**4-3– Machine Learning 3** 

In the beginner course, we described how to blend data by using the join, append, and union tool. This helps us analyze and combine data from different types of sources which in turn unlocks more information about our records. The same can also be said for features. Most of the time, the features you want for your model are not just located on a single file or table. Aside from taking in several data sources, you also need to figure out if new features can be created from your existing columns and add them to the stream. The Build Features tool would be the solution to both of these problems.

**Build Features** is used to create features and establish relationships between data in separate tables. The tool uses “primitives” to build features from the data you provide. This assists in formatting your data in a way that the machine learning model can analyze, increasing the likelihood that the machine learning model will find meaningful patterns. It helps by uncovering variables you might not have considered (or prioritized). For example, you can transform a column for “date of birth” into new features like “age” or “birthday month.”

We are going to use the build feature tool to combine the data of these 2 csv files and check for features that will be used for a predictive model. The first file contains customer information such as the Name, Age, Occupation, annual income, and the number of loans and credit cards. This file also has the field to be predicted called “Credit Score” which indicates whether the customer has Good, Standard, or Poor Credit. The 2nd file contains the customer’s bank transactions and credit-related information which is aggregated by month. We will use these 2 files to segregate the customers into credit score brackets. In order to use the Build Feature tool, we are required to apply the correct data types first and feed them into the feature types tool to apply the correct features. The Feature Types tool is a required input to the Build Features tool for effective feature engineering. From the Preparation tool palette, drag an autofield tool and connect it to the Customer.csv input data. Next, drag another autofield tool but connect this one to the Transaction.csv input data tool. Once the autofield tools are inserted, navigate to the Machine Learning tab then drag feature types tool to the canvas. Connect it after the autofield tool. Do the same for the transaction data stream. Run the workflow to view the output feature types for both data streams.

On the configuration window of the Feature Types Tool, we can see that most fields were now converted to categorical and numeric features, while long strings such as Type\_of\_Loan of the customer data stream were recognized as “Text”. We will leave most of these to the autodetected types except for the ID columns since we will use them on the build features configuration. On this customer data stream, set the change type drop-down for Customer\_ID to “ID”. Then for the 2nd feature tool in the transactions data stream, change both Customer\_ID and ID into the feature type “ID”. With the feature types corrected, we can now link these 2 data streams to the Build Features Tool.

From the Machine Learning tool palette, drag a Build Features Tool into the canvas. This tool can take in multiple inputs and has a single output anchor. Connect the output anchor of both the customer and transactions data stream to the tool. To make data stream identification easier later, we will rename the wired connections. Click the first connection, then on the configuration window, rename it to “customer”. Next, click the 2nd connection, and rename it to “transactions”. The new wire names should be seen as labels on the connections once done. If you go to the configurations window of the Build Features tool under “Manage Relationships” tab, the 2 tables will also have the names “customer” and “transactions”. To configure the Build Features tool, first, select the **target table.** This is the primary table whose columns define the relationships between all of the tables. Since our target column “Credit Score” is on the customer level, we will choose “customer” as the target table.

The next step would be to set the primary keys for the tables. You are only required to place a primary key for the target table, in this case, “customers” but it is still beneficial to add primary keys to other tables if it is available as it could help you create relationships. For Customer table, set the primary key to “Customer\_ID”. As for the transactions table, set it to “ID”. After the Primary Key, the next configuration is to set the table relationships. This portion is similar to that of the Join tool where you need to specify the join fields from both left and right table, except for the fact that in this tool we need to specify which table is the “parent” as well as the “child”. The **parent** **table** would usually be the target table since the output will rely on the target table’s aggregation level. Set the Parent Table to “customer” with the key “Customer\_ID”. Since we only have 2 tables, the other one would automatically be the child table. Set “transactions” as the child table and use the key “Customer\_ID” to match it to the parent’s data. The **Child** **table** features will be aggregated to match the parent level table. In this case, our primitives and features on the transaction table will be aggregated to the customer level. When configuring relationships, make sure that the ID columns being used are the same data type as mismatched data types will cause errors.

Now that we have the keys, tables, and relationships in check. Let’s move to the tab “Manage Primitives”. At a high level, **primitives** are functions applied to raw data that help build features from it. Those functions can either aggregate or transform the data to build features. We can select up to five primitives. (The limit is intended to prevent the Build Feature tool from generating too many features, which could negatively impact performance.) In this example, let's select “Max”, “Sum”, “Mean” and “Mode” which are all under the “aggregation“ category. Alteryx will not apply each primitive aggregation to each feature instead, it will choose the columns or features that will benefit the most with this procedure. Add a select tool after the build features tool so we can easily check and deselect the features that we can use for our model. Then run the workflow.

From the select tool, we can see that 34 new features from the transaction table were found by the Build Features Tool. The first 13 fields from Customer\_ID up to Credit\_Score were the original features from the customer table which were not aggregated since they were from the target table. The 34 features were aggregated to the customer level using either Max, Sum, Mean or Mode.

In the select tool, we are going to remove the features that we will not need for our model. Deselect the following:

* Customer\_ID
* Name
* SSN
* Type\_of\_Loan
* MAX(transactions.Total\_EMI\_per\_month)
* MAX(transactions.Amount\_invested\_monthly)
* SUM(transactions.Total\_EMI\_per\_month)
* SUM(transactions.Credit\_Utilization\_Ratio)
* MEAN(transactions.Total\_EMI\_per\_month)
* MEAN(transactions.Credit\_Utilization\_Ratio)
* MODE(transactions.Credit\_History\_Age)
* MODE(transactions.Month)

These features will not be needed as most of them are unique by customers while some are not simply related to the Credit\_Score. Once done, add an autoML tool after the select tool.

The **AutoML** is used as part of a machine learning pipeline to automatically build a model of your data. The tool provides several algorithms for both classification and regression methods, then evaluates the algorithms against each other before outputting a trained model.

On the configurations window of the tool, you simply need to set the target field or the column to predict, and the machine learning method that you will apply. For the target, set it to “Credit\_Score”. Upon selecting the target, Alteryx has automatically chosen the classification method that is best suited for the column. Since Credit\_Score contains 3 categorical tags; Good, Standard and Bad. It has chosen the method “Classification”. You can still change this to the other method if you that is the better choice.

In the advanced parameters drop down menu, it presents us with different options that you can configure to change how the tool evaluates algorithms. The **objective function** is what you want to use to determine the ranking of models the tool evaluates. Objective functions are measures you can use to determine how optimal a model is for your use-case. We will leave this to the default “log loss”. Next, you can select the types of **algorithms** you want to evaluate as part of the automodeling process. You can select more than 1 option. For this classification autoML we can choose from Random Forest, XGBoost, Linear and CatBoost. Lets keep Random Forest and XGBoost. Do take note that the more types you select, the longer the workflow takes to run. Next, set the **Max Model Pipelines to Evaluate.** Here you can enter the number of pipelines you want the AutoML tool to build, using the chosen algorithms, and then evaluate them, based on the objective function. You can evaluate 1–50 pipelines, any numbers higher than that will cause longer run times. Let’s set this to 10. Finally, you can choose to **Enable Data Checks.** When checked, alteryx will use **EvalML** which provides data checks to help guide you in achieving the highest performing model. Run the workflow and check the output once all configurations are done.

The output is a copy of the model which you can save or use with a predict tool to use the test data. The accuracy of the autoML model was 74% with 75% precision.