodology. The three main types of event logs being analysed are:

1) Normal event log where the Case ID is the company, meaning there is a single timestamp for each event. The case duration lasts for the entire duration of the trial and all the events occur instantaneously.

2) Aggregated event log where the Case ID is still the company but the events have a start and end timestamp which enables event duration. These event logs lose the *feature_action* label as a consequence of the aggregation.

3) Sessionized event logs where the Case ID is the Session ID so the case duration is the length of the short sessions. All long sessions are thrown away as this would be the same as Type 1.

This stage starts with the automated process discovery where sanity checks are performed on the dataset to determine correctness. The structure of the process is understood, locations of reword/repetition, key decision points, and parallel branches are identified. Then depending on the project, either conformance checking or performance mining are performed. This project uses performance mining to identify bottlenecks and wastes in the process. Finally variant analysis is used to dig deeper and understand the performance differences between two countries, regions, types of customers, etc. The steps to achieve each of these components are listed in Table 15.

Automated Process Discovery	1. Flow Analysis 2. Filtered Flow Analysis 3. Frequency Analysis 4. Handoff Analysis
Compliance Checking	1. Flow Compliance Checking 2. Temporal Compliance Checking 3. Resource Compliance Checking 4. Exception Analysis
Performance Mining	1. Bottleneck Analysis 2. Workload & Demand Analysis 3. Rework Analysis 4. Over-processing Analysis
Variant Analysis	1. Flow Comparison 2. Frequency & Rework Comparison 3. Bottleneck Comparison

Table 15. Overview of Analysis Components

A couple different process mining use cases are explained by Marlon [28]:

- **Achieve transparency** to identify your real processes based on actual data from your IT systems.
- **Increase efficiency and productivity** to analyse the actual reasons for their occurrence and discover the potential for automation.
- **Enhance quality and customer experience** to understand the real customer journey and the different outcomes of your processes to sustainably optimize the customer experience.
- **Support governance, risk and compliance** to discover process cases that deviate from your regulations and understand the reasons behind them.

[28] https://courses.cs.ut.ee/LTAT.05.025/2021_spring/uploads/Main/Lecture8.pdf

- **Facilitate agility** to identify undocumented process changes and always keep an eye on your real processes.

Finally, the results are interpreted and validated with the project stakeholders. The hypotheses are formulated and scenarios are discussed. Statistical tests can validate the findings when appropriate. The impact of the potential changes are simulated.

4.1 Automated Process discovery

The first step in automated process discovery is to perform sanity checks to determine if the event log is correct. A problem encountered was when the average case duration lasted the entire period from January to the end of February. The Case ID was the company, therefore the average case duration should be a little more than the length of the trial, which has a starting period of two weeks. What was found instead was an average case duration of 1.5 months. The cause was a missing date delimiter in the SQL on the behavioral user data. After fixing, the case duration came down to 2.27 weeks for T2P as shown in Figure 10. Also shown is the T2E much lower average case duration. This is expected because these users are more likely to stop using Pipedrive at the end of the trial period.

Figure 10. Comparison of Company Case Duration Summary

4.1.1 Event Log Overview

Type 1 and Type 2 event logs are shown in Figures [11, 12, 13]. T2P users have an average case duration of 2.2 weeks with 1.3k cases (companies), while T2E users have an average case duration of 4.0 days with 3.9k cases (companies). There are almost no duplicated cases, meaning that every user flow is unique. Figure 11 shows that active cases build up until they reach the peak at January 14th. Then the trials started on January 1st begin to expire. The T2E cases decline faster than T2P because the trial periods for T2P can be longer than two weeks. In Figure 12, the event activity correlates with active cases as shown in Figure 11. The weekends are clearly visualized with a level of lower activity compared to the weekdays. Figure 13 shows the length of activity on the companies. T2E users have many cases that stop activity after the 1st day meaning that users use Pipedrive for one day and then never use it again. T2P users have a large number of cases ending at the two week period and a smaller portion of cases ending at 30 days and then 45 days. There are a small portion of T2P users that end the case before two weeks. We have not found an explanation for this, however, the proportion of users are small, so the effect should be negligible.

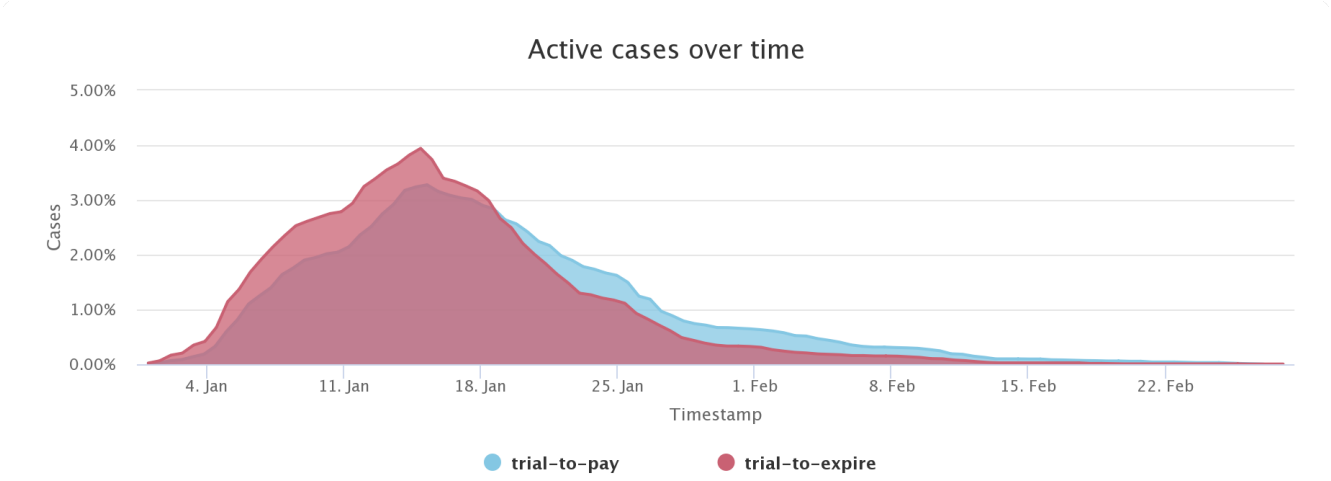


Figure 11. Company Case ID - Active Cases over Time

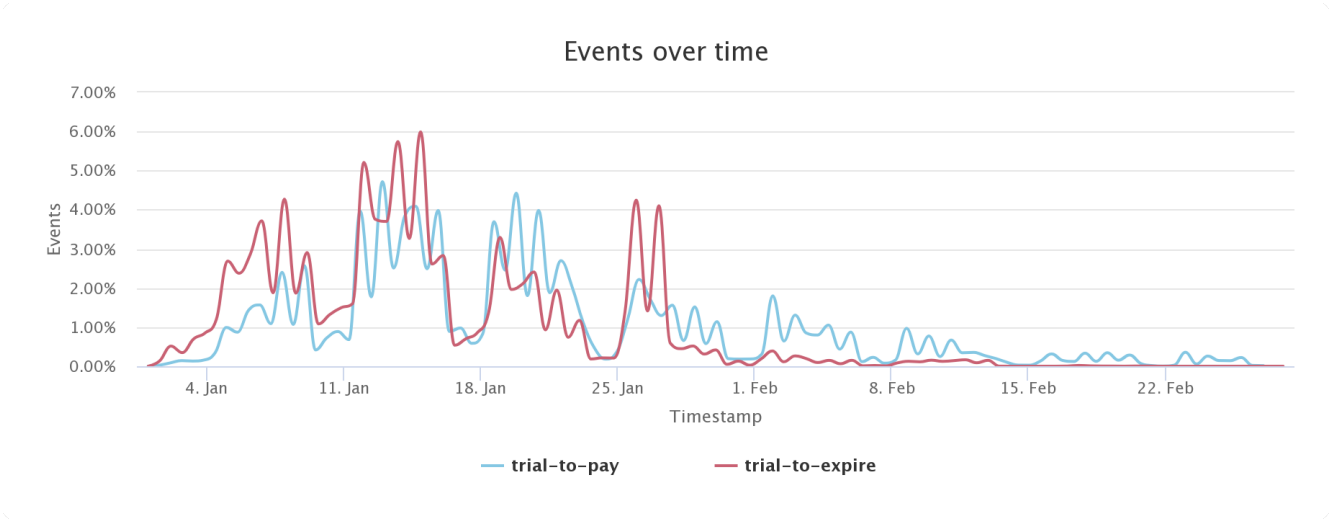


Figure 12. Company Case ID - Events over Time

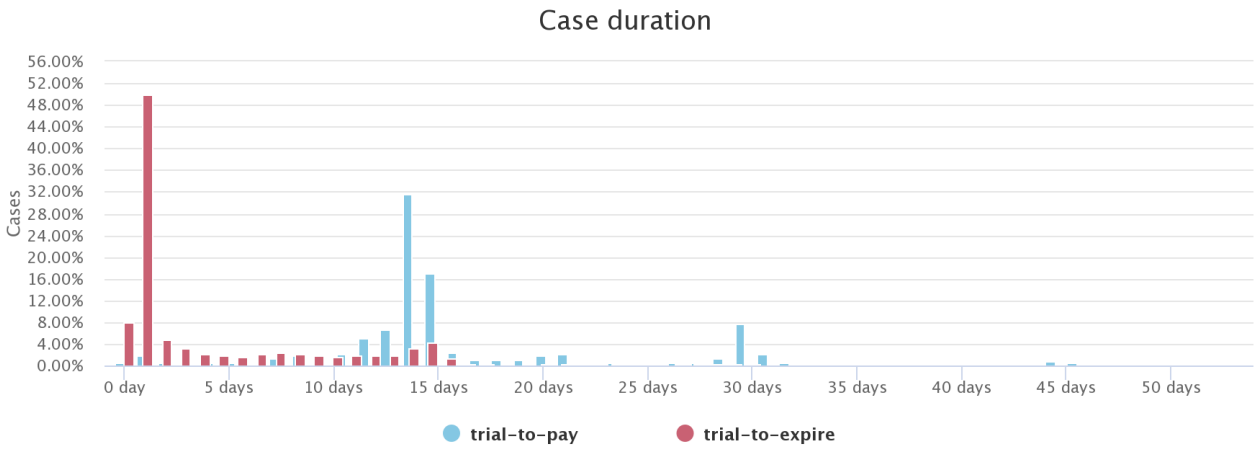


Figure 13. Company Case ID - Case Duration

Type 3 event logs are shown in Figures [14, 15, 16]. T2P users have an average case duration of 3.8 hours with 21k cases (sessions), while T2E users have an average case duration of 2.0 hours with 14k cases (sessions). Since the short sessions have a maximum length of 12 hours, Figure 14 looks very similar in structure to Figure 15. However, looking at Figure 16, it can be seen that the case duration goes up to 20 hours. This shouldn't be possible given how short sessions work. These extra long sessions from 12 to 20 hours should be negligible since they barely occur. This figure also shows that the majority of short sessions last a maximum of one hour.

