Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2020, 7(9):100-108

Research Article ISSN: 2394 - 658X

Controlled Experimentation Setup with Frontend Application to Analyze Effectiveness of an AI/ML Recommendations System

Tanmaya Gaur

Principal Architect, Customer Support, T-Mobile US, Washington, USA
Email address: tanmay.gaur@gmail.com

ABSTRACT

AI has been fueling changes to how the world operates and not many industries are left untouched. While there has been significant talk around AI replacing web developers, that is not yet a fully established space. An area of web development that has really taken off is the ability for AI/ML to power recommendation and insights which help make the web experiences contextual and rewarding. Many enterprises have already started using AI to build more dynamic, contextual, and engaging customer experiences. As we light these new super-powers up, we often are left wondering about how to interpret the results. Is the AI hype and the money spent to power such incredible recommendation engines worth the rewards? This paper will talk about UI Control experiment-based monitoring patterns that can be a tool to answer that question. The paper will use an Enterprise CRM as an example to walk through some scenarios and the solution.

Keywords: Artificial Intelligence, Machine Learning, Control Groups, Controlled Experiments, Web Development

---

INTRODUCTION

Investing in recommendation engines to power web experiences is the latest trend. Many Enterprises are applying AI/ML based recommendation systems across both customer facing and internal channels. A recommendation engine can be broadly defined as a data filtering tool which uses machine learning algorithms to recommend relevant items to a particular user or customer. It operates on the principle of finding patterns in consumer behaviour data often determining a 'propensity'. This 'propensity' would be then linked to a specific user experience tied to a business beneficial outcome. Some of the common use-cases are to increase engagement, reduce churn, and create cross-selling opportunities. An example would be Netflix using customer behaviour data to present content recommendations or Amazon using browsing data to present upsell product recommendations. In CRM(s), it is a common use-case to use end customer behaviour data to present next best actions or opportunities for the agents' assisting customers. What is common across all these use-cases is that they are driving results geared towards the organization's goals. Be it customer engagement for Netflix, boosting sales and revenue for Amazon or lowering cost to serve by providing quicker resolutions for the CRM.

As these recommendation systems drive customer experiences, often at significant cost, it is essential to develop solutions to quantify the outcomes and the business benefit achieved. Is the cost being sunk into executing the models and operational expenses to drive associated experiences generating expected ROI. There are multiple methods to do so all of which rely on collecting data or signals and analysing them. In this paper we will touch on some of these patterns for an enterprise use-case.

The example use-case cited in this paper uses a recommendation system to deliver dynamic insights to a Telco CRM's. AI and machine learning over the data signals available for a customer are used to determine next best actions for a subscriber calling customer care. The agent attending the customer's call are presented with these 'next best actions' which are unique to each customer's needs, preferences, and context. The goal is to either drive quicker call resolution by presenting deep insights or to deliver insights for upsell or churn depending on customer's propensity, essentially compute and present relevant insights in real-time which otherwise required elaborate and time-consuming analysis and deduction by the care agent. We will use this example to discuss the

considerations for constructing a control group and interpret the collected data to determine recommendation engine's business value.

PROBLEM STATEMENT

Before we dive into the research considerations, let's understand our sample use-case.

