“Model Refinement” Module Transcript

Chapter 1

Intro, Topics Covered, & Learning Outcomes

Hey. I'm Hayley, and I'm on the One AI team here at One Model. In previous modules, you learned the foundations of machine learning and how to build models in One AI. In this module, we will explore how to make small adjustments to these models for improved performance and better fit. This module is designed specifically for beginners in machine learning and One AI modeling. Our goal is to establish a baseline understanding of model refinement that will prepare you for more advanced techniques later. We'll focus on quick and easy ways to refine your models in One AI. Please note that this isn't an exhaustive guide to every refinement method, and the module is lengthy due to the variety of techniques covered. Feel free to skip around sections as needed.

We will cover an introduction to machine learning model refinement, using the EDA and Results Summary reports to inform model refinement, and some easy yet powerful methods for refining models in One AI.

After completing this module, you will understand the purpose and goals of model refinement in machine learning to prepare for advanced model configuration later. You will leverage the EDA and Results Summary reports to determine the appropriate refinement techniques for different models. You will master the configurations and overrides in the One AI tool to refine your models effectively, and you will understand their impacts to guide future model refinement efforts.

Chapter 2

Intro to Model Refinement

Section 2 - Introduction to Model Refinement

One AI is designed to take the guesswork out of creating good models. However, to develop a model we can confidently use for strategic decisions, some modifications and iteration may be necessary. This is where model refinement comes into play.

Model refinement involves improving the performance and accuracy of a model after its initial development. This process often includes making adjustments to enhance the model's predictive accuracy and fit for a specific task or dataset. The refinement techniques used depend on the type of model, its goals, and the current features selected on performance. These factors, along with the model's results, guide which refinement methods to apply.

Generally, model refinement can help in improving model performance, accuracy, and reliability; enhancing relevance and scalability to adapt to new or increasing data over time; preventing overfitting and enhancing generalization; aligning the model with business goals and allowing for targeted improvements; and increasing interpretability, making the model's behavior more understandable and explainable. Ultimately, regular refinement creates a feedback loop for continuous improvement.

Chapter 3

The EDA & Results Summary Reports Inform Model Refinement

Section 3 - The EDA & Results Summary Reports Inform Model Refinement

Although we technically could make all of these configurations and overrides for model refinement before running the model, we typically perform most of them afterward. This approach allows us to use the EDA and Result Summary reports to inform our decisions about which refinements are best-suited for the specific model we're working on.

Let's start with how we use the EDA report. The EDA report provides information on data preprocessing and feature selection. This guides adjustments to settings to allow different features to be processed and potentially selected by the model. Understanding why variables of great interest to you and the model's problem domain were automatically dropped helps determine whether to adjust settings to potentially include them or allow more or fewer features to be used by the model.

An example of the EDA report informing feature selection is if the cost center column was selected by the model, but we know it shouldn't be used due to lack of validation or relevance, we can exclude this variable in the core attributes section of the recipe to ensure it won't be selected in the next run.

Alternatively, if the performance review column was dropped because it had more than 5% null values, we could either increase the null drop threshold to allow it to potentially be included, apply a per column intervention to ignore the null drop threshold, or provide a null-filling strategy for this column. If we don't understand why a column was excluded, we can't make the necessary changes to avoid it from being dropped by One AI.

The Results Summary report provides the general configuration of the model selected by One AI as the best fit, ranks the selected features by importance and details the model's performance. This information is crucial for refining our model for several reasons. We can review the selected configuration and decide if modifications are needed, such as including upsampling for imbalanced classes or selecting an algorithm specifically designed to handle outliers. We can also examine the ranked features and choose to remove the lowest ranking ones by adjusting the maximum number of features allowed to be included in the model with a dimensionality reduction reconfiguration. Finally, we can monitor the model's performance after each run, comparing F1, precision, and recall scores to see how our changes impact the model. The ultimate goal of model refinement is to create a better model with each run, maximizing the insights from One AI.

Once you have a fairly good model, you should make adjustments one at a time to observe their impact on performance and output. This approach helps identify which changes are beneficial and which are not. For instance, if performance increases by 1% overall, one change might have increased it by 4% while another decreased it by 3%. Making these changes incrementally allows you to revert those changes that negatively affect performance.

You may need to make changes that decrease performance to improve model interpretability and usefulness. For example, if the model selected a placeholder column with no actual value, like a randomized ID key, it should be dropped. Even if this decreases performance, it's necessary because the column is random and will not be helpful for predictions on new data and makes the model less explainable. We should not sacrifice interpretability for a few performance points.

In the next few sections, I will cover some easy yet powerful methods for refining models in the One AI tab of One Model.

Chapter 4

Recipe Reconfiguration

Section 4 - Recipe Reconfiguration

Chapter 5

Recipe Reconfiguration: Adjust History Training Intervals

The first recipe reconfiguration strategy we'll discuss is adjusting the history training intervals for the model dataset. Let's head over to One Model to take a look. This setting is configured in the One AI Query Builder during the "How much history do you want to use to train your predictive model?" step. To access this, click the 'Edit' button on your model and then select 'Configure One AI Recipe'.