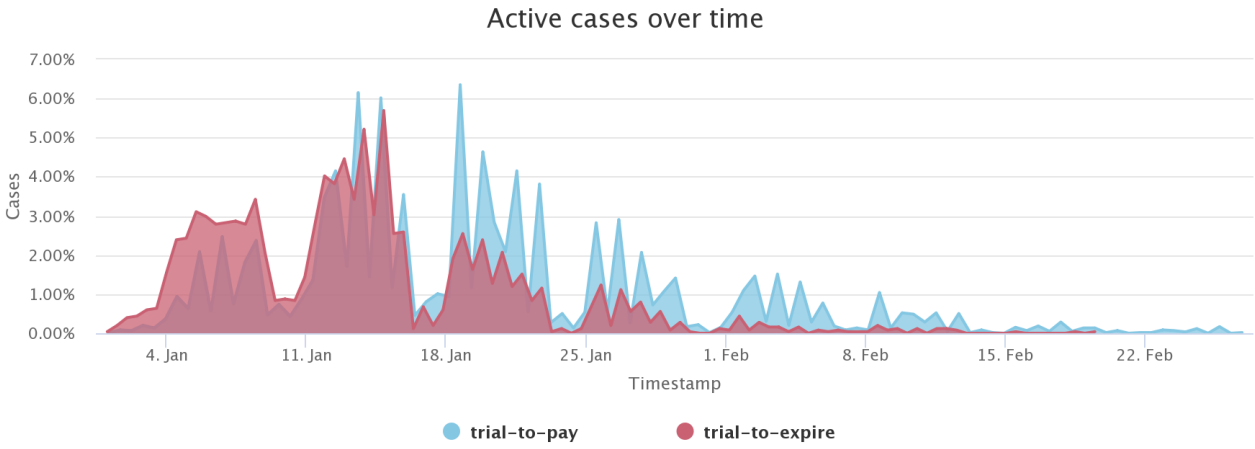


Figure 14. Session Case ID - Active Cases over Time

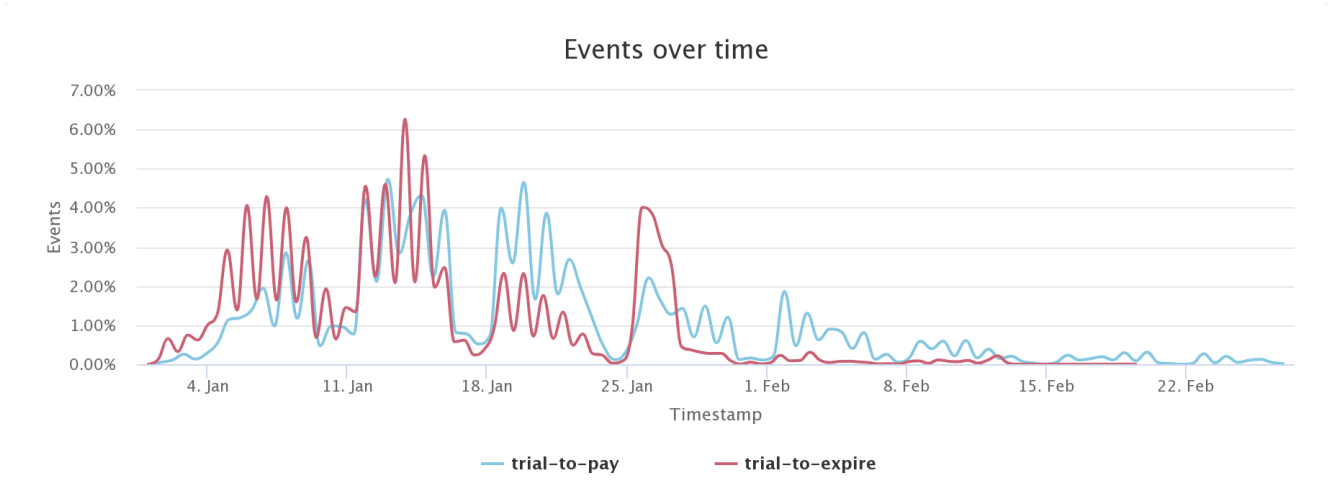


Figure 15. Session Case ID - Events over Time

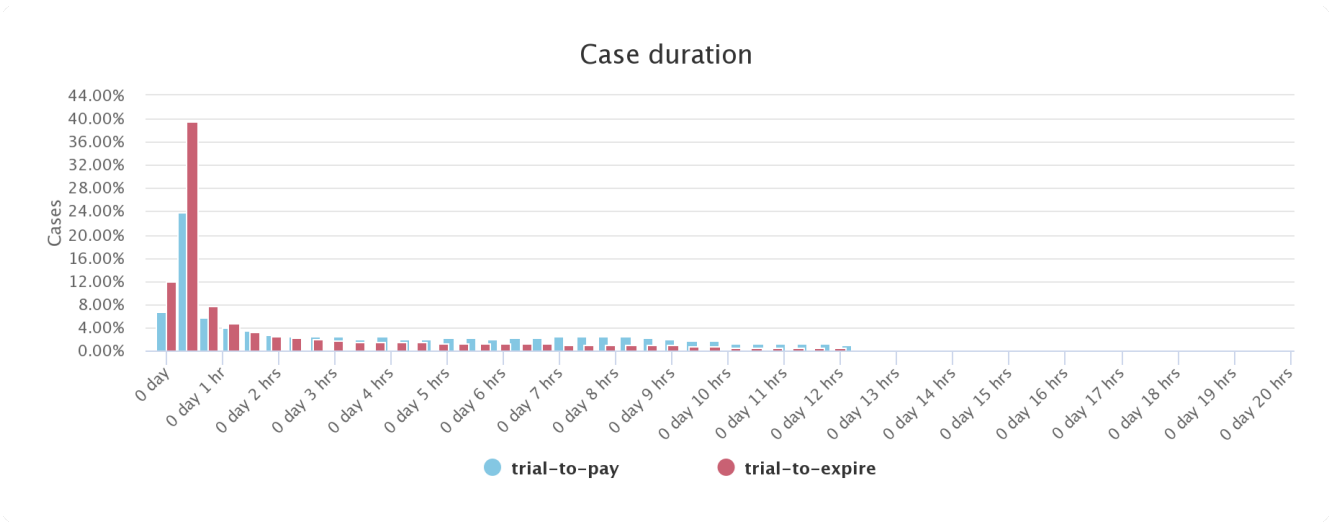


Figure 16. Session Case ID - Case Duration

4.1.2 Flow Analysis

Flow Analysis consists of first understanding the process structure and main case variants which consists of:

- Identify parallelism, branching points, and rework loops
- Analyse case entry and exit points
- Check for incomplete cases

We start off by showing the process flow of Trial-to-Pay and Trial-to-Expire in Figures [17, 18] for Type 1 event logs. These process flows show the frequencies of

each activity for all activities. The arcs are filtered down to show the most common arcs so that the flows are understandable.

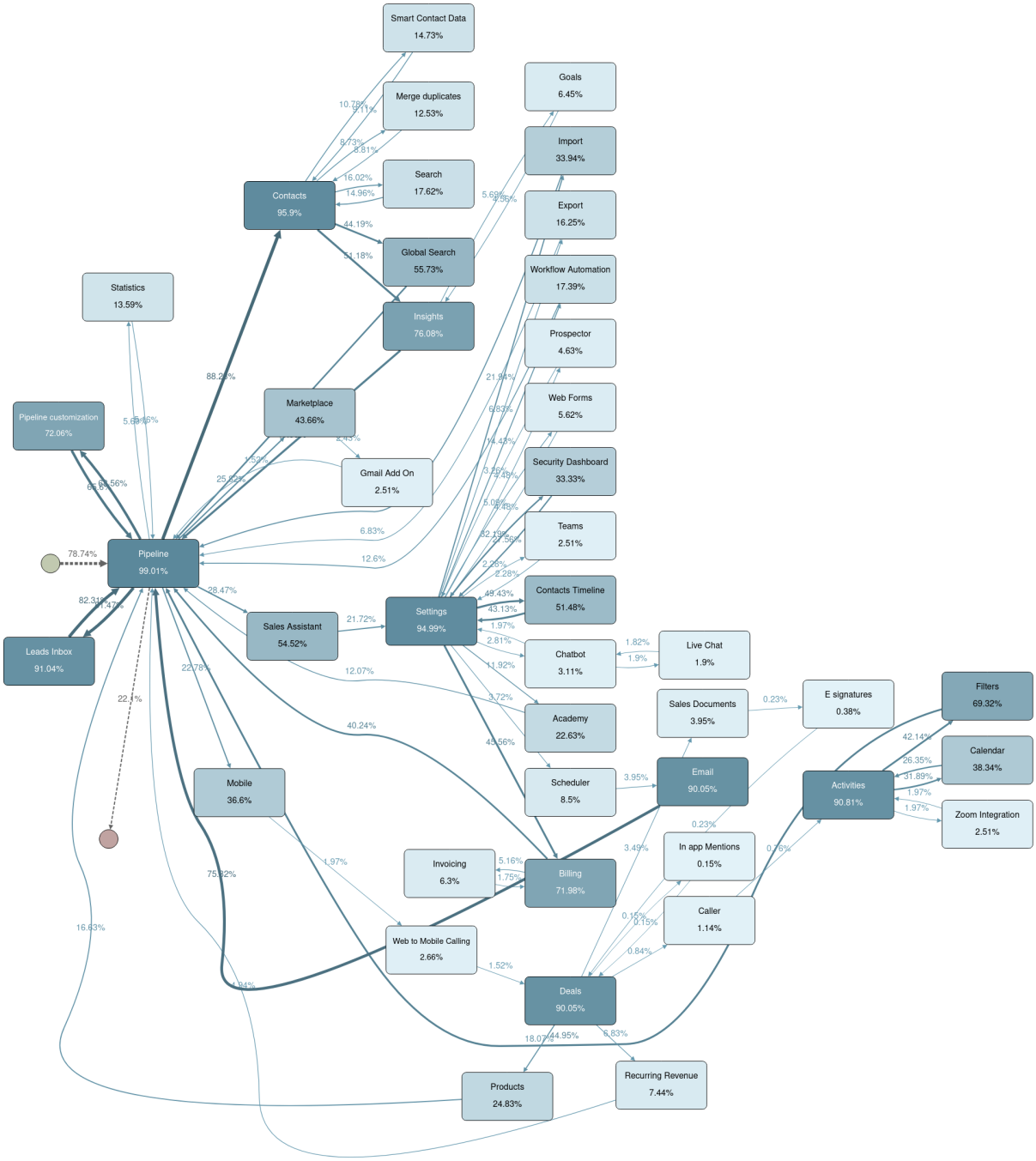


Figure 17. Company Case ID - Trial-to-Pay Process Flow Overview

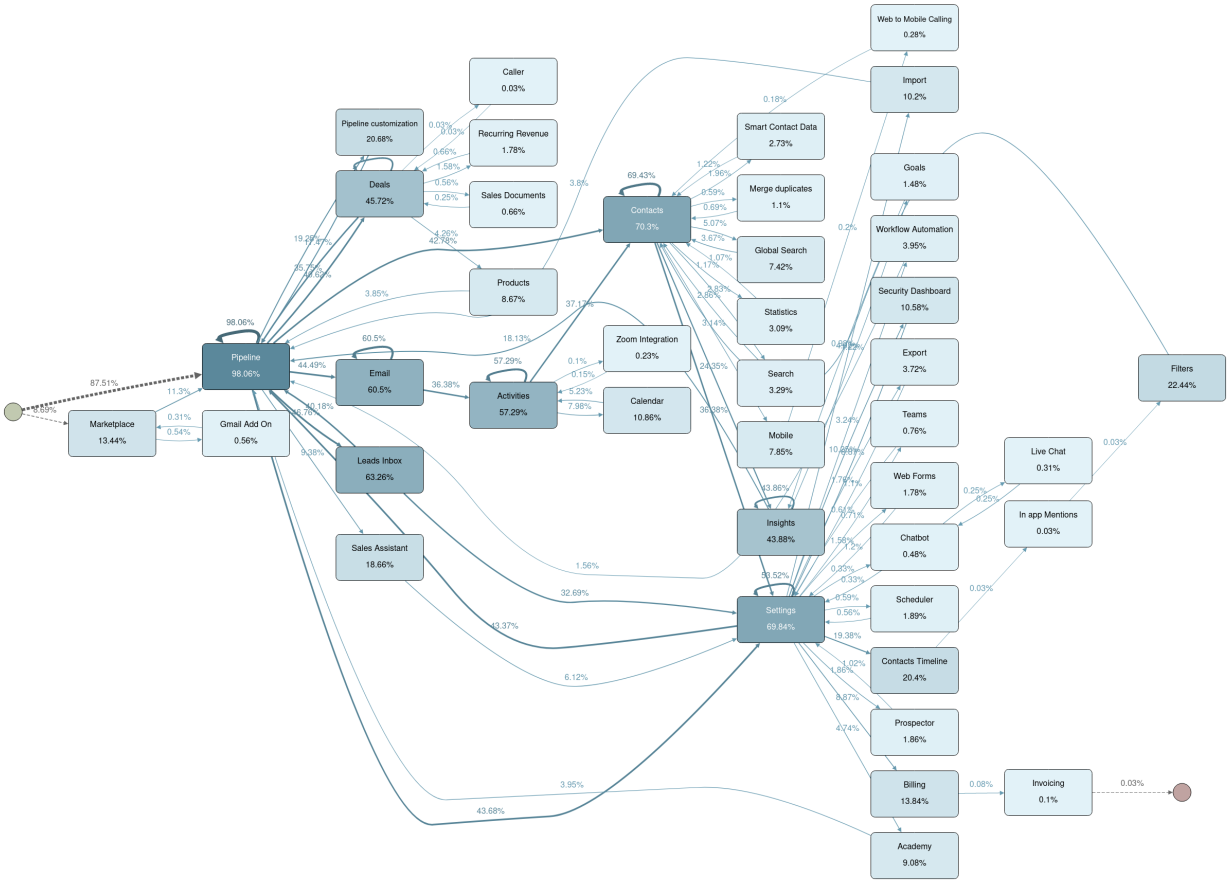


Figure 18. Company Case ID - Trial-to-Expire Process Flow Overview

The structure of between the Trial-to-Pay and Trial-to-Expire process flows are very similar. The top entry point is the “Pipeline”, the first page viewed upon login. The key branching points are “Pipeline”, “Contacts”, “Settings”, “Deals”, and “Activities”. These features are full pages and are used to access other features. All of these except “Settings”, are located on the left hand quick navigation bar so they are easily found by users. From these branching points, the behavior is described as a flower pattern. The flower is formed by the shape of the arcs and nodes which form petals. This shows the most common behavior is to navigate to another page and then back to the root page. This is easily seen in Figure 17 on the “Pipeline”, “Settings”, and “Contacts” nodes.