The Use-Case

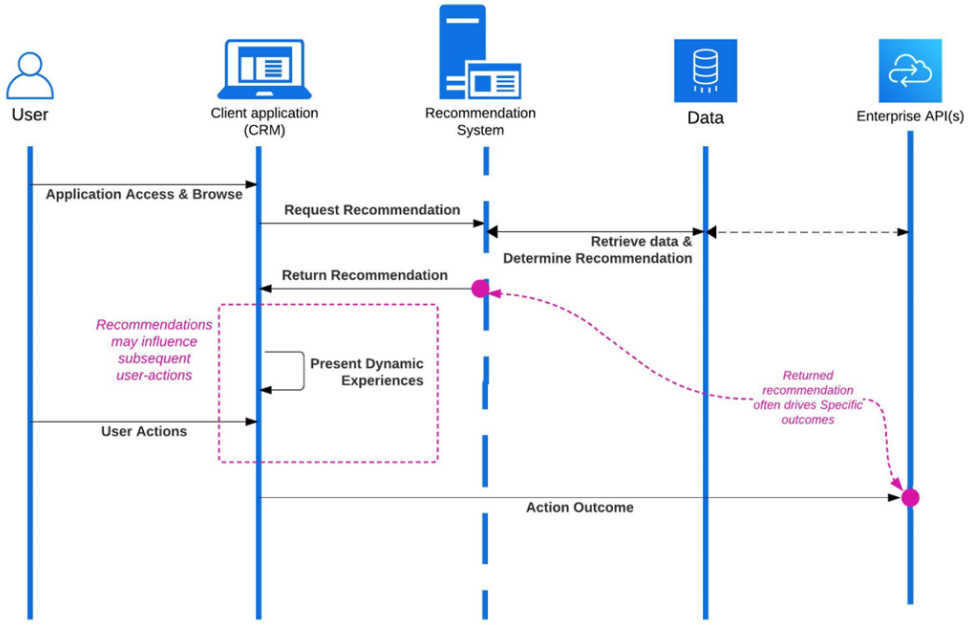

Fig. 1 *Pattern for Recommendation systems driving end user experience*

In our example, we have an enterprise CRM which is being used by a care agent to assist a customer calling into customer care. The application on load of the customer account retrieves and presents API responses from a recommendation system. The recommendation system internally sources customer data signals from various enterprise data domains. These data signals may be simple facts like which plan the customer is on to complex derived metrics like network health scores which perceived network health for that customer. The recommendation system utilizes AI and machine learning over all of these data signals and offers recommendation based on each customer's unique needs, preferences, and context. The recommendation so presented are then used by agents to assist the calling customer needs and may result in subsequent actions if acceptable.

In most scenarios, recommendations are configured based on tangible benefits to the enterprise. Ideally each recommendation system is looking to achieve some goal, or a business metric that it is trying to improve. These outcomes are normally tied to some outcome event the recommendation is trying to drive the user towards. In the case of our telco example, these is promoting sales or reducing churn, also known as customer attrition. Recommendations configured for these business results are tied to specific desired outcome events. As an example, if the recommendation is for a customer has a device incompatible with their current plan which may be causing network issues and hence a churn risk, the recommendation may be a change of plan or a device upgrade. Knowing these relationships, we can monitor if we did see the outcome activity of rate plan change or device upgrade activity on the account post the recommendation being presented. This is one way to help determine adoption of the recommendation engine and the applicability and accuracy of the recommendations being provided.

Business may will also want to monitor if with this recommendation applied to a given number of calling user population, did indeed help the company achieve the desired business result. E.g., in case of the churn driven recommendation, did the churn recommendations being returned lead to a reduction in churn for the calling group compared to the normal? How did this number compare to a group of users who were not provided the recommendation? This can help determine the success of the overall business case. However, businesses can't wait to confirm these results out only after the recommendation has been applied across all users and groups, this is where we can utilize controlled experiments to validate the recommendation before scaling across the entire user-base.

Key Terminology

There are few important terms we should clarify.

• **Propensity** is a statistical technique that uses data to predict the likelihood of a customer performing a certain action in the future. E.g. in our use-case, the recommendation system determined a customer's propensity to churn and then returned business configured churn offers.
• **Eligibility** This is the actual eligibility of a customer to be eligible for a specific offer. E.g. in a churn scenario, the calling customer may have a propensity to churn which has business configured recommendations around moving to specific rate plans, the customer however may only be eligible for specific rate plans out of the list. We set up the recommendation system to also validate eligibility and only return recommendations the customer is eligible for. This can be separated out as its own concern in other implementations.
• **Confidence** is the level of certainty that the recommendation system has on the recommendation's desirable effects outweigh its undesirable effects. This can be driven by propensity scores which in turn depend on data and feedback. In our example implementation, we decided to show only recommendations which had high confidence score.
• **Accuracy** is the percentage of times a recommendation system makes a correct recommendation.
• **Usability** This validates the usability of user-experience in presenting the recommendation and confirms if customer service agents can successfully navigate and complete actions advised by the recommendation and how long it takes.
• **Adoption** This answers if customer service agents are utilizing the recommendations when available. Adoption rate can be measured by calculating number of users following up on the recommendation as a percentage of the total number of users who were provided the recommendation.
• **Disposition** This can be a manual or automated submission on how the recommendation was handled by the agent. This can be used to capture the qualitative feedback about the recommendation. Was the recommendation relevant to the conversation? Did they bring it up with the customer? If so, did the customer engage or not. This can serve as important feedback to the recommendation system.
• Business KPIs Business teams will also want to monitor that with this recommendation applied to a given number of callers, did the company achieve the desired outcome. E.g., in case of the churn related recommendation, did it eventually lead to a reduction in churn for the calling group compared to the normal. This can help determine the success of the overall business case. This is another important feedback for the system.