In step four, you can adjust how much history to use for training your model. The length of the training interval is determined by the "From your Headcount date how far into the future do you want to predict?" step, so let's start there. The selected time interval defines the training interval length in step four. For example, if the interval is set to 1 year, which is the default, the training interval will be 1 year.

To increase the training interval to 2 years, enter "2" in the designated field for two 1-year intervals. Setting this to 2 or 3 years can significantly increase performance by providing more data for the model to learn from. However, the tradeoff is that going further back in history may introduce data that doesn't reflect the current state of your organization and will increase the model's run time slightly.

Consider the age and recent changes in your organization. For newer organizations or those with recent significant changes, such as acquisitions, extreme growth, or layoffs, less history may be better. In our experience, one or two years is optimal, but it's useful to try different values to see which yields the best results.

Chapter 6

Recipe Reconfiguration: Improve Core Attribute Selection

The next strategy is improving core attribute selection, which determines which input features are available in the model dataset as potential drivers for predictions.

This is configured in the One AI Query Builder in the “Which core attributes do you want to use in your prediction?” step of the recipe. You can configure this by selecting one of the four available scopes or manually selecting columns by table or a combination of both.

Just a quick reminder, the available scopes are 'None,' where the user must individually pick columns to include in the model dataset by moving columns from the Excluded section to the Included section.

'Narrow', where the model dataset includes all columns from the same table as your unique identifier from the headcount related step two. 'Balanced' where the model dataset includes all columns up to one join away from the table of the unique identifier. And 'Broad', where the model dataset includes all columns regardless of the number of joins from the table of the unique identifier as long as a join path is available.

Additionally, you can hand select specific columns currently excluded from the model by clicking the checkmark button next to any column in the excluded section, which moves them into the Included section. This allows for adding columns without selecting a larger scope.

You can also exclude columns in the included section by clicking the 'X' button, moving them back to the Excluded section.

Exclude columns that are unvalidated or irrelevant, such as key ID columns or random columns identified in the EDA report and Results Summary.

Adjusting the scope can impact model performance. Increasing the scope from Narrow to Balanced allows the model to try more attributes, potentially improving performance. Conversely, reducing the scope from Broad to Narrow or Balanced can help avoid selecting irrelevant features, leading to a more understandable and effective model. In our experience, a Balanced or Narrow scope typically results in the most optimal models. Remember, we don't want to sacrifice interpretability for performance.

Chapter 7

Recipe Reconfiguration: Improve Generative Attribute Selection

The final recipe configuration strategy that we will discuss is improving generative attribute selection. This can be configured in the One AI Query Builder in the "Which generative attributes do you want to use in your prediction?" step. Generative attributes are new input variables derived from the original model dataset that are contenders for being selected as features.

Providing models with a variety of generative attributes is a great way to increase model performance and robustness because they transform raw data into more meaningful features that can be richer in predictive value than what's available on individual employee records.

There are a few different things we can try here. First, you can create and select new generative attributes to provide the model with more powerful attributes to try.

Next, you can use the selection boxes to choose any additional, relevant generative attributes that have not yet been selected.

And finally, you can edit your generative attributes to improve their likelihood of being selected. For example, if you see that "Demotions for All Time" is being automatically dropped due to null values, click the pencil icon and check the box to fill nulls with zeros and save.

If you need help trying the suggestions above, refer to the Generative Attribute module.

Chapter 8

Create Separate Models for Different Parts of the Organization

Section 5 - Create Separate Models for Different Parts of the Organization

Creating individual models for different parts of an organization can be highly useful due to varying behaviors across subgroups. While understanding drivers and risk factors for the entire organization is valuable, it's not the complete picture. Focusing on these smaller subgroups tailors the model to their unique characteristics, providing deeper insights that would be missed if we were to run a global model.

Employee motivations and behaviors differ often based on factors like location, department, manager, and employment type. We should consider these differences when grouping your model population into separate models. For example, attrition drivers for hourly call center employees might differ significantly from those for salaried office workers. If the training data is dominated by one group, the model might favor that group, leading to inaccurate predictions for others.

Creating multiple models is easy in One AI by copying completed recipes into new machine learning models. In order to copy a model, click 'Add Machine Learning Model' in the upper right hand corner and provide a unique display name. Then select 'Using Data from One AI Recipe' and choose the existing machine learning model to copy from the dropdown.

Click 'Copy Existing One AI Recipe', and then 'Configure One AI Recipe'. This brings you to the One AI Query Builder Recipe screen. To create subgroups, change the headcount population in step two of the recipe by either using different headcount metrics or filtering with the filter icon. For example, to change a manager-only model to non-managers, adjust the managerial dimension. Repeat this process to create models for each subgroup. While relative to the size of your organization and preference, subgroups should have 500 or more instances to provide sufficient data for the model to learn from.

When you are finished creating your subgroup model, click 'Save' and run your new model.

