Technical Design Document - Therapeutic Specialist Pre-Call Plan Use Case

Authors: Google PSO TS team

Prepared for: Insmed Inc.

Document type: Technical Design Document

Version: Final

Last Updated: Nov 11, 2024

About this document

Document highlights	
Purpose	The purpose of this document is to provide Insmed with a good technical understanding of the solution for the use case of therapeutic specialist pre-call plan through AI and ML, which includes but is not limited to use case, requirements, architecture design, code/component structures, and the recommended path to production along with code samples and artifacts.
Intended audience	Customer technical stakeholders, customer development and infrastructure team.
Key assumptions	The audience is expected to be very familiar with Insmed's use case of therapeutic specialist pre-call plan and its objectives.
Delivery note	This document can support Insmed teams by providing design decisions and user guide references.

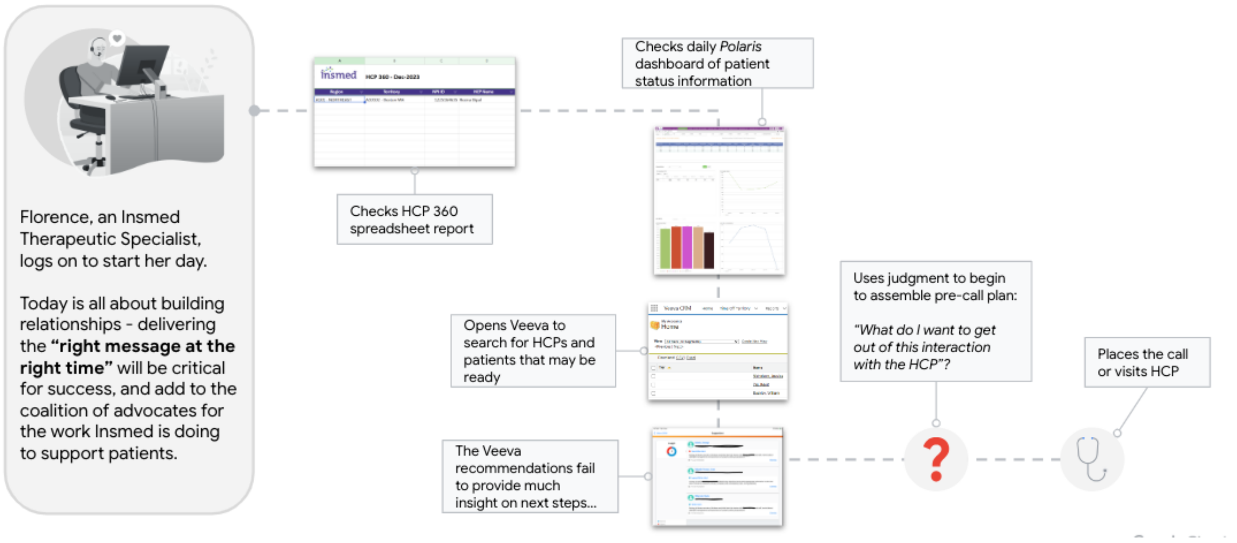
1. Introduction

Insmed seeks to implement an AI-powered solution that supports the organization's business and technical growth priorities. Insmed completed an AI advisory and scoping engagement in February 2024 with Google Cloud PSO to support the objectives of accelerating new drug application (NDA) submissions and speed to market by co-creating an actionable AI roadmap. This technical design document (TDD) outlines the technical specifications, design considerations, and implementation details for one of the use cases that emerged during these discussions. Specifically, this document describes the Therapeutic Specialist (TS) pre-call plan use case, which aims to provide TSs with actionable information to more effectively and efficiently engage with healthcare providers (HCPs) using insights from internal data.

2. Use Case

At Insmed, Therapeutic Specialists play a vital role in connecting healthcare professionals (HCPs) with the resources they need to provide optimal care to their patients. However, the process of identifying where HCPs are, understanding their patients' journeys, and determining the most effective actions to take can be time-consuming and complex.

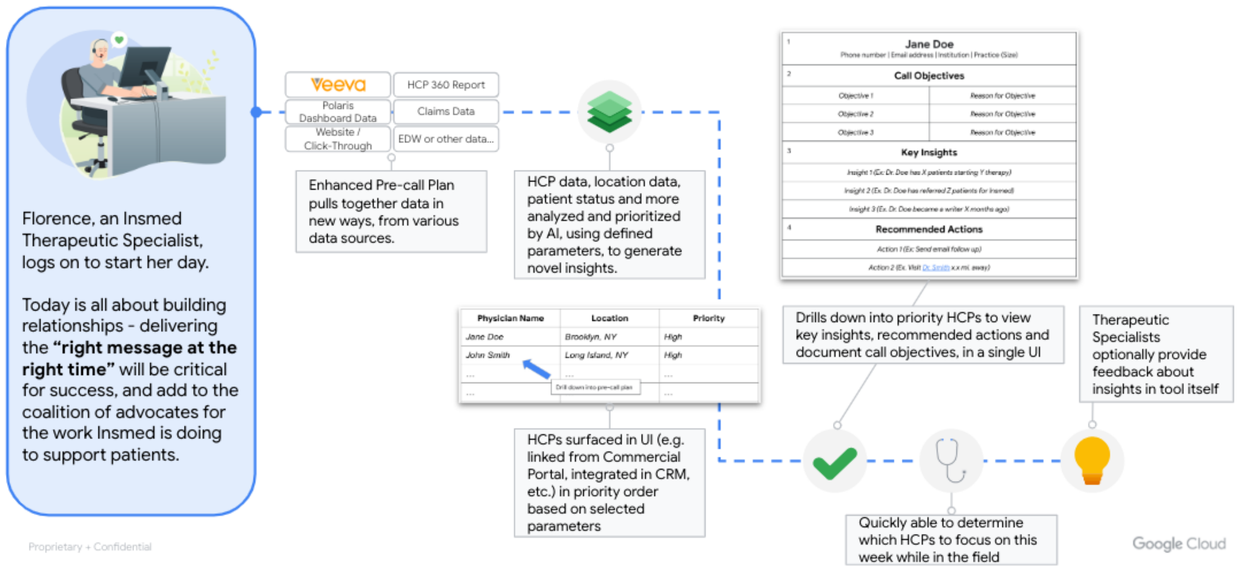
Currently, TSs must navigate through multiple sources of data to make informed decisions about which HCPs to engage with and how best to support them.


Florence, an Insmed Therapeutic Specialist, logs on to start her day.
Today is all about building relationships - delivering the **"right message at the right time"** will be critical for success, and add to the coalition of advocates for the work Insmed is doing to support patients.
Checks HCP 360 spreadsheet report
Checks daily *Polaris* dashboard of patient status information
Opens Veeva to search for HCPs and patients that may be ready
The Veeva recommendations fail to provide much insight on next steps...
Uses judgment to begin to assemble pre-call plan:
"What do I want to get out of this interaction with the HCP"?
Places the call or visits HCP

Picture 2.1: Current Workflow

Together with the Google PSO team, Insmed is developing an AI solution to help its TSs quickly identify and engage with HCPs. This solution will streamline the process of qualifying HCPs, contextualizing their patients' treatment journeys, and recommending the best course of action for each interaction, ultimately leading to more efficient and effective education about relevant treatments.

By proactively collaborating with HCPs, Insmed aims to improve patient outcomes and accelerate the time-to-diagnosis, which will also speed up the market adoption of Insmed's products. This AI-driven approach will empower TSs with actionable insights and recommendations, enhancing their interactions with HCPs and ultimately benefiting both patients and HCPs.



Florence, an Insmed Therapeutic Specialist, logs on to start her day.

Today is all about building relationships - delivering the **"right message at the right time"** will be critical for success, and add to the coalition of advocates for the work Insmed is doing to support patients.

Enhanced Pre-call Plan pulls together data in new ways, from various data sources.

HCP data, location data, patient status and more analyzed and prioritized by AI, using defined parameters, to generate novel insights.

Physician Name	Location	Priority
Jane Doe	*Brooklyn, NY*	*High*
John Smith	*Long Island, NY*	*High*
...	...	...
...	...	...

Drill down into pre-call plan

HCPs surfaced in UI (e.g. linked from Commercial Portal, integrated in CRM, etc.) in priority order based on selected parameters

1 **Jane Doe** Phone number | Email address | Institution | Practice (Size)	
2 **Call Objectives**	
Objective 1	*Reason for Objective*
Objective 2	*Reason for Objective*
Objective 3	*Reason for Objective*
3 **Key Insights**	
Insight 1 (Ex: Dr. Doe has X patients starting Y therapy)	
Insight 2 (Ex. Dr. Doe has referred Z patients for Insmed)	
Insight 3 (Ex. Dr. Doe became a writer X months ago)	
4 **Recommended Actions**	
Action 1 (Ex: Send email follow up)	
Action 2 (Ex. Visit Dr. Smith x.x mi. away)	

Drills down into priority HCPs to view key insights, recommended actions and document call objectives, in a single UI

Therapeutic Specialists optionally provide feedback about insights in tool itself

Quickly able to determine which HCPs to focus on this week while in the field

Picture 2.2: AI-powered workflow

3. Solution Models

The overall deliverable from the AI Models used in the solution – both discriminative and generative aim to provide recommendations that do not shift dramatically across refreshes and allows TSs to plan activities while benefiting from timely and relevant data-driven suggestions.

This solution comprises three different AI models, wherein the results from the first two models serve as input into the third AI model. The models are listed below:

- **Accurate HCP Targeting Model (Model 1)**: Predicting the probability of HCPs becoming enrollment form writers.
- **Improved Patient Journey Model (Model 2)**: Surfacing HCPs with a recent enrollment form; and identifying HCPs with a recent enrollment form that have not been contacted within 48 hours of an enrollment.
- **Personalized Recommendations Model (Model 3)**: Providing actionable insights to TSs in the form of standardized messages describing individual HCPs and engagement details. The results from models (1) and (2), in addition to other data sources, serve as inputs to model (3).

3.1 Foundation Prediction Model: Accurate HCP Targeting

This machine learning classification model predicts the likelihood of an HCP completing an enrollment form in the next 90 days. It uses two main data sources: the HCP's antibiotic prescription history and their patients' AFB lab test results over the past 12 months. The model analyzes factors like the number of MAC positive tests, the types and combinations of antibiotics prescribed, and whether patients have been on certain antibiotics for at least 6 months. The output is a probability score assigned to each HCP, indicating their likelihood of enrollment.

3.1.1 HCP 360 Statistics

To carry out the exploratory data analysis, the total available population of HCPs has been selected from the HCP 360 report from January 2023 to April 2024. The table below shows the total number of HCPs by month and the number of actual enrollment forms in the next 90 days (the target variable for the ML model in model 1). Across all the months, there were an average of 2.3% of enrollments.

MONTH	HCPS	EFS_90_DAYS	% EFS

2024-05	22,033	529	2.40%
2024-04	22,617	531	2.35%
2024-03	22,348	509	2.28%
2024-02	21,961	494	2.25%
2024-01	21,667	494	2.28%
2023-12	21,258	454	2.14%
2023-11	20,753	416	2.00%
2023-10	20,229	419	2.07%
2023-09	19,941	439	2.20%
2023-08	20,073	456	2.27%
2023-07	20,071	452	2.25%
2023-06	20,070	491	2.45%
2023-05	20,015	480	2.40%

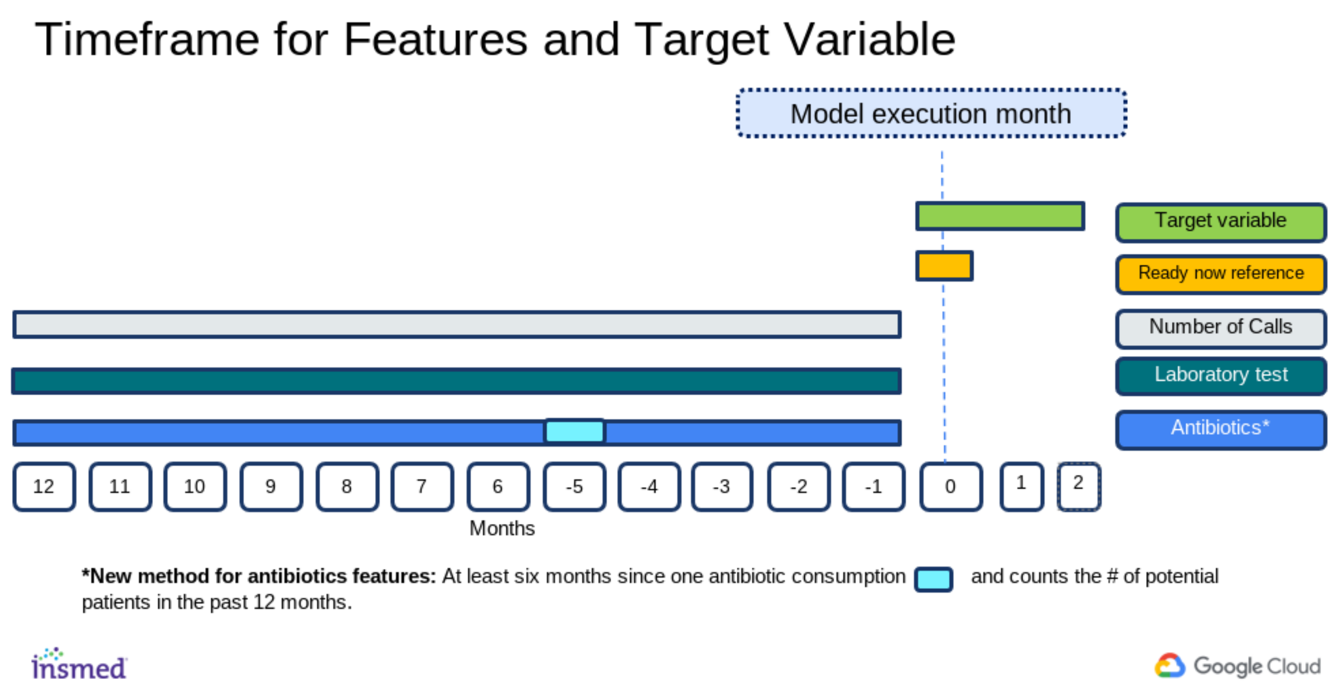
2023-04	17,943	501	2.79%
2023-03	20,201	490	2.43%
2023-02	20,138	456	2.26%
2023-01	20,335	450	2.21%

3.1.2 Model Input Features

The table below lists the names, data source locations, and brief descriptions of the features used for this model. Some columns come directly from HCP 360 reports, while others are created using a custom SQL logic for antibiotics. New features were created in order to capture different combinations and cross scenarios between antibiotics consumption and MAC test results, such as:

FEATURE	DATA SOURCE	DESCRIPTION
AFBTESTSMAC3MONTHS	ADHOC_TBLSR_PROGNOS_HISTORY	Number of MAC positive tests results by NPI
AFBTESTSMAC6MONTHS	ADHOC_TBLSR_PROGNOS_HISTORY	Number of MAC positive tests results by NPI months
AFBTESTSMAC_3_to_6_M	ADHOC_TBLSR_PROGNOS_HISTORY	Number of MAC positive tests results by NPI months
SUM_AZI_CLAR	NEW SQL	Sum of patients taking AZITHROMYCIN plus the last 12 months with the condition of at lea consumption.
SUM_RIF_RIF	NEW SQL	Sum of patients trying RIFAMPIN plus RIFABU months with the condition of at least 6 mont consumption.
SUM_MOX_LEV	NEW SQL	Sum of patients trying MOXIFLOXACIN plus

		LEVOFLOXACIN,OFLOXACIN and CIPROFLC months with the condition of at least 6 montl consumption.
AMIKACIN	NEW SQL	Sum of patients trying AMIKACIN in the last 1 condition of at least 6 months in one consum
FLAG_EDA	NEW SQL	Flag (1,0) where 1 indicates that the HCP has than two different type of antibiotics
SUM_MAC_TEST	NEW SQL	Sum of AFBTESTSMAC3MONTHS, AFBTESTSI AFBTESTSMAC_3_to_6_M
FLAG_EDA_MAC	NEW SQL	Flag (1,0) indicates if the HCP has patients try antibiotics and at least one patient has a MAC last 12 months.



***New method for antibiotics features:** At least six months since one antibiotic consumption and counts the # of potential patients in the past 12 months.

3.1.3 Model Output

The primary result of model 1 is the PROBABILITY_EF column, which indicates the probability of each HCP submitting an enrollment form in the next 90 days. This probability, along with additional HCP profile data, is stored in the **AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS** table.

Full output from Model 1 is as follows:

COLUMN	SOURCE	DESCRIPTION
NPIID	ADHOC_TBLSR_HCP_DIM	HCP identifier
Probability_EF	Generated as model 1 output	Probability in the range [0,1] where 1 indicates a high probability of an enrollment form in the next 90 days.
AREA	ADHOC_TBLSR_HCP_DIM	HCP's geographic area
FIRSTNAME	ADHOC_TBLSR_HCP_DIM	HCP's first legal name
LASTNAME	ADHOC_TBLSR_HCP_DIM	HCP's last name
MONTH_EF_OK	ADHOC_TBLSR_HCP_DIM	Timestamp for model execution
REGION	ADHOC_TBLSR_HCP_DIM	HCP's geographical region
TERRITORY	ADHOC_TBLSR_HCP_DIM	HCP's territory

3.1.4 Model Training

3.1.4.1 Data Preparation

Below table shows the data distribution of the overall training data prepared.

STAGE	HCPs
Initial population size (HCP's Jan 2023 - Apr 2024)	351,653
Target size (EFs in the next 90 days=1)	8,061
% Target size (Target size / Initial population size)	2%

Train-Test Split and Class Balancing

The data is split into training and testing sets using a 70/30 split.

STAGE	# HCPs	# Positive label HCPs (EFs in the next 90 days=1)	# Negative label HCPs (EFs in the next 90 days=0)
Train (70%)	246,157	5,657 [2.2%]	240,500
Test (30%)	105,496	2,404 [2.2%]	103,092

To handle class imbalance data in the training set for the target variable (FLAG_EF_90_DAYS), we reduced the number of instances in the majority class (HCPs who didn't submit enrollment forms) while keeping all instances of the minority class (HCPs who did submit forms). This increased the proportion of positive cases in the training data from 2.2% to 21.6%, enabling the model to learn more effectively from the underrepresented class and improve its overall predictive performance. The table below shows the new distribution of the training set.

STAGE	HCPs
Train (EFs in the next 90 days=0)	20,419
Train (EFs in the next 90 days=1)	5.657
Train % Target size	21.6%

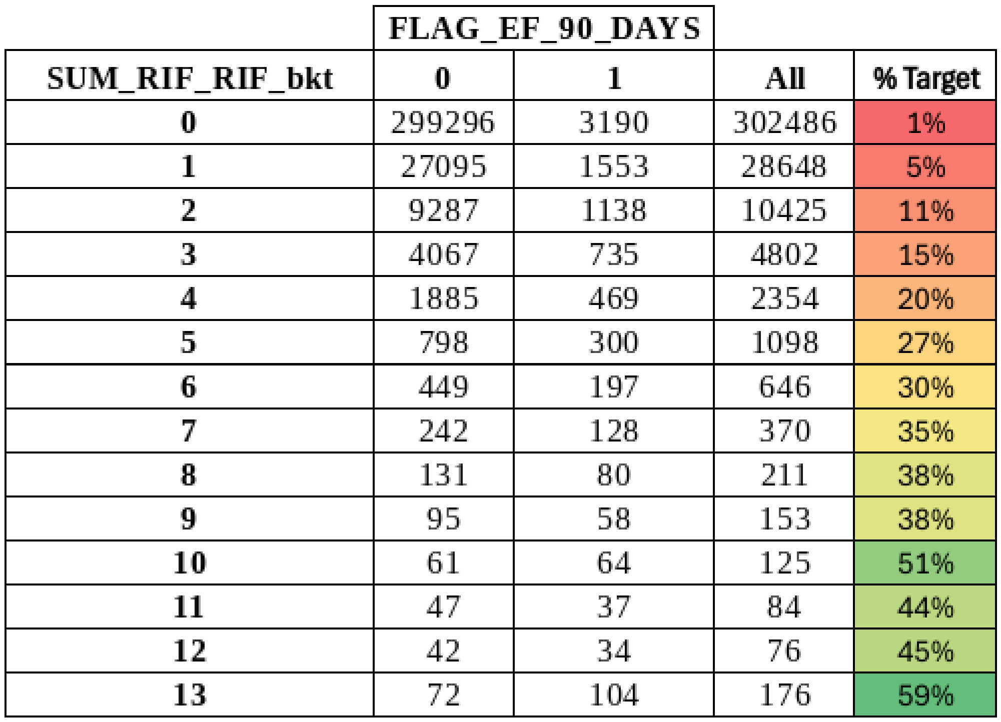
3.1.4.2 Data Preprocessing

Next, the following preprocessing steps were taken:

1. **Missing Value Imputation:** Missing (NaN or Null) values were replaced for zero values in all the antibiotics and Lab test results.
1. **Variable Binning:** The value range for input variables was bucketed to avoid extreme values impact on target probability result.
2. **Standardization:** The standard scaler method $z = (x - u) / s$ from the Python Scikit-learn library was applied, where X is the original value and U is the mean of training samples and S is the standard deviation.

