**4-2– Machine Learning 2** 

In order to make your dataset suitable for modeling, you need to be familiar with the features that you are going to use as well as the state of its data. As discussed in the previous lessons, nulls and outliers take a great toll on the quality of the model or the ML algorithm. Outliers can lead to vague or misleading predictions due to the extreme changes in data while missing data leads to inconsistency and can cause errors in some types of algorithms. To assist you on how your data and features fare from a data modeling standpoint, alteryx has included the data health and feature types in the intelligence suite toolset.

**Data Health** tool from the name itself is used to check on the health of your data. To determine how healthy your data is, the tool analyzes missing values, outliers, and sparsity. By doing so, it tells us if the feature is ready to be used for analytics or if it needs additional treatment and cleaning. This tool also requires at least 30 rows of data and relies on column data types for evaluation.

For this example, we are going to check the health of the bank marketing dataset that will be used for training the predictive model. This dataset contains social and economic features that will be used to predict the value of “y” which is the tag that indicates if the client has subscribed to a time deposit. Since the data health tool relies on the data type, we need to convert them to their proper data type before feeding the data into the tool. To make the conversion faster, navigate to the preparation tool palette and drag an auto field tool to the canvas. Connect it to the input data tool. Since our input is a CSV which automatically makes all of the data types string by default, all 17 columns were listed on the auto field. Select all then run the workflow.

Next, from the machine learning palette, drag a data health tool and connect it after the select tool. This tool only has 2 configurations. First, you need to set the scale. This scale will be the score of each field based on the tool's assessment of missing values, outliers, and sparsity. You can choose to either score by “percentage” which can range from 0 to 100 or “normalized” which ranges from 0.00 to 1.00 and includes 2 decimal places for scores in between. It would be easier to look at percentages as a whole so we will choose the percentage option. Next is the option to “Output Recommendations Based On Score”. Check the box if you want the tool to give you recommendations for how to improve the health of the data, based on the score it receives. The recommendation appears as an additional column in the S output anchor. Let’s enable this option to check alteryx recommended actions later. Now that we have all configurations complete, add a browse tool to all 3 anchors by pressing CTRL + SHIFT + B upon selecting the tool. Once done, run the workflow.

The first output anchor **S** contains each column's associated data-health scores. Each field is scored on a specific metric and tagged with its respective rating label based on the score and the recommended action for each. You can then use this data downstream to filter on fields that were rated as “very poor” or “needs improvement” so you can make further corrections on how your data was collected and stored. Back to our output, the first field “age” was evaluated using 5 metrics, the first metric **Column Score** is the overall score for the specific field. Age was tagged as “needs improvement” so let’s look at the rest of its metrics to see where it fares poorly. The age field’s rating for the metrics **Missing**, **Unique**, and **Unary** were “Very Good” so this means that it has minimal to no nulls, has an acceptable number of members, and can be used as a feature. The metric where it did bad was “Outliers” as it was given the lowest score of 0. Alteryx recommended action also says that “This column has a large number of outliers and the data appears poorly distributed, so it might not be useful for analysis. Investigate the column thoroughly, or drop it entirely.” Understandable, since this is the Age field so we might opt to categorize the age into groups or bins later on. If you are deciding on features to include in your model, it is best to check on each metric to help you clean and normalize your data before starting on a model.

The output of the 2nd anchor **R** is more high level. On the report tab of the browse tool, it shows a comprehensive report about the data's health overall. Since this is a report object, you can only view this chart if you’ve inserted a browse tool after the R anchor. This report can also be exported to a file by using the render tool. The first chart shows a dial that shows the total data health. You can see the exact number of this figure by navigating to the output anchor S and looking at the score for the field “Total Data Health”. In this case, our Bank marketing dataset was rated as “Needs Improvement” with a score of 72. The report anchor also shows a breakdown of 3 additional metrics. **Missing Value by health column** which shows how many fields have null value issues. Our dataset did great on this first metric as all fields were tagged “Very Good”. The 2nd metric is **Missing Value by Row**. This shows if the missing values are limited to specific records of data or if it is pervasive throughout the dataset. All our 45211 rows have been marked as “Very Good” which means no records were found to be purely nulls or can be considered as blank rows. Finally, the last metric **Outlier Health by Column** shows how many columns have outliers. In this case, our dataset has 6 fields that were marked as “Very Poor”.

The **O** output anchor contains all of the outliers that the tool has detected. Its data is standardized into 3 columns, the **RecordID**, **Field** or the column that contains the Outlier, and the **Outlier** value. The RecordID is based on the sorting of the original data source, so you only need to add a RecordID tool and join it back to find the specific records that were tagged as outliers. We can also use this output anchor to check which field has the most outlier value. Simply drag a summarize tool and connect it to the O output then configure the summarize tool to group it by Field, then count distinct the RecordID. Run the workflow once done. As we can see, “pdays” and “previous“ has the most outliers. **Pdays** is the number of days that passed by after the client was last contacted from a previous campaign while **Previous** is the number of contacts performed before this campaign and for this client.

**Feature Types tool** is used to automatically identify what types of features are in your data. Do take note that Feature type and data type are not the same. Feature type is primarily used with the Machine Learning tools while data type is used throughout alteryx designer. The Feature Types tool performs “semantic data typing,” which adds real-world context to the base data type. For example, a field called “CustomerID” might be stored as an Integer data type, but semantic data typing can map this Integer field as an “ID” feature type to better leverage this field when it comes to feature engineering. Here's a complete list of feature types available in the alteryx intelligence suite:

Autodetect

Discrete

Categorical

ID

Zip Code

Country Code

Sub-region Code

Ordinal

Boolean

Numeric

Numeric Time Index

Index

Datetime

Datetime Time Index

Date Of Birth

Time Index

Time Delta

Text

LatLong

IP Address

Full Name

Email Address

URL

Phone Number

For this workflow, notice how the feature type output differs if we take data directly from the source, and compare it to the autofield data stream. Drag a feature type tool and connect it to the input data tool. Then drag another feature type tool and connect it to the output anchor of the autofield tool. No configuration is required to use the Feature Types tool. It autodetects the feature types in your data. But you can also use the Change Type dropdown to manually select the feature type you want to assign to a column in the data. Let’s leave this to the default autodetect then run the workflow to compare the output types. For the Feature Type that was connected directly to the input data, all of its fields were automatically tagged as “Categorical” since our original data type was V\_String. But if we look at the feature type that was connected to the output of the autofield tool, other columns were tagged as “Numeric” which is the correct type since these columns or features contain integers. This shows how feature type can be reliant on the data types as well as the logical aspect of the characters in the data. Like our earlier example with the CustomerID, it looks both at its data type, which can be integer64 or smaller, as well as the composition of numeric characters. Since it has found CustomerID to be unique for each row, it has decided to tag it as “ID” instead of the generalized feature type “numeric”. By inserting a feature types tool, you can enrich the data and increase its effectiveness when used in an analytical model.