**Guided ML and AutoAI**

Elena Lowery, WW Team, Data Science and IA

A close up of a logo

Description automatically generated

Lab: AutoML in Watson Studio

*August 18, 2019*

*Author: Elena Lowery elowery@us.ibm.com*

**Table of contents**

Contents

[Overview 1](#_Toc60650672)

[Required software, access, and files 2](#_Toc60650673)

[Required skills 3](#_Toc60650674)

[Part 1: AutoML Pipeline Generation 4](#_Toc60650675)

[Step 1: Create an AutoML Model 5](#_Toc60650676)

[Part 2: Guided ML with Modeler Flows 27](#_Toc60650677)

[Auto Modeling in Modeler Flows 28](#_Toc60650678)

[AutoML and Guided ML Summary 49](#_Toc60650679)

[Part 3: Advanced Guided ML and Integration with AutoML 50](#_Toc60650680)

[Flow Import 50](#_Toc60650681)

[Running Flows 52](#_Toc60650682)

[Sample flows 53](#_Toc60650683)

[Healthcare Cost Prediction 53](#_Toc60650684)

[Customer Segmentation 55](#_Toc60650685)

[Predicting Campaign Response with Structured and Unstructured Data 57](#_Toc60650686)

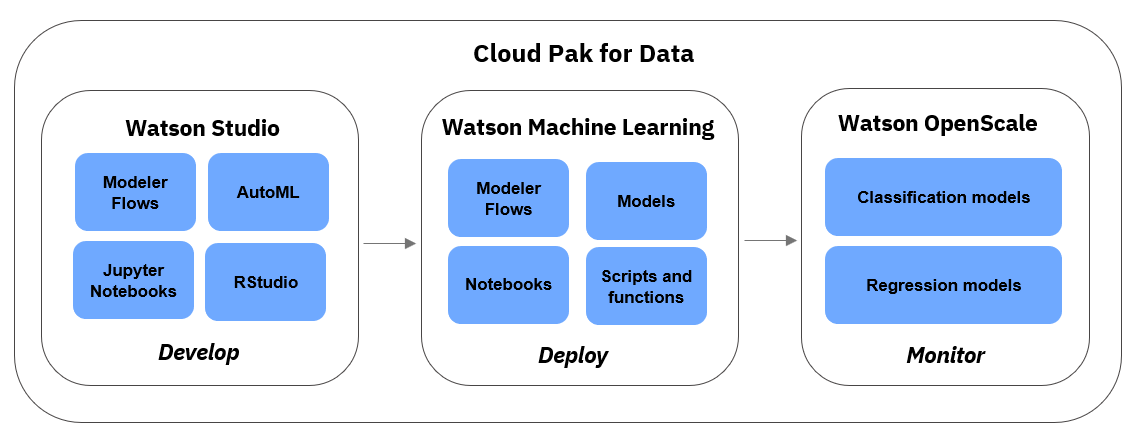
[Additional Resources 60](#_Toc60650687)

# Overview

In this lab you will learn how to use Auto and Guided Machine Learning in **Cloud Pak for Data as a Service (CPDaaS)**. The main benefits of using Guided ML and Auto ML are improving data scientists’ productivity and quality of models. Instructions in this document are written for **CPDaaS**, but you can also use a **Cloud Pak for Data** instance to complete this lab.

It is worth pointing out at this point that both Guided and Auto ML are concepts indicating the amount of technology support that the analyst receives when performing machine learning tasks, and the capabilities within the Cloud Pak for Data environment which support these concepts are AutoAI (Auto ML) and Modeler Flows (Guided ML).

Cloud Pak for Data includes over 30 services (applications) which can be used for data science, data management, and AI projects. Guided ML (Modeler Flows) and Auto ML (AutoAI) capabilities are a part of **Watson Studio** service. In addition to these tools, Watson Studio includes open source IDEs – Jupyter Notebook and RStudio. Models developed in **Watson Studio** can be deployed in **Watson Machine Learning** and monitored in **Watson OpenScale**.



The sections in the lab are organized by skill level and use case:

* **Part 1 and Part 2:** beginner
* **Part 3:** intermediate (Guided ML).

If you have a working knowledge of Modeler flows, you can skip **Part 2** and work on **Part 3.**

For more information about Guided ML and AutoAI see the **Additional Information** section.

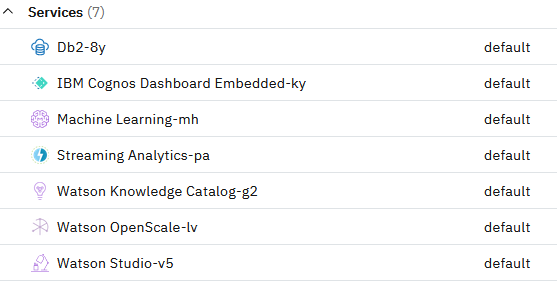
# Required software, access, and files

1. To complete this lab, you will need a **Cloud Pak for Data as a Service** (**CPDaaS**) account: <https://dataplatform.cloud.ibm.com>

* If you don’t have a CPDaaS account, use the same URL to sign up for a free trial. The account will be activated in approximately 5 minutes.

1. If you already have an **IBM Cloud** account, make sure that you provisioned the required services – **Watson Studio** and **Watson Machine Learning.**

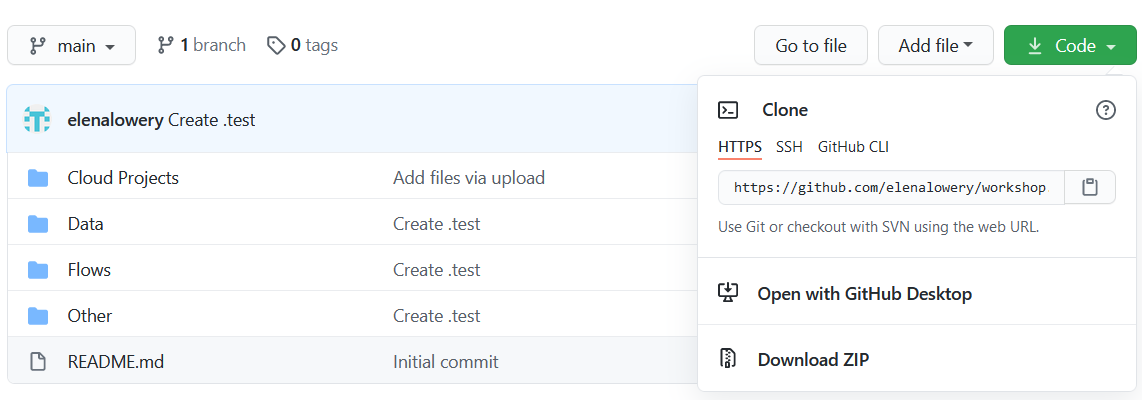
* Navigate to your *Services Dashboard* in your **IBM Cloud** dashboard: <https://cloud.ibm.com/login>
* Check if the mentioned services are displayed under **Services**. If not, search for the services in the **Catalog** and add them.



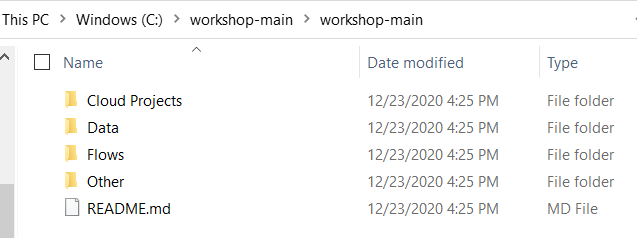
1. You will also need files from this *GitHub* page: <https://github.com/elenalowery/workshop>

*Note: If you downloaded this file for the* ***ModelOps-Deployment*** *lab, you don’t need to download it again.*

* In the Github repo page, click **Code** and select **Download ZIP**.



* Unzip the downloaded file until you get to this directory structure:



In the lab we will refer to this folder as the git repo folder.

# Required skills

We recommend that users who work through this lab:

* **Parts 1 and 2:** understand data science lifecycle and steps in creating a model
* **Part 3:** Have a working knowledge of Modeler flows.

# Part 1: AutoML Pipeline Generation

**Watson Studio** provides several AutoML capabilities. In this section you will learn how to use *AutoAI* to automatically build *classification* models. Classification models are used for data science use cases in which we need to predict the *class* of data. The class of data is a specific *value* that we want to predict.