Feature Selection

For feature selection, we considered the relationship between the input distribution and the target variable. This involved checking the business logic and analyzing the statistical distribution of variables. For example, we examined how the target behavior changed across different ranges of the Rimaficin class variable and the rest of input variables.



	FLAG_EF_90_DAYS			
SUM_RIF_RIF_bkt	0	1	All	% Target
0	299296	3190	302486	1%
1	27095	1553	28648	5%
2	9287	1138	10425	11%
3	4067	735	4802	15%
4	1885	469	2354	20%
5	798	300	1098	27%
6	449	197	646	30%
7	242	128	370	35%
8	131	80	211	38%
9	95	58	153	38%
10	61	64	125	51%
11	47	37	84	44%
12	42	34	76	45%
13	72	104	176	59%

3.1.4.3 Model Development Experimentations

Once the data was standardized, several models for binary classification problems were tested, including logistic regression, XGB, and Vertex AI AutoML. The key metrics for the binary classification model were evaluated over the test data with similar thresholds. The results showed that there was no significant improvement using either the boosted model or the AutoML model. The logistic regression model was chosen for its suitability to binary classification problems and its direct interpretability based on the coefficient sign and value. The table below shows the evaluation results across the three types of models used.

	Xgboost	Logistic Regression	AutoML
Precision	0.15	0.15	0.15
Recall	0.54	0.55	0.54

Additionally, we incorporated feedback from subject matter experts to refine and update the features used in each iteration. We conducted three iterations of model development, refining and updating features in each iteration based on the feedback. In each iteration, we experimented with several models for binary classification problems.

3.1.5 Model Inference process

1. Select all the available HCPs in the ADHOC_TBLSR_HCP_DIM for the relevant execution month.
2. Run the feature creation routine for Antibiotics and Lab test results.
3. Fetch the Ready Now HCPs for the execution month.
4. **Prediction:** Use the predict_proba method of the logistic regression model to generate predicted probabilities of enrollment.
5. Filter HCPs with information in at least one non-zero input variable, because otherwise the EF probability will be the lowest possible value.
6. **Thresholding:** A threshold (0.20 in the current implementation) is applied to convert probabilities into binary predictions (0 or 1). This threshold can be adjusted to prioritize precision or recall based on the application's requirements (refer to: precision-recall curve). This threshold is to filter out the HCPs which are significantly less likely to write EF based on model probability, essentially reducing the number of HCPs sent to the UI.

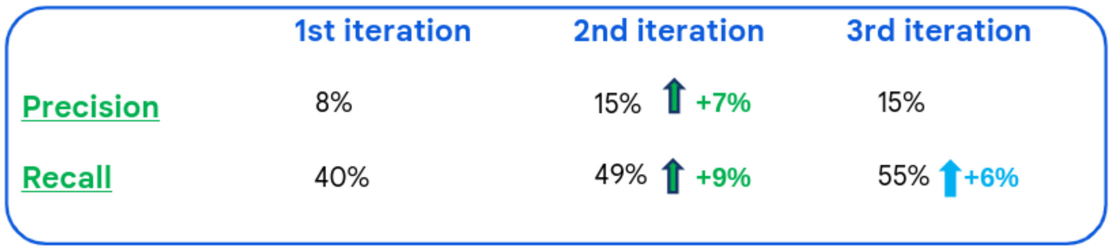
We experimented with bi-weekly vs monthly model inference cadences. The statistics show little differences between the two cadences:

- Model inference was performed on September 15th and 30th with 16,479 and 14,538 HCPs respectively.
- 14,360 HCPs were common to both inferences, with 6,262 of these showing feature changes between the two dates.
- Only for 60 HCPs (around 1%), the difference in enrollment probability between the two dates was greater than 0.05.
- The final output label changed for 27 HCPs, shifting from a positive (1) on September 15th to a negative (0) on September 30th.

Therefore, after discussion with the Insmed team, a monthly inference schedule was determined to be sufficient for our needs.

3.1.6 Model Evaluation

The following shows the model quality improvement through the 3 model iterations.



	1st iteration	2nd iteration	3rd iteration
Precision	8%	15% ⬆ +7%	15%
Recall	40%	49% ⬆ +9%	55% ⬆ +6%

*2nd iteration vs. 1st iteration *3rd iteration vs 2nd iteration

ReadyNow Confusion Matrix

In order to evaluate our model, we compared it against historical results from ReadyNow. The table below illustrates, for Ready Now, the counts of true positives, true negatives, false positives, and false negatives.

	Predicted No Enrollment	Predicted Enrollment
Actual No Enrollment	101399	1693
Actual Enrollment	2138	266

To summarize the results above:

- **True Negative (TN):** 101399 - The model correctly predicted that these HCPs would *not* fill out the form.
- **False Positive (FP)**: 1693 - The model incorrectly predicted that these HCPs *would* fill out the form (when they actually didn't).
- **False Negative (FN)**: 2138 - The model incorrectly predicted that these HCPs would *not* fill out the form (when they actually did).
- **True Positive (TP):** 266 - The model correctly predicted that these HCPs *would* fill out the form.

Overall, ReadyNow is good at predicting which HCPs will *not* fill out the form (high TN). However, it struggles with identifying the HCPs who *will* fill it out (low TP). There is a relatively high number of false negatives, suggesting that it is missing a significant number of HCPs who are actually likely to enroll.

Model 1 Confusion Matrix

For comparison, the table below presents the confusion matrix for our model in one of the iterations of model training. Note, that as we continue to retrain the model over time, preferably on a monthly basis, the results will be different even if there is a fixed `random_state` used in the model training stage.

	Predicted No Enrollment	**Predicted Enrollment**
Actual No Enrollment	95573	7519
Actual Enrollment	1091	1313

To summarize the results above:

- **True Negative (TN):** 95573 - The model correctly predicted that these HCPs would *not* fill out the form.
- **False Positive (FP)**: 7519 - The model incorrectly predicted that these HCPs *would* fill out the form (when they actually didn't).
- **False Negative (FN)**: 1091 - The model incorrectly predicted that these HCPs would *not* fill out the form (when they actually did).
- **True Positive (TP):** 1313 - The model correctly predicted that these HCPs *would* fill out the form.

Comparison: Model 1 vs. ReadyNow

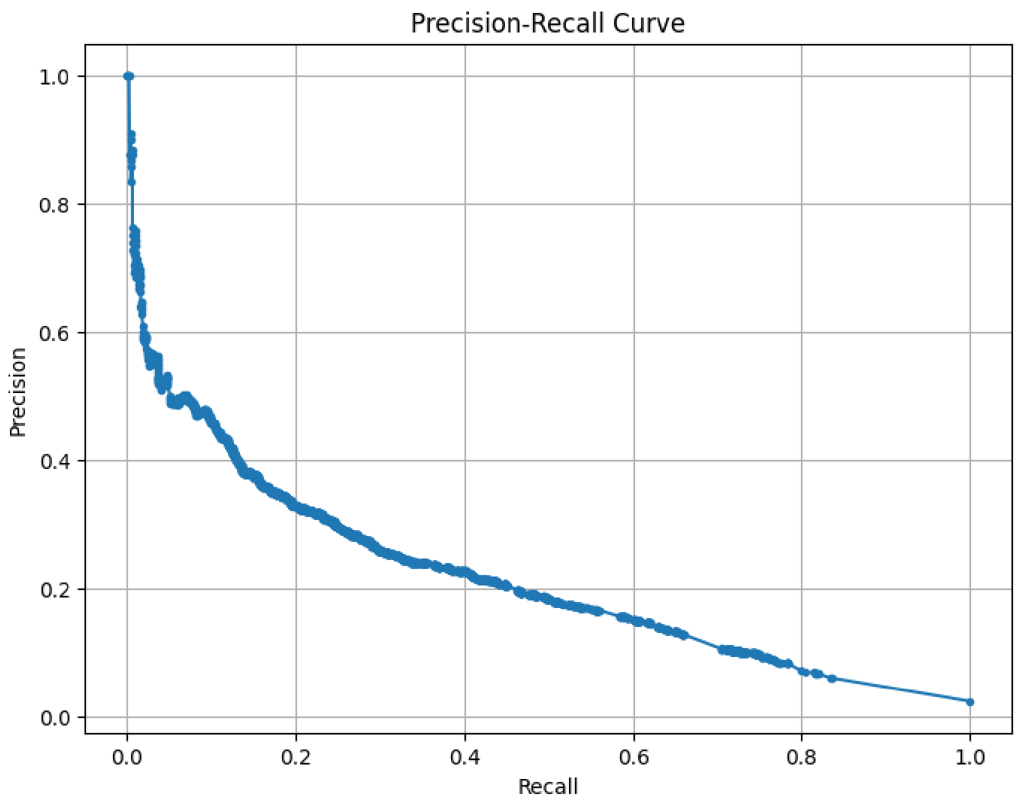
Model 1 shows improvement over ReadyNow in identifying HCPs who will fill out the form:

- **Increased True Positives:** Model 1 has a much higher TP rate (1313 vs. 266), meaning it's better at identifying potential enrollees.
- **Reduced False Negatives:** Model 1 has a lower FN rate (1091 vs. 2138), meaning it misses fewer potential enrollees.

Precision and Recall

Precision and recall scores are calculated to assess the model's accuracy and ability to identify positive cases. The precision-recall curve below provides a visual representation of the trade-off between precision and recall for model 1. For example, from this curve, when we pick recall at 80%, we got precision around 10%.

Note that in the production pipeline, in order to resolve the duplicate HCP issue, filtering is applied and that removes potential candidates, and recall rate is depressed. We recommend removing these filters once the duplicate HCP issue is resolved from the data side.



ROC Curve and AUC

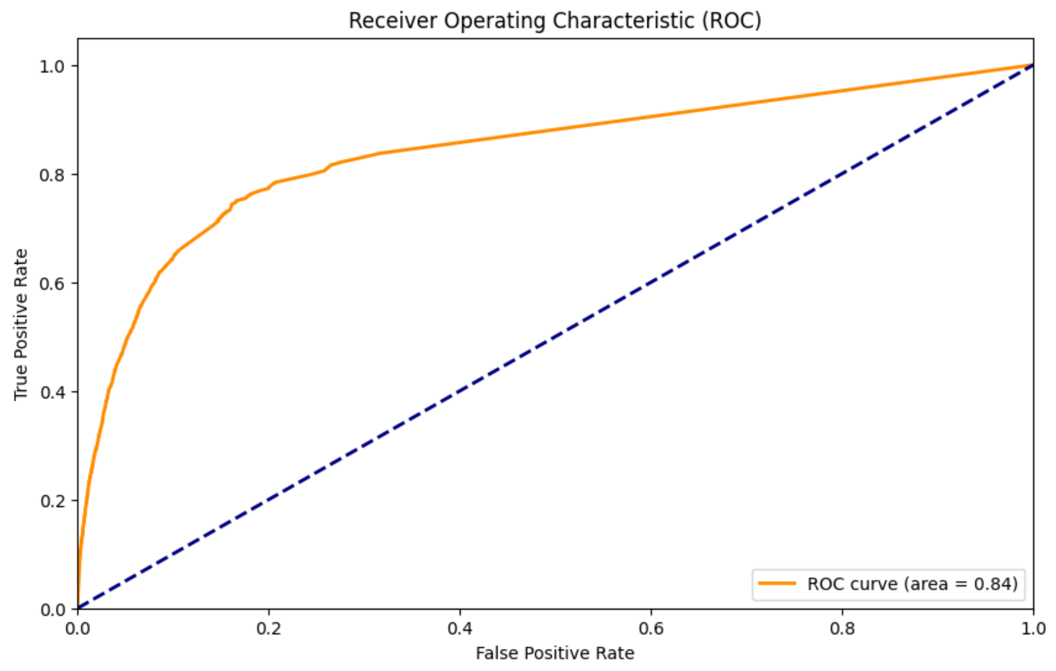
This Receiver Operating Characteristic (ROC) curve below shows the model's performance in distinguishing between HCPs who will enroll and those who won't. It plots the True Positive Rate (sensitivity) against the False Positive Rate (1-specificity) at various threshold settings.[1]

The curve is well above the diagonal line, indicating that the model performs better than random chance in predicting HCP enrollment. The area under the curve (AUC) is 0.84. This is a good AUC score, generally indicating a strong ability to discriminate between the two classes.[2]

[1] Interpretation of the ROC curve:

- X-axis (False Positive Rate): Shows the proportion of actual negative cases (HCPs who did not enroll) that are incorrectly predicted as positive (predicted to enroll).
- Y-axis (True Positive Rate): Shows the proportion of actual positive cases (HCPs who enrolled) that are correctly predicted as positive.
- Diagonal Line: Represents a random classifier (like flipping a coin). A good model should be significantly above this line.

[2] For context, an AUC of 0.5 means the model is no better than random, while an AUC of 1.0 represents a perfect classifier.



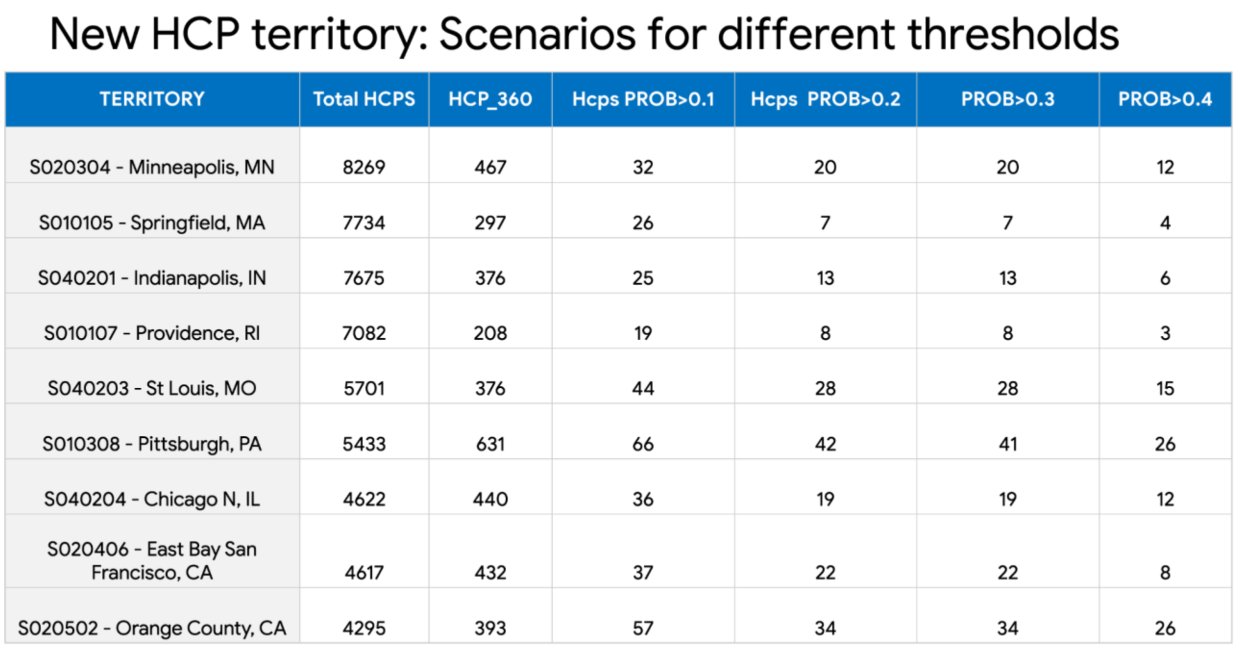
KS Statistic

The KS statistic is a measure of how well the classification model separates the HCPs who are likely to enroll from those who are not. It essentially quantifies the maximum difference between the cumulative distribution of the two groups. The model had **a KS statistic of 0.58** indicating**:**

- **Good discriminatory power:** The model demonstrates a good ability to distinguish between HCPs who are likely to enroll and those who are not.[3]
- **Not perfect:** While 0.58 is a decent score, it's not perfect (a perfect score would be 1.0). This means there's still some overlap between the predicted probabilities for the two groups, and the model isn't perfectly separating them.

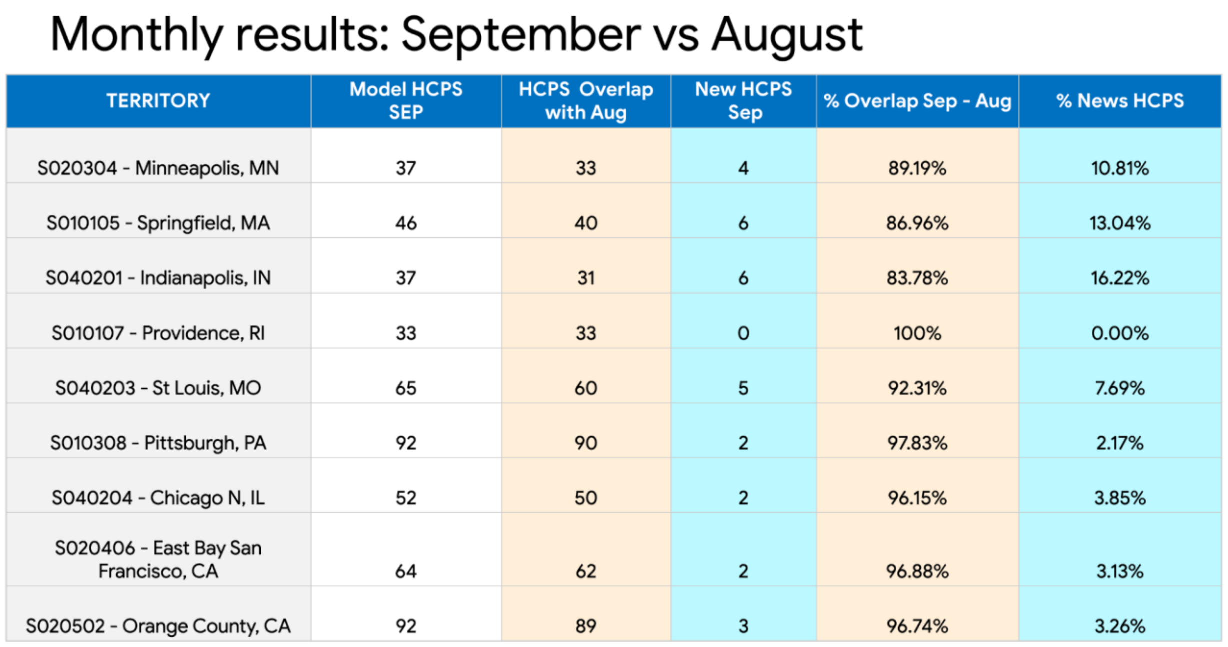
[3] A higher KS statistic generally indicates better separation.

3.1.7 New Territory Statistics of Model Output



New HCP territory: Scenarios for different thresholds

TERRITORY	Total HCPS	HCP_360	Hcps PROB>0.1	Hcps PROB>0.2	PROB>0.3	PROB>0.4
S020304 - Minneapolis, MN	8269	467	32	20	20	12
S010105 - Springfield, MA	7734	297	26	7	7	4
S040201 - Indianapolis, IN	7675	376	25	13	13	6
S010107 - Providence, RI	7082	208	19	8	8	3
S040203 - St Louis, MO	5701	376	44	28	28	15
S010308 - Pittsburgh, PA	5433	631	66	42	41	26
S040204 - Chicago N, IL	4622	440	36	19	19	12
S020406 - East Bay San Francisco, CA	4617	432	37	22	22	8
S020502 - Orange County, CA	4295	393	57	34	34	26



Monthly results: September vs August

TERRITORY	Model HCPS SEP	HCPS Overlap with Aug	New HCPS Sep	% Overlap Sep - Aug	% News HCPS
S020304 - Minneapolis, MN	37	33	4	89.19%	10.81%
S010105 - Springfield, MA	46	40	6	86.96%	13.04%
S040201 - Indianapolis, IN	37	31	6	83.78%	16.22%
S010107 - Providence, RI	33	33	0	100%	0.00%
S040203 - St Louis, MO	65	60	5	92.31%	7.69%
S010308 - Pittsburgh, PA	92	90	2	97.83%	2.17%
S040204 - Chicago N, IL	52	50	2	96.15%	3.85%
S020406 - East Bay San Francisco, CA	64	62	2	96.88%	3.13%
S020502 - Orange County, CA	92	89	3	96.74%	3.26%

3.1.8 Data Validation and Filtered HCP Data

The Google team used the HCP 360 SQL for exploratory data analysis (EDA) and model training purposes. This replication of HCP 360 tables covered the period from January 2023 to April 2024. We selected the first available date for each month from the **ADHOC_TBLSR_HCP_DIM_HISTORY** table. For joining operations with the tables listed below, we selected the closest corresponding dates.