Monitoring Requirements

Below are the requirements we wanted to monitor. I have on purpose skipped operational, performance and availability monitoring requirements for the API and the application to instead focus on product, business, and usability requirements.
• **Usability Requirements** Usability Requirements for the recommendation interface needed to quantify Efficiency of use (how quickly were the agents able to leverage the recommendation system, how many user errors), Intuitiveness (was the interface easy to learn and navigate; messaging and outcome events easy to understand) and perceived workload (did the interface appears seem intimidating, demanding, and frustrating).
• **Adoption Requirements** These feature adoption metrics were to quantify if agents were using the recommendations when available. There were also requirements to determine qualitative answers where agent adoption was not as per expectations. These broke down into Feature adoption rate (recommendations used v/s total available), Adoption Breadth (specific to different recommendations, which ones were agents adopting and which ones were left untouched), Adoption Depth (How frequently was the feature being used), Time to adopt (how quickly were agents adopting the recommendations from first availability calculated by number of calling customer sessions).
• **KPI Impacts** These requirements were to answer how the recommendation engine impacted Customer care KPI(s) like Call Resolution Time (CRT), Average Handle Time (AHT), First Call Resolution (FCR), Net Promoter Score (NPS) and Customer Satisfaction (CSAT). While the recommendation engine's key goals were to positively impact FCR, NPS and CSAT, there were still requirements to co-relate impacts to CRT and AHT to understand impacts.

These care KPI(s) are discussed in more detail in the next sub-section**.**

CONSIDERATIONS

Before we dive into the research method, let's understand our sample use-case as well as talk through the various considerations involved**.**

Determining data to be collected.

There are few unique steps (refer Fig. 2) in an agent interaction with the recommendations. We ideally want to collect these individual data points in a way where we can tie it back to the agent, customer, the session and be able to co-relate the sequence for a session.