Here are a few examples of use cases that can be solved with classification models:

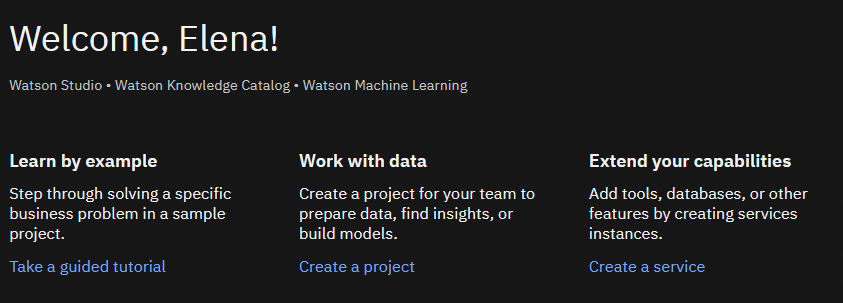
|  |  |  |
| --- | --- | --- |
| **Use case** | **What are we predicting?** | **Sample values (classes)** |
| Customer churn | Will the customer churn? | *Yes* or *no* |
| Credit card fraud detection | Is this a fraudulent transaction? | *Yes* or *no* |
| Predictive maintenance | What is the cause of mechanical failure? | *Overheating, pressure, friction* |
| Customer promotions | Which promotion should we send to the customer? | *Free shipping, buy 1 get 1 free, 20 percent discount* |
| Healthcare | Is this customer at risk for readmission? | *Yes* or *no* |

As you can see from the examples, classification models can be used for many use cases in various industries. In order to create a classification model, we need to have *historical* (*training*) data with the values we are trying to predict. For example, if we want to predict credit card fraud, we need to have records of transactions that were fraudulent and non-fraudulent. Models are created from *statistical algorithms*, and a data scientist can choose from over a dozen of available classification algorithms.

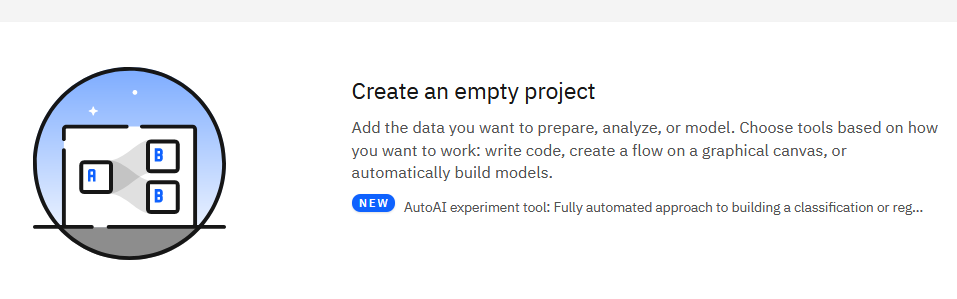
If you would like to learn more about classification models, we recommend that you take an **IBM Data Science Course** from Coursera. Please see the **Additional Information** section.

## Step 1: Create an AutoML Model

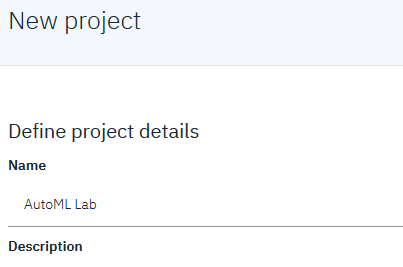
1. Log in to **CPDaaS**.
2. On the *Welcome* screen click **Create a project**



1. Select **Create an empty project**.

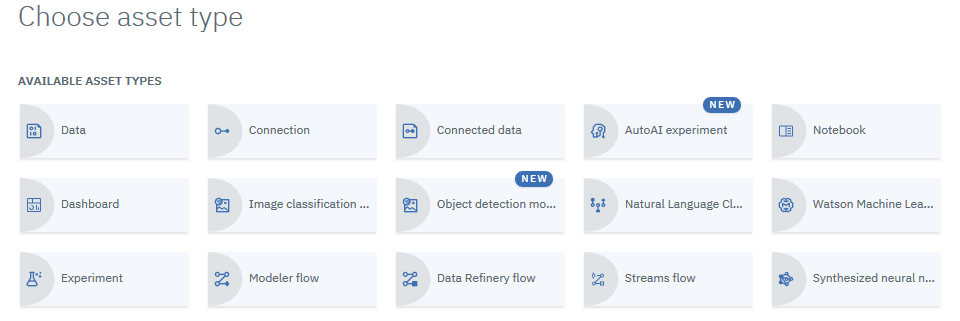


1. Name the project **AutoML Lab** and click **Create**.

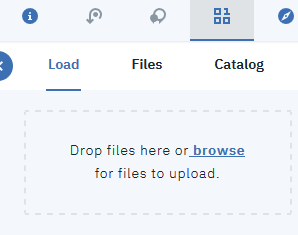


1. Next, we will load data assets that will be used for modeling. The .csv files are located in the */data* directory of the file that you downloaded from GitHub.
   1. In the **Project** view select **Add to Project -> Data**.



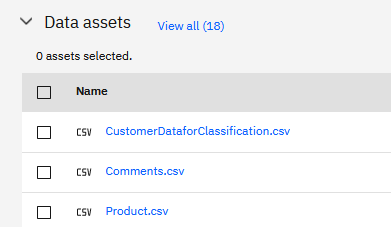


* 1. Click **Browse** and upload all .csv files from the **/data** directory.



* 1. Click on the **Assets** tab. All uploaded files will be displayed under **Data Assets**.

*Note: your list of files may look different from this screenshot.*

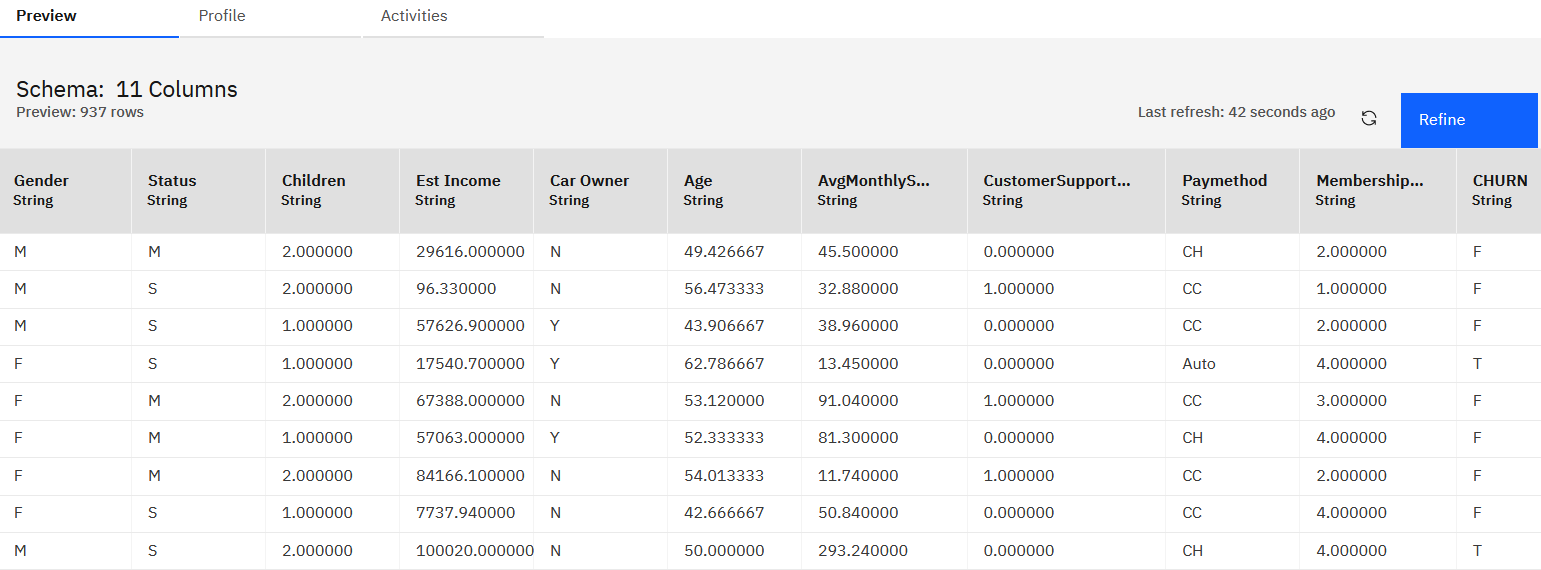


1. For our first *AutoAI* model we will use the *customer\_churn.csv* file. Click on the file to preview the data.

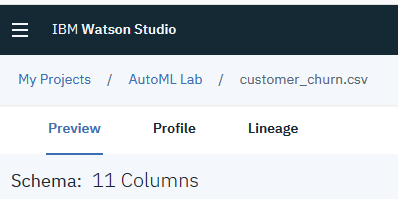
The file contains demographic and historical transaction information for each customer. It also contains the churn flag. If the flag value is set to *T (True),* that means that the customer has churned.