- **ADHOC_TBLSR_PROGNOS_HISTORY**
- **ADHOC_TBLSR_ANTIBIOTICS_HISTORY**
- **ADHOC_TBLSR_READYNOW_HISTORY**
- **ADHOC_TBLSR_RESTARTS_HISTORY**

Data Quality Checks

In the first data quality check, we verified the total number of HCPs considered for model development. Specifically, we compared the number of HCPs against the official numbers present in the available HCP 360 reports. There was an almost exact match between the two (see the table below for the comparison). The Insmed IT team confirmed that the minor differences were due to discrepancies between the production environment and UAT environment used by the Google AI team.

Segment	HCP_360 Jan - 2024	AI team Replication Jan - 2024	% Coincidence
Current Writer	811	817	101%
Lapsed Writer	3072	3040	99%
Non-Writer	17851	17810	100%

Filtering Criteria

The filtering criteria to select HCPs for model 1 matches the filtering criteria in the HCP 360 SQL report. Specifically, the following filters were applied:

- `PDRPOptOut <> 'Y'`
- `HCPStatus ='Active'`
- `PrimarySpecialty not in (select Specialty from TS_DATABASES.MARKETING_TBLSHYFT_HCPEXCLUSION_XREF)`
- `SecondarySpecialty not in (select Specialty from TS_DATABASES.MARKETING_TBLSHYFT_HCPEXCLUSION_XREF)`
- `NTMTier in ('Tier 1', 'Tier 2', 'Tier 3', 'Tier 4')) or ((NTMTier is null or NTMTier = 'Tier 5')`
- `EFsYTD>0 or H.CallsLast12Months>0 or P.AFBTests6Months>0 or P.AFBTestsMAC6Months>0 or .Azithromycin>0 or A.Ethambutol>0 or A.Rifampin>0 or CountofPatients>0 or InsmedPatientID>0`
- `Region is not NULL`

Data Validation on New Data

Since September 2024 Insmed has established new SQL logic for the HCP 360 report and a new territory structure. The Google team has received this document and has replicated the code getting the same results for all the variables used in the process. The results for this exercise are available in Snowflake in the COMMERCIAL_AI database in the table **HCP_360_SEPTEMBER_GOOGLE_F**.

3.2 Priority Models

Priority models include **model 2 (Improved Patient Journey)** and **model 3 (HCP Priority)**. Model 2 queries for new enrollment forms on a daily basis, and creates notifications if new enrollments have not been followed up within 24-48 hours. Model 3 takes in as inputs the outputs from model 1 and model 2, adding TS engagement data insight, to prioritize HCPs for the TSs. Model 3 also provides insights,reasoning, and a standard message type for each HCP it processes.

3.2.1 Improved Patient Journey Model (Model 2)

The **Improved Patient Journey** model provides early identification of new enrollment forms and their follow-ups done by TSs. The model provides alerts or notifications based on a new enrollment form and a *no-call* following the EF. Key objectives include:

- **Early Identification:** Promptly identifying HCPs with patients who have recently submitted enrollment forms (EFs) for Arikayce, specifically within the initial 24-48 hours of the onboarding process.
- **Proactive Outreach:** Identifying and creating opportunities for TSs to proactively engage with HCPs who have patients with recent EF submissions. This will facilitate timely

intervention and support, potentially improving patient outcomes and adherence.

Initially, we explored various aspects of the patient journey model, including discontinuity analysis and patient status journey. Extensive exploratory data analysis (EDA) was conducted on both approaches; however, it did not translate to a direct requirement for the TSs.

Ultimately, we focused on delivering a solution that prioritizes the TS's efforts by providing a filtered list of EFs with relevant status and call activity data, enabling them to efficiently track the early stages of the patient journey and identify potential blockers.

Below are the two notifications/alerts that were agreed to.

3.2.1.1 Notification 1: New Enrollment Form Generated

- **Frequency:** This notification is generated daily.
- **Description:** Identifies all new enrollment forms (EFs) received on the previous day by querying the ADHOC_TBLPATIENT_360_MASTER table and filtering on the DATE_START_FORM_RECEIVED column.
- **Fields:**
  - **NOTIFICATION_DATE:** {CURRENT_DATE}
  - **NOTIFICATION_NAME:** New Enrollment Form Generated
  - **NOTIFICATION_DESCRIPTION:** A new EF has been generated on {DATE_START_FORM_RECEIVED} by HCP {EF_PHYSICIAN_FIRST_NAME} {EF_PHYSICIAN_LAST_NAME}

3.2.1.2 Notification 2: No HCP calls for 48 hours after new EF generation

- **Frequency:** This notification is generated daily.
- **Description:**
  - Identifies enrollment forms (EFs) submitted two days prior to the current date using the **ADHOC_TBLPATIENT_360_MASTER** table and filtering on the **DATE_START_FORM_RECEIVED** column.
  - Checks for any in-person meetings (using **INSMED_CALL_TYPE** filter) with the associated HCPs in the **ADHOC_TBLDF_SHYFT_CALL_ADHOC** table within the two days following EF submission.
- **Notification Trigger:** If no in-person meetings are found, a notification is generated to alert relevant teams.
- **Fields:**
  - **NOTIFICATION_DATE:** {CURRENT_DATE}
  - **NOTIFICATION_NAME:** No HCP calls for 48 hours after new EF generation
  - **NOTIFICATION_DESCRIPTION:** A new EF has been generated on {DATE_START_FORM_RECEIVED} by HCP {EF_PHYSICIAN_FIRST_NAME}

{EF_PHYSICIAN_LAST_NAME}. 48 hours have passed since then, and no calls have been initiated.

3.2.1.3 Model 2 Artifacts

The generation of the two notifications outlined above is achieved through a combination of SQL queries and a Python script.

- Github Link
- SQL Scripts

3.2.1.3 Model 2 Snowflake Tables Considered

Below are the tables and relevant columns used for generating alerts.

Tables	Important Columns	Table/Column Description
ADHOC_TBLPATIENT_360_MASTER	DATE_START_FORM_RECEIVED, PRODUCT_DESCRIPTION	Contains details about enrollment forms, including date of enrollment and product specification.
ADHOC_TBLDF_SHYFT_CALL_ADHOC	INSMED_HCP_ID, CALLDATE, INSMED_CALL_TYPE	Stores information about calls made to HCPs, including the date, type of call, and the associated HCP's Insmed ID.
ADHOC_TBLSR_HCP_DIM	INSMEDID, NPIID	Provides a mapping between HCPs' Insmed IDs and their NPIIDs or EF_PHYSICIAN_NPI.

3.2.1.4 Model 2 Output

Model 2 generates outputs that are used as input for model 3, which creates personalized recommendations for TSs, and uses models 1 and 2 outputs to adjudicate priority. The data points generated by Model 2 are stored in a Snowflake table with below mentioned details.

- **Database Name:** COMMERCIAL_AI.SALES
- **Table Name: AI_TS_NOTIFICATIONS**
- **Schema/Columns :**
  - `'INSMED_PATIENT_ID',`
  - `'DATE_START_FORM_RECEIVED',`
  - `'EF_PHYSICIAN_NPI',`
  - `‘EF_PHYSICIAN_FIRST_NAME',`
  - `'EF_PHYSICIAN_LAST_NAME',`

- 'EF_PHYSICIAN_FACILITY_NAME',
- 'EF_PHYSICIAN_ZIP_CODE',
- 'SP_PHARMACY_ID',
- 'EF_SALES_FIELD_REGION',
- 'EF_SALES_FIELD_TERRITORY',
- 'LATEST_SP_STATUS',
- 'LATEST_SP_SUB_STATUS',
- 'LATEST_SP_STATUS_DATE',
- 'LATEST_CASE_MANAGER_STATUS',
- 'LATEST_CASE_MANAGER_SUBSTATUS',
- 'LATEST_CASE_MANAGER_STATUS_REASON',
- 'LATEST_CASE_MANAGER_STATUS_DATE',
- 'NOTIFICATION_DATE',
- 'NOTIFICATION_NAME',
- 'NOTIFICATION_DESCRIPTION'

- **Important columns:**
  - "EF_PHYSICIAN_NPI",
  - "INSMED_PATIENT_ID",
  - "NOTIFICATION_DATE",
  - "NOTIFICATION_NAME",
  - "NOTIFICATION_DESCRIPTION"

3.2.2 Model 3 - HCP Priority Model (Model 3)

Model 3 comprises an algorithm that is used to assign a priority value to each HCP and three pieces of textual guidance (*insights, reasoning*, and *messages*). This priority value is used to rank HCPs and guide TSs in their outreach efforts, ensuring that those with the highest potential for business impact are prioritized. The textual guidance provides additional context in natural language.

The ranking and priority score model synthesizes outputs from models 1 and 2 and other data to prioritize HCPs in rank order. This model outputs a priority score (used for ranking HCPs) and narrative text that provides insights and reasoning relevant to the generated priority score.

3.2.2.1 Model Input Features

Model 3 uses several inputs from different sources, including models 1 and 2 outputs; HCP profile information, and HCP engagement data. These are described in greater detail in the sections below.

Model 1 Output and HCP Profile Information

DESCRIPTION	● Probability of an enrollment form (with range between 0 and 1). ● General information about the HCP and aggregated information about patients, including the number of patients on antibiotics, the number of AFB tests conducted, and the number of MAC positive tests.

SOURCE	The following SQL query is used to retrieve both model 1 outputs and HCP profile information
	`SELECT`
	`  *`
	`FROM`
	`  AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS`
	`WHERE`
	`  RUNID= (SELECT MAX(RUNID) FROM AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS)"`

ADDITIONAL NOTES	● HCP profile information is included in the same table as the model 1 outputs, so a single query is used to retrieve both the model 1 output and the HCP profile information ● For more details, see hcp_priority_data_creation

Model 2 Output

DESCRIPTION	● An indicator for whether there has been a recent enrollment form and an indicator for whether or not the HCP has been contacted within 48 hours following the enrollment.

SOURCE	The following SQL query is used to retrieve model 2 outputs:
	`SELECT`
	`  EF_PHYSICIAN_NPI,`
	`  INSMED_PATIENT_ID,`
	`  NOTIFICATION_DATE,`

	`NOTIFICATION_NAME,`
	`NOTIFICATION_DESCRIPTION`
	`FROM`
	`AI_TS_NOTIFICATIONS;`
	
	▯

ADDITIONAL NOTES	● The output from model 2 is subset to records where NOTIFICATION_DATE is less than or equal to the base date. ● Further pivots and transformations are made to identify the latest EF date and the number of recent EFs.

HCP Arikayce Patient details

DESCRIPTION	● The details of all the Arikayce patients enrolled by a HCP in last 1 year and also finds out total enrollments in last 6 months which helps in categorization of the HCP being non-writer, lapsed writer, or current writer.

SOURCE	The following SQL query is used to retrieve Arikayce patients details:
	
	▯`SELECT`
	`*`
	`FROM`
	`ADHOC_TBLPATIENT_360_MASTER;`
	
	▯

ADDITIONAL NOTES	● The output is subset to records where DATE_START_FORM_RECEIVED is less than 1 year or equal to the base date. ● Further uses the patients enrollment counts to categorize the HCP writer status

HCP Engagement Data

DESCRIPTION	● Information related to recent engagement with the HCP (primarily used for determining whether an HCP was visited recently).

SOURCE	The following SQL query is used to retrieve HCP engagement data:
	`SELECT`
	`  *`
	`FROM`
	`  ADHOC_TBLDF_SHYFT_CALL_ADHOC;`

ADDITIONAL NOTES	● Additionally, use ADHOC_TBLSR_HCP_DIM table to map the calls with NPIID based on INSMEDID and to obtain SEGMENT for the HCPs. ● The following columns are used from the table: ○ CALLDATE ○ CALLTYPE ○ INSMED_CALL_TYPE ○ COMMENTS ○ DISCUSSION_TOPICS ○ DETAIL_PRIORITY ○ ATTENDEES ○ TERRITORY ○ INSMED_HCP_ID ○ NPIID (appended for further mapping to other tables) ● Filtering call data for only in person visit with specific following entries in INSMED_CALL_TYPE: ○ "Live – Staff Only" ○ "Live - HCP and Staff" ○ "Live - HCP Only" ○ "In Person" ○ "Phone Call" ○ "Phone Call w/Staff" ○ "HCP Only" ○ "Phone Call w/HCP" ○ "HCP and Staff" ○ "Staff Only" ○ "Virtual – HCP Only" ○ "Virtual - Staff Only" ○ "Virtual Call"

	○ "Face to Face" ● For more details on transformations, see MLOps code

3.2.2.2 Model Output

Model 3 outputs include the following:

- **Priority Score -** Generated using the HCP prioritization algorithm (described in the
- GCC - Insmed TS Pre-Call Plan TDD section below)
- **Insights -** Generated using rules-based logic (described in the GCC - Insmed TS Pre-Call Plan TDD section below)
- **Reasoning -** Generated using rules-based logic (described in the GCC - Insmed TS Pre-Call Plan TDD section below)
- **Messages -** Generated using rules-based logic (described in the GCC - Insmed TS Pre-Call Plan TDDsection below)

The code that generates the model outputs and stores them in Snowflake can be found here: hcp_priority_data_creation. The outputs are stored in the database table COMMERCIAL_AI.SALES.AI_TS_HCP_PRIORITY_RANKING and can be retrieved using the following query:

```
SELECT
  *
FROM
  COMMERCIAL_AI.SALES.AI_TS_HCP_PRIORITY_RANKING;
```

3.2.2.3 HCP Prioritization Algorithm

The prioritization algorithm utilizes a formula that incorporates the following key factors:

- **HCP enrollment status -** An indicator for whether there has been a recent enrollment.
- **Recent engagements -** An indicator for whether there has been a recent engagement between an HCP and a TS since enrollment.
- **Probability of an enrollment -** The probability of the HCP having an enrollment (also the output of the ML model1).
- **HCP writer status -** An indicator for whether the HCP is an existing writer, a non-writer, or a lapsed writer.

These factors were selected because of their relevance to a TSs' considerations when deciding which HCPs to prioritize to increase the likelihood of desired outcomes.

The formula for prioritization is as follows:

$$\begin{aligned} Expression_1 &= Status_{\text{Enrollment}} + Engagement_{\text{Recent}} \\ Expression_2 &= (0.8 * Prob_{\text{Enrollment}}) + (0.2 * Status_{\text{Writer}}) \\ \textbf{PrioritizationScore} &= \max(\textbf{Expression}_\textbf{1}, \textbf{Expression}_\textbf{2}) \end{aligned}$$

where:

$$Status_{\text{Enrollment}} = \begin{cases} 1, & \text{if there is an enrollment} \\ 0, & \text{otherwise} \end{cases}$$

$$Engagement_{\text{Recent}} = \begin{cases} -1, & \text{if there has been a recent engagement} \\ 0, & \text{otherwise} \end{cases}$$

$$Status_{\text{Writer}} = \begin{cases} 0, & \text{if Non-Writer} \\ 1, & \text{if Current Writer} \\ 0.5, & \text{if Lapsed Writer} \end{cases}$$

$$0 \leq Prob_{\text{Enrollment}} \leq 1$$

Formula Rationale

The formula is designed to capture the interplay of factors that influence HCP prioritization.

- *Expression₁* prioritizes HCPs that have had a recent enrollment and have not been visited since that enrollment. The formula ensures that these HCPs receive the highest possible score of 1. If the HCP has had a recent enrollment, and has been engaged with since the enrollment, then the HCP receives a score of 0 - in this case, the recent engagement effectively nullifies the impact of a recent enrollment.
- *Expression₂* has an effect when actual enrollments and engagement are not relevant. In this case, HCPs are prioritized based on their *probability* of a future enrollment as well as their writer status. Current writers are the highest priority, lapsed writers are second highest priority, and non-writers are the lowest priority. *Expression₂* gives a higher weight to the probability of enrollment, and a smaller weight to the HCP writer status, reflecting the higher importance of the former.

The MAX function ensures that HCPs who score high on either expression are prioritized, recognizing that different combinations of factors (namely whether or not there is an HCP with an actual enrollment that has not been visited) can impact an HCP's priority.

More information on the implementation is provided in Appendix 9.1 (code) and Appendix 9.2 (configuration).3.1.3.4 Natural Language Guidance

To help TS understand what factors influence an HCP's prioritization score and inform next steps, three textual outputs are generated for display on the front-end. These are:

GUIDANCE ITEM	DESCRIPTION
Insights	A concise summary describing HCPs, including: ● **HCP status:** Whether they are a new, existing, or lapsed prescriber of Arikayce. ● **Patient enrollment activity:** Number of new patients enrolled in the last month. ● **Enrollment probability:** Model 1 output ● **Antibiotic use:** Types of antibiotics being used by their patients. ● **Lab test results:** Number of AFB and MAC tests ordered and the number of positive MAC results.
Reasoning	Indicates what factors influenced the HCP's prioritization score, focusing primarily on: ● Recent enrollment activity ● Enrollment probability ● Visit post Enrollment ● HCP writer status
Messages	The Message selects and displays based on 11 approved messages and also stages of the HCP on the Arikayce journey. Additionally it also showcases any enrollment and no calls made within 48hrs of enrollment event messages. The mapping can be found in the below section.

We experimented with two approaches to determine which was the most effective for generating these textual outputs.

1. First, we tried using Generative AI, trying to instruct the **Gemini LLM** to generate output based on some examples and data inputs.
2. After several iterations and review with Insmed, we converged on a fairly templatized output, which strengthened the case for using a rules-based approach to generating the text, rather than the Gemini API.
    a. Using a rules-based approach in code is faster, less costly, and sufficient for the outputs needed.

However, there may be a need in the future to switch back to using a generative AI approach - hence, both the rules-based logic and the generative AI logic is described below.

3.2.2.3.1 Future Changes to Writer Segment

Should there be any future changes to the writer segment (e.g., new categories emerge), this may require changes in the HCP prioritization codebase. These changes would be made in the following locations:

FILE	DESCRIPTION
src/priority_models/prioritization_calculator.py	In the `PrioritizationCalculator` class, there is logic to set the `HCP_WRITER_STATUS_WEIGHT` based on different values for the writer. If there are new categories for writers, this area should be updated to reflect the new mapping of weights.
src/priority_models/hcp_priority_reasoning_utils.py	In the `insight_generation` function, there is logic to generate insights based on the writer status. Similarly, there is logic in `no_val_insight_generation` for generation insights based on the writer status.

3.2.2.4 Rules-based generation

Rule-based generation considers TS UI requirements and aligns with Insmed's business logic and needs. Three types of text are generated:

- The **Insights** section summarizes HCP data and engagement details
- The **Reasoning** section explains the priority score and its influencing factors.

- The **Messages** section provides the list of approved messages for the TS to use when engaging with the HCP (based on the data attributes).

The logic for each of these sections is described below.

3.2.2.4.1 Insight

The code snippet below shows how Insights are generated. The function generates a textual insight report based on input data about the HCP. The report includes information on:

- **Enrollment:** Number of enrollments and whether the HCP was contacted within 48 hours.
- **Writer Status:** The HCP's writer status (current, lapsed, non-writer) and probability of a new enrollment.
- **Antibiotic Prescriptions:** Details on antibiotic prescriptions for Azithromycin, Clarithromycin, Ethambutol, Rifampin, and Rifabutin.
- **MAC Test Results:** Number of MAC positive AFB tests ordered by the HCP in the last 6 months.

Input parameters:

- x: A dictionary containing HCP data (e.g., enrollment counts, writer status, prescription information, test results).
- enrollment_interval: Time interval for enrollment data (e.g., "30 days", "90 days").
- writer_name_to_int: A dictionary mapping writer status names to numerical values.
- writer_status_text: A dictionary mapping numerical writer status values to textual descriptions.

Output:

- A formatted string containing the generated insights.

Logic:

The function uses conditional statements to analyze the input data and construct the insight report. It incorporates data from the input dictionary (x) and utilizes the provided dictionaries (writer_name_to_int, writer_status_text) to generate descriptions.