There are no clear exit points for these processes. As shown in the figures above, the process ends when the user converts or abandons on a day time scale. For the Type 3 event log, the process end is the user clicking log out or the session expiring at the end of the day. Therefore, the process end is the feature the user last used, which does not have any real meaning for analysis. As seen in Figure 17 the “Pipeline” was the most common feature to be used before the user logged out or the session expired. This makes sense as the log out button can be accessed from the “Pipeline” view and it is the most

commonly used features.

Rework loops are also shown in the processes. These are identified as loops that start and end on the same process such as in Figure 19. In Figure 18, the rework loops are shown on “Pipeline”, “Email”, “Activities”, “Contacts”, “Insights”, and “Settings” activities. These loops show multiple actions being done on these large pages. The event log is divided up into more detailed event usages with the *feature_action* label as seen in Section 3.5.5. For Figure 17, the rework loops do not appear because the arcs are filtering to a narrow 5% while the other figure has the arc filter at 10%. This means that repeating a node is not in the top 5% of arcs.

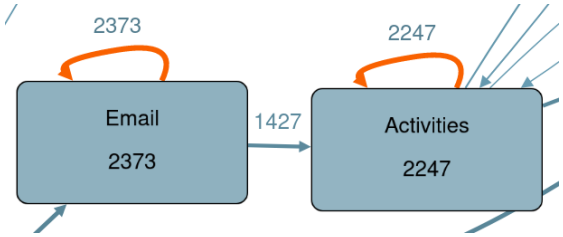


Figure 19. Rework Loops Example

These processes have almost infinite parallelism. This means that from one feature, a user is able to jump to any other feature currently on the screen. The entire navigation menu and the features on the page are all valid paths to the user. This makes it a challenge to analyse these processes as they gain complexity very fast.

The “Mobile” feature stood out, as “Mobile” is not a page or feature inside of the Pipedrive app. Figure 20 shows the *feature_actions* of “Mobile”. This shows that this feature is used to indicate whether an Android or IOS mobile phone was used to access Pipedrive. This is basically a status feature as described in Section 3.8.5 and cannot be used in future analysis. This was not caught in the data cleaning because the timestamps for this feature were not missing the time data.

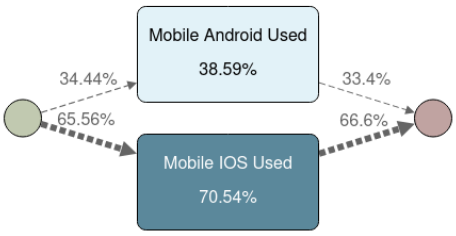


Figure 20. Mobile Feature Actions

4.1.3 Frequency Analysis

Frequency analysis analyses the most frequent activities and the relationships between them. Figure 21 gives an overview of the relative distribution of activity frequency. It can be seen that T2P users use more varied features than T2E users. T2E users have a much higher percentage of usage of general page features such as “Pipeline”, “Contacts”, “Settings”, “Leads Inbox”, “Email”, “Activities”, “Deals”, and “Insights”. This suggests that the usage of the general pages does not correlate positively to user conversion. The “Billing” feature has higher usage for T2P users, which makes sense, because to become a paying user they need to go through the Billing page. From our observation, features that encompass customization and integration stand out with their higher usages among T2P. Features that customize Pipedrive are “Pipeline customization”, “Filters”, and “Custom Fields”. Features that integrate Pipedrive with other tools are “Email Sync”, “Calendar Sync”, and “Contact Sync”. The “Global Search” and “Search” features have a higher usage for T2P, probably because they use the software more, therefore searching becomes a more valuable tool.

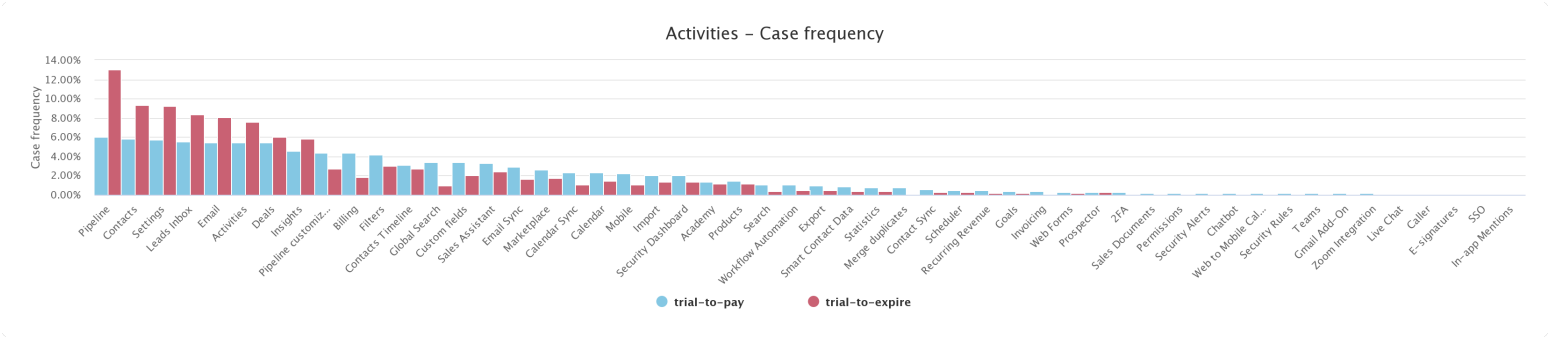


Figure 21. Activity Frequencies

We observed that many advanced features are used more by users that convert. Many of these features are accessed through the “Settings” page. Figure 22 shows a comparison of these usages using a Type 1 event log. On the left is T2P and on the right is T2E. The activity usages are shown in relative scale to enable comparing event logs that have different number of cases. This figure shows that on a per case basis for the entire duration of a trial, T2P uses at least twice as many settings features than T2E. The settings features with a significant difference in usage (more than twice) are “Billing”, “Export”, “Import”, “Live Chat”, “Marketplace”, “Scheduler”, and “Workflow Automation”. The “Export” feature is used to export the Pipedrive data into CSV or Excel. The “Import” feature is used to import the data from the “Export” feature or to import data from another CRM system. These ensure the continuity of the user’s data so that they can continue with the old data and backup the new data.

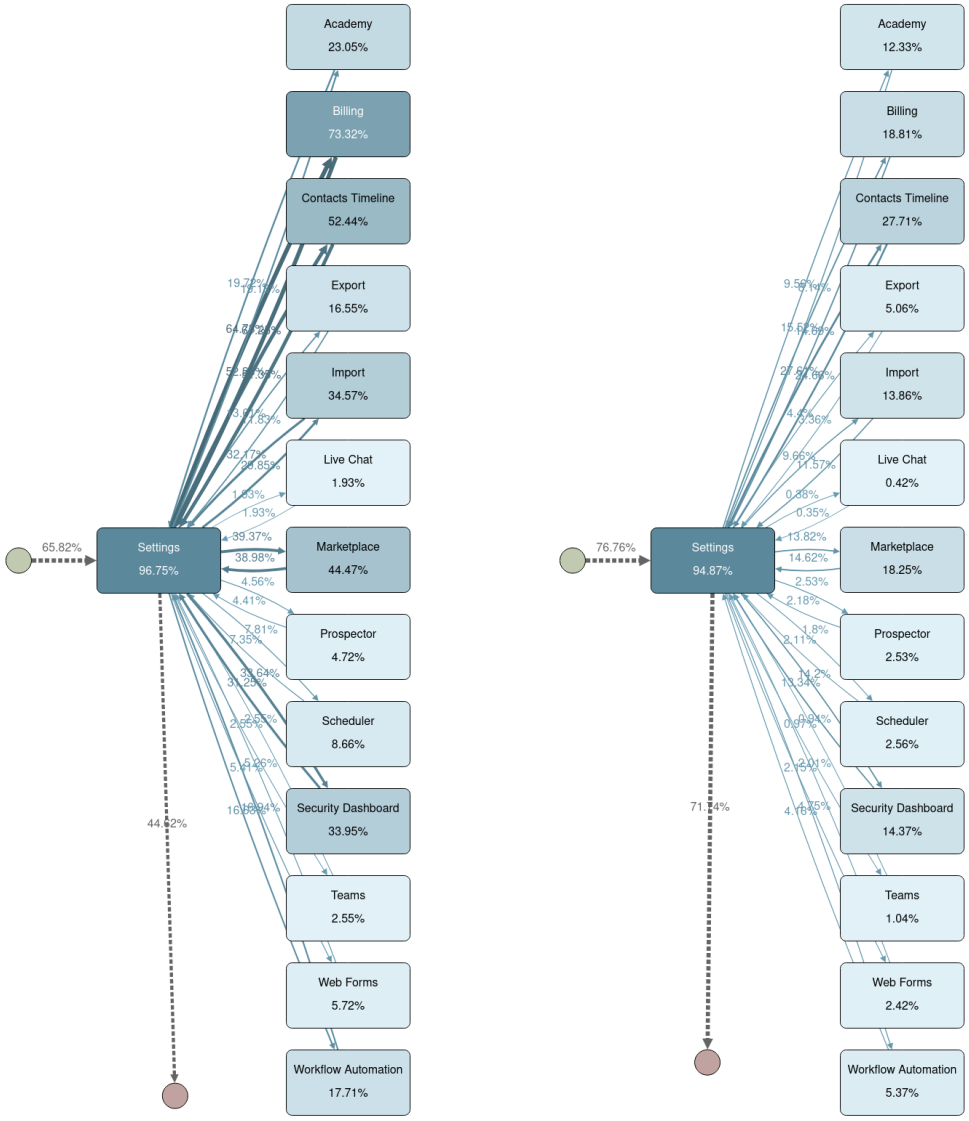


Figure 22. Settings Activity Frequency Comparison

4.1.4 Handoff Analysis

This section analyses the handoffs between workers, teams, groups, and organization units. However, with these event logs, it is not possible as no other resources exist for the handoff. There is only one resource and that is the user.

4.2 Performance Analysis

The next stage of analysis is performance mining, which consists of bottleneck, workload, and rework analysis.

4.2.1 Bottleneck Analysis

Activity bottlenecks are high-effort activities which are identified by a long processing time. Waiting bottlenecks are slow handoffs between activities which are identified by arcs with long waiting times. Resource bottlenecks are not an issue as these event logs do not have multiple resources to switch between.