Using AutoAI we will build a model that will predict churn for new customers. A marketing department can use this information to proactively prevent churn.

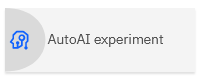
In data science terminology the columns that we use to predict churn are called *features*, and the column that we are trying to predict is called *target*. In our dataset the target column name is *CHURN*.



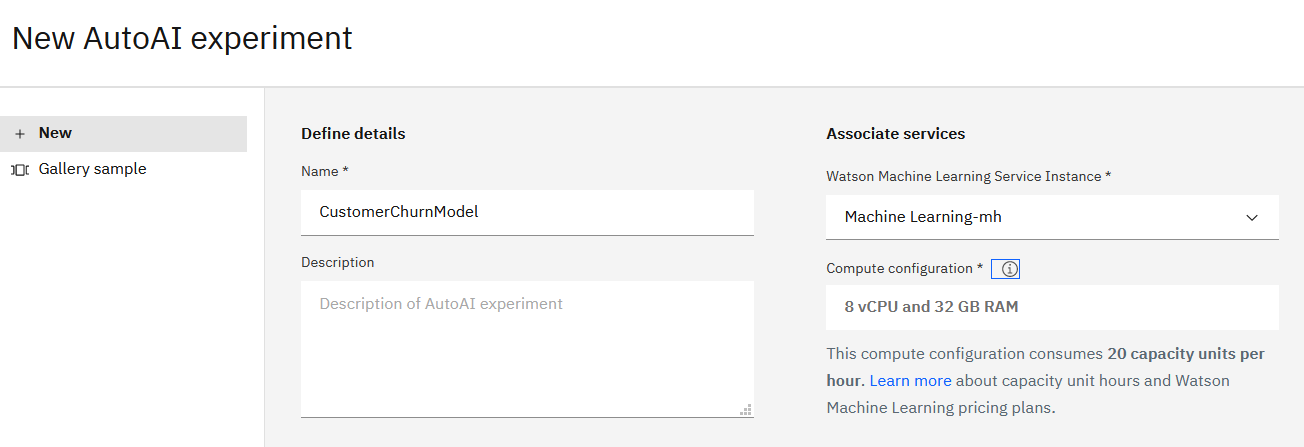
1. Click on **Auto ML Lab** to return to the **Project** view.



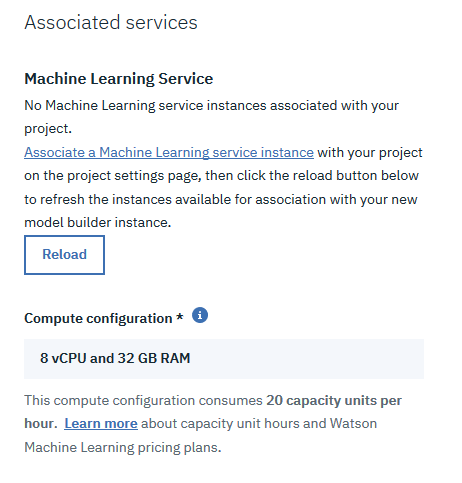
1. In the **Project** view select **Add to Project -> AutoAI Experiment.**



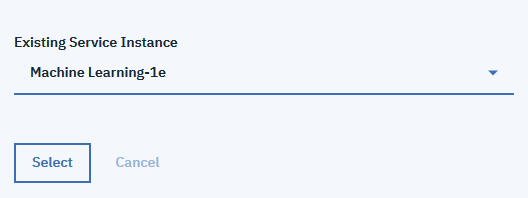
1. Name the experiment *CustomerChurnModel.*



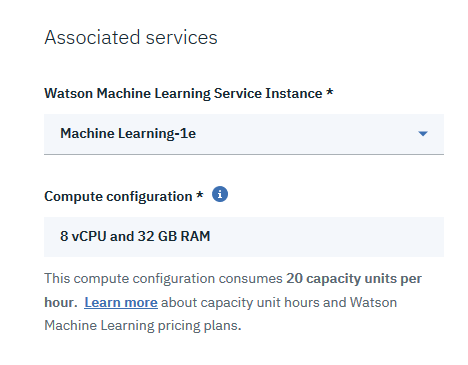
1. Next, we need to add the **WML** service to our project.
   1. Click the **Associate…** link.



* 1. Select the **WML Service**

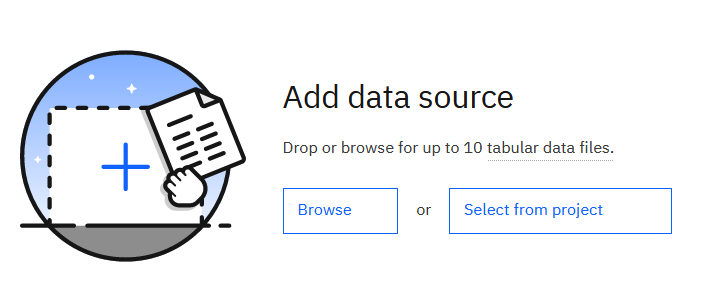


* 1. Click the **Reload** button to make sure the that WML Service has been associated with the project.

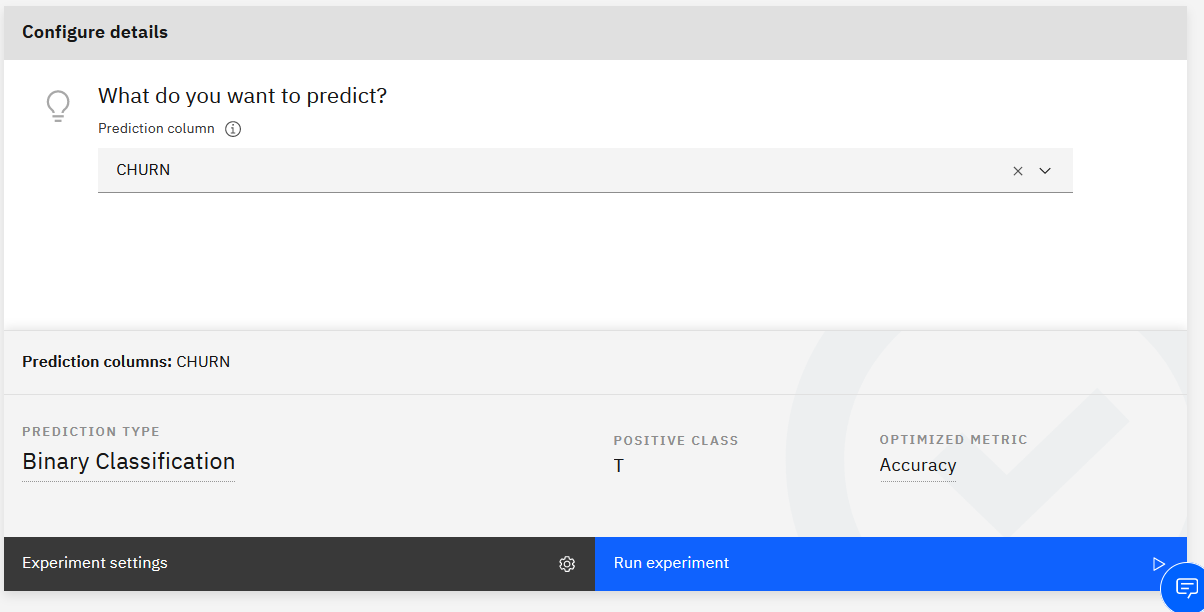


* 1. Click **Create** to create the AutoAI asset.