Code snippet:

Insight generation without numerical value shown on UI

```
writer_status_text = {
    "1.0": "The HCP is a current writer",
    "2.0": "The HCP has recently converted from non-writer to a current writer",
    "0.5": "The HCP is a lapsed writer",
    "0.0": "The HCP is a non-writer"
}

writer_name_to_int = {
    "Current Writer": 1.0,
    "Lapsed Writer": 0.5,
    "Non-Writer": 0.0
}


def no_val_insight_generation(

    x, enrollment_interval, use_segment, writer_name_to_int, writer_status_text

):

    insight = ""

    ##Enrollment

    if (x["NO_OF_EF_ENROLLED"] > 0) & (x["NO_OF_NO_HCP_CALL"] > 0):

        insight += f"There were {int(x['NO_OF_EF_ENROLLED'])} enrollment and {int(x['NO_OF_NO_HCP_CALL'])} occurence where the HCP was not
contacted within 48 hours of the enrollment in last {enrollment_interval} days.\n"

    elif x["NO_OF_EF_ENROLLED"] > 0:

        insight += f"There were {int(x['NO_OF_EF_ENROLLED'])} enrollment in last {enrollment_interval} days.\n"

    ## Writer status

    writer_status = writer_name_to_int[x["HCP_WRITER_STATUS"]]

    if (

        (x["HCP_WRITER_STATUS"] == 1)

        & (x["NO_OF_EF_ENROLLED"] == x["ARIKAYCE_TOTAL_USER"])

        & (x["NO_OF_EF_ENROLLED"] > 0)

    ):

        writer_status += 1.0

    if use_segment and (type(x["SEGMENT"]) == str):

        insight += f"The HCP is a {x['SEGMENT'].lower()}.\n"

    else:

        insight += writer_status_text[str(float(writer_status))]

        insight += "\n"

    if writer_status == 2.0:

        insight += (
            "The HCP has recently converted from non writer to a current writer.\n"

         )

    ## Number of patients

    if writer_status > 0.5:

        insight += f"The HCP does have new patient enrollments for Arikayce in the last 6 months.\n"

    ##First line antibiotics


    if (int(x["SUM_AZI_CLAR"]) + int(x["ETHAMBUTOL"]) + int(x["SUM_RIF_RIF"])) > 0:

        insight += f"The HCP has patients on antibiotics.\n"

    else:

        insight += "The HCP has no patients on antibiotics(Azithromycin,Clarithromycin,Ethambutol,Rifabutin or Rifamin).\n"

    ##Mac test details

    if int(x["AFBTESTSMAC6MONTHS"]) > 0:

        insight += f"There were some MAC positive results for all the AFB tests ordered in the last 6 months by the HCP."

    else:

        insight += f"There were no AFB tests ordered by the HCP."

    return insight
```

Insight generation with numerical value shown on UI

```
writer_status_text = {
    "1.0": "The HCP is a current writer",
    "2.0": "The HCP has recently converted from non-writer to a current writer",
    "0.5": "The HCP is a lapsed writer",
    "0.0": "The HCP is a non-writer"
}

writer_name_to_int = {
    "Current Writer": 1.0,
    "Lapsed Writer": 0.5,
    "Non-Writer": 0.0
}

def insight_generation(
    x, enrollment_interval, writer_name_to_int, use_segment, writer_status_text
):
    insight = ""

    ##Enrollment
    if (x["NO_OF_EF_ENROLLED"] > 0) & (x["NO_OF_NO_HCP_CALL"] > 0):
        insight += f"There were {int(x['NO_OF_EF_ENROLLED'])} enrollment and {int(x['NO_OF_NO_HCP_CALL'])} occurence where the
HCP was not contacted within 48 hours of the enrollment in last {enrollment_interval} days.\n"
```

```
elif x["NO_OF_EF_ENROLLED"] > 0:
    insight += f"There were {int(x['NO_OF_EF_ENROLLED'])} enrollment in last {enrollment_interval} days.\n"

## Writer status
writer_status = writer_name_to_int[x["HCP_WRITER_STATUS"]]

if (
    (x["HCP_WRITER_STATUS"] == 1)
    & (x["NO_OF_EF_ENROLLED"] == x["ARIKAYCE_TOTAL_USER"])
    & (x["NO_OF_EF_ENROLLED"] > 0)
):
    writer_status += 1.0

if use_segment and (type(x["SEGMENT"]) == str):
    insight += f"The HCP is a {x['SEGMENT'].lower()}"  # writer_status_text[str(float(writer_status))]
else:
    insight += writer_status_text[str(float(writer_status))]

insight += (
    f" and has a {int(x['PROBABILITY_EF']*100)}% probability of a new enrollment.\n"
)

if (writer_status == 2.0) and (use_segment):
    insight += (
        "The HCP has recently converted from non writer to a current writer.\n"
    )

## Number of patients
if writer_status > 0:
    if x["ARIKAYCE_TOTAL_USER"] == x["ARIKAYCE_ACTIVE_USER"]:
        insight += f"The HCP has total {int(x['ARIKAYCE_TOTAL_USER'])} patient enrollment and it all has been in the last 6 months.\n"
    else:
        insight += f"The HCP has total {int(x['ARIKAYCE_TOTAL_USER'])} patient enrollment in last year and
{int(x['ARIKAYCE_ACTIVE_USER'])} out of it has been in the last 6 months.\n"

##First line antibiotics

if (int(x["SUM_AZI_CLAR"]) + int(x["ETHAMBUTOL"]) + int(x["SUM_RIF_RIF"])) > 0:
    insight += f"The HCP has patients on antibiotics (Azithromycin or Clarithromycin : {x['SUM_AZI_CLAR']}, Ethambutol:
{x['ETHAMBUTOL']}, and Rifampin or Rifabutin: {x['SUM_RIF_RIF']}).\n"
else:
    insight += "The HCP has no patients on antibiotics(Azithromycin,Clarithromycin,Ethambutol,Rifabutin or Rifamin).\n"

##Mac test details
if (int(x["AFBTESTSMAC6MONTHS"]) + int(x["AFBTESTS6MONTHS"])) > 0:
    insight += f"There were {x['AFBTESTSMAC6MONTHS']} MAC positive results out of total {x['AFBTESTS6MONTHS']} AFB tests
ordered in the last 6 months by the HCP."
else:
    insight += f"There were no AFB tests ordered by the HCP."

return insight
```

3.2.2.4.2 Reasoning

This function generates a textual explanation of the factors contributing to the HCP's priority score. The reasoning focuses on:

- **New Enrollments:** Highlights new enrollments as a major contributor to the score.

- **Enrollment Probability:** If no new enrollments, emphasizes the probability of future enrollments.
- **Writer Status:** Notes if the HCP is a "Current Writer" or "Lapsed Writer" as a contributing factor.
- **Recent Visits:** Indicates if a recent visit has reduced the priority score or if the absence of recent visits increases the score.

Input parameters:

- x: A dictionary containing HCP data (e.g., enrollment counts, writer status, visit information, enrollment probability).

Output:

- A formatted string explaining the factors influencing the priority score.

Logic:

The function uses conditional statements to analyze the input data (x) and construct the reasoning text.

Code snippet:

Reasoning generation without numerical value shown on UI comes from a logic like the following:

```
def no_val_reasoning_generation(x):
    reasoning = ""

    if x["NO_OF_EF_ENROLLED"] > 0:
        reasoning += (
            f"The major contributor to the priority score is the new enrollments.\n"
        )

        if x["VISITS_LAST_2_WEEKS"] > 0:
            reasoning = "Due to recent visit the priority score has been reduce."

    else:
        reasoning += f"The major contributing factor for the priority score is the new enrollment probability chances.\n"

        if x["HCP_WRITER_STATUS"] == "Current Writer":
            reasoning += "Another contributing factor for the priority is the HCP being a Current Writer.\n"
        elif x["HCP_WRITER_STATUS"] == "Lapsed Writer":
            reasoning += "Another contributing factor for the priority is the HCP being a Lapsed Writer.\n"

        if x["VISITS_LAST_2_WEEKS"] > 0:
            reasoning = "Due to recent visit the priority score has been reduce."
        else:
```

```
        reasoning += "Another contributor to the prioirty score is no recent inperson visits in last 2 weeks."
    return reasoning
```

Reasoning generation with numerical value shown on UI comes from a logic like the following:

```
def reasoning_generation(x):
    reasoning = ""

    if x["NO_OF_EF_ENROLLED"] > 0:
        reasoning += (
            f"The major contributor to the priority score is the "
            f"{int(x['NO_OF_EF_ENROLLED'])} new enrollments\n"
        )

        if x["VISITS_LAST_2_WEEKS"] > 0:
            reasoning = "Due to recent visit the priority score has been reduce"

    else:
        reasoning += (
            f"The major contributing factor for the priority score is "
            f"{np.round(float(x['PROBABILITY_EF'])*100, decimals=3)}% "
            f"chances for new enrollment\n"
        )

        if x["HCP_WRITER_STATUS"] == "Current Writer":
            reasoning += (
                "Another contributing factor for the priority is the HCP "
                "being a Current Writer\n"
            )
        elif x["HCP_WRITER_STATUS"] == "Lapsed Writer":
            reasoning += (
                "Another contributing factor for the priority is the HCP "
                "being a Lapsed Writer\n"
            )

        if x["VISITS_LAST_2_WEEKS"] > 0:
            reasoning = "Due to recent visit the priority score has been reduce"
        else:
            reasoning += (
                "Another contributor to the priority score is no recent "
                "inperson visits in last 2 weeks\n"
            )
    return reasoning
```

3.2.2.4.3 Messages

This function generates a list of relevant messages based on the HCP data. The messages are categorized into different stages:

- `initial`
- `pre-enrollment`
- `on-enrollment`
- `post-enrollment`
- `enrollment`
- `Nocall_48hrs`

Input parameters:

- x: A dictionary containing HCP data (e.g., enrollment counts, writer status, FLAG_EDA value).

Output:

- A list of strings representing relevant messages for the HCP.

Logic:

The function uses conditional statements to determine the appropriate message category based on the input data (x). It starts by checking if there are new enrollments (NO_OF_EF_ENROLLED). Based on this and other factors like NO_OF_NO_HCP_CALL, HCP_WRITER_STATUS, and FLAG_EDA, it selects messages from the NBA_message_list dictionary. The function prioritizes "enrollment" messages if there are new enrollments, and then considers other factors like "nocall_48hrs" and "on-enrollment." If the HCP is a "Non-Writer," it checks FLAG_EDA to determine whether to use "initial" or "pre-enrollment" messages. Otherwise, it defaults to "post-enrollment" messages.

Code snippet:

```
NBA_message_list = {
  "initial": ["DSA", "Guidelines"],
  "pre-enrollment": [
    "Arikayce patient profile",
    "Arikayce indication",
    "Arikayce efficacy and safety",
    "6-month urgency",
    "Arikayce MOD",
  ],
  "on-enrollment": ["Arikayce access"],
  "post-enrollment": [
    "Arikares",
    "Arikayce patient counseling",
    "Arikayce AE management",
  ],
  "enrollment": ["New Enrollment event", "Arikayce AE management"],
  "nocall_48hrs": ["No Call on 48hrs of Enrollment event"],
}


def message_generation(x):
  message = []
  if x["NO_OF_EF_ENROLLED"] > 0:
    message = NBA_message_list["enrollment"].copy()

    if x["NO_OF_NO_HCP_CALL"] > 0:
      message.extend(NBA_message_list["nocall_48hrs"])
    if x["HCP_WRITER_STATUS"] == "Non-Writer":
      message.extend(NBA_message_list["on-enrollment"])
  elif x["HCP_WRITER_STATUS"] == "Non-Writer":
    if x["FLAG_EDA"] == 0:
      message = NBA_message_list["initial"].copy()
    else:
      message = NBA_message_list["pre-enrollment"].copy()
  else:
    message = NBA_message_list["post-enrollment"].copy()
  return message
```

3.2.2.5 Updating Text for Insights, Reasoning, and Messages

This section outlines how to update the text for insights, reasoning, and messages within this Python file: `src/priority_models/hcp_priority_reasoning_utils.py`.

Understanding the Code

The code defines several functions:

- **`insight_generation()`:** Generates insights on HCPs based on their Arikayce usage data and other factors.
- **`reasoning_generation()`:** Provides reasoning behind the priority score assigned to an HCP for Arikayce usage and other factors.
- **`message_generation()`:** Generates relevant messages for the TS based on the HCP's stage in the Arikayce adoption process.
- **`no_val_insight_generation()`:** Similar to `insight_generation()` but generates insights without mentioning specific values.

- **`no_val_reasoning_generation()`:** Similar to `reasoning_generation()` but provides reasoning without mentioning specific values.

Updating the Text

To update the text for insights, reasoning, and messages, you'll need to modify the strings within these functions. Here's a breakdown:

1. Insights

- **`insight_generation()` and `no_val_insight_generation()`:**
  - Locate the `insight += ...` lines within these functions.
  - Modify the f-strings (e.g., `f"There were {int(x['NO_OF_EF_ENROLLED'])} enrollment in last {enrollment_interval} days.\n"`) to change the text. You can rephrase sentences, add or remove information, and adjust the formatting as needed.
  - Ensure that the placeholders (`{int(x['NO_OF_EF_ENROLLED'])}`, `{enrollment_interval}`, etc.) remain correctly integrated within the f-strings.

2. Reasoning

- **`reasoning_generation()` and `no_val_reasoning_generation()`:**
  - Similar to the insights, locate the `reasoning += ...` lines.
  - Modify the f-strings to update the reasoning text.

3. Messages

- **`message_generation()`:**
  - This function uses a `message_stage_mapper` dictionary, which maps HCP stages to lists of pre-approved messages.
  - To update the messages, you'll need to modify the contents of this dictionary, which can be found in the `src/priority_models/hcp_ranking_data_creation_config.json` file, inside the `messages_stage_list_mapper` key.

Example

Let's say you want to change the insight generated when an HCP has new enrollments but no calls within 48 hours. Currently, the text is:

```
insight += f"There were {int(x['NO_OF_EF_ENROLLED'])} enrollment and
{int(x['NO_OF_NO_HCP_CALL'])} occurence where the HCP was not contacted within 48
hours of the enrollment in last {enrollment_interval} days.\n"
```

You could update this to:

```
insight += f"The HCP had {int(x['NO_OF_EF_ENROLLED'])} new enrollment{'s' if
int(x['NO_OF_EF_ENROLLED']) > 1 else ''} in the last {enrollment_interval} days, but
wasn't contacted within 48 hours for {int(x['NO_OF_NO_HCP_CALL'])} of them.\n"
```

This revised text provides the same information but with some rephrasing.

3.3 The Generative Model

While the solution currently uses a rules-based approach for generating natural language outputs, we experimented with a generative model as well. Specifically, we used the Gemini large language model (LLM) to contextualize and summarize HCP data and the outputs from the models above. This approach was deprioritized because of the highly templatized nature of the text, which could easily be generated using rules and conditions (rather than using a large language model). Should Insmed decide to revisit this approach because of evolving needs, this section describes how to implement the LLM-based text generation and this section talks about the prompts used for the same.

3.3.1 Turning on the LLM-based Approach

To use the LLM-based approach:

1. Find the `use_llm` toggle in `src/priority_models/hcp_ranking_data_creation.py`, and set the value to `True`.
2. This will route the workflow to the LLM summary generation, largely contained in this file: `src/priority_models/llm_utils/llm_hcp_summary.py`
3. Note that the LLM approach creates new columns - these will need to be added to the Snowflake tables:
    a. **For Reasoning:**
        i. `llm_reasoning_response`
        ii. `llm_reasoning_error`
    b. **For Insights:**
        i. `llm_insights_response`
        ii. `llm_insights_error`
    c. **For Messages:**
        i. `llm_messages_response`
        ii. `Llm_messages_error`

The LLM prompts for each of the above are stored here in the following locations:

- **Reasoning:** `src/priority_models/prompts/reasoning.txt`

- **Insights:** `src/priority_models/prompts/insights.txt`
- **Messages:** `src/priority_models/prompts/messages.txt`

3.3.2 Model Setup

In our experimentation, we used the following model, parameters, and safety settings:

Model	gemini-1.5-flash-002
Temperature	0.5
Top P	0.9
Top K	30
Max Output Tokens	2000
HARM_CATEGORY_HARASSMENT	BLOCK_ONLY_HIGH
HARM_CATEGORY_HATE_SPEECH	BLOCK_ONLY_HIGH
HARM_CATEGORY_SEXUALLY_EXPLICIT	BLOCK_LOW_AND_ABOVE
HARM_CATEGORY_DANGEROUS_CONTENT	BLOCK_ONLY_HIGH

In terms of code, this can be set up using the following:

```
from vertexai.generative_models import (
   GenerativeModel,
   GenerationConfig,
   HarmBlockThreshold,
   HarmCategory,
)

# Configure generation parameters.
generation_config = GenerationConfig(
   temperature=0.5,
   top_p=0.9,
   top_k=30,
   max_output_tokens=2000,
)

# Configure safety settings.
safety_settings = {
   gemini.HarmCategory.HARM_CATEGORY_HARASSMENT: gemini.HarmBlockThreshold.BLOCK_ONLY_HIGH,
  gemini.HarmCategory.HARM_CATEGORY_HATE_SPEECH: gemini.HarmBlockThreshold.BLOCK_ONLY_HIGH,
  gemini.HarmCategory.HARM_CATEGORY_SEXUALLY_EXPLICIT:
gemini.HarmBlockThreshold.BLOCK_LOW_AND_ABOVE,
  gemini.HarmCategory.HARM_CATEGORY_DANGEROUS_CONTENT:
gemini.HarmBlockThreshold.BLOCK_ONLY_HIGH,
}

model = GenerativeModel(
  model_name="gemini-1.5-flash-002",
  generation_config=generation_config,
  safety_settings=self.safety_settings,
)
```

3.3.3 Generative Insights Output

The prompt used for experimenting with generating **Insights** is in Appendix 9.3. The following table shows illustrative examples (and actual predictions from the LLM) using this prompt.

NPIID	Sample Insights output
1275700478	This HCP is a non-writer, and has a 96% probability of a new enrollment. There was 1 enrollment last month and there was 1 occurrence where the HCP was not contacted within 48 hours of the enrollment. The HCP has 1 patient on antibiotics (Azithromycin or Clarithromycin :1 and ETHAMBUTOL: 1 and RIFAMIN or RIFABUTIN:1). There were 5 MAC positive results out of 104 AFB tests ordered in the last 6 months by the HCP.
1710306832	This HCP is a lapsed writer, and has a 99% probability of a new enrollment. The HCP has 2 patients on antibiotics (Azithromycin or Clarithromycin :4 and ETHAMBUTOL: 6 and RIFAMIN or RIFABUTIN:2). There were 24 MAC positive results out of a total 72 AFB tests ordered in the last 6 months by the HCP.
1598794026	This HCP is a non-writer, and has a 98% probability of a new enrollment. The HCP has 11 patients on antibiotics (Azithromycin or Clarithromycin :11 and ETHAMBUTOL: 8 and RIFAMIN or RIFABUTIN:3). There were 4 MAC positive results out of total 20 AFB tests ordered in the last 6 months by the HCP.
1831517622	This HCP is a non-writer, and has a 49% probability of a new enrollment. The HCP has 1 patient on antibiotics (Azithromycin or Clarithromycin :2 and ETHAMBUTOL: 2 and RIFAMIN or RIFABUTIN:1). There were no AFB tests ordered in the last 6 months.

3.3.4 Generative Reasoning Output

The prompt used for experimenting with generating **Reasoning** is in Appendix 9.4. The following table shows illustrative examples (and actual predictions from the LLM) using this prompt.

NPIID	Sample Reasoning output
1275700478	The HCP has a high priority, because it had a recent enrollment event and has not been visited recently.
1710306832	The HCP has a high priority, because it has a high probability of enrollment.
1598794026	The HCP has a high priority, because it has a high probability of enrollment.
1831517622	The HCP has a medium priority, because it has a medium probability of an enrollment and is an existing writer.

3.3.5 Generative Messages Output

The prompt used for experimenting with generating **Messages** is in Appendix 9.5. The following table shows illustrative examples (and actual predictions from the LLM) using this prompt.

NPIID	Sample Messages output
1275700478	New Enrollment event, No Call on 48hrs of Enrollment event
1710306832	Arikayce indication, Arikayce efficacy and safety
1598794026	Arikayce indication, Arikayce efficacy and safety
1831517622	Arikayce patient profile, Arikayce indication

4. MLOps Design and Architecture

This section dives into the technical design and architecture for MLOps (machine learning. Specifically, it covers the logical components and their interactions, as well as the physical infrastructure. Finally, we'll explore the features and signals considered for training the Accurate HCP Targeting machine learning model, and delve into the Improved Patient Journey Model model and Personalized Recommendations model and Large Language Model (LLM) architecture that forms the core of the system's intelligence.

The MLOps is applicable specifically to model 1. Models 2 and 3 have dedicated operational pipelines which run as separate stand-alone component(s) separate from the model 1 pipeline. In the case of model 1, the MLOps pipeline comprises Vertex AI pipelines powered by Kubeflow. The operational pipelines for models 2 and 3 are executed as scheduled jobs (using Cloud Run and Cloud Scheduler).