Chapter 9

Adjust Global Settings

Section 6 - Adjust Global Settings

The next strategy we will discuss is adjusting global settings. Global settings refer to overarching configuration parameters and settings that affect the behavior and performance of the entire model, not individual specific columns. Think about these as the rules for which columns One AI automatically drops. These settings can be unique to each model and do not impact every model on your site.

The most common reason for a column to be automatically dropped is due to excessive missing or null data determined by the null drop threshold. The null drop threshold specifies the percentage of missing values in a column that will trigger its exclusion from the model. For One AI models, the default threshold is 0.05, meaning columns with 5% or more null values will be dropped. Note that zeros are not considered nulls. Nulls are missing or blank data. In the EDA report, a column drop due to excessive null values will have a grey 'Missing' label next to the red 'Dropped' label. It will also indicate the number and percentage of null values, helping you decide whether to adjust the threshold to include the column or to leave it dropped.

Let's jump into One Model to demo how to adjust this.

To adjust the null drop threshold, click the 'Edit' button on your model and scroll down and expand the "Global Settings" section. Toggle the override slider to "On" next to the Null Drop Threshold to reveal the designated field where this threshold is configured.

Input any value ranging from 0 - 1. For example, entering 0.1 means that columns can have up to 10% null values before being automatically dropped by One AI during preprocessing. Then, save and rerun your model.

This adjustment is useful if there are columns that are intentionally partially null, such as performance review data, which might be more null for newer employees than other columns. Alternatively, you can perform a per column intervention to prevent a single column from being automatically dropped by One AI, allowing you to maintain the null drop threshold for all other columns. Additionally, null fill strategies can be used to replace missing values with estimated or calculated ones. We will discuss both of these options in the next section.

Chapter 10

Perform Per Column Interventions

Section 7 - Performing Per Column Interventions

A per column intervention refers to making changes specifically for individual columns or features in a model dataset. Unlike global settings, which apply to the entire model, these changes are tailored to the selected columns. Different columns may have distinct characteristics, such as varying data types, scales, distributions, and degrees of missing data or outliers. Therefore, applying the same preprocessing to all features may not be optimal. The per column intervention approach allows for tailored preprocessing, cleaning, and transformation for each column, optimizing the model's ability to learn and generalize patterns from the data. This can result in a more performant and understandable model.

In One AI, there are a few per column interventions available. We will examine droppability and null filling in this module.

To configure per column interventions, click 'Edit' on the model and scroll down to the "Per Column Intervention" section. Toggle the override slider to "On" and click the caret to expand the options. Collapse the column intervention section and use the dropdown and the plus button to add the columns you wish to modify.

For example, we can add the bonus column from the employee table like so. You can add as many columns as needed. The dropdown will only display columns included in the model. To get started on a specific column, click the caret next to it to expand the intervention options.

The first option is to modify that column's droppability using the droppable drop down menu. There are three options here. First, is 'Droppable' means the column can be selected if it's predictive, or it can be dropped by One AI if it doesn't improve performance or if it violates a global setting. This is the default treatment for all columns in the model dataset.

Next, we have 'Not Droppable', which means that the column cannot be dropped from the model by One AI. This doesn't necessarily mean that the column will be selected for a given model, but it will always be processed and considered regardless of if it violates any global settings. For example, if it's more null than your null drop threshold, but it's predictive, it will be selected regardless.

This is useful for trying columns that don't meet global settings without altering the settings for the entire model.

And finally, we have 'Always'. Always means the column will always be dropped and not tested in any way. This is more easily done in the core attributes section of the One AI Query Builder.

Next, you'll see the Null Fill section. Toggle the override slider to "On" and and expand to configure null filling for your selected columns. Null values can negatively impact performance, so addressing them is helpful for accurate predictions. A good null fill strategy allows a column to be used in the model dataset despite exceeding the null drop threshold, making more attributes available for the model to try. This is a good alternative to changing a column's droppability.

First configure the null fill strategy to choose how you want to handle the missing data. Toggle the override slider to "On" and expand the dropdown to reveal several options.

Mean, median, and mode replace missing values with the overall columns mean, median, or mode. This is a simple straightforward option.

B fill is short for backwards fill, and f fill is short for forwards fill. These methods are particularly useful for time series data. For b fill, null values will be filled starting from the last value working toward the first with nulls being filled with the next populated value in the column. For f fill, it's just the opposite. Null values will be filled starting from the first value working toward the last. Any null will be filled with the previous populated value. Pad is just another term for f fill. And custom is a custom strategy that lets you use a specific value for all nulls.

Toggle the override slider to "On" for the custom fill value box and enter your desired value in the designated field. This field accepts numerical date or categorical values. For example, you could fill all nulls with the number 1, like so. When you are finished with your per column interventions, save your model and rerun to see how it impacts.

Chapter 11

Conclusion & Thanks

Mastering model refinement is important for enhancing the performance and accuracy of your machine learning models in One AI. By leveraging EDA and Results Summary reports, performing targeted configurations, and understanding the impact of your adjustments, you can create more reliable and insightful models. If you're looking to deepen your knowledge on the topic, check out the Advanced Configuration and Global Settings module series. Happy modeling!