1. On the **Add Data Source** screen click **Select from project**



1. Select *customer\_churn.csv*. Then click **Select asset**.
2. In the *Configure Details* dialog select *CHURN* as the *Prediction column*.



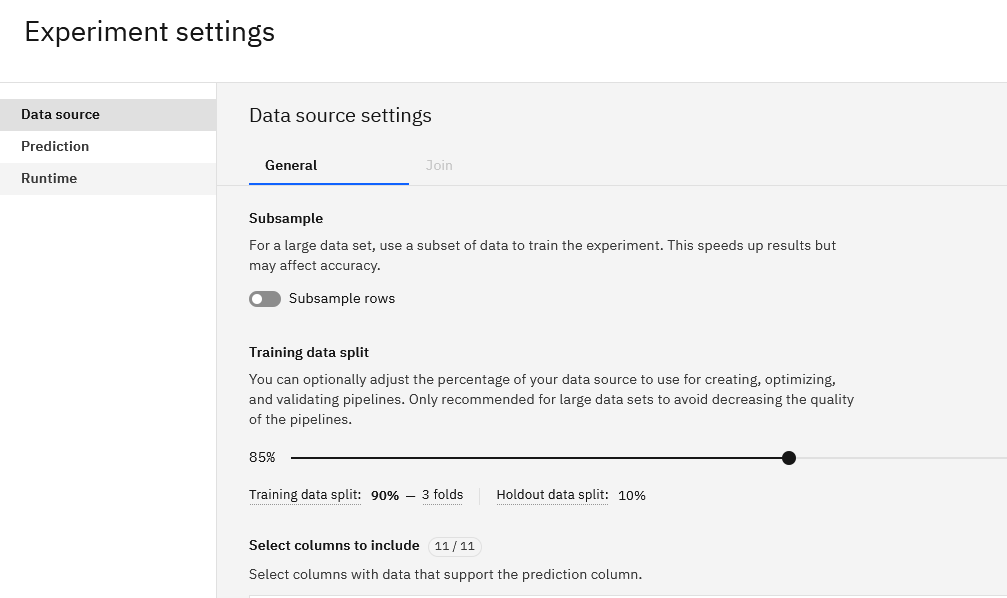
When you clicked on the column to predict, several values were selected on the bottom of the screen. Let’s review them.

* *Binary Classification* is the type of classification that’s the best fit our target variable, which has *2 values*, T (True) and F (False) and the “T” value has been set as the POSITIVE CLASS, which is the group of interest.
* *Accuracy* is one of the model evaluation metrics. Since there are several evaluation metrics, we need to configure *AutoAI* to optimize the model for the specific metric. For classification problems, *Accuracy* is often used as the primary evaluation metric.

Select a different column name, for example, *AvgMonthlySpend*. Notice that the *Prediction type* has changed to *Regression* and the *Optimized metric* has changed to *RMSE*. AutoML changed these values because *AvgMonthlySpend* is a continuous (numeric) variable.



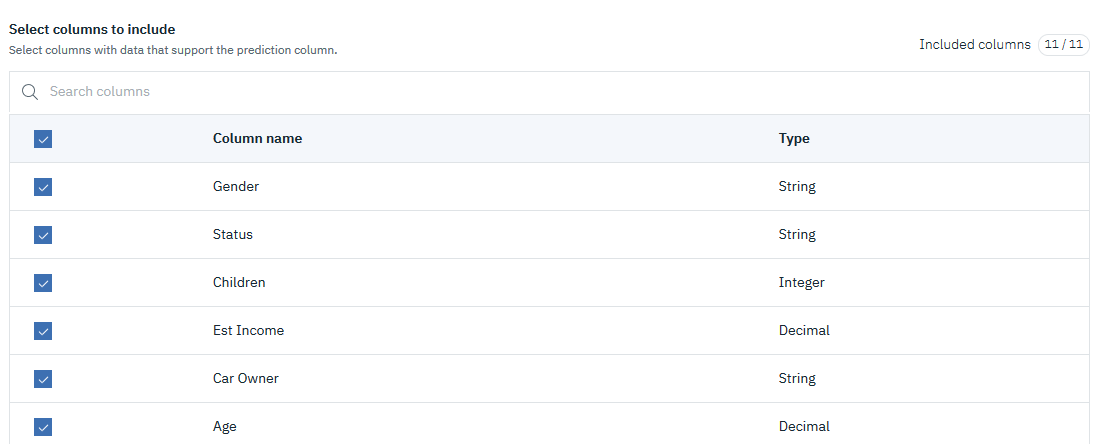
1. Select *CHURN* again and click **Experiment Settings**.



When creating a model, the data is split into training and testing data. Testing data is also called “holdout data”. By default, 10% of data is used for testing. Let’s leave the default value.

*Folds* are parts of the input dataset that will alternatively be used for training and testing. By specifying 3 folds, we are creating 3 parts in the dataset. We will use the default value of folds in this lab.

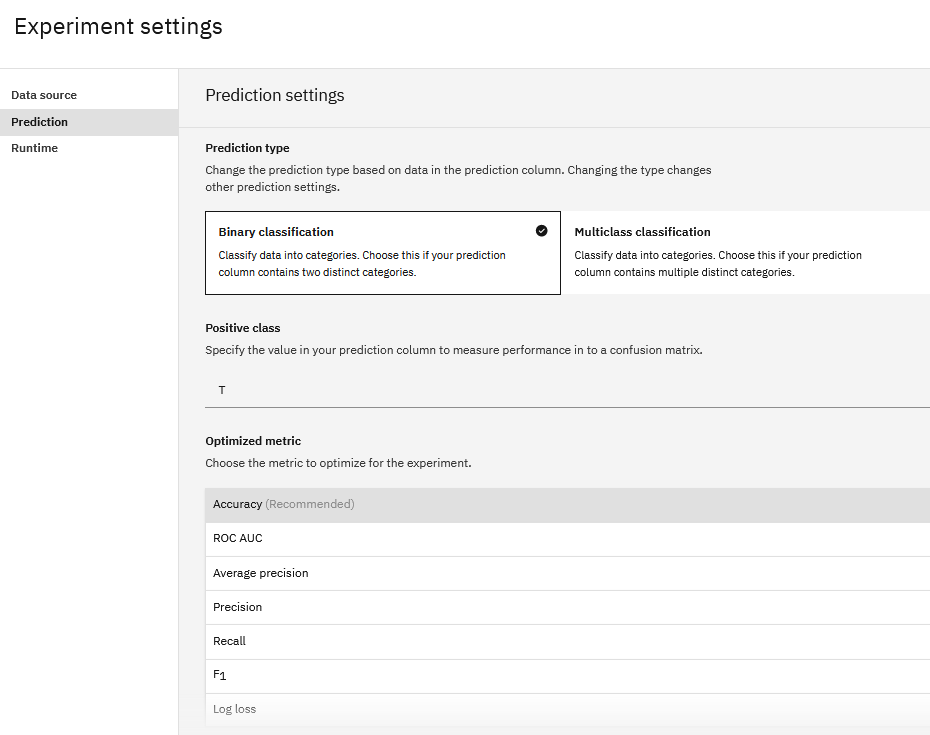
When needed, we can also deselect fields to be used for training. In this lab we will use all fields.



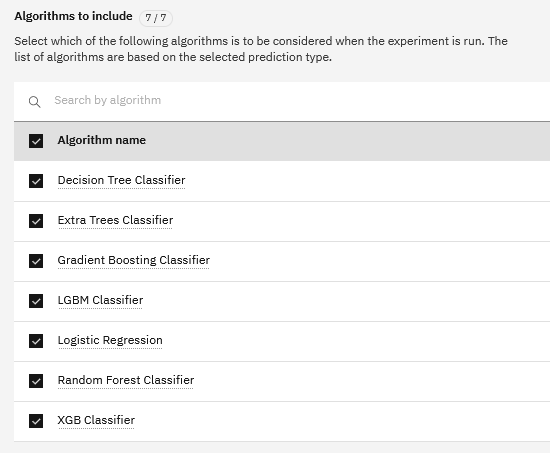
1. Click on the **Prediction** tab.