4.1 Key Principles

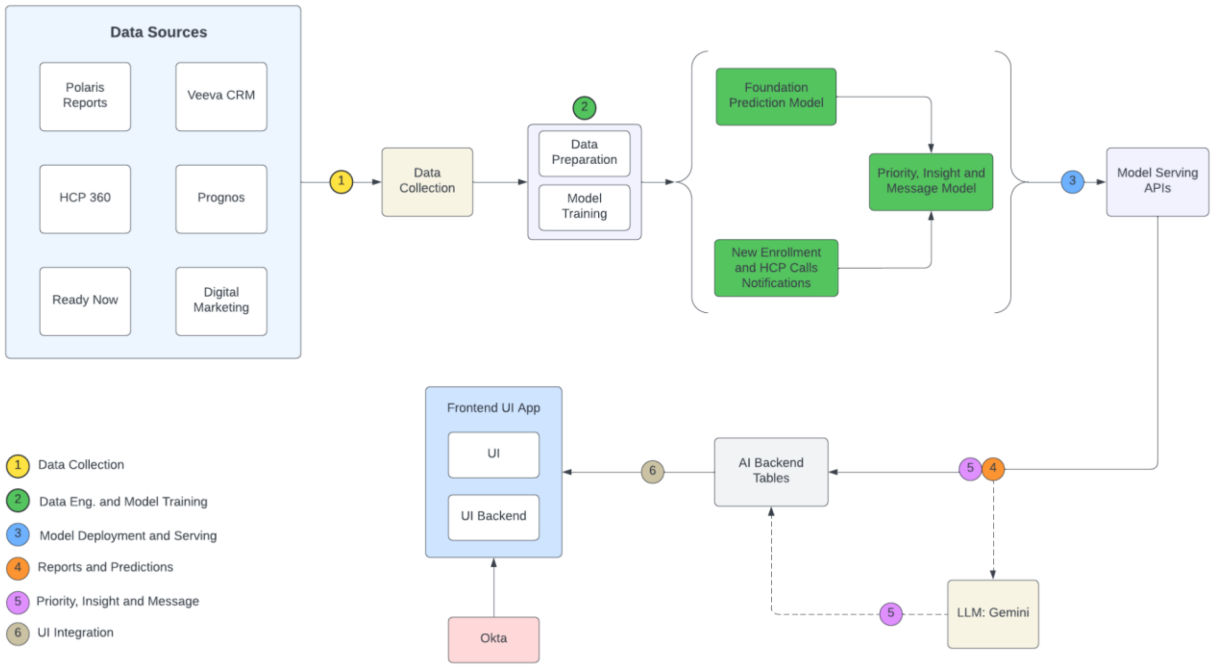
The operational set up is guided by 3 principles:

1. **Modular Components:** Break down the machine learning workflow into independent Vertex AI Pipeline components (using Kubeflow Pipelines) for each stage: data collection, preprocessing, model training, evaluation, etc. This enhances modularity and maintainability.

2. **Scheduled Model Training and Inference:** Use Vertex AI Scheduler to automatically trigger the ML pipeline on a recurring monthly basis. This ensures your models are consistently retrained with the latest data.

4.2 Logical Architecture

The logic architecture diagram represents the operational system.



Picture 4.1 Logic Architecture Diagram

4.2.1 Logical Structures

We use lightweight Kubeflow components to run the machine learning pipelines (specifically for model 1 for TS EF Predictions). These components are just-in-time initiated containers that execute our predefined logic. We orchestrate these components using Vertex AI Pipelines.

The INSMED GitHub repository houses the pipeline logic. Using Cloud Build, developers can push new and updated versions of this logic, which will be automatically built into container images stored in Google Container Registry. Eventually, the pipeline(s) will be submitted for one execution as well, optionally, in the non-production environments.

Below is the overall design of the Kubeflow components that form the standard ML pipeline for Model 1.

1. **Data Collection Operation:**
    a. Connects to Snowflake database in the select model operation
    b. Collect all the required data, and save it to a GCS bucket (centralized per model)
    c. Save the entire training dataset into Snowflake in this table:
        i. **AI_TS_EF_PREDICTION_TRAINING_ALL_DATA**
2. **Data Preparation Operation:**

   a. Load the saved data from the GCS bucket (from Data Collection collect operation)
   b. Prepare and clean data (e.g., remove nulls, feature engineering)
   c. Save prepared data into the GCS bucket.
   d. Later in the model training stage, the model training data will also be staged into a particular table in Snowflake.
3. **Model Training Operation:**
   a. Take processed CSV from GCS bucket
   b. Run the final model training (using the agreed Logistic Regression model)
   c. Save trained model in GCS bucket & Vertex AI Model Registry
      i. Name of the model and its versioning will be self explanatory
      ii. Description of the model to contain the date/time when the model was trained
   d. Save all training features in Snowflake in table:
      i. **AI_TS_EF_PREDICTION_TRAINING_FEATURES**
   e. Save model details in GCS/Snowflake (optional) (note: by model details, we refer to coefficients and metrics evaluated on test data during training process) in table: **AI_TS_EF_MODEL_METRICS**
      i. This table containing the metrics of each model training iteration will be used by the Batch prediction component to pick the best model as of any of its execution (which has the best Precision value) and use this best-as-of-now model to take the predictions.
   f. While the model is trained, the features are scaled for batch prediction(s).
      i. The model training step also additionally exports the StandardScaler object as an artifact with the model.
      ii. This information is then used in the batch prediction utility/application, while fetching the best model, as explained in the next section.
4. **Model Batch Prediction Operation:**
   a. Read delta/specific data from Snowflake tables (almost exactly similar as the data collection operation, but with limited date range!)
      i. During data preparation operation, a separate dataset will be saved in GCS for the applicable month for which we require batch predictions.
   b. Prepare data using the Data preprocessing component
   c. Scale the data similar to the Data preprocessing component
      i. To do so, read the previously exported StandardScaler object (during model training step) from a locally accessible storage or GCS Directory.
   d. Run batch inferences using Vertex AI Batch Predictions on trained model
   e. Save Inference inputs and outputs to Snowflake tables:
      i. **AI_TS_EF_PREDICTION_INFERENCE_INPUTS**
      ii. **AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS**

f. Delete any locally created artifacts or GCS buckets during this process.

Note, that all the run operations for all the models are put into the **AI_TS_RUN_DETAILS** table in the target Snowflake database/schema.

4.2.2 Model Deployment and Serving

We use monthly (or scheduled) batch predictions for model 1 (as opposed to online predictions), given TS needs for the solution. Below describes how this will work:

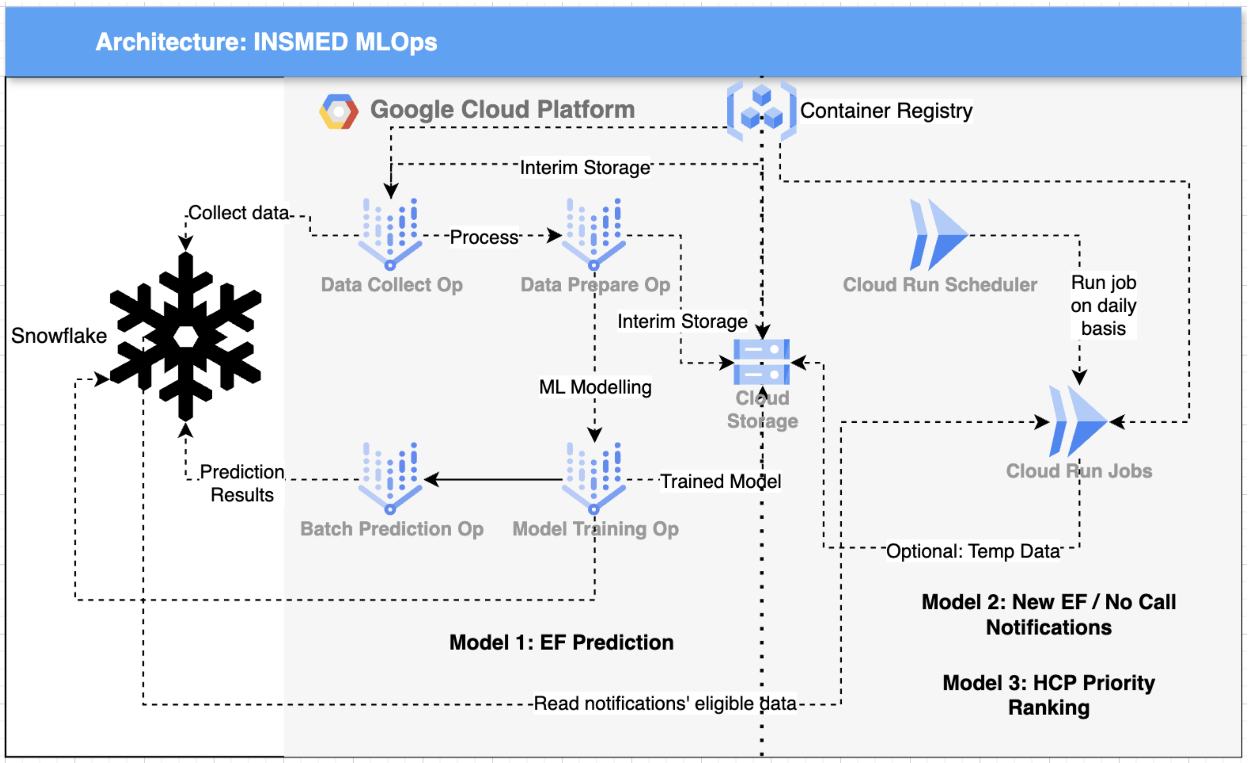
1. Whenever a new version of the model is trained, we perform batch predictions using Vertex AI Batch Prediction.
    a. The trained models are saved (as part of the training component, in the MLOps pipelines) in the Vertex AI Model Registry.
2. The batch inference results are stored in Snowflake database in the following 2 tables:
    a. **AI_TS_EF_PREDICTION_INFERENCE_INPUTS**: This stores the training dataset features used for inference from the latest model.
    b. **AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS**: This stores the batch prediction results into the database for future/downstream references and integrations as applicable.
3. As of the current version of the solution, there is a date and RUN_ID based data retention planned for in the design. In later stages, it is optionally recommended that we may have *_HIST like table(s) for each of the aforementioned tables, so that anything older than last iteration of the model training or batch predictions can be stored in the *_HIST table(s) in future while the latest data ONLY is available in the main tables.
    a. The main tables (mentioned in this and the last section of this document) should be “JOIN”-able using the RUN_ID column to get the entire picture of any particular training and inference execution.

Note, that the batch prediction component is (or will be) designed to execute as a separate component. This is to enable or ensure that the batch prediction/inference can run on a different schedule than the training pipeline. The training pipeline can then be scheduled to run every month using Vertex AI Pipeline Scheduler, while the batch prediction pipeline, designed to run as a Cloud Run job, can be run bi-weekly or other, using Cloud Scheduler.

4.3 System Architecture

We use Kubeflow Pipelines to orchestrate model 1's EF prediction pipeline, while model 2 and 3 rely on a simpler job scheduler. Both pipelines utilize container images stored in Container

Registry for their respective components. Note, that model 2 and 3 are synchronized to run one after the other, wherein Model 2 runs first, followed by Model 3.



4.4 CI/CD and Cloud Build Components

This project uses Google Cloud Build for continuous integration and continuous delivery (CI/CD). Cloud Build automatically builds and pushes Docker images whenever changes are made to the code repository.

Cloud Build Triggers

The following Cloud Build triggers are configured to respond to code changes in specific branches:

- **dev-ai-be-batch-prediction:** This trigger is activated when code changes are pushed to the `dev` branch.
  - **Included files:**
    - src/ts_ef_predictions/vertex_ai/batch_prediction/**
  - **Excluded files:**
    - src/priority_models/**

    - src/ts_ef_predictions/vertex_ai/data_collect/**
    - src/ts_ef_predictions/vertex_ai/pipeline/**
  - **Cloud Build YAML file:** src/ts_ef_predictions/cloud_build/cloud_build-batch-prediction.yaml
- **dev-ai-be-data-collection:** This trigger is activated when code changes are pushed to the `dev` branch.
  - **Included files:**
    - src/ts_ef_predictions/vertex_ai/data_collect/**
  - **Excluded files:**
    - src/priority_models/**
    - src/ts_ef_predictions/vertex_ai/batch_prediction/**
    - src/ts_ef_predictions/vertex_ai/pipeline/**
  - **Cloud Build YAML file:** src/ts_ef_predictions/cloud_build/cloud_build-data-collection.yaml
- **dev-ai-be-prioritization:** This trigger is activated when code changes are pushed to the `dev` branch.
  - **Included files:**
    - src/priority_models/**
  - **Excluded files:**
    - src/ts_ef_predictions/**
  - **Cloud Build YAML file:** src/priority_models/cloud_build/cloud_build.yaml
- **dev-ai-be-training-pipeline:** This trigger is activated when code changes are pushed to the `dev` branch.
  - **Included files:**
    - src/ts_ef_predictions/vertex_ai/pipeline/**
  - **Excluded files:**
    - src/ts_ef_predictions/vertex_ai/data_collect/**
    - src/ts_ef_predictions/vertex_ai/batch_prediction/**
    - src/priority_models/**
  - **Cloud Build YAML file:** src/ts_ef_predictions/cloud_build/cloud_build-training-pipeline.yaml

Cloud Build YAML File

Each trigger utilizes a Cloud Build YAML file to define the build steps. Here's an example of a typical `cloudbuild.yaml` file:

```
# Copyright 2024 Google LLC. This software is provided as-is, without warranty
# or representation for any use or purpose. Your use of it is subject to your
# agreement with Google

#Subsitute environment variables for training.
substitutions:
  _MLOPS_PIPELINE_VERSION: '${BRANCH_NAME}-${SHORT_SHA}'
  _BATCH_PRED_CONTAINER_IMAGE : 'us-east4-docker.pkg.dev/therapeutic-precall-plan-
dev/dev-ai-be-batch-prediction/ts-ef-pred-batch-pred:latest'

steps:
# Step 0 : Build training  container image.
- name: 'gcr.io/cloud-builders/docker'
  args: ['build', './src/ts_ef_predictions/vertex_ai/batch_prediction/','-t',
        '${_BATCH_PRED_CONTAINER_IMAGE}_${_MLOPS_PIPELINE_VERSION}',
        '-t', '${_BATCH_PRED_CONTAINER_IMAGE}']

# Step 1 : Push training container image.
- name: 'gcr.io/cloud-builders/docker'
  args: ['push', '${_BATCH_PRED_CONTAINER_IMAGE}']


timeout: 21600s

options:
  logging: CLOUD_LOGGING_ONLY
```

Explanation:

- **`substitutions`:** Defines environment variables that can be used in the build steps. In this case, it sets the pipeline version and the container image name.
- **`steps`:** This section defines the sequence of steps to be executed in the build.
    - **Step 0:** Builds the Docker image using the provided Dockerfile and tags it with the pipeline version and "latest."
    - **Step 1:** Pushes the built Docker image to the specified container registry.
- **`timeout`:** Sets the maximum time allowed for the build process to complete.
- **`options`:** Configures additional options for the build, such as logging.

Benefits of using Cloud Build:

- **Automated Builds:** Eliminates manual intervention and ensures consistent builds.
- **Scalability:** Cloud Build can handle a large number of builds concurrently.
- **Integration with Google Cloud:** Seamlessly integrates with other Google Cloud services.
- **Detailed Logs:** Provides comprehensive logs for debugging and monitoring.

By using Cloud Build and its triggers, this project achieves a streamlined CI/CD pipeline, enabling rapid and reliable code deployment.

4.5 Model Metric Modification

Our current pipeline picks the best performing model based on Precision of trained model under a threshold. This can be changed, for example by using F1 score to balance between Precision and Recall, when picking the best model.

5. Backend Interface for UI

The `**AI_TS_HCP_PRIORITY_RANKING**` table is used for downstream consumption by the UI. The schema and SQL to create the **AI_TS_HCP_PRIORITY_RANKING** table is

```
create or replace AI_TS_HCP_PRIORITY_RANKING (
      NPIID NUMBER(38,0),
      HCPNAME VARCHAR(16777216),
      PROBABILITY_EF NUMBER(38,17),
      RANKING NUMBER(38,0),
      READY_NOW_STATUS NUMBER(38,0),
      PRIORITY VARCHAR(16777216),
      PRIORITY_SCORE NUMBER(38,17),
      INSIGHT VARCHAR(16777216),
      REASONING VARCHAR(16777216),
      MESSAGES ARRAY,
      UPDATE_TIME TIMESTAMP_NTZ(9)
);
```

The table contains details for the UI sections:

Column	**Description**
NPIID	Unique NPIID for each HCP
HCPNAME	Name of HCP (FirstName LastName)
PROBABILITY_EF	HCP's probability of a future EF
RANKING	Ranking detail based on priority score
READY_NOW_STATUS	1 if HCP is in ReadyNow; 0 otherwise
PRIORITY	The category for the Prioritization (High,Medium and Low)

PRIORITY_SCORE	Priority score for HCP (model 3 output)
REASONING	Contributing factors for the priority score
INSIGHT	HCP profile and relevant data attributes
MESSAGES	List of messages for the TS as talking points
UPDATE_TIME	UTC no timezone time when the record was created/appended.

5.1 Key Principals

- **Real-time Data Updates**: The system should enable continuous data ingestion and model retraining to adapt to evolving trends and HCP behaviors.
- **Transparency and Explainability**: Model outputs should be transparent and understandable by TSs to build trust and facilitate adoption.
- **User Feedback Loop**: The system should capture user feedback on the effectiveness of recommendations to refine models and improve accuracy.

5.2 AI Backend Through Snowflake Tables

These tables serve as an interface between the underlying models:

- **AI_TS_EF_PREDICTION_TRAINING_FEATURES**
- **AI_TS_EF_PREDICTION_INFERENCE_INPUTS**
- **AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS**
- **AI_TS_NOTIFICATIONS**

These tables serve as interface between the AI models and the UI:

- **AI_TS_HCP_PRIORITY_RANKING**
- **AI_TS_HCP_PRIORITY_RANKING_STAGING**

The logic for generating these tables are discussed in the sections above. In addition, for the UI, we add these FLAGs:

- **ReadyNow:** To indicate if the HCP comes from ReadyNow rather than from models.
- **New EF:** To indicate whether the HCP has submitted a new EF during the current inference month. This information is used in the UI to identify HCPs who have already generated an EF and may no longer be candidates for potential EF generation.

- **Inference month:** To distinguish between HCPs identified by model 1's main inference process and those added by model 2's daily updates. This clarifies the origin of each HCP despite combining all model outputs into a single table.

6. Path to Production

This section describes the recommended steps for taking the models and overall solution into a production state. It outlines the steps involved in infrastructure setup, deploying and operationalizing the AI models, encompassing pipeline execution, and data management.

6.1 GCP Project Setup and Vertex AI Enablement

6.1.1 Roles and Permissions

The following permissions are required for the Service Account to run the end-to-end flow:

- Artifact Registry Administrator
- Artifact Registry Reader
- Artifact Registry Repository Administrator
- Artifact Registry Writer
- BigQuery Admin
- Cloud Run Developer
- Cloud Scheduler Job Runner
- Cloud Scheduler Viewer
- Container Registry Service Agent
- Logs Writer
- Secret Manager Secret Accessor
- Service Account Token Creator
- Service Account User
- Storage Admin
- Storage Object Creator
- Storage Object Viewer
- Vertex AI Service Agent
- Vertex AI User

6.1.2 Service Account

Instead of assigning permissions directly to individual users, Google PSO recommends creating a custom role encompassing all the necessary permissions outlined above. This custom role can then be assigned to a dedicated Service Account. This approach simplifies user management, making it easier to maintain and update permissions. Detailed instructions on how to create this custom role and Service Account, using scripts or "gcloud" commands, will be provided in the README file within the Insmed GitHub repository.