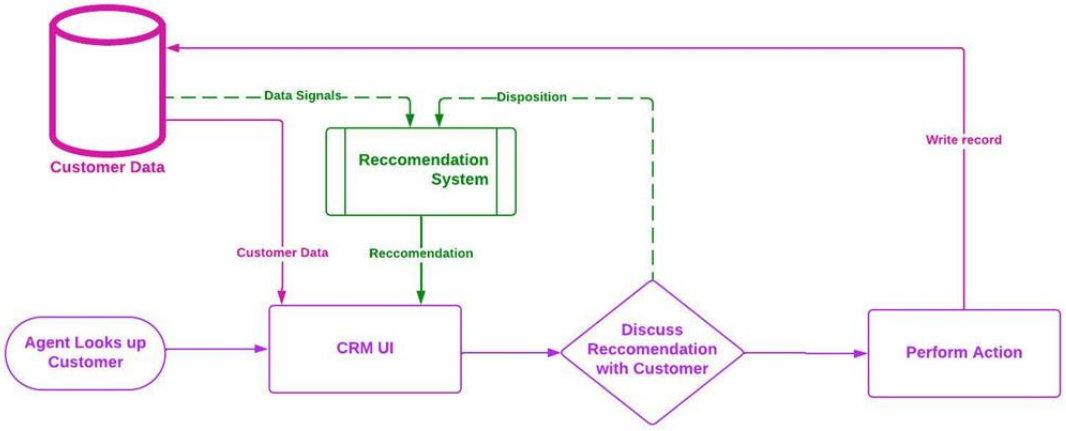

Fig. 2 *CRM interacting with the recommendation system*

1. Agent looks up Customer. Every new customer lookup should generate a unique sessionID that can be logged.
2. Customer Account loaded on CRM. There are unique aspects of agent, and customer that need to be logged with co-relation sessionID.
3. The application retrieves recommendations providing the customer identity. The recommendations returned (ID) needs to be logged with co-relation sessionID. Each unique recommendation returned needs to be tracked.
4. For each recommendation, we need to log which recommendations were presented to the agent. There were scenarios with more than a pre-set number of recommendations being returned where we only presented up to the configured count (sorted based on confidence score high to low)
5. For each unique recommendation ID, we wanted to log actions the agent took against the recommendation. Did the agent access the recommendation, access additional insights (clickable CTA), click co-related action CTA.
6. There was manual disposition requested from the agent which was submitted by the agent. We wanted to be able to track the disposition unique to each recommendationID.
7. There was an ask to track subsequent actions taken by the agent after being presented with a recommendation. An example would be where a change plan was recommended, was the change plan applied to the user account. This could be thought of as conversion. We will discuss conversion modeling in this section which applies to this data being tracked in the actual backend system where the change of plan was applied instead of via UI activity.
8. In addition to tracking data on the recommendationID and associated activity, we had requirements to track and co-relate this data against business KPI(s) detailed in the next section.

Relevant Business KPI(s)

There was few specific customer care and service desk KPI(s) which mattered to the business case, these will be the ones we will focus on. While this list is specific to a telco use-case example, it is ideal to understand KPI(s) that matter to a recommendation engine implementation. The list pertinent to our use-case were.

• Call Resolution Time Call resolution time is how long it takes to solve a support ticket from call start to finish, and the length of time usually varies based on the complexity of the issue. Next Best Action initiative understood that it may negatively impact the Call resolution time as agents were often asked to perform actions beyond the customer call reasons. That said, the team did want to understand how different offers impact this KPI.

• Average Handle Time AHT represents the average time an agent (or agents) spends in resolving a customer's issue. This may be a summation of the time taken by the agent handling the call as well as any additional time spent after finishing the call to perform follow up actions. Next Best Action initiative understood that it may negatively impact the Call resolution time as agents were often asked to perform actions beyond the customer call reasons. That said, the team did want to understand how different offers impact this KPI.

• First Call Resolution FCR is a KPI used to represent the ability of an IT team to meet a customer's needs fully the first time they contact them. This is calculated by checking if the customer called back within a stipulated period. Next Best Action believed it would positively influence FCR.

• Customer Satisfaction Score Typically after a customer care call, sample set of customers are encouraged to fill out a survey to provide feedback. Customer Satisfaction Scores(CSAT) show how happy customers are with the service provided and how well customer service team members handle customer issues and complaints. Next Best Action believed it would positively CSAT scores.

• Net Promoter Score the Net Promoter Score (NPS) is a satisfaction benchmark that measures how likely your customers are to recommend your business to someone else. You get an NPS score by surveying customers and asking them "on a scale of 1 to 10, how likely are you to recommend us to a friend?" Customers that rate you a 9 or 10 would be considered promoters, and as the name implies, most likely to promote your brand. The rest of the scores break down as follows:

O Detractors: scores 0 to 6

O Passives: scores 7 and 8

O Promotors: scores 9 and 10.

To calculate Net Promoter Score, subtract the percentage of detractors (wouldn't recommend you) from the percentage of promoters (would recommend you).

Determining Data collection Approaches.

There are many options to collect the data required. Below list contains some possible considerations.

• **Manual dispositions** This involves building required data collection as a manual step for the user of the recommendation.