On this tab we can make changes in the algorithm type, the positive class, the model evaluation metric for which the model can be optimized, and the list of algorithms that will be considered for building the final model.

By default, *AutoAI* already selected the values that are the best fit for the input dataset and the target variable. The **Prediction type** is set to *Binary classification* (we need to predict *T* or *F*) and the metric for selecting the best model is *Accuracy*.

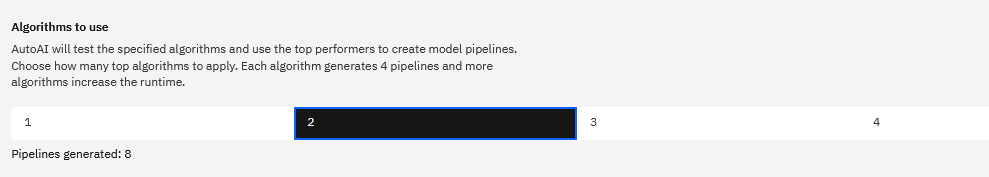


We can deselect the algorithms that will be considered for building the model. For this lab we will use the default settings.



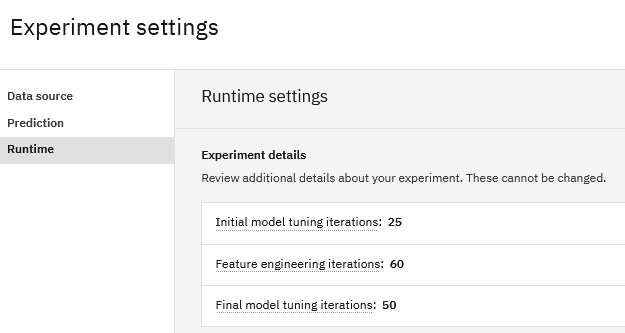
Below the algorithms you can specify how many pipelines (models) you would like to build. By default, 2 top performers out of the 7 available algorithms will be used to build a total of 8 pipelines (4 for each algorithm). We will explain why 4 pipelines are built later in the lab.

For this lab we will leave the default value.



1. Click on the **Runtime** tab.

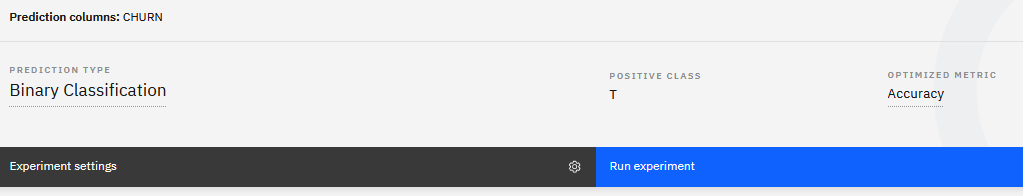
On this tab we see a list of settings which are being applied to the experiment. It is not possible to change these settings, but they are displayed so that the user can see the configuration that will be used for building the model.



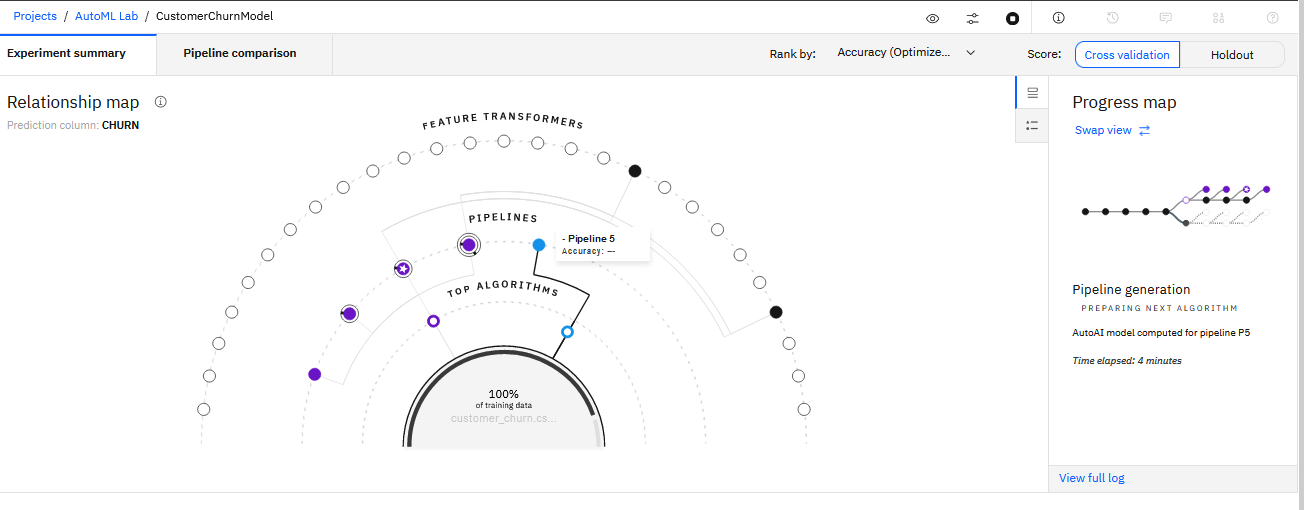
Since we don’t need to make any changes, click **Cancel** to close **Experiment Settings**.

Now we are ready to build the model.

1. Click **Run experiment**.



AutoAI displays the build status.



During model building phase we can switch between two types of output using the **Swap View** button:

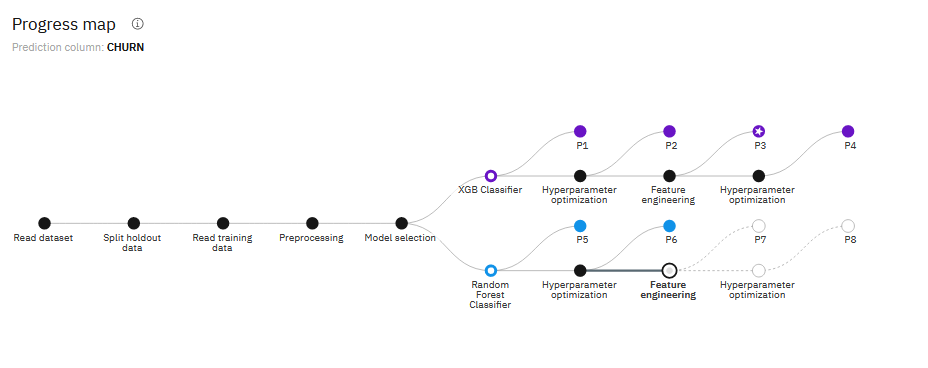
**Progress Map**: This shows the current stage of processing where *P1, P2,*etc. refer to the 8 pipelines that will be built.

A *pipeline* is a term used to describe various steps in creating a model. A pipeline always contains the generated model. It can also contain steps to generate new columns of data to use as input variables (*features*) and steps to tune the model (*hyperparameter optimization*).

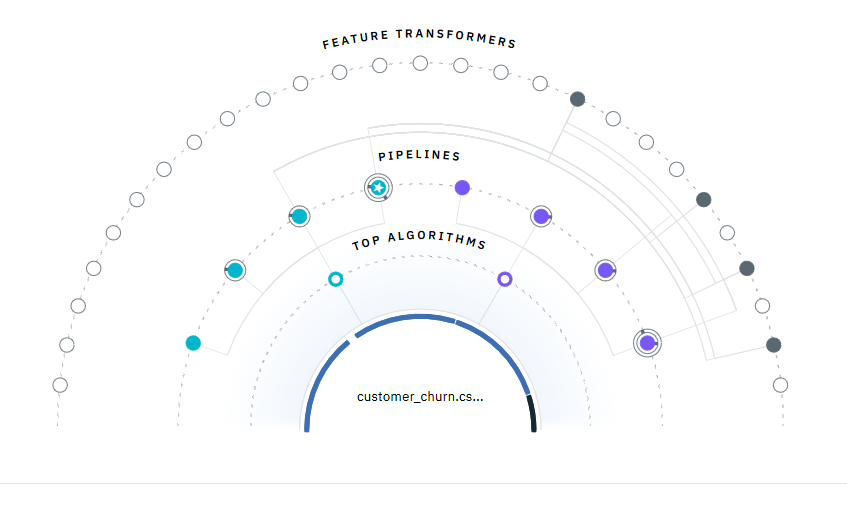
* As we can see in the graph, *P1* and *P5*build a pipeline using the *estimators* (algorithms) which AutoAI has determined to be the best fit for the data, in this case *XGB classifier* and *Random Forest*
* *P2*and*P6* perform hyperparameter optimization (HPO). Hyperparameters are “settings” (parameters) that are specific to each algorithm. Hyperparameter optimization means that we are building the model using different settings in the algorithm. *AutoAI* tries several combinations and determines the combination which will produce the best result.
* *P3*and*P7* perform feature engineering (derives new features) and build a model with these features.
* *P4* and *P8* perform hyperparameter optimization for the model that uses the derived features.