6.1.3 Enabling APIs

The following APIs needs to be enabled for using all the service in the end-to-end flow:

- Vertex AI API
- Artifact Registry API
- BigQuery API
- BigQuery Connection API
- BigQuery Data Policy API
- BigQuery Data Transfer API
- BigQuery Reservation API
- Cloud Scheduler API
- Container Registry API
- IAM Service Account Credentials API
- Cloud Logging API
- Cloud Run Admin API
- Secret Manager API
- Cloud Storage API

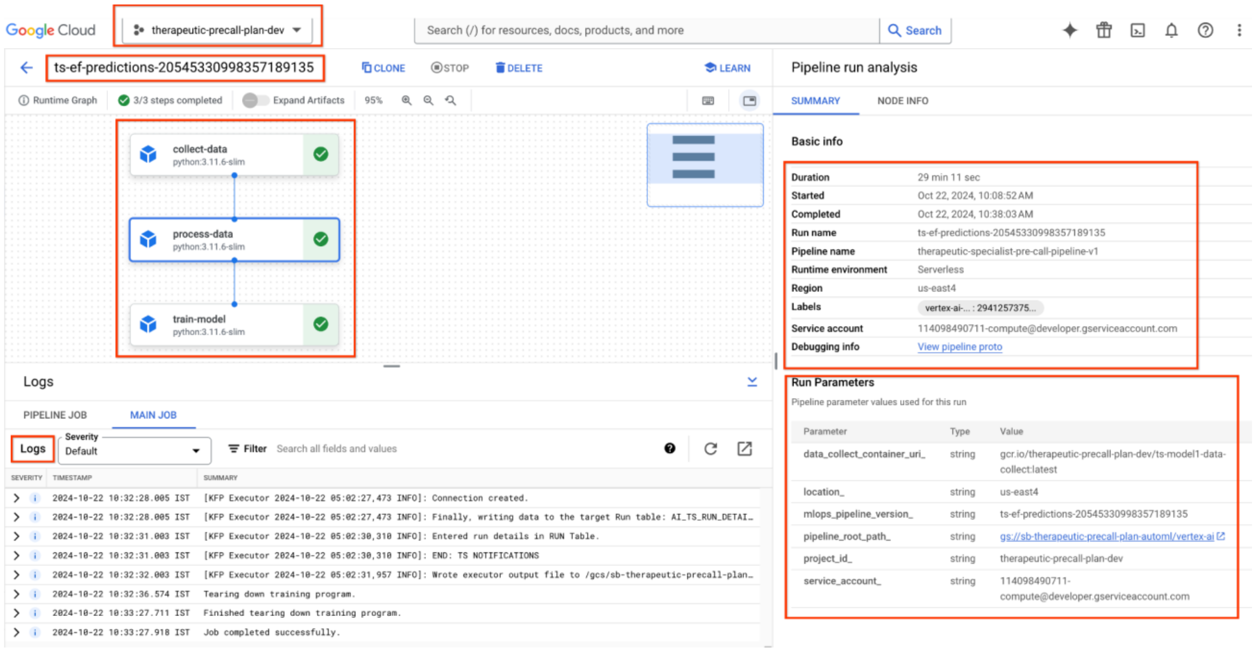
6.2 Operationalizing Model Pipelines

- *Describe how the pipelines work, and the components with their dependencies*
    - *Training pipeline for model 1*
    - *Inference pipeline for models 1, 2 and 3*
    - *Data refresh dependency*
    - *Any other steps, e.g. use Cloud Build to package the code into container images for pipeline components*
- *How to use the pipeline (Operation Guide)*
    - *How to schedule a pipeline run*
    - *How to run a pipeline manually*
    - *How to check the pipeline run status and how to check the log info, etc.*

6.2.1 Pipeline Details

Model 1 Training Pipeline

Model 1 (EF predictions) pipeline runs can be viewed in the Vertex AI Pipelines section of your GCP console. All pipelines, jobs, and services are primarily deployed in the us-east4 (North Virginia) region within the designated GCP project. A typical run will look like this:



Basic info

Duration	29 min 11 sec
Started	Oct 22, 2024, 10:08:52 AM
Completed	Oct 22, 2024, 10:38:03 AM
Run name	ts-ef-predictions-20545330998357189135
Pipeline name	therapeutic-specialist-pre-call-pipeline-v1
Runtime environment	Serverless
Region	us-east4
Labels	vertex-ai-... : 2941257375...
Service account	114098490711-compute@developer.gserviceaccount.com
Debugging info	View pipeline proto

Run Parameters

Pipeline parameter values used for this run

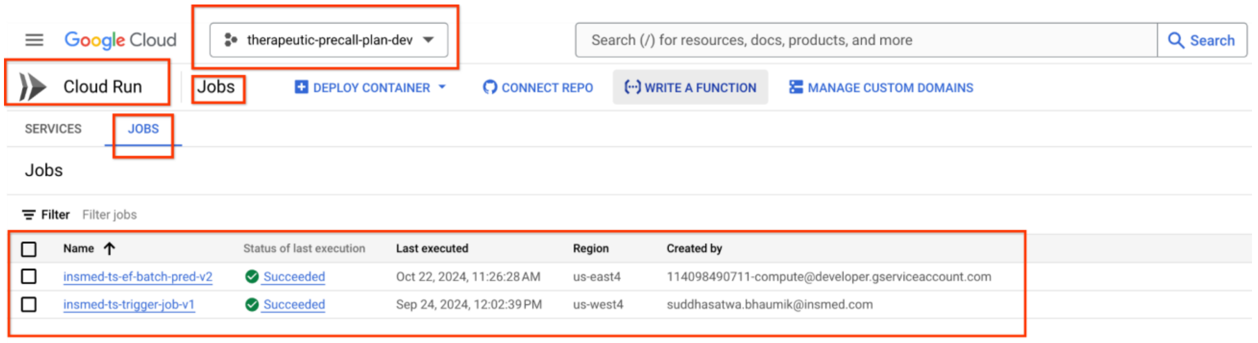
Parameter	Type	Value
data_collect_container_uri_	string	gcr.io/therapeutic-precall-plan-dev/ts-model1-data-collect:latest
location_	string	us-east4
mlops_pipeline_version_	string	ts-ef-predictions-20545330998357189135
pipeline_root_path_	string	gs://sb-therapeutic-precall-plan-automl/vertex-ai
project_id_	string	therapeutic-precall-plan-dev
service_account_	string	114098490711-compute@developer.gserviceaccount.com

```
SEVERITY TIMESTAMP                    SUMMARY
   i     2024-10-22 10:32:28.005 IST  [KFP Executor 2024-10-22 05:02:27,473 INFO]: Connection created.
   i     2024-10-22 10:32:28.005 IST  [KFP Executor 2024-10-22 05:02:27,473 INFO]: Finally, writing data to the target Run table: AI_TS_RUN_DETAI…
   i     2024-10-22 10:32:31.003 IST  [KFP Executor 2024-10-22 05:02:30,310 INFO]: Entered run details in RUN Table.
   i     2024-10-22 10:32:31.003 IST  [KFP Executor 2024-10-22 05:02:30,310 INFO]: END: TS NOTIFICATIONS
   i     2024-10-22 10:32:32.003 IST  [KFP Executor 2024-10-22 05:02:31,957 INFO]: Wrote executor output file to /gcs/sb-therapeutic-precall-plan…
   i     2024-10-22 10:32:36.574 IST  Tearing down training program.
   i     2024-10-22 10:33:27.711 IST  Finished tearing down training program.
   i     2024-10-22 10:33:27.918 IST  Job completed successfully.
```

Note about new HCP 360 logic: For future model training, we suggest having a manual verification because the training set will be based on new HCP 360 logic.

Models 2 and 3 Execution Pipeline/Schedule

The Cloud Run pipeline will be scheduled and run both the Model 2 and Model 3 everyday at a specific point in time from the scheduler. We can find more details here. Both these models are designed to run as a Cloud Run Batch job, which will look similar to the following screenshot in the applicable target GCP Project:



Name	Status of last execution	Last executed	Region	Created by
insmed-ts-ef-batch-pred-v2	✓ Succeeded	Oct 22, 2024, 11:26:28 AM	us-east4	114098490711-compute@developer.gserviceaccount.com
insmed-ts-trigger-job-v1	✓ Succeeded	Sep 24, 2024, 12:02:39 PM	us-west4	suddhasatwa.bhaumik@insmed.com

6.2.2 Pipeline Execution

This section outlines how to schedule and manually execute pipelines, as well as how to monitor their operation.

Scheduling

- **Model 1:** This pipeline is scheduled to run on a monthly (inference) or quarterly (training) basis using Vertex AI Scheduler, providing flexibility in configuration. The pipeline itself leverages Vertex AI Pipelines built on Kubeflow.
  - For Model-1, the Batch prediction pipeline is realized as a separate component running on Cloud Run on Schedule or as and when we want it to execute.
- **Models 2 and 3 Operations Pipelines:** These pipelines are scheduled for daily execution using Cloud Scheduler.

Manual Execution

Both types of pipelines can be triggered manually for ad-hoc runs or testing purposes. This can be done programmatically using gcloud API calls or through the Google Cloud console.

Monitoring

- **Cloud Logging:** All services, including Vertex AI, Vertex AI Scheduler, and Cloud Run jobs for models 2 and 3, send operational logs to Cloud Storage.
- **Cloud Monitoring:** Cloud Monitoring provides built-in tools for monitoring and analyzing pipeline performance.
- **Log Sinks and BigQuery:** For advanced analytics and visualization, log sinks can be configured in Cloud Logging to route logs to BigQuery. This data can then be analyzed and visualized using tools like Looker.

6.2.3 Snowflake Production

- *How to switch to Snowflake production instance using a config file or info stored in Secret Manager (Snowflake Migration Guide)*

This section outlines the process of transitioning the AI solution from the UAT (User Acceptance Testing) environment to the production Snowflake instance.

Environment-Agnostic Design

The SQL queries within the application logic are designed to be environment-agnostic. This means that they do not hardcode database names, schemas, or any other environment-specific references. This allows for easy switching between environments without code modification.

Secret Management

Snowflake credentials for both UAT and production environments are securely stored in Cloud Secret Manager. The applications responsible for MLOps (model 1) and operational schedules (models 2 and 3) retrieve the appropriate credentials from Secret Manager to establish connections to the respective Snowflake databases.

Migration Steps

Due to the environment-agnostic design and centralized secret management, migrating the pipelines from UAT to production involves the following straightforward steps:

1. **Image Creation:** Build the required container images and store them in Google Container Registry within the target production project. This can be automated using Cloud Build jobs.
2. **Installation:** Install the pipeline components in the production environment. This can be facilitated using Terraform scripts to automate infrastructure provisioning.

This streamlined approach ensures a smooth transition to the production Snowflake instance without requiring code changes or manual configuration adjustments.

7. Code and Artifacts

7.1 Code repository

The code for the AI backend can be accessed at this Github repository.

There are three environments for the AI backend, corresponding to the following branches and resources:

BRANCH	GCP PROJECT HOSTED	DATABASES USED
`dev`	therapeutic-precall-plan-dev	`COMMERCIAL_AI` & `UAT_INSMED_CLAIMS`
`uat`	ts-test	`COMMERCIAL_AI_UAT` & `UAT_INSMED_CLAIMS`

`main`[4]	therapeutic-precall-plan-prod	`COMMERCIAL_AI_PROD` & `PROD_SHYFT_INSMED_CLAIMS`

7.2 GCP Resources

All GCP resources follow a common naming format:

- For the `dev` environment, the workflow resources are prefixed with `dev-ai-be` (the `ai-be` is in reference to the AI backend)
- For the `uat` environment, the resources are prefixed with `uat-ai-be`

7.2.1 DEV Environment

The relevant GCP resources for the `dev` environment are listed in the table below.

RESOURCE TYPE	RESOURCE NAME	NOTES
Cloud Build triggers	dev-ai-be-batch-prediction	Trigger for repo code changes to batch prediction
	dev-ai-be-data-collection	Trigger for repo code changes to data collection
	dev-ai-be-prioritization	Trigger for repo code changes to prioritization and ranking
	dev-ai-be-training-pipeline	Trigger for repo code changes to training pipeline
Artifact Registry	dev-ai-be-batch-prediction	Docker repo for batch prediction images
	dev-ai-be-data-collection	Docker repo for data collection images
	dev-ai-be-prioritization	Docker repo for prioritization and ranking images
	dev-ai-be-training-pipeline	Docker repo for training pipeline images
Cloud Run Jobs	dev-ai-be-batch-prediction	Job for running batch prediction

[4] As of this writing, the branch is currently `master`, but Insmed aims to use the `main` branch for parity with front-end and back-end repos for the UI

	dev-ai-be-prioritization	Job for running prioritization and ranking
	dev-ai-be-training-pipeline	Job for executing Vertex training pipeline
GCS Bucket	dev-ai-be-automl-artifacts	Bucket for MLOps artifacts

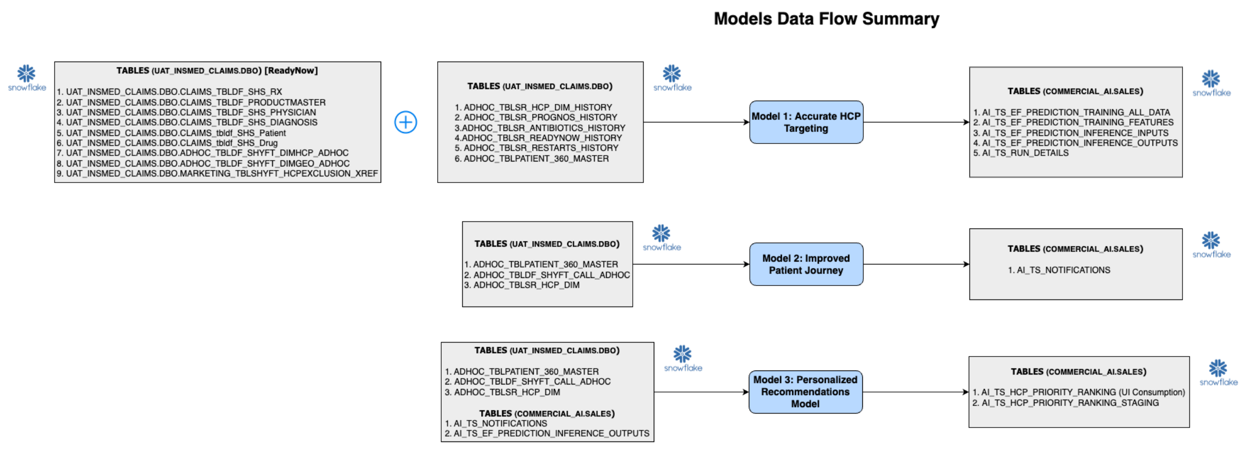
7.2.1 UAT environment

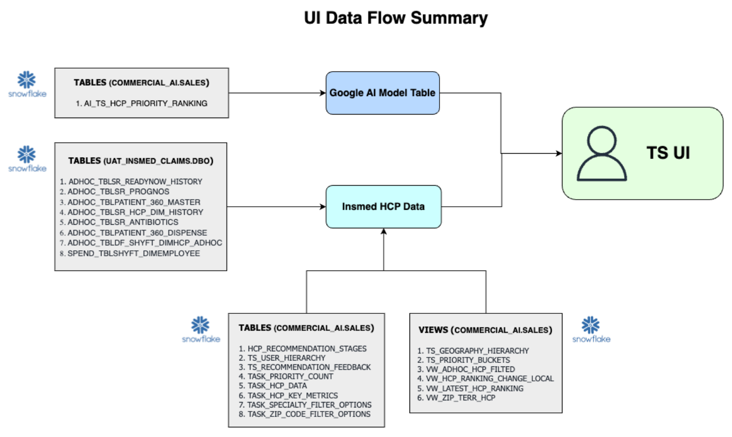
The relevant GCP resources for the `uat` environment are listed in the table below.

RESOURCE TYPE	RESOURCE NAME	NOTES
Cloud Build triggers	uat-ai-be-batch-prediction	Trigger for repo code changes to batch prediction
	uat-ai-be-data-collection	Trigger for repo code changes to data collection
	uat-ai-be-prioritization	Trigger for repo code changes to prioritization and ranking
	uat-ai-be-training-pipeline	Trigger for repo code changes to training pipeline
Artifact Registry	uat-ai-be-batch-prediction	Docker repo for batch prediction images
	uat-ai-be-data-collection	Docker repo for data collection images
	uat-ai-be-prioritization	Docker repo for prioritization and ranking images
	uat-ai-be-training-pipeline	Docker repo for training pipeline images
Cloud Run Jobs	ts-ef-pred-batch-pred	Job for running batch prediction
	priority-model	Job for running prioritization and ranking
	ef-prediction-training-pipeline	Job for executing Vertex training pipeline
GCS Bucket	uat-ai-be-automl-artifacts	Bucket for MLOps artifacts

8. Snowflake Database Flow

The below flow shows how the Snowflake database tables are used.


Models Data Flow Summary
snowflake
TABLES (UAT_INSMED_CLAIMS.DBO) [ReadyNow]
1. UAT_INSMED_CLAIMS.DBO.CLAIMS_TBLDF_SHS_RX
2. UAT_INSMED_CLAIMS.DBO.CLAIMS_TBLDF_PRODUCTMASTER
3. UAT_INSMED_CLAIMS.DBO.CLAIMS_TBLDF_SHS_PHYSICIAN
4. UAT_INSMED_CLAIMS.DBO.CLAIMS_TBLDF_SHS_DIAGNOSIS
5. UAT_INSMED_CLAIMS.DBO.CLAIMS_tbldf_SHS_Patient
6. UAT_INSMED_CLAIMS.DBO.CLAIMS_tbldf_SHS_Drug
7. UAT_INSMED_CLAIMS.DBO.ADHOC_TBLDF_SHYFT_DIMHCP_ADHOC
8. UAT_INSMED_CLAIMS.DBO.ADHOC_TBLDF_SHYFT_DIMGEO_ADHOC
9. UAT_INSMED_CLAIMS.DBO.MARKETING_TBLSHYFT_HCPEXCLUSION_XREF
TABLES (UAT_INSMED_CLAIMS.DBO)
1. ADHOC_TBLSR_HCP_DIM_HISTORY
2. ADHOC_TBLSR_PROGNOS_HISTORY
3.ADHOC_TBLSR_ANTIBIOTICS_HISTORY
4.ADHOC_TBLSR_READYNOW_HISTORY
5. ADHOC_TBLSR_RESTARTS_HISTORY
6. ADHOC_TBLPATIENT_360_MASTER
snowflake
Model 1: Accurate HCP Targeting
TABLES (COMMERCIAL_AI.SALES)
1. AI_TS_EF_PREDICTION_TRAINING_ALL_DATA
2. AI_TS_EF_PREDICTION_TRAINING_FEATURES
3. AI_TS_EF_PREDICTION_INFERENCE_INPUTS
4. AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS
5. AI_TS_RUN_DETAILS
snowflake
TABLES (UAT_INSMED_CLAIMS.DBO)
1. ADHOC_TBLPATIENT_360_MASTER
2. ADHOC_TBLDF_SHYFT_CALL_ADHOC
3. ADHOC_TBLSR_HCP_DIM
snowflake
Model 2: Improved Patient Journey
TABLES (COMMERCIAL_AI.SALES)
1. AI_TS_NOTIFICATIONS
snowflake
TABLES (UAT_INSMED_CLAIMS.DBO)
1. ADHOC_TBLPATIENT_360_MASTER
2. ADHOC_TBLDF_SHYFT_CALL_ADHOC
3. ADHOC_TBLSR_HCP_DIM
TABLES (COMMERCIAL_AI.SALES)
1. AI_TS_NOTIFICATIONS
2. AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS
snowflake
Model 3: Personalized Recommendations Model
TABLES (COMMERCIAL_AI.SALES)
1. AI_TS_HCP_PRIORITY_RANKING (UI Consumption)
2. AI_TS_HCP_PRIORITY_RANKING_STAGING
snowflake


UI Data Flow Summary
snowflake
TABLES (COMMERCIAL_AI.SALES)
1. AI_TS_HCP_PRIORITY_RANKING
Google AI Model Table
TS UI
snowflake
TABLES (UAT_INSMED_CLAIMS.DBO)
1. ADHOC_TBLSR_READYNOW_HISTORY
2. ADHOC_TBLSR_PROGNOS
3. ADHOC_TBLPATIENT_360_MASTER
4. ADHOC_TBLSR_HCP_DIM_HISTORY
5. ADHOC_TBLSR_ANTIBIOTICS
6. ADHOC_TBLPATIENT_360_DISPENSE
7. ADHOC_TBLDF_SHYFT_DIMHCP_ADHOC
8. SPEND_TBLSHYFT_DIMEMPLOYEE
Insmed HCP Data
snowflake
TABLES (COMMERCIAL_AI.SALES)
1. HCP_RECOMMENDATION_STAGES
2. TS_USER_HIERARCHY
3. TS_RECOMMENDATION_FEEDBACK
4. TASK_PRIORITY_COUNT
5. TASK_HCP_DATA
6. TASK_HCP_KEY_METRICS
7. TASK_SPECIALTY_FILTER_OPTIONS
8. TASK_ZIP_CODE_FILTER_OPTIONS
VIEWS (COMMERCIAL_AI.SALES)
1. TS_GEOGRAPHY_HIERARCHY
2. TS_PRIORITY_BUCKETS
3. VW_ADHOC_HCP_FILTED
4. VW_HCP_RANKING_CHANGE_LOCAL
5. VW_LATEST_HCP_RANKING
6. VW_ZIP_TERR_HCP
snowflake

Below is the list of columns used from the tables mentioned above

1. ADHOC_TBLSR_HCP_DIM_HISTORY
2. ADHOC_TBLSR_PROGNOS_HISTORY
3. ADHOC_TBLSR_ANTIBIOTICS_HISTORY
4. ADHOC_TBLSR_READYNOW_HISTORY
5. **ADHOC_TBLPATIENT_360_MASTER** - INSMED_PATIENT_ID, DATE_START_FORM_RECEIVED, EF_PHYSICIAN_NPI, PHYSICIAN_NPI, EF_PHYSICIAN_FIRST_NAME, EF_PHYSICIAN_LAST_NAME, EF_PHYSICIAN_FACILITY_NAME, EF_PHYSICIAN_ZIP_CODE, SP_PHARMACY_ID, EF_SALES_FIELD_REGION, EF_SALES_FIELD_TERRITORY, LATEST_SP_STATUS,

LATEST_SP_SUB_STATUS, LATEST_SP_STATUS_DATE, LATEST_CASE_MANAGER_STATUS, LATEST_CASE_MANAGER_SUBSTATUS, LATEST_CASE_MANAGER_STATUS_REASON, LATEST_CASE_MANAGER_STATUS_DATE, PRODUCT_DESCRIPTION
6. ADHOC_TBLSR_RESTARTS_HISTORY
7. **ADHOC_TBLDF_SHYFT_CALL_ADHOC** - VEEVA_CALL_ID, CALLDATE, CALLTYPE, TERRITORY, INSMED_HCP_ID, INSMED_CALL_TYPE,"INSMED_HCP_ID"
8. **ADHOC_TBLSR_HCP_DIM**- NPIID,INSMEDID.

