**1-4– Cleaning**

Cleaning is a necessary step wherein the data is transformed in order to get meaningful results. Uncleansed Quantitative fields may contain nulls that can significantly affect the outcome of predictive models and analysis. To add to that, some algorithms does not support datasets with missing values. Null or missing rows can be due to corrupt data, failure to load information or incomplete extraction. To handle this problem, we can utilize the imputation tool in Alteryx.

**The Imputation Tool replaces null or specified value of one or more numeric data fields with another specified value or a derived value.**

A common use of the imputation tool is replacing null values on numeric fields. For our first example, we are going to impute the null values of this Iris dataset with a specified value of zero. As we can observe from the results windows data quality bar, the columns sepal length, sepal width, petal length, and petal width have yellow bars which indicates that some of their records contains null values. To replace them, navigate to the Preparation tool tab and drag the imputation tool to the canvas. In configuring the tool, first, we need to indicate which fields to impute. Only numeric fields will be listed so you need to convert them beforehand. Since all 4 columns needs imputation, we will leave this selection as is. The next option lets you set the incoming value to replace. You can either choose Null or type a specific value that you want to be changed. We will set this to Null to impute the missing values. Next, you need to set the value that will replace the original. **The Imputation tool lets you choose from 4 choices, Average, Median, Mode or User specified value.** Choose “User specified value” then type “0” on the textbox. The last 2 configurations are optional. Enabling “Include imputed value indicator field” adds a new field for each imputed column that specifies whether or not a value has been imputed. The next option ”Output imputed values as a separate field” retains the original field and creates a new field with the imputed value. We will leave the 2 unchecked then run the workflow. The results from the output anchor of the imputation tool now have a full green quality bar for all columns since the nulls were replaced with zeroes.

Next, lets try to replace nulls using a derived value. Drag a 2nd imputation tool to the canvas and place it below the first one. Connect it to the output anchor of the select tool. For the imputation options, we are still going to use all 4 numerical fields, and the incoming value to replace will still be Null. To replace null values, choose Average, then enable Output imputed values as a separate field so we can check which rows were added. Once done, run the workflow. From the output, a new field with suffix Imputed Value is appended. Let’s check how the nulls were imputed. For sepal length, all of the null values were replaced with the average value of 5.925. The average was from the total sum of all sepal length divided by the count of records that are not null, which is 142.2 divided by 24 non-null records. Next, lets try the median derived value. Change the configuration to replace the value to Median then run the workflow. On this output, alteryx calculated the imputed value by separating the higher half from the lower half, or simply put, the middle value. For Sepal length, its middle value was 6.05. Finally, change the replace value to mode then run the workflow. As we can see from the result, sepal length null values were replaced with 6.3 which is the mode, or the value that appears most often. Each derived value differs from each column, so the sepal width has its own mode. Using the derived values has its own pros & cons. **It is a better approach when the data size is small, and it can also prevent data loss from removal of rows or columns. On the downside, using either average, median or mode can add variance and bias to the data.**

**Auto Field Tool reads through all of the records of an input and set the field type to the smallest possible size relative to the data contained within the column. In addition, the tool correctly assigns a numeric field to a string data type where any record starts with zero and not a number.**

For this example, we are going to use this insurance csv dataset which has age, sex, bmi, number of children, smoker tag, region and insurance charges which are all V\_String datatype with size 254 since the columns were saved in a comma delimited text file. To correct the data types and reduce the size needed for each field, we will apply the auto field tool into the workflow. From the preparation tool tab, drag Auto Field and place it into the canvas. Connect it to the output anchor of the select tool. The Auto field tool has a single configuration option. You only need to specify the fields that will be autoconfigured. You can also use the All button to select all fields, and clear to uncheck all. We will retain the default selection where all columns are checked. One important note about this tool is that its resulting autoconfigured datatype and size is not automatically recognized downstream unless you run the workflow first. So, if you place a select tool after it, it will still read the original datatypes from the csv. But if you run the workflow, new data types will be assigned. As we can see from the output, age and children were converted to Byte since they have smaller integer values, while bmi and charges were changed into Double since they have decimal places and are composed of larger numbers. Sex, smoker and region were turned to strings with smaller sizes. The auto field tool optimizes workflows by reducing resource consumption which in turn improves the performance. But before placing this tool to a workflow, you will still need to consider where to appropriately use it and think of how it may affect the rest of the tools downstream.

The **Generate Rows** **tool** is used to create new rows of data at the record level. It follows a process to generate rows of data. That process consists of an initial expression (applied to record 1), then a loop expression is applied (such as an increment) that builds subsequent rows, based on a condition (true or false) that ultimately build rows until the condition is false when it terminates the loop.

Drag a generate rows tool from the preparation tool tab and place it into the canvas. The tool has an optional input anchor and a single output anchor. Since the input is optional, you can use this tool to generate its standalone dataset. In the tool’s configuration, the first 2 radio buttons let you either update an existing field or create a new field. The new field was selected by default since we did not use any inputs, we will also leave the default configuration as is then discuss how it affects the output. Below it is the “Initialization Expression” where you can input a value or an expression to start the creation of rows. Since 1 was indicated as the default initialization expression, the first record for the new field “RowCount” will be 1. After the initialization would be the “Condition Expression”. This should contain an expression that will be answered by either True or False. Additional fields will be generated and will only stop once the condition returns false. The default condition is set to “RowCount <= 10” which means new rowcount will be generated until it reaches the value 10 and stops there. It will not create the rowcount value 11. After the condition expression is the “Loop Expression” which is usually the increment value. In this default setting, every rowcount record is incremented by 1. Run the workflow to show the standalone dataset that the tool produced. As we can see from the output, 10 records in the field RowCount were generated. The record started with 1 and incremented by 1 until it reached the condition that it should be less than or equal to 10.

Next, lets try to apply the tool with an input dataset. We will create a new field called “dependents” and create rows based on the indicated number of children on the same insurance dataset we used for auto field earlier. So, if a customer has 3 children, it will generate 3 rows of dependents. Drag a new generate rows tool into the canvas and connect it to the output of the auto field tool. Set it to Create a New Field named “Dependents” with datatype Int32. Set the Initialization Expression to “0” since some customers might have no children. Next, click the ellipses button next to the Condition Expression to open the Expression Editor. Type the expression “[Dependents]<=[children]”. The loop will only stop if the Dependents value is greater than equal to that of the children column. For the Loop Expression type “[Dependents] + 1” so each dependent increases by one on each new row. Run the workflow to view the results. Our first customer only has 1 record since it does not have any children, but the 2nd record male 18, has generate a single duplicate of its row since it has 1 child thus having 1 dependent. You can also observe the incrementation of the dependents field. Male 28 has 3 children so it generated 3 additional records aside from its original row. The dependents also started from the value 0 and stopped when it reached 3.