In summary, for each algorithm AutoML creates 4 pipelines:

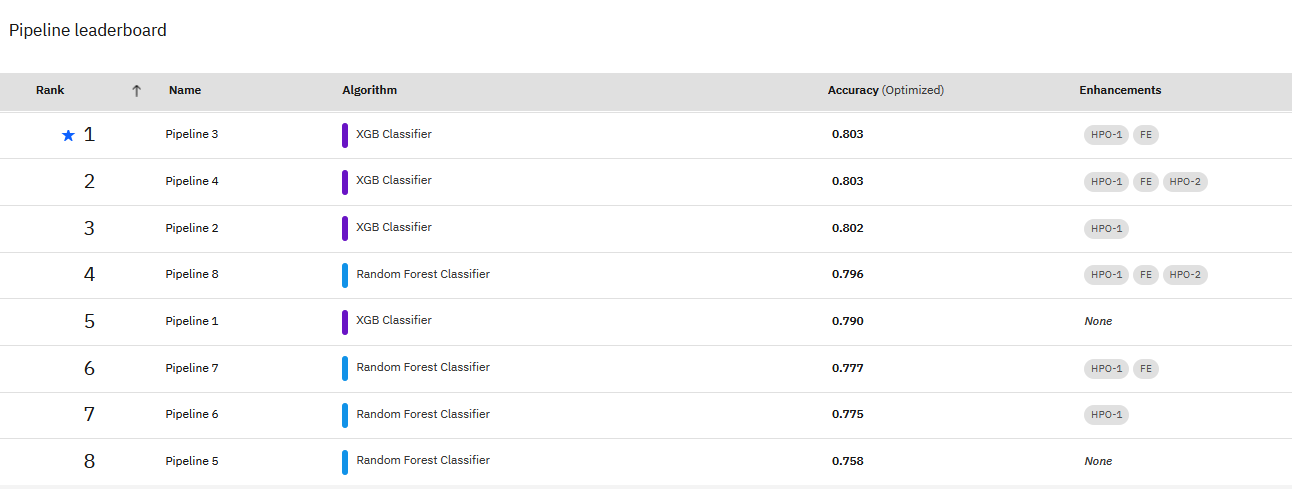
* Pipeline 1 contains just the model
* Pipeline 2 contains the algorithm and performs HPO
* Pipeline 3 generates additional features and builds a model that includes these features
* Pipeline 4 generates the same features as pipeline 3 and performs HPO.



**Relationship Map**: This shows which transformations have been applied to the various pipelines. The chart is interactive – if you position the cursor over one of the pipelines, you can review the map that’s associated with it.



1. After pipelines have finished building, *AutoAI* will display the results similar to this screenshot:



1. Let’s review each pipeline. We’ll start with *Pipeline 1*, which uses the default features.

Click on *Pipeline 1*.

In the **Pipeline** view we can find various information about the model. The right side of the screen describes evaluations metrics that are used by data scientists.

The **Holdout Score** is the score for the *test* dataset that was not used for modeling (by default, 10 percent of the data was left out for testing). The **Cross Validation Score** is the score when the cross validation technique was used. Cross validation uses different parts of the same dataset first for training, then for testing.

An experienced data scientist may look at a combination of evaluation metrics before determining if the model is ready for production, as these measures provide information on how the model performs in difference areas of “good-ness”.

For example:

* Precision tells you: “Out of all the records in the hold-out sample that the model predicted to be Churners (T), what percentage actually churned…”. For Pipeline 1, from all the records that the model predicted were churners, 0.774 (77.4%) were actual churners.
* Recall tells you: “Out of all the churners in the hold-out sample, what percentage can your model actually find?”. For Pipeline 1, the model managed to correctly identify 0.667 (67%) of the total churners in the hold-out sample.

Many data scientists use *Area Under ROC Curve.* On a high level, you can use the following guidelines for *Area Under ROC Curve*:

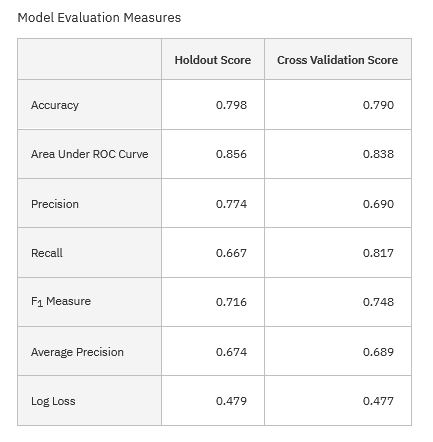
*.90-1 = excellent (A)*

*.80-.90 = good (B)*

*.70-.80 = fair (C)*

*.60-.70 = poor (D)*

*.50-.60 = fail (F)*



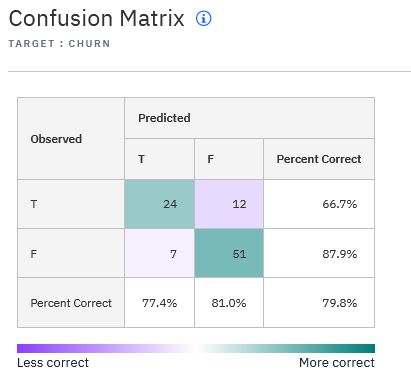
1. Click on **Confusion Matrix** on the left.

Confusion matrix is another typical metric for model evaluation. It shows *true positives, false positives, true negatives, and false negatives*.

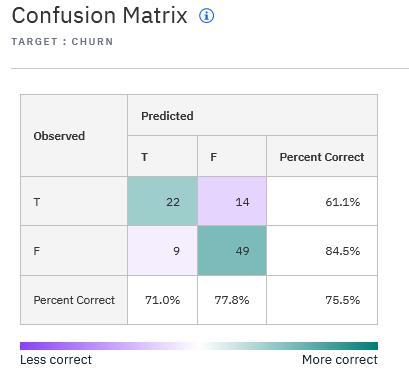
For example, the model predicted 51 of *F* (*False* – customer did not churn) records as *F* (correct prediction). At the same time, it predicted 7 records as T (the customer will churn), but the actual value was *F*. Overall, for the *False* prediction the accuracy is 87.9%. The accuracy for *True* prediction is only 66.7 percent.

If we accept that very few models are 100% accurate, and therefore will have errors in their predictions, often the goal of the confusion matrix is to help the business user under the “cost” of a misclassification (the 12 and 7 in the pink cells) as there is often a “preferred” type of error for your specific use-case.

This idea will be revisited after the next section when the model has been executed a second time.

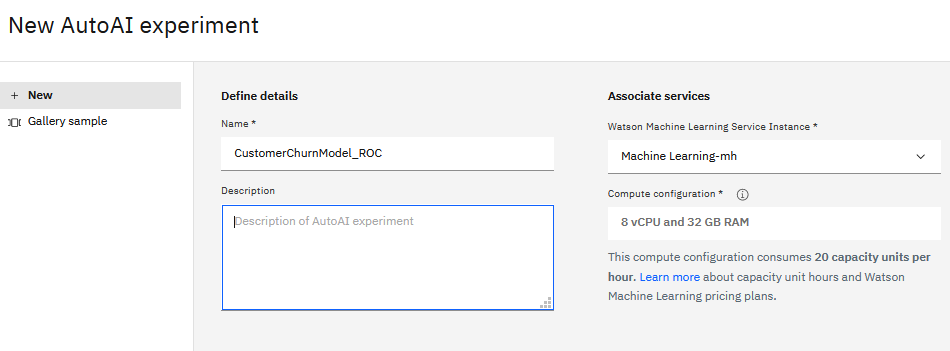


Click on *Pipeline 3*, which is the best-performing model based on accuracy, and navigate to **Confusion Matrix** view. We have a similar issue in this model, the accuracy for *T* prediction is low.