9. Appendix

9.1 HCP Prioritization Implementation

The algorithm is implemented in the PrioritizationCalculator class within the prioritization_calculator.py file in [TODO] and will automatically calculate the priority value for each HCP. This value will be dynamically updated everyday, ensuring that the prioritization remains relevant and accurate.

To execute the prioritization calculator, you must:

- Import the PrioritizationCalculator object
- Instantiate the PrioritizationCalculator object with the file path to the config.json file
- Load in the dataset that contains the relevant input columns into a Pandas dataframe object
- Call the calculate_prioritization_score method of the PrioritizationCalculator object, passing in the Pandas dataframe.

An example of the code can be seen below:

```
# Import custom scripts
from src.prioritization_calculator import PrioritizationCalculator

# Instantiate calculator with location of config.json file
calculator = PrioritizationCalculator('config/config-max-model.json')

# Load dataset - the prioritization calculator expects a pandas DF
dataset = pd.read_csv('data/hcp_data.csv')

# Run prioritization calculator
dataset = calculator.calculate_prioritization_score(dataset)
```

The method calculate_prioritization_score is the entrypoint for starting the calculation, but the core formula is in the _calculate_overall_score method. It returns an overall score which is then appended to the dataframe as a new column.

Other methods used in the PrioritizationCalculator class are described in the table below.

METHOD NAME	DESCRIPTION
calculate_prioritization_score	Entrypoint for calculation orchestration
_create_new_features	Generates new features if needed for calculation
_calculate_category_scores	Calculates scores categories of features. This method can also be used to calculate scores for individual features, as long as the category in the config.json file specifies just an individual feature. The ability to use either individual features or categories of features provides additional flexibility and convenience. The categories property in the config file contains a list of categories, the corresponding features for each category, and the weight for the categories.
_calculate_overall_score	Uses prioritization score formula (in this case, MAX function over two expressions) to generate final score. Any changes to the prioritization formula can be made here.
_add_weight_summary	Helper method to generate a rules-based summary for each feature. Summary logic is defined in the config.json file.
_add_range_summary	Helper method to generate a rules-based summary for each feature where a range is involved (i.e., a value between two other values). Summary logic is defined in the config.json file.
visualize_evaluation_results	Helper method to graphically display how each data point falls with respect to other data points in terms of prioritization.

9.2 HCP Prioritization Config JSON file

The config.json file is a required component for the HCP prioritization process that provides flexibility and maintainability, and allows for easy experimentation with different parameters without having to change the processing codebase. By clearly defining the categories, factors, thresholds, and evaluation rules, it allows for easy customization and adaptation of the prioritization logic to meet evolving business needs. An example of this file is located at priortization_calculator_file.

The purpose of the config.json file is to:

- **Define Prioritization Categories:** Specify the different categories used to assess HCPs, such as their enrollment status, engagement level, and probability of enrollment.
- **Assign Weights to Categories:** Determine the relative importance of each category in the overall prioritization score.
- **Configure Individual Factors:** Define the specific factors within each category and how they contribute to the category score (e.g., weight-based, range-based, etc.).

The HCP prioritization script (PrioritizationCalculator class) expects this configuration JSON during instantiation, and uses its configurations to:

1. **Calculate Category Scores:** Calculate scores for each category based on the defined factors and their types.
2. **Calculate Overall Score:** Calculate the overall prioritization score by combining the category scores according to the defined weights and formula.
3. **Generate Summary Explanations:** Create summary explanations for each HCP in the influencing_factors column based on the summary_instructions configured for the factors.

Structure

The config.json file is structured as a JSON object with the following main sections:

CONFIG SECTION	DESCRIPTION
categories	This section defines the different categories used in the prioritization process. Each category is an object with the following properties: ● **factors:** A list of factor names that belong to this category. These factors must be defined in the factors section. ● **weight:** A numerical value representing the weight of this category in the overall prioritization score calculation.
factors	This section defines the individual factors used within each category. Each factor is an object with the following properties:

	<ul><li>**type:** The type of calculation to be used for this factor. Supported types include:<ul><li>WEIGHT: The factor value is multiplied by a weight.</li><li>RANGE-BASED VALUE: The factor value is assigned a value based on the range it falls into.</li><li>MIN-WEIGHT: The minimum of the factor value and a defined maximum value is multiplied by a weight.</li></ul></li><li>**weight:** (Optional) A numerical value representing the weight of this factor within its category (used for WEIGHT and MIN-WEIGHT types).</li><li>**ranges:** (Optional) A list of ranges used for RANGE-BASED VALUE type. Each range is an object with min, max, and value properties.</li><li>**max_value:** (Optional) The maximum value to be considered for MIN-WEIGHT type.</li><li>**summary_instructions:** (Optional) A list of objects defining how to generate summary explanations for this factor in the influencing_factors column. Each object has:<ul><li>condition_type: The type of condition to check (range, below, or above).</li><li>min, max, value: Parameters for the condition.</li><li>instruction: The summary instruction template to use if the condition is met.</li></ul></li></ul>
thresholds	This section defines the thresholds for categorizing HCPs into high, medium, and low priority groups based on their overall scores. It has the following properties:<ul><li>**high:** The threshold for the high priority group (e.g., 0.67 for the 67th percentile).</li><li>**low:** The threshold for the low priority group (e.g., 0.34 for the 34th percentile).</li></ul>
evaluation_rules	This section defines rules for evaluating the effectiveness of the prioritization logic. Each rule is an object with the following properties:

	● **description:** A human-readable description of the rule. ● **condition:** A condition expression to be evaluated using pandas eval(). ● **category:** The expected priority category (high, medium, or low) if the condition is met.

9.3 Generative Model (LLM) Insight Prompt

The Insight provides the details for the HCP based on the input data for usage of different antibiotics,test details and engagement details data.

We utilize below prompt for the generation of the output from the LLM

```
You are an AI assistant tasked with generating concise insights for Healthcare Professionals (HCPs).  Strictly adhere to the following instructions to create a 2-4 sentence summary for each HCP:

**Data Points:**

You will be provided with the following data for each HCP:

* `HCP_WRITER_STATUS`:  Indicates whether the HCP is an existing Arikayce writer (1),Lapsed writer(0.5) or nonwriter (0).
* `NO_OF_EF_ENROLLED`: The number of enrollments in the last month.
* `PROBABILTY_EF`: The probability of a new enrollment (if `NO_OF_EF_ENROLLED` is 0).
* `NO_OF_NO_HCP_CALL`: The number of "no-calls" within 48 hours of enrollments.
* `ETHAMBUTOL`, `SUM_AZI_CLAR`, `SUM_RIF_RIF`:  Data related to first-line antibiotic use.
* `AMIKACIN`, `SUM_MOX_LEV`: Data related to second-line antibiotic use.
* 'AFBMACTESTS`: Data related to the number of AFB and MAC tests prescribed in the last 6 months.
* `AFBTESTSMAC6MONTHS`: Data related to the number of AFB and MAC tested resulted  positive in the last 6 months.
* `TOTAL_ARIKAYCE_USER`: Data related to number of Arikayce patients enrolled till date.

* If number of enrollment(NO_OF_EF_ENROLLED) in last month is same as total arikayce user(TOTAL_ARIKAYCE_USER) and non zero then use the statement that the HCP has recently converted from new writer to current writer.
* Additonally If number of no hcp call(NO_OF_NO_HCP_CALL) is non zero then mention There was no call to the HCP within 48 hours of enrollment and do not mention the occurence.

**Instructions:**
* Combine the above insights into a 2-4 sentence summary.
```

* Present the information in the order listed above (enrollment status, enrollment activity, no-calls, antibiotics, lab tests).
* Ensure the summary flows naturally and avoids repetition.
* If there are any decimal numbers that should be integers, convert them to integers.
* Use proper singular/plural text

Example 1: (Singular enrollments with conversion from New to current Writter)

Input Data:

* `HCP_WRITER_STATUS`: 1
* `NO_OF_EF_ENROLLED`: 1
* `PROBABILTY_EF`: 0.92301
* `NO_OF_NO_HCP_CALL`: 0
* `ETHAMBUTOL`: 2
* `SUM_AZI_CLAR`: 3
* `SUM_RIF_RIF`: 0
* `AMIKACIN`: 1
* `SUM_MOX_LEV`: 0
* `AFBMACTESTS`: 4
* `AFBTESTSMAC6MONTHS`: 2
* `TOTAL_ARIKAYCE_USER` : 1

Output Summary:
This HCP has recently converted from new writer to a current writer, and has a 92% probability of a new enrollment.
There were 1 enrollment in last month.
The HCP has patients on antibiotics (Azithromycin or Clarithromycin :3 and ETHAMBUTOL: 2).
There were 2 MAC positive results in the last 6 months out of total 4 AFB test ordered by the HCP.

Example 2: (Plural enrollments with conversion from New to current Writter)

Input Data:

* `HCP_WRITER_STATUS`: 1
* `NO_OF_EF_ENROLLED`: 3
* `PROBABILTY_EF`: 0.753423
* `NO_OF_NO_HCP_CALL`: 0
* `ETHAMBUTOL`: 3
* `SUM_AZI_CLAR`: 1
* `SUM_RIF_RIF`: 0
* `AMIKACIN`: 0
* `SUM_MOX_LEV`: 0
* `AFBMACTESTS`: 9
* `AFBTESTSMAC6MONTHS`: 5
* `TOTAL_ARIKAYCE_USER` : 3

Output Summary:

This HCP has recently converted from new writer to a current writer, and has a 75% probability of a new enrollment.
There were 3 enrollments in last month.
The HCP has patients on antibiotics (Azithromycin or Clarithromycin :1 and ETHAMBUTOL: 3).
There were 5 MAC positive results in the last 6 months out of total 9 AFB test ordered by the HCP.

Example 3: (Singular enrollments for current writer)

Input Data:

* `HCP_WRITER_STATUS`: 1
* `NO_OF_EF_ENROLLED`: 1
* `PROBABILTY_EF`: 0.687435
* `NO_OF_NO_HCP_CALL`: 1
* `ETHAMBUTOL`: 7
* `SUM_AZI_CLAR`: 4
* `SUM_RIF_RIF`: 1
* `AMIKACIN`: 0
* `SUM_MOX_LEV`: 1
* `AFBMACTESTS`: 12
* `AFBTESTSMAC6MONTHS`: 4
* `TOTAL_ARIKAYCE_USER` : 4

Output Summary:
This HCP is a current writer,and has a 68% probability of a new enrollment.
There were 1 enrollment in last month and there was 1 occurence where the HCP was not contacted within 48 hours of the enrollment.
The HCP has patients on antibiotics (Azithromycin or Clarithromycin :4 and ETHAMBUTOL: 7 and RIFAMIN or RIFABUTIN:1).
There were 4 MAC positive results in the last 6 months out of total 12 AFB test ordered by the HCP.

Example 4: (Singular enrollment)

Input Data:

* `HCP_WRITER_STATUS`: 1
* `NO_OF_EF_ENROLLED`: 1
* `PROBABILTY_EF`: 0.4234311
* `NO_OF_NO_HCP_CALL`: 2
* `ETHAMBUTOL`: 3
* `SUM_AZI_CLAR`: 1
* `SUM_RIF_RIF`: 0
* `AMIKACIN`: 0
* `SUM_MOX_LEV`: 0
* `AFBMACTESTS`: 0
* `AFBTESTSMAC6MONTHS`: 0
* `TOTAL_ARIKAYCE_USER` : 3

Output Summary:

This HCP is a current writer,and has a 42% probability of a new enrollment.
There were 1 enrollment in last month and there were 2 occurences where the HCP was not contacted within 48 hours of the enrollment.
The HCP has patient on antibiotics (Azithromycin or Clarithromycin :1 and ETHAMBUTOL: 3).
There were no AFB tests ordered in the last 6 months.

Example 5: (No enrollment, new writer)

Input Data:

* `HCP_WRITER_STATUS`: 0
* `NO_OF_EF_ENROLLED`: 0
* `PROBABILTY_EF`: 0.953433
* `NO_OF_NO_HCP_CALL`: 0
* `ETHAMBUTOL`: 3
* `SUM_AZI_CLAR`: 1
* `SUM_RIF_RIF`: 1
* `AMIKACIN`: 0
* `SUM_MOX_LEV`: 0
* `AFBMACTESTS`: 6
* `AFBTESTSMAC6MONTHS`: 3
* `TOTAL_ARIKAYCE_USER` : 0

Output Summary:
This HCP is a non-writer,and has a 95% probability of a new enrollment.
The HCP has 1 patient on antibiotics(Azithromycin or Clarithromycin :1 and ETHAMBUTOL: 3 and RIFAMIN or RIFABUTIN:1).
There were 3 MAC positive results out of total 6 AFB tests ordered in the last 6 months by the HCP.

Example 5: (No enrollment, existing writer)

Input Data:

* `HCP_WRITER_STATUS`: 0
* `NO_OF_EF_ENROLLED`: 0
* `PROBABILTY_EF`: 0.4614525
* `NO_OF_NO_HCP_CALL`: 0
* `ETHAMBUTOL`: 5
* `SUM_AZI_CLAR`: 1
* `SUM_RIF_RIF`: 0
* `AMIKACIN`: 0
* `SUM_MOX_LEV`: 0
*`AFBMACTESTS`: 12
* `AFBTESTSMAC6MONTHS`: 3
* `TOTAL_ARIKAYCE_USER` : 0

Output Summary:
This HCP is a non-writer, and has a 46% probability of a new enrollment.
The HCP has 3 patients on antibiotics(Azithromycin or Clarithromycin :1 and ETHAMBUTOL: 5).

There were 3 MAC positive results out of total 12 AFB tests ordered in the last 6 months by the HCP.

Now, I will provide you with the data for different HCPs. Generate a concise and informative summary for each HCP based on the instructions above.
** Make sure for the summary the new information starts on new line.**

**Data*
* `HCP_WRITER_STATUS`: {HCP_WRITER_STATUS}
* `NO_OF_EF_ENROLLED`: {NO_OF_EF_ENROLLED}
* `PROBABILTY_EF`: {PROBABILITY_EF}
* `NO_OF_NO_HCP_CALL`: {NO_OF_NO_HCP_CALL}
* `ETHAMBUTOL`: {ETHAMBUTOL}
* `SUM_AZI_CLAR`: {SUM_AZI_CLAR}
* `SUM_RIF_RIF`: {SUM_RIF_RIF}
* `AMIKACIN`: {AMIKACIN}
* `SUM_MOX_LEV`: {SUM_MOX_LEV}
* `AFBMACTESTS`: {AFBTESTS6MONTHS}
* `AFBTESTSMAC6MONTHS`: {AFBTESTSMAC6MONTHS}
* `TOTAL_ARIKAYCE_USER` : {TOTAL_ARIKAYCE_USER}

▯

9.4 Generative Model (LLM) Reasoning Prompt

Below is the prompt for the generation of the Reasoning.

▯You are an AI assistant tasked with prioritizing Healthcare Professionals (HCPs) based on their likelihood of enrolling in a program. You must strictly adhere to the following logic to determine the priority and reasoning for each HCP:

Rules:

* **If `NO_OF_EF_ENROLLED` > 0 and `VISITS_LAST_2_WEEKS` = 0:**
   * Priority: High
   * Reasoning: "The HCP has a high priority, because it had a recent enrollment event and has not been visited recently."

* **If `NO_OF_EF_ENROLLED` > 0 and `VISITS_LAST_2_WEEKS` > 0 and `PROBABILITY_EF` > 0.6:**
   * Priority: High
   * Reasoning: "The HCP has a high priority, because it has a high probability of an enrollment."

* **If `NO_OF_EF_ENROLLED` > 0 and `VISITS_LAST_2_WEEKS` > 0 and `PROBABILITY_EF` < 0.6 and `PROBABILITY_EF` > 0.3:**
   * Priority: Medium

   * Reasoning: "The HCP has a medium priority, because it has a medium probability of an enrollment."

* **If `NO_OF_EF_ENROLLED` > 0 and `VISITS_LAST_2_WEEKS` > 0 and `PROBABILITY_EF` < 0.3:**
   * Priority: Low
   * Reasoning: "The HCP has a low priority, because it has a low probability of an enrollment."

* **If `NO_OF_EF_ENROLLED` = 0 and `PROBABILITY_EF` > 0.6 and `HCP_WRITER_STATUS` = 1:**
   * Priority: High
   * Reasoning: "The HCP has a high priority, because it has a high probability of an enrollment and is not an existing writer."

* **If `NO_OF_EF_ENROLLED` = 0 and `PROBABILITY_EF` > 0.6 and `HCP_WRITER_STATUS` = 0:**
   * Priority: High
   * Reasoning: "The HCP has a high priority, because it has a high probability of an enrollment."

* **If `NO_OF_EF_ENROLLED` = 0 and `PROBABILITY_EF` < 0.6 and `PROBABILITY_EF` > 0.3 and `HCP_WRITER_STATUS` = 1:**
   * Priority: Medium
   * Reasoning: "The HCP has a medium priority, because it has a medium probability of an enrollment and is not an existing writer."

* **If `NO_OF_EF_ENROLLED` = 0 and `PROBABILITY_EF` < 0.6 and `PROBABILITY_EF` > 0.3 and `HCP_WRITER_STATUS` = 0:**
   * Priority: Medium
   * Reasoning: "The HCP has a medium priority, because it has a medium probability of an enrollment and is an existing writer."

* **If `NO_OF_EF_ENROLLED` = 0 and `PROBABILITY_EF` < 0.3 and `HCP_WRITER_STATUS` = 0:**
   * Priority: Low
   * Reasoning: "The HCP has a low priority, because it has a low probability of an enrollment and is an existing writer."

* **If `NO_OF_EF_ENROLLED` = 0 and `PROBABILITY_EF` < 0.3 and `HCP_WRITER_STATUS` = 1:**
   * Priority: Low
   * Reasoning: "The HCP has a low priority, because it has a low probability of an enrollment, but is not an existing writer."

Examples:

* **Input:** `NO_OF_EF_ENROLLED` = 2, `VISITS_LAST_2_WEEKS` = 0
   * **Output:**
      The HCP has a high priority, because it had a recent enrollment event and has not been visited recently.

* **Input:** `NO_OF_EF_ENROLLED` = 2, `VISITS_LAST_2_WEEKS` = 1, `PROBABILITY_EF` = 0.7,
  * **Output:**
    The HCP has a high priority, because it has a high probability of an enrollment.

* **Input:** `NO_OF_EF_ENROLLED` = 2, `VISITS_LAST_2_WEEKS` = 1, `PROBABILITY_EF` = 0.4,
  * **Output:**
    The HCP has a high priority, because it has a medium probability of an enrollment.

* **Input:** `NO_OF_EF_ENROLLED` = 0, `PROBABILITY_EF` = 0.7, `HCP_WRITER_STATUS` = 0
  * **Output:**
    The HCP has a high priority, because it has a high probability of an enrollment.

* **Input:** `NO_OF_EF_ENROLLED` = 0, `PROBABILITY_EF` = 0.4, `HCP_WRITER_STATUS` = 1
  * **Output:**
    The HCP has a medium priority, because it has a medium probability of an enrollment and is not an existing writer.