• **Did the agent even bring up the recommendation with the customer?** Or was the recommendation irrelevant to the customer need and the agent decided to ignore it. Such feedback can be collected manually or and is relevant to identify adoption and accuracy of the recommendations. If a recommendation is not applied in expected numbers, we know it is not going to matter when plotted against the business case. It is important to identify and diagnose adoption and accuracy early as it may point towards lack of credible data, issues with the model or a user adoption challenge.

• **End-user monitoring or application Logs** This provides a way to automate the disposition collection and works for scenarios where recommendation is tied to specific subsequent actions or API calls the application can be aware of. This works by confirming a subsequent action as requested by the recommendations. Click tracking and application logs cannot solve all scenario or type of recommendations and may often prove challenging depending on specific enterprise scenarios.

• **Conversion** Modeling Conversion modeling refers to an analysis technique where we analyze the effectiveness of the recommendation by confirming actual conversion event at the backend enterprise system of record. If the recommendation was to change rate plan, this could refer to looking at billing system logs and confirming if the requested action did indeed take place. There are few advantages to this approach.

O Agnostic of the UI, recommendation system could be employed over multiple agent and customer facing applications which all don't need to be instrumented.

O This is the only option that can solve for complex enterprise scenarios like return policies. As an example, if your recommendation system powers sales scenarios, it may be causing impulse buys which could be getting returned during the return period. Attribution analysis like manual disposition or click tracking may suggest positive adoption but the actual impact to the business case is not possible to be deduced just from that data.

Data Types

- **Qualitative vs. quantitative** Qualitative data is descriptive and may be interpretation-based, while quantitative data is quantifiable and numbers-based. Qualitative data can help understand why behind certain behaviors, while quantitative data can answer what.
- **Primary vs. Secondary** While not pertinent to our use-case, this is still an important enough topic to mention. Primary sources are original, first-hand event data, while secondary sources analyze, interpret, or evaluate primary sources.
- **Descriptive vs. experimental** Another crucial aspect, especially to the approach being discussed in the paper. Descriptive research refers to research which describes a phenomenon or else a group under study. It is mainly useful in gathering data on a certain population, situations, and events. Descriptive research is more towards collecting data and try to find out some insight out of that data using statistical analysis.
- **Experimental research** on the other hand refers to research where the researcher manipulates the variable to conclude. This is useful in finding out the cause effect of a causal relationship and correlation.

Data collection methodologies

- **A/B testing** A/B testing is a methodology for comparing two versions of a webpage or app against each other to determine which one performs better. A/B testing is essentially an experiment where two or more variants of a page are shown to users at random, and statistical analysis is used to determine which variation performs better for a given conversion goal.
- **Beta testing** Beta testing is an opportunity for real users to use a product in a production environment to uncover any bugs or issues before a general release.

  Beta testing is the final round of testing before releasing a product to a wide audience. The objective is to uncover as many bugs or usability issues as possible in this controlled setting. The purpose of having a control is to rule out other factors which may influence the results of an experiment. Not all experiments include a control group, but those that do are called "controlled experiments."

- **Control Treatment** Testing This is like A/B Testing where control group and experimental group are compared against each other in an experiment. The only difference between the two groups is that the independent variable is changed in the experimental group. The independent variable is "controlled" or held constant in the control group. The purpose of having a control is to rule out other factors which may influence the results of an experiment. Not all experiments include a control group, but those that do are called "controlled experiments.". while both controlled experiments and A/B tests involve comparing groups to make informed decisions, controlled experiments are more focused on understanding fundamental principles and causality, while A/B tests are practical tools for optimizing specific aspects of products or marketing efforts in real-world scenarios.
- **Blue-Green deployment** Completely unrelated yet often confused, Blue-green deployment is a release management technique that reduces risk and minimizes downtime. It uses two production environments, known as Blue and Green, to provide reliable testing, continuous no-outage upgrades, and instant rollbacks.
- **Multivariate testing** Multivariate testing (MVT) is a controlled experiment that involves changing multiple elements of a website to determine which combination of changes produces the most conversions. MVT is useful when multiple elements on a page can be changed at once to improve a single conversion goal.