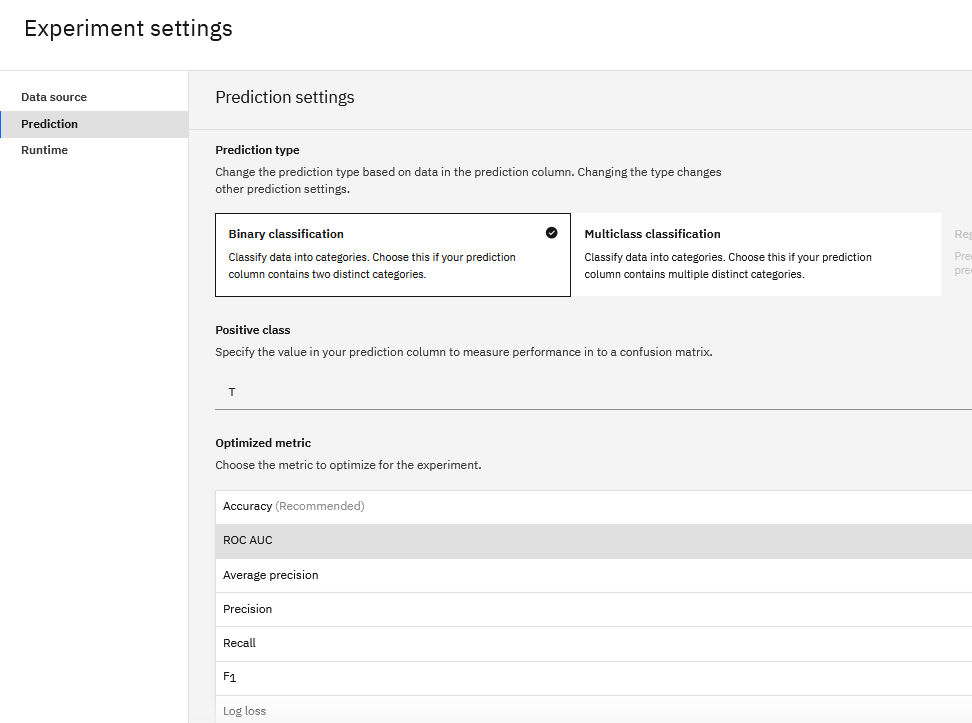


To solve this issue, we will create another model, but this time we will optimize it not for *Accuracy*, but for *Area Under ROC Curve* (abbreviated to *AUC ROC*).

1. Repeat steps 8-17 with the following modifications:
   * Name the model *CustomerChurnModel\_ROC*



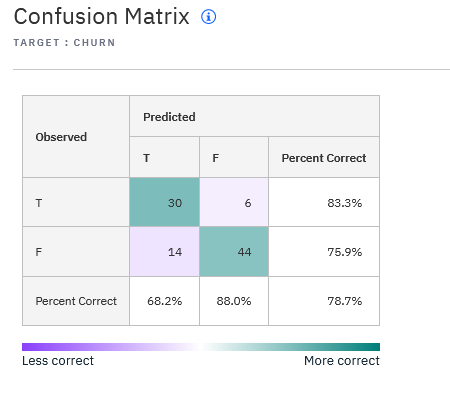
* + In **Experiment Settings** select ROC AUC as the optimization metric.



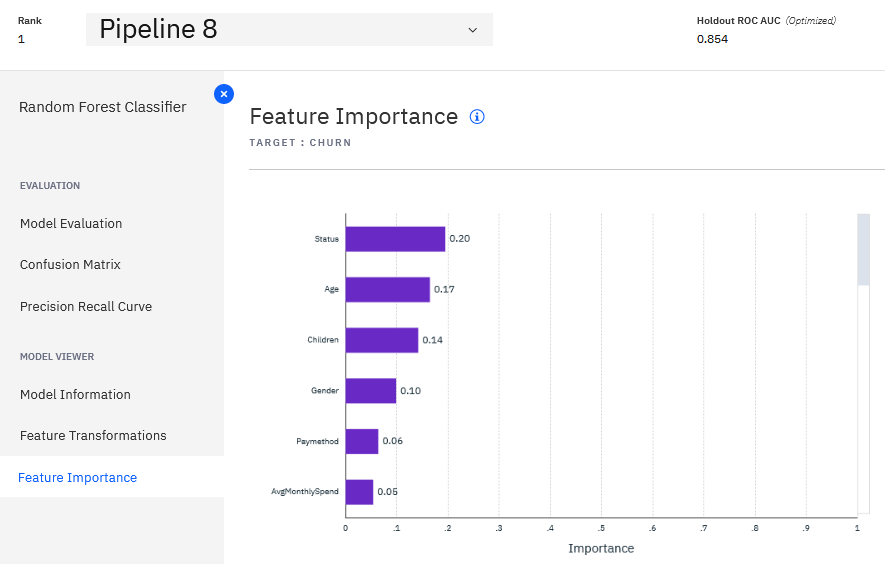
1. Click on the top pipeline in your leader board.

The accuracy of the *T* (churn) prediction has increased to 83.3 percent. However, this model has more *false positives* compared to the previous models: 14 incorrect predictions compared to 7. In machine learning this concept is described as a *cost*.

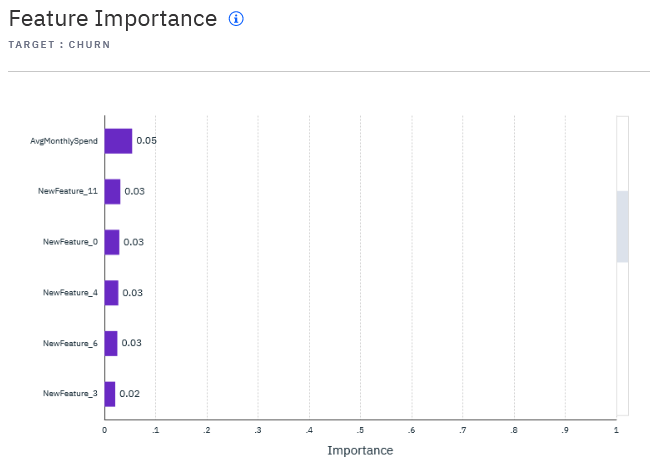
It’s usually a business decision to determine what’s more important – better accuracy in identifying true positives or reducing the number of false positives. For example, in a customer churn scenario identifying a false positive may mean that a customer will receive a promotion when they should not have received it. If there is no business impact of this prediction, then the model with the highest accuracy of true positives should be selected.



1. Click on **Feature importance**. Here we can see which features make the most impact on prediction.

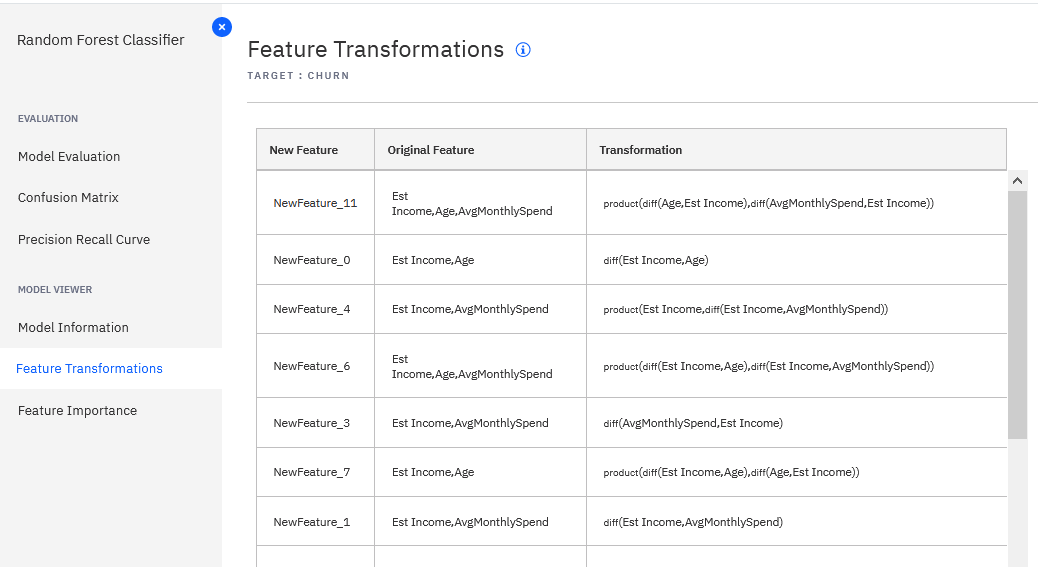


Scroll down the list of features. Notice that the model uses both features that were provided with the input data set and the features that were generated by AutoAI (start with *NewFeature*)



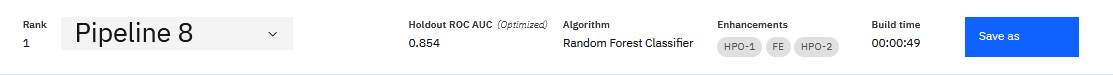
1. Click on **Feature Transformations**.