Figures [23, 24] show the processing time of all activities for average duration and total duration in a logarithmic scale. For the average duration figure, the top activity is being measured in hours, while the smallest activity is measured in seconds, which is why the logarithmic scale is needed. Our observations are:

- For most activities, T2E has longer average duration. For total duration, T2E and T2P close to equal. T2P has slightly longer activity with total duration. This is caused by activity frequency being higher on T2P than T2E.
- "Marketplace" has the longest average duration and the second longest total duration. This feature is a separate page outside of the Pipedrive app where a user can browse for plugins and apps to integrate with Pipedrive. Browsing these can take a long time so the high duration makes sense.
- "Pipeline" has the longest total duration but is 5^{th} on average duration. This activity has the highest frequency so it makes sense to be number one on total duration. The activity does not take as much processing time, which is why average duration is lower.
- "Academy" is second on average duration, but is 12^{th} on total duration. This feature is where users can watch education videos about Pipedrive to learn how to use the product, which is very important for trial users. This page is also outside of the main Pipedrive app.
- The page features are at the top of total duration because they all have very high frequency usage.

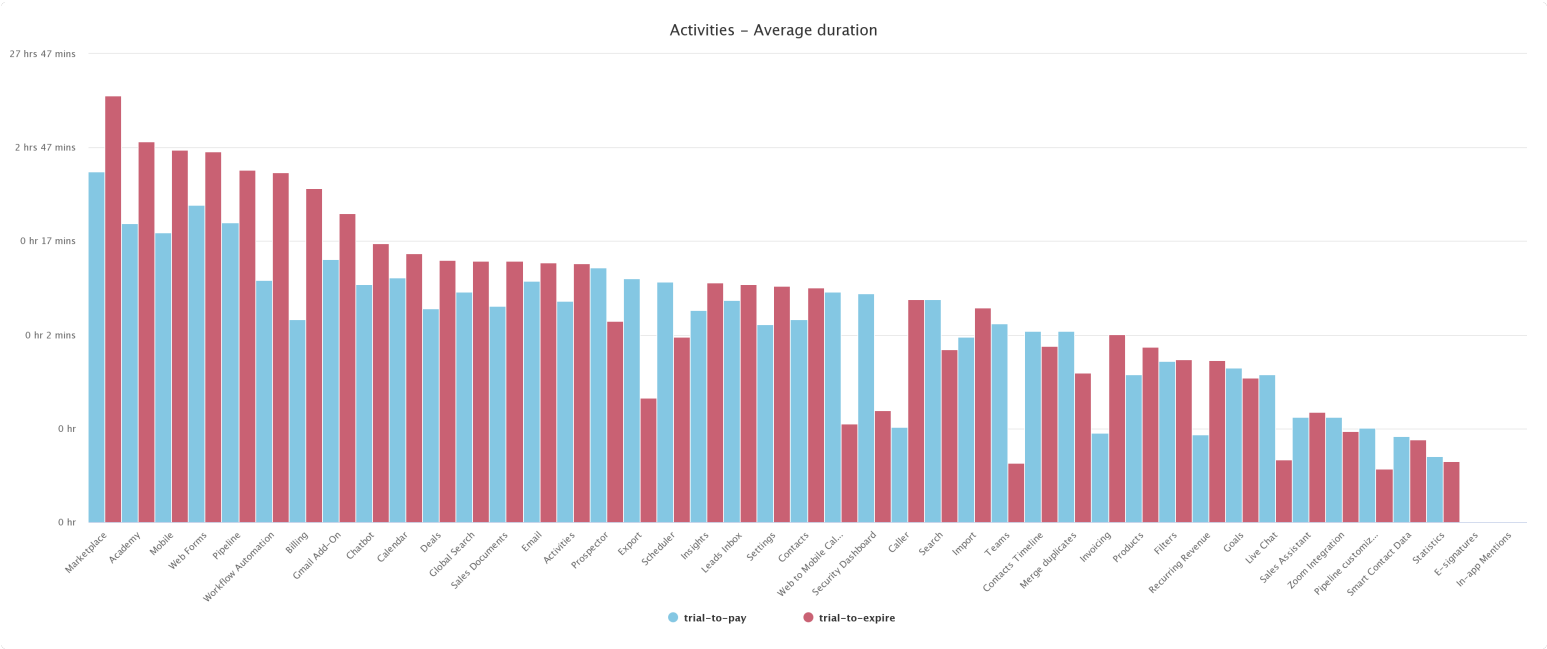


Figure 23. Activity Average Duration

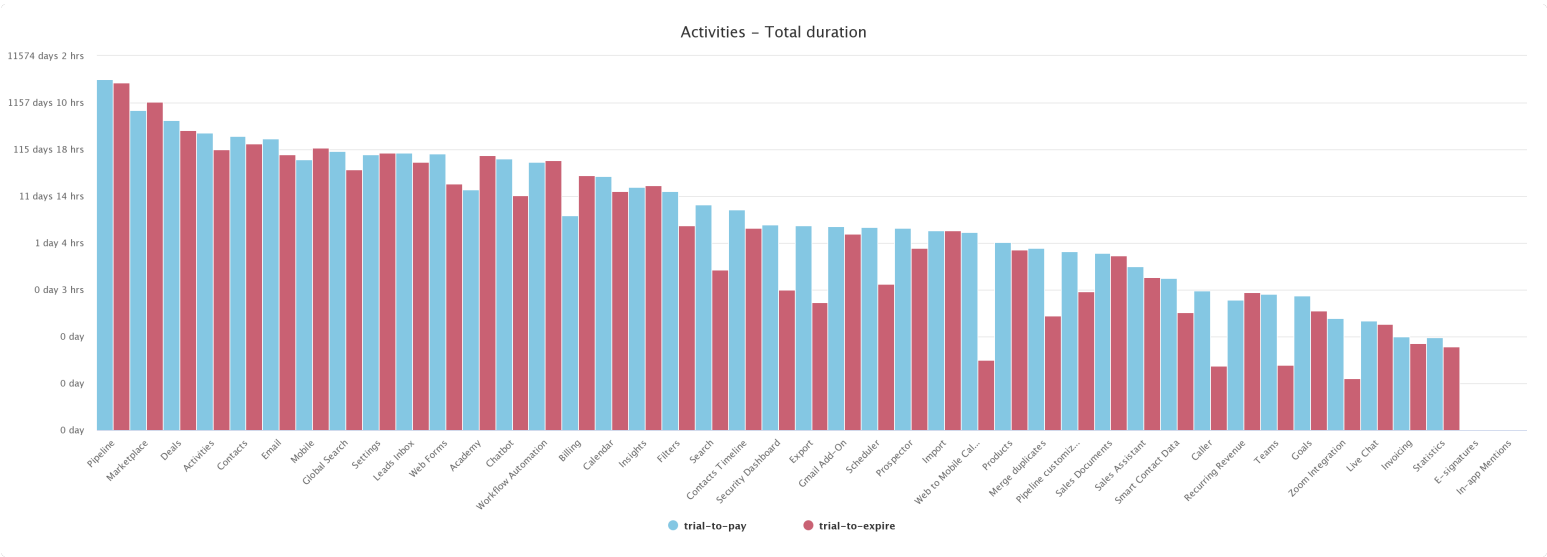


Figure 24. Activity Total Duration

Figures [25, 26] show the slowest average duration with the usage frequency on a Type 2 event log, filtered down to top 30% longest processing time (nodes) and top 10% waiting times (arcs). Our observations are listed below:

- “Marketplace” is 6 times longer for T2E than T2P, while the frequency usage is 3 times longer for T2P than T2E. This means that the users who convert, use “Marketplace” more but for shorter periods of time.

- "Academy" is 6 times longer for T2E than T2P, while the frequency usage is 2.5 times longer for T2P than T2E. This can mean that a small portion of T2E users need a lot of education on how to use Pipedrive. While T2P users do not need as much education because they may already come from a CRM background and just need to see how a couple features work.
- Since the process flows are so unique, there are no significant differences in waiting times.

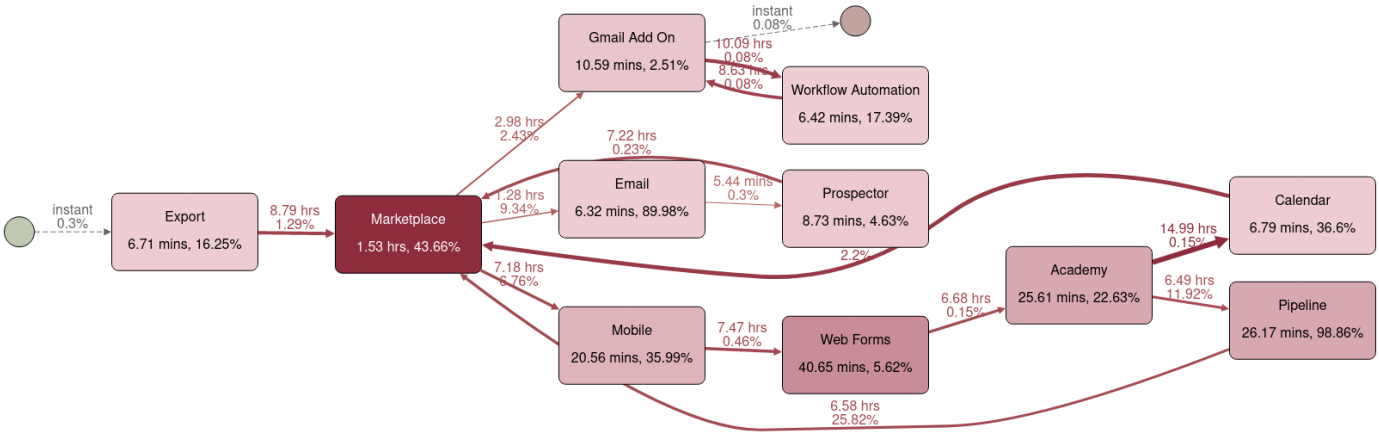


Figure 25. Trial-to-Pay Longest Average Duration

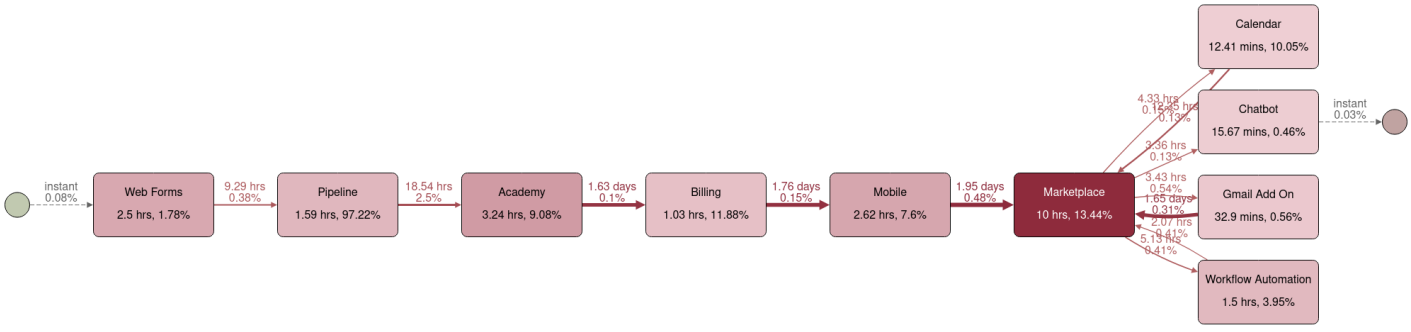


Figure 26. Trial-to-Expire Longest Average Duration

4.2.2 Workload Analysis

This section is to identify overused versus underused resource waste. Overused resources have high total frequency or long waiting times on incoming arcs. Underused resources have low total frequency. It is not possible to do workload analysis on these event logs as this analysis is only relevant for workers who perform activities. In our case, the process is only performed by the user.

4.3 Variant Analysis

Variant Analysis comes in two different ways: process variant and case variant. A process variant is a sequence of process activities with a start and final event. A case variant is the collection of process variants that share common case attributes.