Other Misc. Data considerations

Teams also need to be mindful of their Data storage and analysis needs. Diving deep into these topics is beyond the scope of this paper. We need to ensure we are asking the right questions and setting up the experiment for success. Is the data granular enough as is required for analysis? What is the frequency or urgency of analysis? You also want to confirm key data analysis frequency needs and choose appropriate approach and stack.

- Real-time analysis Data is processed as it is created.
- Batch analysis Data is processed periodically.
- Near-real-time analysis Data is processed in minutes instead of seconds when you don't need it immediately.

SETUP OF CONTROLLED EXPERIMENT AND GROUP IN AN ENTERPRISE CRM

Now that we understand the base use-case, lets discuss the setup specific to our CRM use-case.

The Controlled Experiment

A controlled experiment involves comparing two or more groups to make data-driven decision. Controlled experiments are broader than A:B and can encompass a wider range of scope. Another key difference is the aim to understand cause-and-effect relationships when manipulating one or more variables while keeping everything else constant. These experiments are often used to test hypotheses.

It is important to understand and setup context of the controlled experiment. Controlled experiments are typically conducted in controlled environments where we can manipulate specific variables precisely. Controlled experiments may involve smaller sample sizes and can be conducted over long period of time to gather comprehensive data, as is useful to test out a hypothesis. The results may often require more complex statistical analysis to draw conclusions about causality and generalizability.

Next sub-section will talk through our process to setup a controlled environment for our CRM. While a specific example, this sub-section talks through the considerations we had in our setup which could be applied to other scenarios as well. Subsequent sub-section will dive into the data collected for analysis. Last sub-section will then talk about how this all comes together for analysis.

Control and Treatment Groups

Control groups are essential to experimental design. When application team are interested in the impact of a new recommendation, they randomly divide their study participants into at least two groups:

- The treatment group (also called the experimental group) receives the recommendation or recommendations whose effect we are interested in.
- The control group receives either no recommendation, a standard recommendation whose effect is already known, or a placebo (a fake recommendation to control for placebo effect).

Setup of control and treatment Groups for a CRM

To setup a controlled experiment, we need to be able to setup a control & treatment group of users (refer Figure 3) with differing experiences. A controlled environment for a web application then returns recommendations to the treatment group while the control group receives no recommendation, standard or placebo depending on your setup. Creating such experience variations builds on a base capability of the application to test and deploy variations to specific user groups without affecting the general production environment. This allows application owners to test specific changes to the application without disrupting all users. Some important considerations to be mindful of

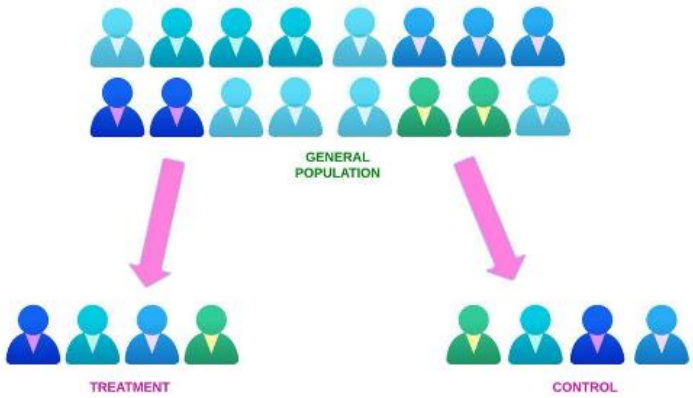

Fig. 3 *Control Treatment samples should be identical and representative of general population.*

- It may be easier to just create one logical isolated user group which could be treatment and keep everything else as control, it may however make it difficult to keep all variables the same across such a large control group. It may be best to create 2 or more logical isolated groups from the general audience (all users) and treat them as control and treatment.
- It is ideal that a user assigned to one of the control or treatment groups continues to be provide the same experience for the duration of the study. This may complicate your setup based on the specific need of your application. In the case of the CRM, this required us to ensure our control and treatment groups were assigned not just for the agents but also for our end customers. We achieved this by ensuring customers who connected with a treatment group the first time, continued to have their call routed to the same group.