* **Input:** `NO_OF_EF_ENROLLED` = 0, `PROBABILITY_EF` = 0.2, `HCP_WRITER_STATUS` = 0
  * **Output:**
    The HCP has a low priority, because it has a low probability of an enrollment and is an existing writer.

Instructions:

It is imperative that you follow the logic defined above precisely when determining the priority and reasoning for an HCP. Do not deviate from these rules.

Now, I will provide you with the input values for different HCPs. For each HCP, determine their priority (High, Medium, or Low) and provide only the exact reasoning as specified in the rules above.

`NO_OF_EF_ENROLLED` = {NO_OF_EF_ENROLLED}
`PROBABILITY_EF` = {PROBABILITY_EF}
`HCP_WRITER_STATUS` = {HCP_WRITER_STATUS}
`VISITS_LAST_2_WEEKS` = {VISITS_LAST_2_WEEKS}

9.5 Generative Model (LLM) Messages Prompt

Below is the prompt used for generation of the messages

You are an AI assistant designed to select the most relevant messages for Healthcare Professionals (HCPs) based on their current situation.

```
**Task:**

* Select **two** messages from the predefined list in the **Messages** section.
* Base your selection **exclusively** on the provided **Data** and the message descriptions.
* **Output only the message names.**

**Selection Logic:**

1. **Mandatory Messages:**
  * **Always** select "New Enrollment event" if `NO_OF_EF_ENROLLED` > 0.
  * **Always** select "No Call on 48hrs of Enrollment event" if `NO_OF_NO_HCP_CALL` > 0.

2. **Inferential Selection:**
  * **If neither of the mandatory messages applies**, select two messages based on the remaining data and message descriptions. Use the following guidelines to infer the best fit:
    * **`HCP_WRITER_STATUS`:** Consider whether the HCP is a new or existing Arikayce writer.
    * **Antibiotic Usage:**  If the HCP has patients on antibiotics (`ETHAMBUTOL`, `SUM_AZI_CLAR`, `SUM_RIF_RIF`, `AMIKACIN`, `SUM_MOX_LEV` > 0), prioritize messages related to Arikayce's role in treatment.
    * **Lab Tests:** If `AFBTESTSMAC6MONTHS` > 0, consider messages related to Arikayce's efficacy and relevance to those tests.
    * **Enrollment Probability:** If `NO_OF_EF_ENROLLED` = 0, but `PROBABILTY_EF` is high, consider messages that support enrollment.
    * **Recent Engagement:**  Use data like `DAYS_BEFORE_LAST_VISIT` and `VISITS_LAST_2_WEEKS` to infer the HCP's current engagement level and choose messages accordingly.
    * **Priority Score:**  Consider the overall `PRIORITY_SCORE` as a general indicator of the HCP's potential need for Arikayce.

**Message Options:**

(Provide the full list of messages from the `<MESSAGES>` section here)

**Example:**

**Input Data:**

* `HCP_WRITER_STATUS`: 0
* `ETHAMBUTOL`: 5
* `SUM_AZI_CLAR`: 2
* `SUM_RIF_RIF`: 0
* `AMIKACIN`: 1
* `SUM_MOX_LEV`: 0
* `AFBTESTSMAC6MONTHS`: 10
* `NO_OF_EF_ENROLLED`: 1
```

* `PROBABILTY_EF`: 0.8
* `NO_OF_NO_HCP_CALL`: 0
* `DAYS_BEFORE_LAST_VISIT`: 30
* `VISITS_LAST_2_WEEKS`: 0
* `PRIORITY_SCORE`: 0.7

Output:

New Enrollment event, Arikayce patient profile

Explanation:

"New Enrollment event" is selected because `NO_OF_EF_ENROLLED` > 0.
"Arikayce patient profile" is selected because the HCP has patients on various antibiotics and has recent lab test results, suggesting they are actively managing patients with potential NTM infections.

Now, I will provide you with data for different HCPs and the pool of available messages. Select the two most appropriate messages for each HCP based on the data and the instructions above.

Data
* `HCP_WRITER_STATUS`: {HCP_WRITER_STATUS}
* `NO_OF_EF_ENROLLED`: {NO_OF_EF_ENROLLED}
* `PROBABILTY_EF`: {PROBABILITY_EF}
* `NO_OF_NO_HCP_CALL`: {NO_OF_NO_HCP_CALL}
* `ETHAMBUTOL`: {ETHAMBUTOL}
* `SUM_AZI_CLAR`: {SUM_AZI_CLAR}
* `SUM_RIF_RIF`: {SUM_RIF_RIF}
* `AMIKACIN`: {AMIKACIN}
* `SUM_MOX_LEV`: {SUM_MOX_LEV}
* `AFBTESTSMAC6MONTHS`: {AFBTESTSMAC6MONTHS}

Messages
Message Name	Used for	Life Cycle Stage
DSA	Disease State Awareness	Before any antibiotic usage
Guidelines	NTM treatment guidelines	Before any antibiotic usage
Arikayce patient profile	Different patient types	During antibiotic usage or MAC/AFB test positive for 6 months
Arikayce indication	FDA info on how to properly prescribe	When there is probable enrollment
Arikayce efficacy and safety	Primary benefits and safety profile	When there is probable enrollment
6-month urgency	6-month MAC test positive while on antibiotics	Probable enrollment
Arikayce MOD	How the drug treats the disease and mechanism	During enrollment
Arikayce access	Insurance coverage of medicine	During enrollment

Arikayce AE management	Marketing tool for common adverse events and side effects	Later stage of enrollment
Arikares	Patient support program and next steps on enrollment submission	On enrollment
Arikayce patient counseling	Education info for patients	During enrollment
New Enrollment event	When there is new enrollment	On enrollment
No Call on 48hrs of Enrollment event	When there is no call made within 48 hrs after enrollment	After enrollment

▯

9.6 MLOps Tables Metadata

Overall, below are the operational tables updated or used during the MLOps pipeline of Model-1 specifically.

Table Name	Description
AI_TS_EF_MODEL_METRICS	Stores the performance metrics of different trained models.
AI_TS_EF_PREDICTION_INFERENCE_INPUTS	Stores input data used for generating predictions using Model-1.
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	Contains training data for every iteration of Model-1 training.
AI_TS_RUN_DETAILS	Stores the run information of all models' training and runs.
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	Contains the output generated from Model-1.
AI_TS_EF_PREDICTION_TRAINING_FEATURES	Contain features used for training Model-1 along with values.
AI_TS_HCP_PRIORITY_RANKING_STAGING	Contains all the input features used for model 3 prioritization input and output columns for prioritization.
AI_TS_HCP_PRIORITY_RANKING	Contains all the UI consumption based output columns from HCP RANKING model (model 3) with Insights,messages,reasoning and ranking for all HCPs.

Further, below are the specific columns in each of these tables.

Table Name	Column Name	Data Type
AI_TS_EF_MODEL_METRICS	RUN_ID	NUMBER
AI_TS_EF_MODEL_METRICS	MODEL_ID	TEXT
AI_TS_EF_MODEL_METRICS	THRESHOLD	FLOAT

AI_TS_EF_MODEL_METRICS	ACCURACY	FLOAT
AI_TS_EF_MODEL_METRICS	PRECISION	FLOAT
AI_TS_EF_MODEL_METRICS	RECALL	FLOAT
AI_TS_EF_MODEL_METRICS	F1_SCORE	FLOAT
AI_TS_EF_MODEL_METRICS	ROC_AUC_SCORE	FLOAT
AI_TS_EF_PREDICTION_INFERENCE_INPUTS	ETHAMBUTOL_bkt	NUMBER
AI_TS_EF_PREDICTION_INFERENCE_INPUTS	SUM_AZI_CLAR_bkt	NUMBER
AI_TS_EF_PREDICTION_INFERENCE_INPUTS	SUM_RIF_RIF_bkt	NUMBER
AI_TS_EF_PREDICTION_INFERENCE_INPUTS	SUM_MOX_LEV_bkt	NUMBER
AI_TS_EF_PREDICTION_INFERENCE_INPUTS	AMIKACIN_bkt	NUMBER
AI_TS_EF_PREDICTION_INFERENCE_INPUTS	FLAG_EDA	NUMBER
AI_TS_EF_PREDICTION_INFERENCE_INPUTS	SUM_MAC_TEST_bkt	NUMBER
AI_TS_EF_PREDICTION_INFERENCE_INPUTS	FLAG_EDA_MAC	NUMBER
AI_TS_EF_PREDICTION_INFERENCE_INPUTS	RUN_ID	NUMBER
AI_TS_EF_PREDICTION_INFERENCE_INPUTS	current_datetime	NUMBER

AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	NPIID	TEXT
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	FIRSTNAME	TEXT
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	LASTNAME	TEXT
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	MONTH_EF_OK	NUMBER
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	TERRITORY	TEXT
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	REGION	TEXT
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	AREA	TEXT
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	TIER	TEXT
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	FLAG_EF_90_DAYS	TEXT
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	FLAG_RN_VS_TOTAL	FLOAT
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	ETHAMBUTOL	NUMBER
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	SUM_AZI_CLAR	NUMBER
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	SUM_RIF_RIF	NUMBER
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	SUM_MOX_LEV	NUMBER
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	AMIKACIN	NUMBER

AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	FLAG_EDA	NUMBER
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	SUM_MAC_TEST	NUMBER
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	FLAG_EDA_MAC	NUMBER
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	AFBTESTSMAC6MONTHS	FLOAT
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	AFBTESTS6MONTHS	FLOAT
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	PROBABILITY_EF	FLOAT
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	RUN_ID	NUMBER
AI_TS_EF_PREDICTION_INFERENCE_OUTPUTS	current_datetime	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AFBTESTSOVERALL	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AFBTESTSMACOVERALL	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AFBTESTS6MONTHS	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AFBTESTSMAC6MONTHS	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AFBTESTS3MONTHS	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AFBTESTSMAC3MONTHS	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AFBTESTS1MONTH	FLOAT

AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AFBTESTSMAC1MONTH	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AFBTESTS2WEEKS	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AFBTESTSMAC2WEEKS	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	ARIKAYCEEFSLTD	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	NPIID	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	MONTH_EF_OK_OLD	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	MONTH_EF_OK	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	INSMEDID	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	FIRSTNAME	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	LASTNAME	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	HCPNAME	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	TERRITORY	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	REGION	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AREAFILTER	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AREA	TEXT

AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	ADDRESS	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	CITY	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	STATE	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	ZIPCODE	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	CURRENTSEGMENT	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	NTMDECILE	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	NCFBEDECILE	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	SEGMENTLASTREPORT	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	EFSLAST6MONTHS	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	DATEOFLASTEF	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	EFSLTD	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	EFS6MONTH	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	CALLSLAST3MONTHS	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	TIER	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	EFSYTD	FLOAT

AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	PRIMARYSPECIALTY	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	SECONDARYSPECIALTY	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	PRIMARYSPECIALTYDESCRIPTION	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	SECONDARYSPECIALTYDESCRIPTION	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	HCPSTATUS	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	VR_TGT	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	VIRTUALREPTARGET	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	PDRPOPTOUT	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	FLAG_RN_VS_TOTAL	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	HCPNAME_right	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	LAST_FILL_DATE	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	PATIENT_COUNT	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AZITHROMYCIN	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	CLARITHROMYCIN	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	RIFAMPIN	FLOAT

AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	ETHAMBUTOL	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	RIFABUTIN	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	LEVOFLOXACIN	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	MOXIFLOXACIN	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	OFLOXACIN	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	CIPROFLOXACIN	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AMIKACIN	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AZITHROMYCIN_MONTHS	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	CLARITHROMYCIN_MONTHS	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	RIFAMPIN_MONTHS	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	ETHAMBUTOL_MONTHS	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	RIFABUTIN_MONTHS	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	LEVOFLOXACIN_MONTHS	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	MOXIFLOXACIN_MONTHS	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	OFLOXACIN_MONTHS	FLOAT

AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	CIPROFLOXACIN_MONTHS	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AMIKACIN_MONTHS	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AFBTESTSMAC_3_TO_6_M	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AFBTESTS_3_TO_6_M	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	HISTORYDATEDATE	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	NEW_DATE_M	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	RANK_DATE	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	FLAG_EF_90_DAYS	TEXT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	COCKTAIL_FLAG	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	LAB_TEST_FLAG	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	FLAG_EF_6MONTHS	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	FLAG_AZITHROMYCIN6MTH	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	FLAG_CLARITHROMYCIN6MTH	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	FLAG_ETHAMBUTOL6MTH	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	FLAG_RIFAMPIN6MTH	NUMBER

AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	FLAG_RIFABUTIN6MTH	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	FLAG_AMIKACIN	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	FLAG_MOXIFLOXACIN	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	FLAG_LEVOFLOXACIN	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	FLAG_OFLOXACIN	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	FLAG_CIPROFLOXACIN	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	NUMBER_ANTIBIO	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	FLAG_EDA	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	FLAG_EDA_3	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	FLAG_LAB_ANTIB	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	FLAG_EDA_MAC	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	SUM_AZI_CLAR	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	SUM_RIF_RIF	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	SUM_MOX_LEV	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	SUM_MAC_AFB	FLOAT

AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	SUM_MAC_TEST	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	SUM_AFB_TEST	FLOAT
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	FLAG_1st2nd_line	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	SUM_AZI_CLAR_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	ETHAMBUTOL_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AFBTESTSMAC6MONTHS_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AFBTESTSMAC3MONTHS_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	SUM_RIF_RIF_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	SUM_MOX_LEV_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AMIKACIN_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	AFBTESTS6MONTHS_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	SUM_MAC_TEST_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	SUM_AFB_TEST_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	RUN_ID	NUMBER
AI_TS_EF_PREDICTION_TRAINING_ALL_DATA	current_datetime	NUMBER

AI_TS_EF_PREDICTION_TRAINING_FEATURES	NPIID	NUMBER
AI_TS_EF_PREDICTION_TRAINING_FEATURES	FLAG_EDA	NUMBER
AI_TS_EF_PREDICTION_TRAINING_FEATURES	MONTH_EF_OK	NUMBER
AI_TS_EF_PREDICTION_TRAINING_FEATURES	ETHAMBUTOL_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_FEATURES	AFBTESTSMAC1MONTH	FLOAT
AI_TS_EF_PREDICTION_TRAINING_FEATURES	AFBTESTS1MONTH	FLOAT
AI_TS_EF_PREDICTION_TRAINING_FEATURES	AMIKACIN_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_FEATURES	MOXIFLOXACIN	FLOAT
AI_TS_EF_PREDICTION_TRAINING_FEATURES	LEVOFLOXACIN	FLOAT
AI_TS_EF_PREDICTION_TRAINING_FEATURES	AFBTESTS6MONTHS_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_FEATURES	AFBTESTSMAC3MONTHS_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_FEATURES	AFBTESTSMAC6MONTHS_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_FEATURES	LAB_TEST_FLAG	NUMBER
AI_TS_EF_PREDICTION_TRAINING_FEATURES	NUMBER_ANTIBIO	NUMBER
AI_TS_EF_PREDICTION_TRAINING_FEATURES	FLAG_EF_90_DAYS	NUMBER

AI_TS_EF_PREDICTION_TRAINING_FEATURES	FLAG_EDA_MAC	NUMBER
AI_TS_EF_PREDICTION_TRAINING_FEATURES	SUM_AZI_CLAR_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_FEATURES	SUM_RIF_RIF_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_FEATURES	SUM_MOX_LEV_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_FEATURES	SUM_MAC_AFB	FLOAT
AI_TS_EF_PREDICTION_TRAINING_FEATURES	FLAG_1st2nd_line	NUMBER
AI_TS_EF_PREDICTION_TRAINING_FEATURES	SUM_MAC_TEST_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_FEATURES	SUM_AFB_TEST_bkt	NUMBER
AI_TS_EF_PREDICTION_TRAINING_FEATURES	RUN_ID	NUMBER
AI_TS_EF_PREDICTION_TRAINING_FEATURES	current_datetime	NUMBER
AI_TS_RUN_DETAILS	RUN_ID	TEXT
AI_TS_RUN_DETAILS	START_TIME	TEXT
AI_TS_RUN_DETAILS	END_TIME	TEXT
AI_TS_RUN_DETAILS	PROCESS_NAME	TEXT
AI_TS_HCP_PRIORITY_RANKING_STAGING	RUNID	NUMBER

AI_TS_HCP_PRIORITY_RANKING_STAGING	NPIID	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	FIRSTNAME	VARCHAR
AI_TS_HCP_PRIORITY_RANKING_STAGING	LASTNAME	VARCHAR
AI_TS_HCP_PRIORITY_RANKING_STAGING	MONTH_EF_OK	VARCHAR
AI_TS_HCP_PRIORITY_RANKING_STAGING	TERRITORY	VARCHAR
AI_TS_HCP_PRIORITY_RANKING_STAGING	REGION	VARCHAR
AI_TS_HCP_PRIORITY_RANKING_STAGING	AREA	VARCHAR
AI_TS_HCP_PRIORITY_RANKING_STAGING	TIER	VARCHAR
AI_TS_HCP_PRIORITY_RANKING_STAGING	FLAG_EF_90_DAYS	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	FLAG_RN_VS_TOTAL	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	ETHAMBUTOL	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	SUM_AZI_CLAR	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	SUM_RIF_RIF	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	SUM_MOX_LEV	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	AMIKACIN	NUMBER

AI_TS_HCP_PRIORITY_RANKING_STAGING	FLAG_EDA	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	SUM_MAC_TEST	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	FLAG_EDA_MAC	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	AFBTESTSMAC6MONTHS	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	AFBTESTS6MONTHS	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	PROBABILITY_EF	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	LATEST_EF_DATE	DATE
AI_TS_HCP_PRIORITY_RANKING_STAGING	NO_OF_EF_ENROLLED	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	LATEST_NO_HCP_CALL_DATE	DATE
AI_TS_HCP_PRIORITY_RANKING_STAGING	NO_OF_NO_HCP_CALL	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	DAYS_PAST_LAST_EF	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	ARIKAYCE_TOTAL_USER	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	ARIKAYCE_ACTIVE_USER	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	HCP_WRITER_STATUS	VARCHAR
AI_TS_HCP_PRIORITY_RANKING_STAGING	SEGMENT	VARCHAR

AI_TS_HCP_PRIORITY_RANKING_STAGING	INPERSON_DATE	DATE
AI_TS_HCP_PRIORITY_RANKING_STAGING	DAYS_BEFORE_LAST_VISIT	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	VISITS_LAST_2_WEEKS	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	VISITS_LAST_1_WEEK	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	VISITED_AFTER_LATEST_ENROLLMENT	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	INSIGHTS	VARCHAR
AI_TS_HCP_PRIORITY_RANKING_STAGING	REASONING	VARCHAR
AI_TS_HCP_PRIORITY_RANKING_STAGING	MESSAGES	ARRAY
AI_TS_HCP_PRIORITY_RANKING_STAGING	VAL_INSIGHTS	VARCHAR
AI_TS_HCP_PRIORITY_RANKING_STAGING	VAL_REASONING	VARCHAR
AI_TS_HCP_PRIORITY_RANKING_STAGING	NO_ANTIOBIOTICS_USED	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	NEW_ENROLLMENT	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	HCP_WRITER_STATUS_WEIGHT	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	RECENT_ENGAGEMENT	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	INFLUENCING_FACTORS	VARCHAR

AI_TS_HCP_PRIORITY_RANKING_STAGING	EF_STATUS_SCORE	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	RECENT_ENGAGEMENT_SCORE	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	EF_PROBABILITY_SCORE	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	HCP_WRITER_STATUS_WEIGHT_SCORE	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	PRIORITY_SCORE	NUMBER
AI_TS_HCP_PRIORITY_RANKING_STAGING	PRIORITY	VARCHAR
AI_TS_HCP_PRIORITY_RANKING_STAGING	UPDATE_TIME	TIMESTAMP_NTZ
AI_TS_HCP_PRIORITY_RANKING	HCPNAME	Varchar
AI_TS_HCP_PRIORITY_RANKING	INSIGHT	Varchar
AI_TS_HCP_PRIORITY_RANKING	MESSAGES	Array
AI_TS_HCP_PRIORITY_RANKING	NPIID	Number
AI_TS_HCP_PRIORITY_RANKING	PRIORITY	Varchar
AI_TS_HCP_PRIORITY_RANKING	PRIORITY_SCORE	Number
AI_TS_HCP_PRIORITY_RANKING	PROBABILITY_EF	Number
AI_TS_HCP_PRIORITY_RANKING	RANKING	Number

AI_TS_HCP_PRIORITY_RANKING	READY_NOW_STATUS	Number
AI_TS_HCP_PRIORITY_RANKING	REASONING	Varchar
AI_TS_HCP_PRIORITY_RANKING	UPDATE_TIME	Timestamp_NTZ