Process variant analysis has been performed incrementally throughout this analysis. Case variant analysis is used to understand the performance differences between the case attributes. The company attribute comparison is shown in Figures [27, 28, 29, 30, 31]. Approximately 45% of the companies had missing data which was subsequently filtered out of these figures.

Figures [28, 29] show that companies in regions such as North America, and Western/Northern Europe have a higher converting rate than companies located in regions such as South America and Asia. Companies in the higher converting regions generally have more disposable money than companies in the regions with a lower conversion rate. Pipedrive charges the same rate worldwide, however during the time frame this data was extracted, a pricing test was being ran on countries with lower GDP, mainly Africa and Asia. Figure 27 shows that companies with an estimated annual revenue between 10M and 50M (medium sized) have a larger conversion rate than companies less than that. Few companies have an annual revenue of greater than 50M, so deductions cannot be made. Figure 30 shows that almost all of the companies are private and the differences are negligible. Figure 31 shows two pie charts based on company industry, the left one being companies that convert and the right one being companies that abandon. These pie charts look too similar to be able to make any distinctions between them.

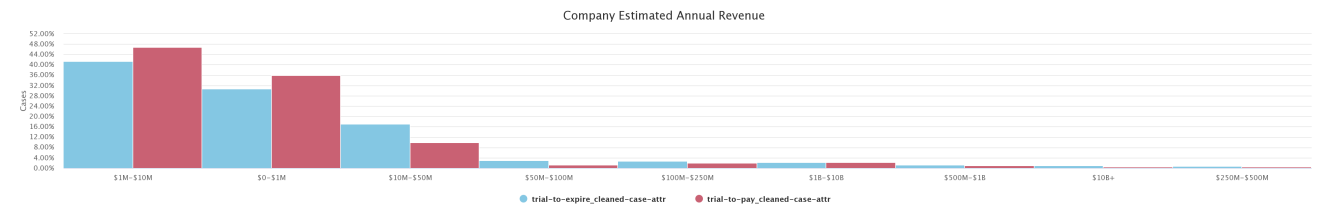


Figure 27. Company Estimated Annual Revenue

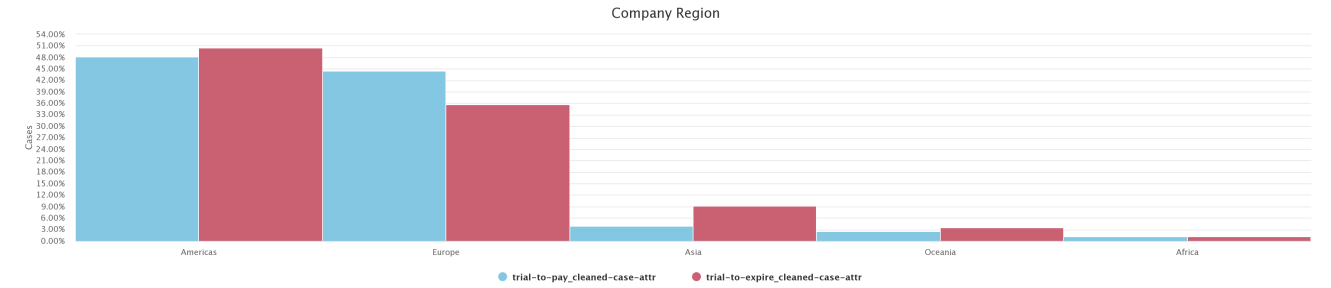


Figure 28. Company Region

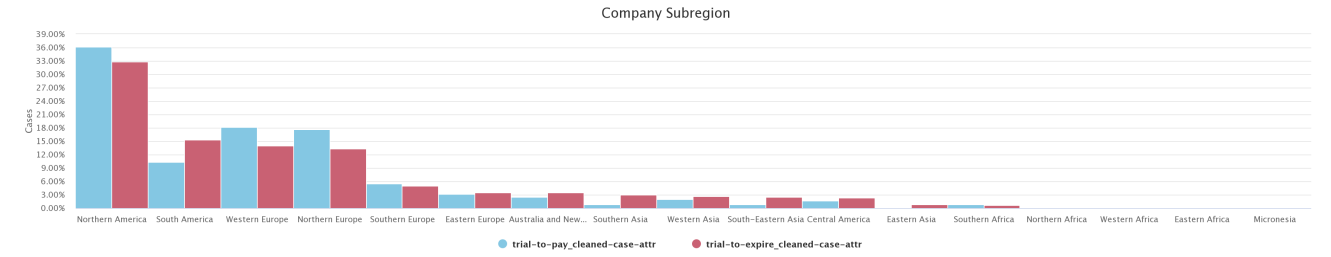


Figure 29. Company Subregion

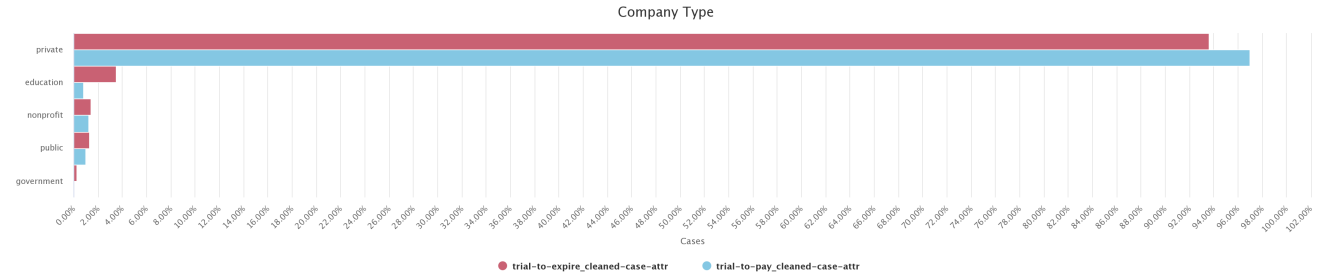
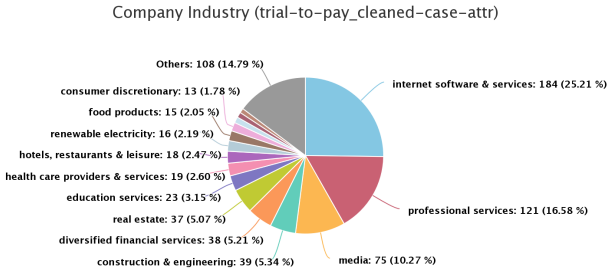


Figure 30. Company Type



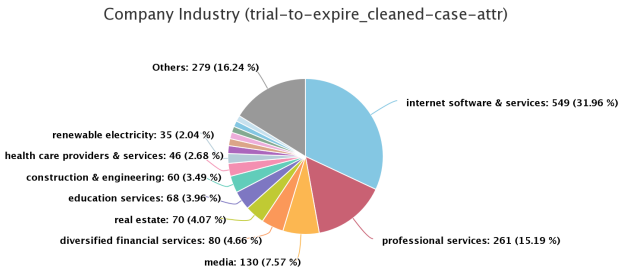


Figure 31. Company Industry

5 Validation

This section uses classification machine learning in order to validate the interpretations in the previous analysis. Listing 14 shows the machine learning code.

5.1 Feature Engineering

In order to begin running machine learning models on our data, we first need to convert the events logs into one multivariate feature table. Each activity feature is a dependent variable that contributes to the outcome of the user converting or abandoning. To organize the table, each feature is a column and each row is a company. The data for each row is the value of the feature for that company. The timestamp each feature event occurs, has been thrown out in order to create this table structure. The Trial-to-Pay is assigned a label of "1" for conversion while Trial-to-Expire is assigned a label of "0" for abandonment. The "label" column is considered the "y" values while the features are considered the "X" values, as described in the code. These two tables are joined and then split into training and testing sets. The machine learning models use the training dataset to learn how to predict the "label". These models are then fed the "X" testing dataset to try to predict the "y" labels. The accuracy of the models can be determined by calculating the correctness percentage in their prediction.

From the event logs, it is possible to construct additional features so that we may gain more understanding. The basic set is feature usage, which is created by the summation of the features used per company. But, this technique is flawed because feature usage is widely different between T2P and T2E. Then we engineered a relative feature usage by dividing the usage by the number of companies in each event log, so that the usages become scaled between the logs. Additional feature sets were added which are the average and total duration for each feature. The features are created by first finding the processing duration for each feature from the Type 2 event log. These were created by applying a *groupby* function to "company" and "feature_name" columns and then aggregate by the *mean* and *sum* functions. In total we ended up with about 120 features. Features that constitute larger pages are more likely to be more important in duration because these will have an overall higher duration usage than smaller features.

5.2 Machine Learning Models

Three machine learning models were compared, Logistic Regression (LR), Random Forest (RF) [13], and XGBoost (XGB) [14]. Logistic Regression [29] is a common model used for binary classification problems. It is a linear method where the predictions are transformed using the logistic function. LR is known to not be precise with a lot of correlated features. Random Forest is a supervised learning algorithm where it builds an ensemble of decision trees. This model is very easy to measure the relative importance of each feature on the prediction. XGBoost [30] stands for Extreme Gradient Boosting

[29]https://analyticsindiamag.com/random-forest-vs-xgboost-comparing-tree-based-algorithms-with-codes/

[30]https://analyticsindiamag.com/random-forest-vs-xgboost-comparing-tree-based-algorithms-with-codes/

Algorithm which is also an ensemble method that boosts decision trees. This makes use of a gradient descent algorithm that can correct the previous mistake done by the model, learn from it, and improve its performance.

We applied these machine learning models to various different configurations in order to see which features caused the highest accuracy. They were compared by measuring the accuracy on the training set and testing set. We watched out for over-fitting, which is when the model learns too much from the training set and cannot generalize into the testing set. This is identified by a high accuracy on the training set and a lower accuracy on the testing set. The machine learning results are presented in Tables [16, 17, 18].

Accuracy	Logistic Regression	Random Forest	XGBoost
Training	0.858	1.0	0.999
Testing	0.851	0.970	0.993

Table 16. Parameters (average usage count, no duration features)

Accuracy	Logistic Regression	Random Forest	XGBoost
Training	0.714	0.998	0.989
Testing	0.706	0.878	0.874

Table 17. Parameters (average usage count, average and total duration features)

Accuracy	Logistic Regression	Random Forest	XGBoost
Training	0.855	0.996	0.984
Testing	0.874	0.964	0.855

Table 18. Parameters (no usage count, total and average duration features)

XGBoost and Random Forest have higher accuracies than Logistic Regression for all different parameters. Random Forest is the model that over-fits the most. XGBoost has a better balance of not over-fitting while also having very high accuracies for different parameters. Table 18 shows the results for duration features only, however the accuracies of XGBoost are not that different from when the frequency features were included in Table 17. From this, it is hard to say which set of parameters are more important. This stands out more in SHAP values.

5.3 SHAP Values

Shapley Additive exPlanations (SHAP) values [15] are used for machine learning explainability to break down a prediction and show how the feature impacts the outcome [31].

[31] https://www.kaggle.com/learn/machine-learning-explainability

SHAP values interpret the impact for a given feature compared to if the prediction was made at a baseline value of that feature. Therefore, the SHAP values of all features sum up to explain why the prediction was different from the baseline. When a SHAP value prediction is made, the result is stored in *shap_values* variable which consists of two arrays. The first array are the SHAP values for a negative outcome (user abandons) and the second array are the SHAP values for a positive outcome (user converts). Normally, feature importance with a model does not give this separation of impact in positive or negative. The summary plot gives an overview of feature importance and what is driving it, as seen in Figure 32. The summary plot is made of many dots. These are described as:

- **Vertical** location shows what feature it is depicting
- **Color** shows what feature was high or low for that row of the dataset
- **Horizontal** location shows whether the effect of that value caused a higher or lower prediction

To create SHAP values, we must select which Explainer to use. There are KernelExplainer, TreeExplainer, and many others. The KernelExplainer works with all models, though it is slower than other Explainers, and offers an approximation rather than exact SHAP values. TreeExplainer only works with Tree Models such as Random Forest or XGBoost, but not with Logistic Regression. We decided the highest accuracy model will be used. Additionally, SHAP values are optimized for XGBoost model and thus runs faster and can create more detailed SHAP value summary plots.