There are several ways to create a controlled environment for a web application. We will discuss two common approaches below

- Multiple Environments One common approach is to setup a whole new environment with the custom code. This approach is like how web applications often setup a staging environment. A staging environment is a replica of the production environment, but it is not accessible to all users. Developers can use the staging environment to test changes to the application before they are deployed to production. For a control/treatment type experiment, staging could be treatment and production could behave like control. That said, we should remember that we want all other variables to remain consistent. Unless staging works like production environment for all possible ways, this wouldn't work.
- Environment Variables Another way to create a controlled experiment for a web application is by using environment variables or configurations to drive the new experiences . Environment variables are variables that are set outside of the code and can be used to configure the application to display different variations. This is the same pattern applied in traditional web development to display development, staging, and production environment variations (e.g. backend API calls made for development environment go to a different endpoint versus staging versus production). In our CRM example, we can similarly create an environment variable or derive one at run-time based on specific agent, user, or session properties. The value of the variable could be tied to specific experiences being displayed.

There may be additional approaches available based on your application stack and patterns. For our use-case, we used specific environment variables derived based on agent identity to populate an environment variable which could take three values (control, treatment, or NA). This was then used to drive the below user-experience.

- If **Treatment**, we generate the recommendation and display it to the agent.
- If **Control**, we generate the recommendation, but it is not displayed to the agent.
- If NA, we do not generate a recommendation.

Once we had the agent groups teams identified, we configured one of the teams to receive the recommendations (treatment) and the other team did not (control). While the recommendation engine was generating recommendation for both set of communities, only the treatment community was presenting the insights to the agents.

Once we had this setup in place, we started monitoring the application telemetry as well as Business KPI(s) relevant to our use-case as discussed in the earlier section.

Manual Dispositions

We had our UI application mandate agents to submit dispositions. While not ideal, this gave us an effective way to gauge qualitative response from the agents. In modern care implementations, we can now gather this data using AI

over call transcription. For our specific scenario, developing that solution was not worth the ROI and hence manual data collection was prioritized. The dispositions that were setup were

- ***Accept*** Agent applied the recommendation on the customer account.
- ***Reject*** Agent did not apply the recommendation because the Customer rejected it after hearing it out.
- ***No_Engagement_Customer*** Agent did not apply the recommendation because the Customer declined to discuss it.
- ***No_Engagement_Agent*** If the agent could not bring up the recommendation OR if no disposition was submitted (default)

The final Monitoring Stack

The final monitoring stack used Application Logs, Logs from the recommendation system and relied on the call routing system for care KPI(s) (refer Figure 4)

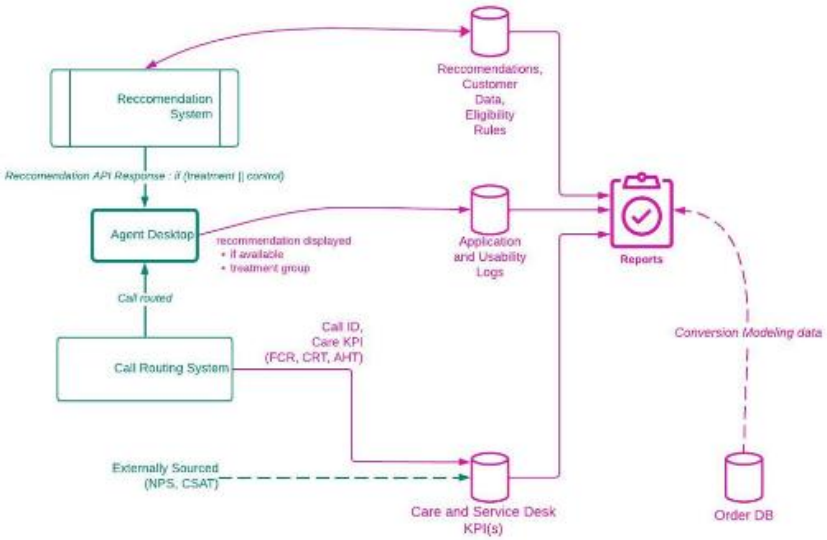

Fig. 4 *Monitoring Stack for Reporting*

The Usability and Adoption data was captured using application logs and a user activity monitoring solution. Both logging systems had sessionID in logs to co-relate activity that occurred during a session. The sessionID was directly related to agent's identity.

