**4-5– Machine Learning 5**

in the previous lesson, we’ve mentioned the machine learning pipeline that is automatically created upon applying the model created using the assisted modeling tool. According to alteryx “**a modeling pipeline** refers to the steps needed to get your data from a raw input data format to a fully trained and useable model, including any encoding or transformation performed on the data to get it into a useable format.” You can think of each tool as a puzzle piece. Each tool contributes to building the whole picture, from cleaning and selecting features up to fitting the model to the data. The most crucial part of a machine learning pipeline is training the model itself. There are 2 tools in the intelligence suite that helps train models; the Regression and Classification Tools. So how will you know which tool to add to your pipeline?

To put it simply, the **regression** tool is used if your target variable (or column to predict) is numeric, but if it is a categorical column, you should use the **classification** tool. For example, if you wanted to predict the number of days until a customer possibly churns, you should apply the regression tool. In other cases, if you wanted to know whether a patient probably has dementia or not, you should apply the classification tool. For this lesson, we are going to show how to apply the regression and classification tool on 2 different machine learning pipelines in alteryx.

For our first example, we are going to create a linear regression machine learning pipeline using this car price dataset. We are going to use the linear regression algorithm in order to predict the price of each car using the 25 inde pendent variables given in the file. So as to prepare the data, we have already inserted an autofield tool to convert it to the correct data type and separated a training stream and a testing stream using the create sample tool. We also used the data health tool to automatically find outliers which were then removed from the training data stream and unioned into the testing data stream. To start the pipeline, drag a feature types tool, and connect it to the select tool of the training data stream. We will leave all of the feature types to autodetect except for car\_ID. Change the feature type of Car ID to “ID”. Next, insert an assisted modeling tool. We will need this tool so that the rest of the pipeline can recognize that we are going to predict the “price” and apply regression accordingly. Under the assisted modeling configuration, choose the option “Expert” since we are building the model without assistance. Once done, proceed with the next tool. Drag a transform tool and connect it after the assisted modeling tool. Set its transformer as “One Hot Encoding”. Some of our features are categorical so this would be a necessary step. Under One Hot Encoding, click hide un-encodable features and make sure that the following columns are selected; carname, fueltype, aspiration, doornumber, carbody, drivewheel, and enginelocation. Once all configurations are done, we can now insert a regression tool. From the machine learning toolset, drag a regression tool and connect it after the transform tool.

The **Regression tool** is used as part of a machine-learning pipeline to identify a trend. The tool provides several algorithms you can use to train a model. The tool also allows you to tune a model using many parameters. In configuring the tool, first, we need to set the algorithm. There are 3 algorithms currently available. **Linear Regression** is the most basic of the regression algorithms and is the easiest and fastest to train. It is great to use if the dataset is normally distributed and its features are mutually independent. **Decision Tree** is great for non-linear solutions and is best to use when there is a large number of categorical independent features. **Random Forest** has a higher training time and is suitable for large datasets where interpretability is not a major concern.

We will choose Linear Regression as the algorithm for this workflow. Once selected, you can see 2 general parameters that have an on-off switch for Fit Intercept and Normalize, do take note that these parameters change depending on the algorithm you’ve chosen. If switched on, **Fit Intercept** will calculate the intercept or “constant” for the linear regression. the intercept is the expected mean value of y where x equals 0. **Normalize** when switched on will normalize the targets. It adjusts your targets in such a way that you can compare them on a common scale with other data, which may help you identify associations in your data. It is recommended to keep both of these parameters ON. Next, we can now insert a fit tool to train the model to the data. Drag a fit tool and connect it after the regression tool. With the model done, insert a predict tool and connect its M anchor after the fit tool. Then connect its D anchor to the select tool of the testing data stream. Finally, add a browse tool after the predict tool, then run the workflow. From the output, the price\_predicted is appended as the last column. You can now compare it to the price column to see the accuracy of the model and check how big of a difference the predicted price value was. In this output, car\_ID “4” was originally priced as 13,950, but the model predicted its price as 11838.81 which is 16% lower than the intended price.

For our 2nd example, we going to continue working with the bank marketing dataset that we’ve used in lesson 4-2 to predict if new clients will subscribe to a time deposit. The target will be the column “y” which has the tag of either yes or no. Same as our first example, we’ve already prepared the data by using an autofield tool to convert datatypes and separated the outliers on the training dataset. We are going to use the assisted modeling option in order to build the 2nd pipeline. Insert an assisted modeling tool and connect it to the training data stream. Run the workflow to feed the data into the tool. Once done, open the assisted modeling window.

For step 1, set the target to “y”. Then, choose “classification” as the algorithm. In step 2, choose “step by step”. In step 3, alteryx recommended dropping 3 features; pdays, previous, and poutcome since they all have unary values. Make sure that the rest of the features are auto-tagged to the correct data type then hit next. Step 4 has no configuration since we don’t have missing values on the dataset, proceed to the next step to continue. For step 5, alteryx has deemed 2 features to be unnecessary. “Default” and “loan” were recognized to be weakly associated with the target, so uncheck the 2 features before hitting next. For the last step of choosing the algorithms to run, let's focus on logistic regression and XGBoost, then uncheck the rest. Run the selected algorithms to check how they fare. According to the leaderboard, XGBoost model’s accuracy is higher by .2% than logistic regression. So let’s choose this model and include it in the workflow. Click the box for XGBoost and press “add models and continue to workflow”. We are already familiar with the rest of the tools used in this pipeline aside from the classification tool.

The **Classification** tool is used as part of a machine-learning pipeline to identify what category a target belongs to. The tool provides several algorithms you can use to train a model. The tool also allows you to tune a model using many parameters.

Since we’ve chosen XGBoost during the assisted modeling stage, its algorithm is also selected automatically. **XGBoost** is a scalable, distributed gradient-boosted decision tree. This model is preferred for non-linear datasets and works well with either binary classification, multi, or even regression. On the XGBoost general parameters, we have 4 configurations that we can tweak. **Max Depth** is the longest path from a root to a leaf for each tree in the forest. **Learning Rate** is the rate at which the algorithm lets new info override old info. **Number of Estimators** which is the number of trees you want to create as part of the forest. And **Gamma** sets the loss reduction required for a decision tree to split into a new node. You can type in a new value for each parameter if you want to increase or decrease its value. Aside from the general parameters, other advanced configurations can also be tweaked if you click the advanced parameters dropdown. As always, all of these parameters will change depending on the chosen algorithm.

Connect the testing stream to the D anchor of the predict tool then add a browse tool after. Once done, run the workflow. Since we’ve created a classification pipeline, more than 1 column is appended to the output. Y\_predicted states if the customer will sign up for a time deposit or not. Y\_no shows the percentage of how likely they will not sign up and on the flip side, Y\_yes shows how likely they are to sign up.