For the following SHAP value summary plots, the XGBoost model will be used. Figure 32 shows the feature importance for total usage while Figure 33 shows the feature importance for relative usage. These plots are similar in that the features are ranked in approximately the same order, however the feature contribution is quite different. The feature values are more extreme on Figure 32. This is likely because counting the total feature usage for a company over the trial period becomes very large numbers with extreme differences. Total frequency usage features as seen in Figure 32 will not be used in the final conclusions because the relative frequency usage features seem to give more interpretable results. Figure 34 shows the feature importance when there are only duration features.

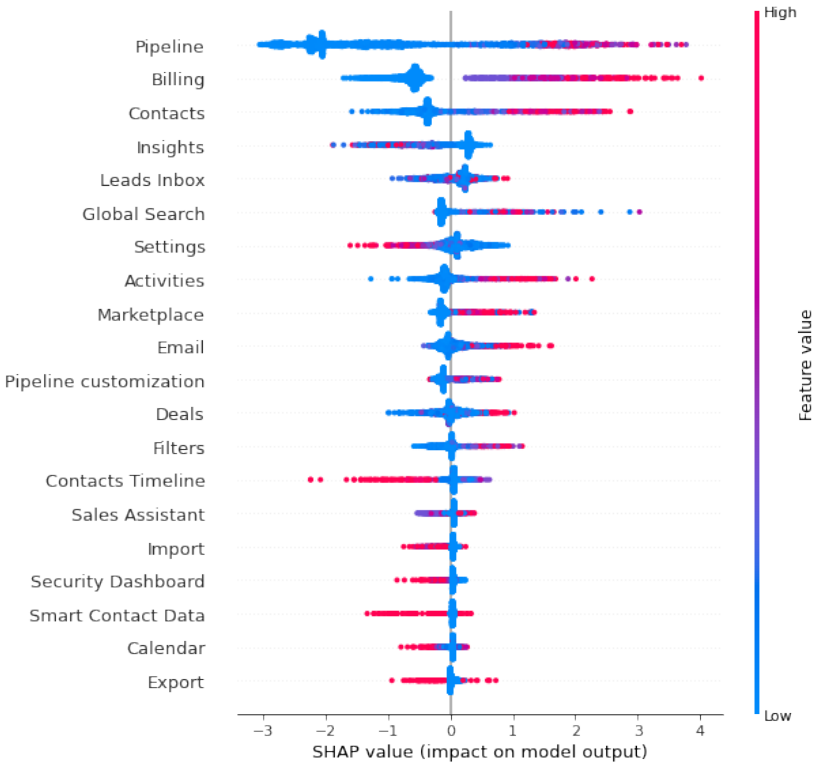


Figure 32. SHAP Values for Total Usage Frequency Features

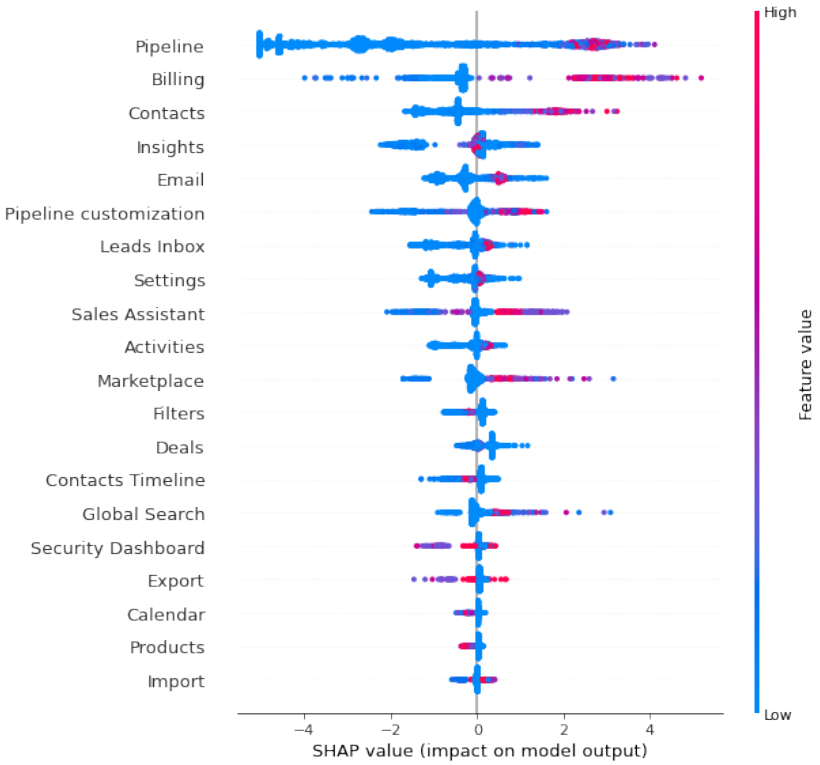


Figure 33. SHAP Values for Relative Usage Frequency Features

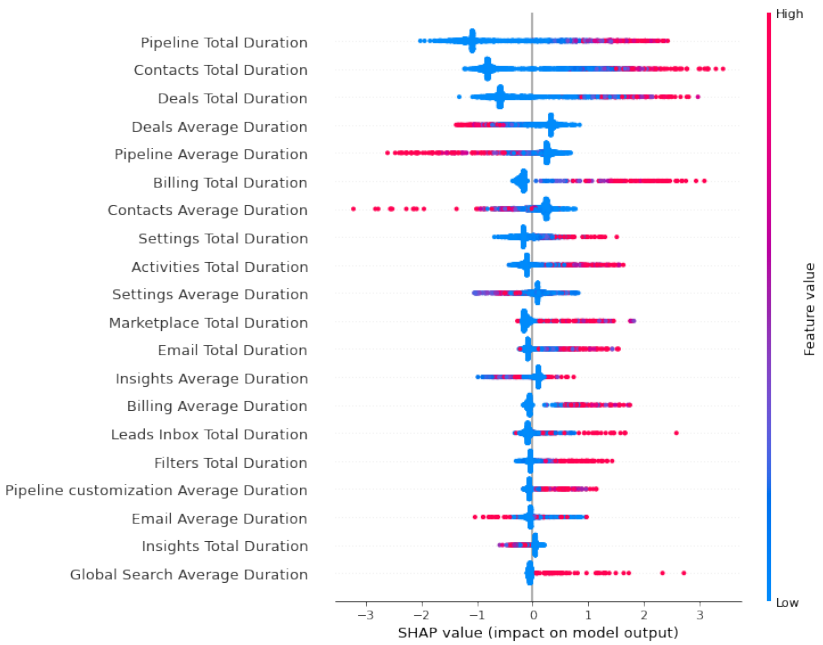


Figure 34. SHAP Values for Total and Average Duration Features

Figure 35 is the SHAP values summary plot for all 120 features. My findings for this plot are listed below:

- High "Pipeline" relative frequency usage values and high "Pipeline Average Duration" values result in a positive model impact. Since "Pipeline" is the first page upon logging in, it can be represented as average user usage of Pipedrive. The more a user uses Pipedrive and the longer they stay on the starting page, the more likely to convert. Also, the less users use Pipedrive, the less likely they are to convert, however the users may still convert even when low, as we can see there are quite a lot of blue dots on the right side of the SHAP impact line.
- High "Billing" usage has a clear large positive model impact. This makes sense as users need to go through "Billing" in order to convert.
- Of the main pages besides "Pipeline", high "Contact" relative frequency usage values correlate the most with positive impact. "Email", "Activities", "Leads Inbox", "Insights", and "Products" pages are not as important to user conversion as "Contacts".
- Duration features are generally not very important, because they do not appear high in the ranking. The highest ranked duration feature is "Settings Total Duration" at the 8th ranking. This shows that a high "Settings Total Duration" value results in a slightly negative model impact.
- The other features have a smaller impact to overall model output but the higher their relative frequency usage are, the more they impact the model positively. These are: "Leads Inbox", "Email", "Pipedrive customization", "Sales Assistant",

"Activities", "Marketplace", "Export", and "Academy". These features encompass customization, integration of Pipedrive, as well as the education of the users.

- "Global Search" feature usage generally lead to positive model output, whether the values are high or low.
- High "Security Dashboard" relative frequency usage values result in both negative and positive model impact. This can mean the feature appeals and also does not appeal to some groups of users. Therefore, room improvement may exist.

We are surprised that "Academy" is not higher on importance and that "Academy Duration" is not anywhere near the top of the list of important features. "Academy" is still important but not as important as other features. Out of the main pages that are not "Pipeline", we expected that "Activity" would be the page to correlate with user conversion the most, since the product is focused around activities a user needs to do in order to close a deal. Pages such as "Email" and "Insights" being lower in important makes sense. However "Contacts" is clearly more important, since salespeople need to manage and keep track of their contact list. When interviewing a real estate agent, he said, "My biggest asset is my contact list. Most of my future deals come from my previous clients". Features that we expected to be important but were not high on the list are "Filters", "Custom Fields", "Import", "Calendar Sync", and "Contact Sync". We are surprised at how much feature importance is dominated by the large whole page features. The models may be more biased to these features as they have more usage associated with them. We would really like to know in more detail how the smaller features impact conversion.

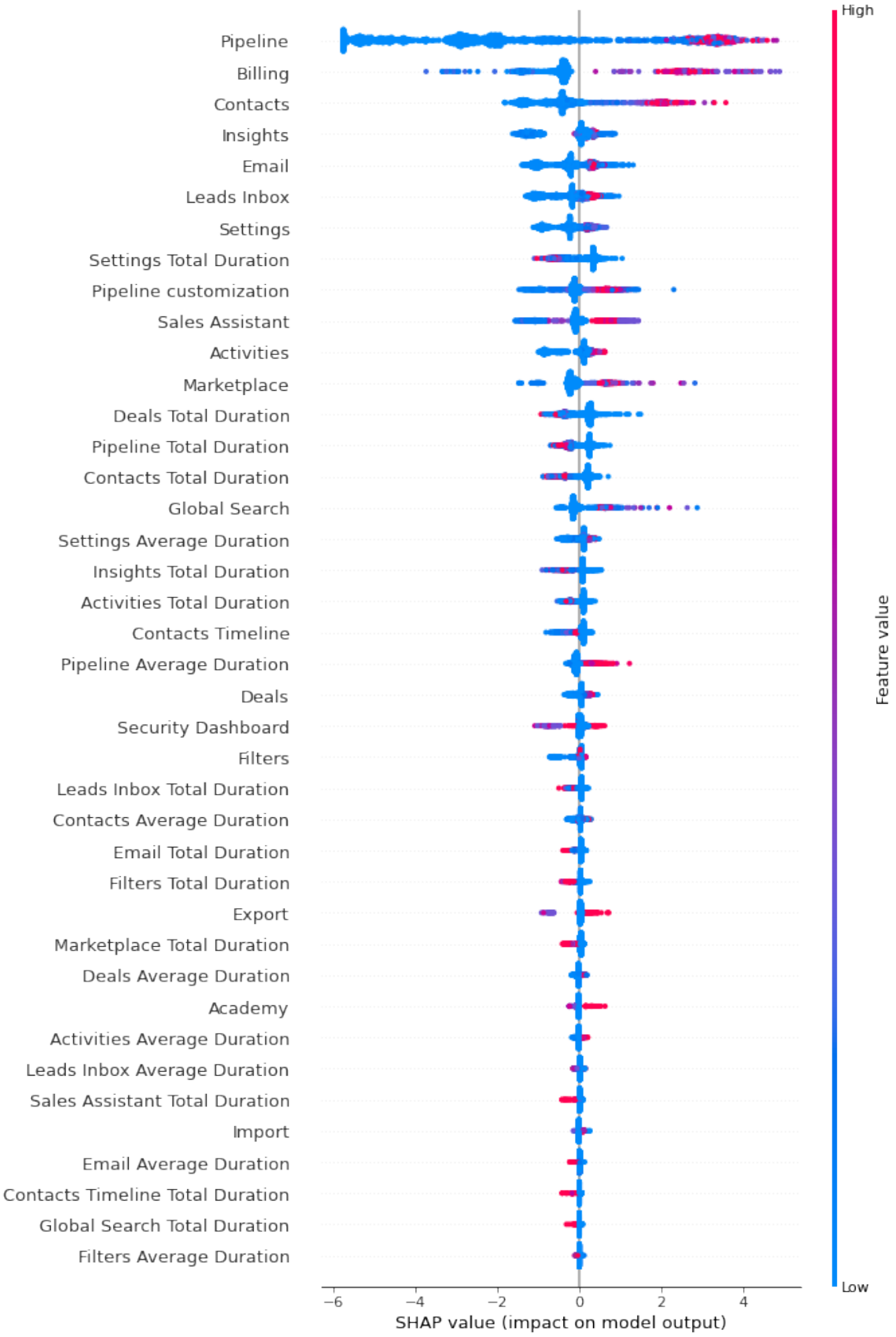


Figure 35. SHAP Values for Relative Usage and Total/Average Duration Features

6 Conclusions

This thesis analyses the customer journey of trialled users from when they signup for a trial until they convert or abandon Pipedrive. We gathered user activity data which was converted into event logs that was subsequently analysed using the process mining tool Apromore. Finally, we applied machine learning models to the event logs to validate the findings from the process mining analyse by looking at the feature importance. Visualizing user activity data with an event log on Apromore did not lead to linear user processes that were easy to analyse. Every user had a unique process variant of how they used Pipedrive.