We decided to utilize manual disposition instead of conversion modelling for the control groups given some of the complexity with conversion modeling. Each manual disposition was also related to agent's identity.

There were existing BAU call routing system monitoring logs that measured the Care KPI(s) granular to an agent's identity. Given our ability to stitch together (refer Figure 5) a treatment agent's activity across usability, adoption logs as well as the disposition and care KPI(s), this helped us successfully co-relate care KPI(s) against control and treatment groups as well as against different usability and adoption data points from within the treatment group.

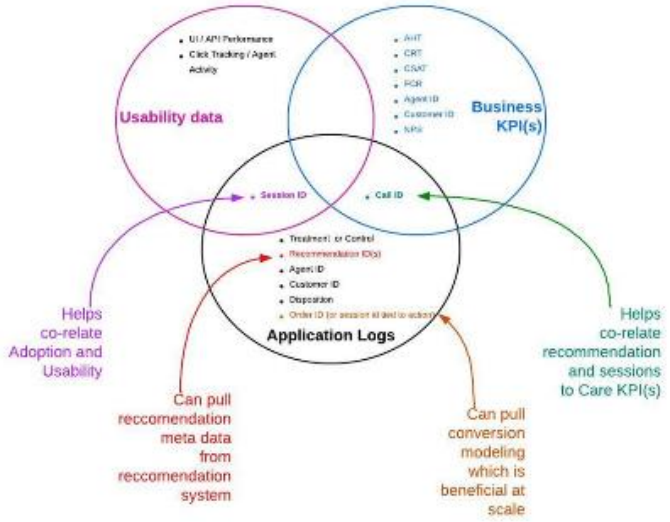

Fig. 5 *Data Co-relation for Reports*

CONCLUSION

Enterprises and smaller businesses are all looking to tap into the advantages provided by AI/ML recommendation systems. These systems if setup right can often help reduce costs, increases sales, or can be trained to positively influence select business goals. They most often do so by helping users find products and services they might not have discovered on their own or in a timely fashion. These systems however and associated processes and operations do come at a significant cost. It is important to realize and adjust your recommendations early and with smaller groups before wider rollouts. To confirm if your recommendation does work the way it was intended, there are areas that need to be tested like usability, adoption apart from finding ways to project impacts against the business goals.

This paper talked about using controlled experiments as a way of testing AI/ML recommendation systems. There are significant considerations which often get overlooked and impact your findings. An example of such a miss is not selecting the right control treatment groups, such that they are either not identical or are not a good representation sample of the total population. These can help dilute the accuracy of your results. This paper used an example of a control/treatment setup for an enterprise CRM to talk through the various considerations and possible implementation options to mitigate such issues. The advantages of these control tests could range from demonstrating early viability to the teams or helping uncover flaws with the team's hypotheses, allowing them to revise or replan.

REFERENCES

[1]. The Difference between Descriptive Research and Experimental Research. [Online]. Available: https://www.geeksforgeeks.org/difference-between-descriptive-research-and-experimental-research/

[2]. What Is a Recommendation Engine and How Does It Work? [Online]. Available: https://www.appier.com/en/blog/what-is-a-recommendation-engine-and-how-does-it-work

[3]. Build a recommendation [Online]. Available: https://help.amplitude.com/hc/en-us/articles/360059625252-Build-a-recommendation

[4]. Control Groups and Treatment Groups | Uses & Examples. [Online]. Available: https://www.scribbr.com/methodology/control-group

[5]. Recommendation Engine for Winning the Personalization Battle [Online]. Available: https://www.comtecinfo.com/rpa/wp-content/uploads/2020/06/white-paper-recommendation-engine-1.pdf

[6]. Online Recommendation Systems: A special case of information filtering systems [Online]. Available: https://www.mphasis.com/content/dam/mphasis-com/global/en/home/innovation/next-lab/thoughtleadership/online-recommendation-systems-whitepaper.pdf