The research questions from 1.4 are restated in this conclusion.

> **RQ1**: What distinguishes customer journeys between trialled customers that convert to a paid account versus those who abandon?

Those likely to convert:

- A customer who uses Pipedrive software consistently and actively during a trial period is obviously a prime indicator they are likely to convert. A high amount of activity correlating to conversion are found with pages Pipeline, Billing, and Contacts.
- Other strong indicators for conversion are high usage of collaboration and customization features of the product. These features provide additional value to customers for them to determine whether Pipedrive suits their needs.
- The Academy feature shows that not all users need education. Those who converted to a paid plan used Academy for shorter time periods, suggesting they could either figure out how Pipedrive works on their own or they had previous knowledge on CRM systems.

Those likely to abandon:

- Activity on the other pages such as Email and Activities did not signal as strong of an intention to convert, as did the other pages previously noted.
- The users spending more time with the Academy feature ended up being more likely to abandon. This suggests less familiarity with CRM systems and/or a need for more education.

The second question is shown below.

> **RQ2**: Are there any particular roadblocks or bottlenecks that prevent trialled customers from converting?

We could not find any conclusive evidence for this question as it relates to conversion. The longer time periods discussed in 4.2.1 pertain to browsing Marketplace or Academic videos, which are outside of the Pipedrive app. Bottlenecks are typically known to occur when a resource is handed off between different tasks. But in this case, there is only one resource, the user. The event log did not show linear processes of users starting from signing up to ending the trial, so no actual bottlenecks for a user converting were found.

The fact that no roadblocks or bottlenecks pertaining to trialled users were found, could be considered a positive attribute toward Pipedrive. This could also be the basis point for a different direction or approach in future studies.

6.1 Feedback

6.1.1 Recommendations to Pipedrive Data Team

We found that the event log granularity did not go deep enough. For example, an area that we would liked to have investigated deeper is the Billing process flow. Figure 36 shows the event granularity on the *feature_action* level. This shows three events, "Billing Overview Opened", "Billing Subscriptions Page Opened", and "Billing Details Opened". If the event level granularity included information such as user clicks on text boxes, then this would show the process as the user fills out their billing information details. Then it may be possible to discover the weakest link in the Billing process. These insights are more useful to PMs and data analysts.

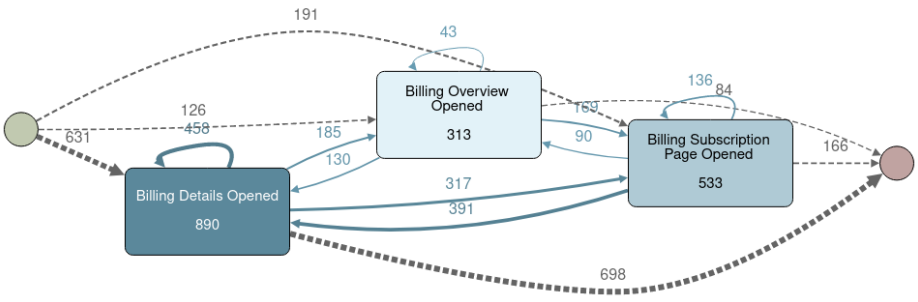


Figure 36. Detailed Billing Process Flow

Events described in Section 3.8.5 should not be included in the *dw.d_behaviorfeature* table. If they must be included, a separate flag column should be added to indicate they are status events. These status features are not very clear to someone doing analysis on the table since status is not behavior. Discovering which events are status events is very cumbersome and prone to change over time.

Company status change events would be useful to include as behavioral events. This can be when a user clicks on the subscription banner to change their plan level within a trial. Features are tied to plan level, so they might not be available to use if the plan level is lower.

6.1.2 Feedback to Apromore

This thesis used the Enterprise Apromore version 7.15 which has all the fancy features available. There is a Community available that has a limited feature set. I could not have achieved this thesis without access to a process mining product and we are happy that Apromore met all of our needs. Apromore is a very powerful tool and has many powerful features that enable drilling down into something specific or expanding to look at overall event log comparison. We noticed that there were a couple of small quality of life features that we would have liked while writing this thesis:

- On the performance dashboard comparison, we would have liked to apply filtering to both event logs and then to display the comparison of the filtered result. I can filter each event log individually and then show the individual filtered result, but not on comparison. To get around this, we could filter the event log on the Process Discoverer, save both event logs, and then go back to dashboard comparison, but this is tedious and requires going back to a previous page.
- The customized performance dashboard sometimes fails to save. Then we have to recreate the custom pages and graphs all over again. Additionally, the customized dashboards we create should work across different event logs that share the same data structure.
- Resizing the screen on the Performance Dashboard page does not automatically adjust the size of the graphs.
- A filtering option to show all the arcs and nodes that connect to one node, would be nice to have. This would be useful when our analysis was to focus on the connections of one node. This can kinda be achieved using the existing filters by all nodes directly follows to the node and then that node directly follows to all other nodes. However this will still include the interaction of all the other nodes.
- "NullPointerException" errors sometimes appear throughout the app in a dialog box. They usually disappear when refreshing the page.

6.2 Limitations

User activity data does not translate very well into defined processes. Activity data tends to go everywhere, even with filtering; therefore, finding trends in user pathways becomes increasingly tricky. Business processes typically become linear when filtered down, and these process mining techniques are more easily applied. In user activity data, some process mining techniques are not applicable. There are no actors/resources in which tasks are handed off from one to another. These handoffs usually are where bottlenecks in the process will occur.

The process of converting activity data into event logs had limitations. The logs did not include any information that said which subscription plan the user was on. The user could be on the "Essential" or the "Advanced" plans, which did not have access to the more advanced features. This can cause lower feature usage for these activities to cause bias in the results. The results show that most of the whole page features were important while the smaller features were less important. We believe that these bigger features overpowered the effect the smaller features have on new users. It would be interesting to take a more focused approach on these smaller individual features rather than whole pages as a feature.

Three types of event logs were created to analyse. The Type 3 event log, the sessionized user activity data, is more useful because that is how event logs are normally represented. However, this log was not utilised very much in the analysis. Part of the reason was that it was challenging to tell differences in user flows from different times. Also, to create the sessionized event log, we had to throw away more than half of the activity data, which renders conclusions from the log not very conclusive. The benefit of sessionized event logs is that they show user flows using Pipedrive over time. Therefore, we can compare users activity per week in order to see how they change. However, it wasn't easy to compare these flows, so the results were not shown. Type 2 event logs, aggregated user activity data, were created to get the features' duration. The problem with this is that the log drops events on a smaller granularity. Therefore, the final results using Type 2 event log are going to be of higher-level features.

6.3 Future Work

6.3.1 Improvements to Event Log

Some general improvements to event logs that could be done on a similar project like this thesis:

- Analyse bigger time frames. This analyse uses two weeks of company signup and two months of activity data. More data leads to more accurate results.
- Analyse multiple time frames. This enables verifying that the results are valid across multiple time frames.
- Create event logs with sessionized data with an entire user lifecycle, from trial signup to conversion or abandonment. Sessionized data means the user activity data when the user is using the software. The sessionized event logs could have holes in them because of how we created them. The users may have turned on "Remember Me" during a trial, triggering their session to become a long session.
- Subscription plan information should be included in the process of creating the event log. The event logs could be created using only users that have the highest subscription plan so that all the features are available to all the users. If the goal is to analyse the user usage when plans are changed, then plan changing can be included as a new behavior event or a status event.

- Event logs of user activity data should include much more detailed information about a users activity. It should be possible to deep dive into the detailed behavior of a user on a page. Layering the behavior data in layers like what is done in this thesis is an excellent idea to keep the appropriate abstraction levels.

6.3.2 Product Suggestions to Pipedrive

The findings from this thesis can be used to influence the decisions in how Pipedrive features are designed and displayed to new users. These findings can also be incorporated into the customer success team approach.

Application features:

- The onboarding screen feature walks the user around on how to use the app with some basic flows. This can be modified so that the walk-through includes collaboration and integration features.
- Set-up ongoing real-time analysis of customer's behavior, so Pipedrive would then know the real time usage stats on the users in a trial. With alert features to signal certain issues, this information could feed instantaneously to the success team.
- For trialled uses that do not convert, follow-up with a how can we improve our product question. Make this an easy input for existing customers. Valid comments should then be forwarded to the tech team.

Customer success team:

This team targets existing customers for retention and/or encourage them into more expensive subscription plans. Their responsibilities could be expanded to a more active role reaching out to converting trialled users.

Pipedrive should have flexibility in offerings, such as an extra two week trial period to encourage more use thereby developing comfort and habits. The occasional call to reach out to see if there are ways to help could assist with someone considered on the fence for converting. Some specific examples include the following.

- Non-use could be a situation of excessive work or unexpected leave, where the user could benefit from an extended trial.
- Excessive time with the Academy feature could trigger an educational assistance call from customer service.
- A highly active user during trial who does not sign-up at the end of a trial, may need further incentive or motivation, such as a call from customer service offering a temporary discount.

References

[1] W. van der Aalst, A. Adriansyah, A. K. A. de Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. van den Brand, R. Brandtjen, J. Buijs, A. Burattin, J. Carmona, M. Castellanos, J. Claes, J. Cook, N. Costantini, F. Curbera, E. Damiani, M. de Leoni, P. Delias, B. F. van Dongen, M. Dumas, S. Dustdar, D. Fahland, D. R. Ferreira, W. Gaaloul, F. van Geffen, S. Goel, C. Günther, A. Guzzo, P. Harmon, A. ter Hofstede, J. Hoogland, J. E. Ingvaldsen, K. Kato, R. Kuhn, A. Kumar, M. La Rosa, F. Maggi, D. Malerba, R. S. Mans, A. Manuel, M. McCreesh, P. Mello, J. Mendling, M. Montali, H. R. Motahari-Nezhad, M. zur Muehlen, J. Munoz-Gama, L. Pontieri, J. Ribeiro, A. Rozinat, H. Seguel Pérez, R. Seguel Pérez, M. Sepúlveda, J. Sinur, P. Soffer, M. Song, A. Sperduti, G. Stilo, C. Stoel, K. Swenson, M. Talamo, W. Tan, C. Turner, J. Vanthienen, G. Varvaressos, E. Verbeek, M. Verdonk, R. Vigo, J. Wang, B. Weber, M. Weidlich, T. Weijters, L. Wen, M. Westergaard, and M. Wynn, "Process mining manifesto," in *Business Process Management Workshops*, pp. 169–194, Springer Berlin Heidelberg, 2012.

[2] W. M. P. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer Publishing Company, Incorporated, 1st ed., 2011.

[3] M. L. van Eck, X. Lu, S. J. J. Leemans, and W. M. P. van der Aalst, "PM ˆ2 : A process mining project methodology," in *Advanced Information Systems Engineering - 27th International Conference, CAiSE 2015, Stockholm, Sweden, June 8-12, 2015, Proceedings* (J. Zdravkovic, M. Kirikova, and P. Johannesson, eds.), vol. 9097 of *Lecture Notes in Computer Science*, pp. 297–313, Springer, 2015.

[4] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, "Crisp-dm 1.0 step-by-step data mining guide," *SPSSInc.*, 2000.

[5] J. C. J. C. Bose, R. S. Mans, and W. M. P. van der Aalst, "Wanna improve process mining results?," in *IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2013, Singapore, 16-19 April, 2013*, pp. 127–134, IEEE, 2013.

[6] G. Bernard and P. Andritsos, "A process mining based model for customer journey mapping," *Consortium Papers Presented at the 29th . . .*, 2017.

[7] A. S. T. Davenport, "What process mining is, and why companies should do it.," 2019. Accessed 25 10 2020.

[8] L. Rosa, J. Mendling, H. Reijers, M. Dumas, and Marcello, *Fundamentals of Business Process Management*. Springer, Berlin, Heidelberg, Feb. 2013.

[9] G. Bernard and P. Andritsos, "CJM-ex: Goal-oriented exploration of customer journey maps using event logs and data analytics," *BPM (Demos)*, 2017.

[10] A. Terragni and M. Hassani, "Analyzing customer journey with process mining: From discovery to recommendations," in *2018 IEEE 6th International*

Conference on Future Internet of Things and Cloud (FiCloud), pp. 224–229, ieeexplore.ieee.org, Aug. 2018.

[11] M. Böhlen, *Temporal Coalescing*, pp. 2932–2936. Boston, MA: Springer US, 2009.

[12] L. Zhuang, Z. Kou, and C. Zhang, "Session identification based on time interval in web log mining," in *Intelligent Information Processing II, IFIP TC12/WG12.3 International Conference on Intelligent Information Processing (IIP 2004), October 21-23, 2004, Beijing, China* (Z. Shi and Q. He, eds.), vol. 163 of *IFIP*, pp. 389–396, Springer, 2004.

[13] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[14] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *CoRR*, vol. abs/1603.02754, 2016.

[15] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," 2017.

Appendix

I. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Brandon Christopher Autrey**,
(author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Customer Journey Analysis at Pipedrive: A Process Mining Approach,
   (title of thesis)

   supervised by Marlon Dumas.
   (supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Brandon Christopher Autrey
05/07/2021

II. Source Code

```sql
select
    case
        when datediff(paying_first_dt, trial_first_expired_dt)
            between 5 and 9 then '5-9'
        when datediff(paying_first_dt, trial_first_expired_dt)
            between 10 and 14 then '10-14'
        when datediff(paying_first_dt, trial_first_expired_dt) >= 15
            then '>15'
        else datediff(paying_first_dt, trial_first_expired_dt) end as
             days_between_paying_and_expire,
    count(distinct key_company) no_of_companies,
    round(count(distinct key_company)*100.0/sum(count(distinct
        key_company)) over (), 2) share
from dwmodel.d_company
where added_dt between '2021-01-01' and '2021-01-14'
and (paying_first_dt between '2021-01-01' and '2021-02-28' and
    trial_first_expired_dt between '2021-01-01' and '2021-02-28')
group by 1
order by 1
```

Listing 9. Days Between Paying and Trial Expired

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pylab import savefig

# rename columns, set date to index, fill missing dates with 0
def prepare_data(df):
    df.columns = ['date','num']
    df = df.set_index('date')
    df.index = pd.DatetimeIndex(df.index)
    idx = pd.date_range('2021-01-01', '2021-02-28')
    df = df.reindex(idx, fill_value=0)
    df = df.rename_axis('date').reset_index()
    return df

t2p = pd.read_csv('trial_length/T2Pay.csv')
t2p = prepare_data(t2p)
t2e = pd.read_csv('trial_length/Trial2Expire.csv')
t2e = prepare_data(t2e)
signup = pd.read_csv('trial_length/Signup-companies.csv')
signup = prepare_data(signup)

t2p['type'] = 'Trial-to-pay'
t2e['type'] = 'Trial-to-expire'
signup['type'] = 'Signup'
df = pd.concat([t2p,t2e,signup])
df = df.sort_values(by='date')

plt.figure(figsize=(16,8))
plt.xticks(rotation=30)
sns.set(font_scale=1.5)
```

```

ax = sns.lineplot(data=df, x='date', y='num', hue='type')
ax.set_xlim(df['date'].min(), df['date'].max())
ax.set_ylabel('Number of companies')
plt.tight_layout()

savefig("company-trial-length.png")
```

Listing 10. Company Trial Length Differences

```
with result as (
    select
        case
            when pl.plan_descr like '%Professional%' then '
                Professional Plan'
            when pl.plan_descr like '%Advanced%' then 'Advanced Plan'
            when pl.plan_descr like '%Essential%' then 'Essential
                Plan'
            else pl.plan_descr
        end as plan_descr,
        count(*) as plan_count
    from dw.f_companychangelog cc
    inner join dw.d_plan pl
        on cc.new_value = pl.id_plan
    inner join (
        select key_company, end_dt from df_companies
    ) as c
        on c.key_company = cc.key_company
    where cc.month_dt between '2021-01-01' and '2021-02-28'
    and cc.field_key == 'plan_id'
    and cc.updated_dt <= c.end_dt
    group by 1
    order by 2 desc
)
select *
from result
union all
select 'Total',
    sum(plan_count)
from result
```

Listing 11. Subscription Plan Statistics

```
select
    case
        when date_format(date_date, "HH:mm:ss") == '00:00:00' then '
            midnight'
        when date_format(date_date, "HH:mm:ss") != '00:00:00' then '
            not midnight'
    end as dataTime,
    count(*) as count
from dwmodel.f_behavior
where month_dt between '2021-01-01' and '2021-02-28'
group by 1
order by 2 desc
```

Listing 12. Investigate Events at 00:00:00

```
select fea.id_behaviorfeature,
    fea.feature_name,
    fea.feature_action
from (
    select
        key_behaviorfeature,
        sum(
            case
                when date_format(date_date, "HH:mm:ss") != '00:00:00'
                    then 1 else 0
            end
        ) as not_midnight_num
    from dwmodel.f_behavior
    where month_dt between '2021-01-01' and '2021-02-28'
    group by 1
) as beh
inner join dw.d_behaviorfeature as fea
    on fea.id_behaviorfeature = beh.key_behaviorfeature
where beh.not_midnight_num == 0
order by 2,1
```

Listing 13. Events Missing Time Attribute

```
# ---- Global Config Variables ----
freq_usage = True
relative = True

duration = True
total_duration = True
average_duration = True

shap_filename = "shap_values"

# ---- Imports ----
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn import metrics

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

import shap
shap.initjs()

import matplotlib.pyplot as plt

from datetime import datetime
fmt = '%Y-%m-%d %H:%M:%S'
```

```
29
30 # ---- Pre-process Data ----
31 def readEventLog(path):
32     df = pd.read_csv(path)
33     return df.rename(columns={'caseId_company': 'company', '
       activity_feature_name': 'feature_name'})
34
35 # converts event log to company feature value
36 # Output df: index = company, column name = feature name, value
37 # df_el - event log dataframe
38 # relative - bool - set whether the feature usage should be relative
    to how many companies
39 def convertEventLog(df_el, relative):
40     df = df_el[['company', 'feature_name']]
41
42     # count the feature usage per company
43     df = df.groupby('company')['feature_name'].value_counts().
       reset_index(name='count')
44
45     if (relative):
46         num_cases = len(df.company.unique())
47         df['count'] = df['count'] / num_cases
48
49     # create new df to format output
50     companies = df.company.unique()
51     features = df.feature_name.unique()
52     out = pd.DataFrame(0, index=companies, columns=features) #
       default values to 0
53
54     if freq_usage:
55         for _, row in df.iterrows():
56             out.loc[row.company, row.feature_name] = row['count']
57
58     # Drop columns not for analysis
59     out = out.drop(['E-signatures', 'Mobile'], axis=1, errors='ignore
       ')
60
61     return out
62
63 # Calculates and adds total and average duration for the features
64 # df_feature - feature dataframe
65 # df_el - event log dataframe
66 def addFeatureDuration(df_feature, df_el):
67     df_el['duration'] = df_el.apply(lambda row:
68                                     (datetime.strptime(row['end_time'], fmt
                                        ) -
69                                      datetime.strptime(row['start_time'],
                                         fmt)
70                                     ).total_seconds(), axis=1)
71     df_duration = df_el[['company', 'feature_name', 'duration']]
72     df_new_duration = df_duration.groupby(['company', 'feature_name'
       ]).sum().reset_index().rename(columns={'duration': '
       total_duration'})
73     df_new_duration['average_duration'] = df_duration.groupby(['
       company', 'feature_name']).mean().reset_index().duration
```

```
74
75      # remove features that do not exist
76      df_new_duration = df_new_duration.query('feature_name != "Mobile"
            and feature_name != "E-signatures"')
77
78      # create new feature columns
79      features = df_new_duration.feature_name.unique()
80      func_total_duration = lambda f: f + ' Total Duration'
81      func_avg_duration = lambda f: f + ' Average Duration'
82      new_features = np.vectorize(func_total_duration)(features)
83      new_features = np.concatenate((np.vectorize(func_avg_duration)(
            features), new_features))
84
85      # initialize new features to 0
86      for feature in new_features:
87          df_feature[feature] = 0
88
89      if total_duration:
90          for _, row in df_new_duration.iterrows():
91              df_feature.loc[row.company, row.feature_name + ' Total
                    Duration'] = row['total_duration']
92
93      if average_duration:
94          for _, row in df_new_duration.iterrows():
95              df_feature.loc[row.company, row.feature_name + ' Average
                    Duration'] = row['average_duration']
96
97      return df_feature
98
99  # Load data trial-to-pay
100 t2pay_el = readEventLog('~/thesis/event_log/aggregated/t2p_no-
        midnight.csv')
101 t2pay = convertEventLog(t2pay_el, True)
102 if duration:
103     t2pay = addFeatureDuration(t2pay, t2pay_el)
104
105 # Load trial-to-expire
106 t2expire_el = readEventLog('~/thesis/event_log/aggregated/t2e_no-
        midnight.csv')
107 t2expire = convertEventLog(t2expire_el, True)
108 if duration:
109     t2expire = addFeatureDuration(t2expire, t2expire_el)
110
111 # Combine event logs together
112 t2pay['label'] = 1
113 t2expire['label'] = 0
114 events = pd.concat([t2pay, t2expire])
115
116 # Splitting data set to train and test
117 y = events.pop('label')
118 X_train, X_test, y_train, y_test = train_test_split(events, y,
        test_size=0.25, train_size=0.75, random_state=0)
119
120 # ---- Models ----
121 def stats(model):
```

```
    y_pred_train= model.predict(X_train)
    print('Accuracy on training set: {}\n'.format(metrics.
        accuracy_score(y_train, y_pred_train)))

    y_pred = model.predict(X_test)
    print('Accuracy on test set: {}\n'.format(metrics.accuracy_score(
        y_test, y_pred)))

    print("Confusion Matrix")
    print(metrics.confusion_matrix(y_test, y_pred))

# Logistic Regression
model_lr = LogisticRegression(max_iter=10000).fit(X_train, y_train)
stats(model_lr)

# RandomnForest
model_rf = RandomForestClassifier(random_state=0).fit(X_train,
    y_train)
stats(model_rf)

# XGBoost
model_xgb = XGBClassifier().fit(X_train, y_train)
stats(model_xgb)

# ---- SHAP ----
# Create object that can calculate shap values
explainer = shap.TreeExplainer(model_xgb)

# Calculate Shap values
shap_values = explainer.shap_values(X_test)
fig = shap.summary_plot(shap_values, X_test, show=False, max_display
    =50)
plt.savefig(shap_filename + '.png', bbox_inches='tight')
```

Listing 14. Feature Importance with SHAP Values using Machine Learning Code