Here you can review the details of generated features. The description of these transformations is available in AutoAI documentation (see the **Additional Information** section.



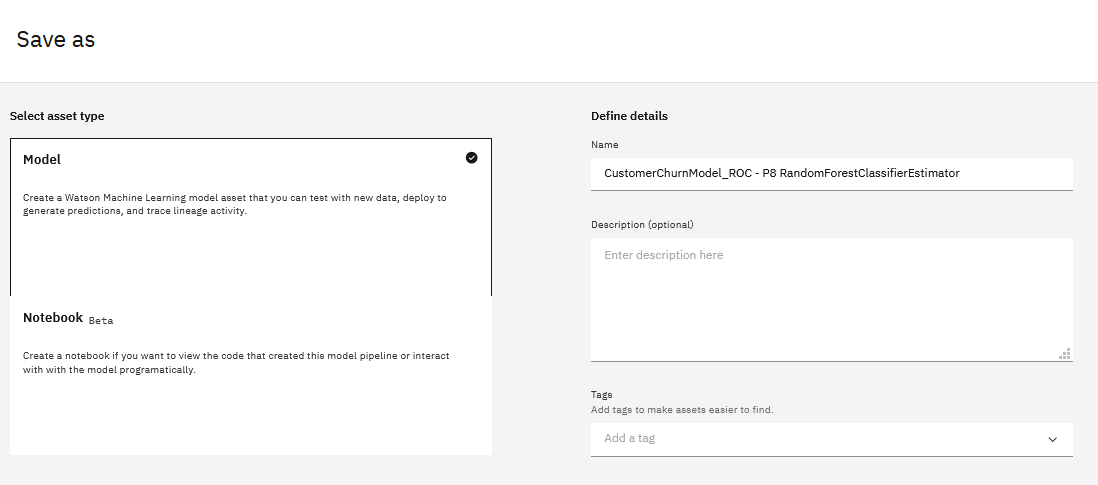
1. If you wish, review other pipelines.
2. Next, we will deploy the AutoML model (pipeline) so that it can be integrated with line of business applications.

Click **Save as** next in the *Pipeline 8* details view

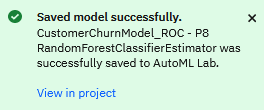


Select *Model* and keep the default *Name*. Click **Create**.

*Note: The Notebook option can be used by data scientists who want to modify generated code.*



Confirmation message is displayed in the top right corner of the screen. Click the **View in project** link or navigate back to the project.



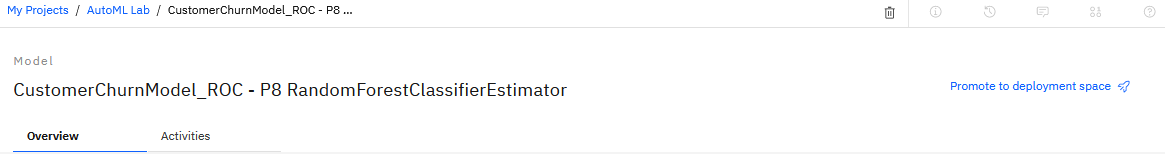
1. Next, we will configure deployment.

Deployments are configured in *Deployment Spaces*, a project type that’s dedicated to online and batch deployments of models. Separate development and deployment environments provide better security and governance. Deployments are often configured by ModelOps team, and only authorized users are given access to *Deployment Spaces*.

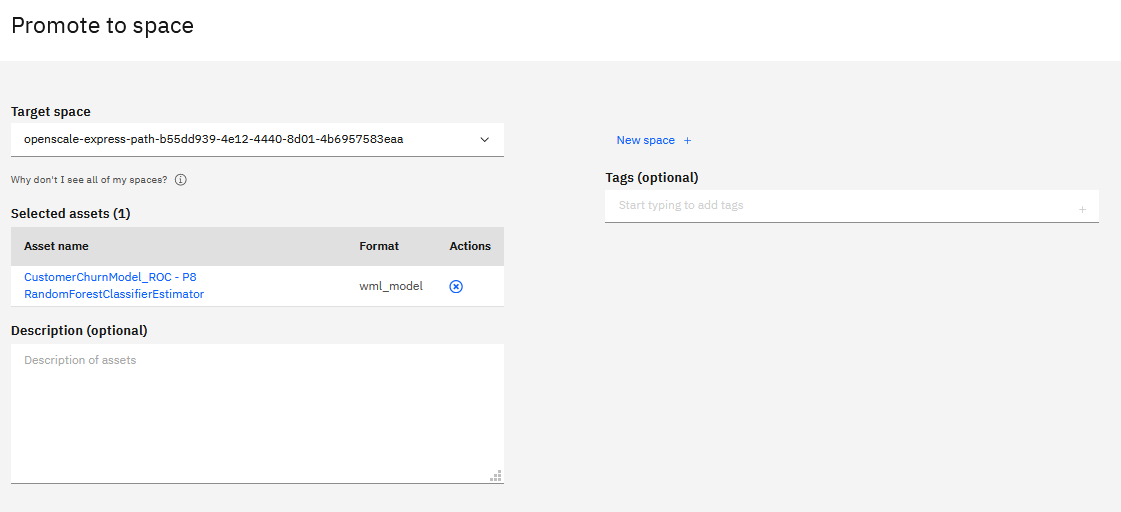
In the lab environment you have the necessary permissions to deploy models to a deployment space.

If you clicked the link in the previous step, you will see **Model details** screen. If you do not see this screen, click on the model in the project.

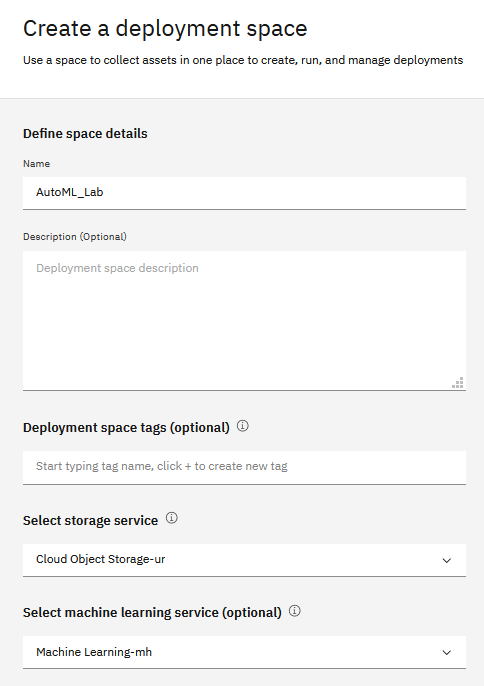
Click *Promote to deployment space*.



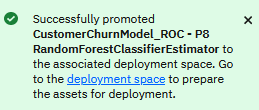
1. Click **New Space.**

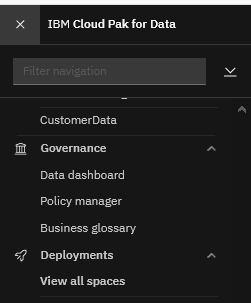


1. Provide a *Name*, for example, *AutoML\_Lab*, and select a *Machine Learning Service* from the dropdown. Click **Create**.

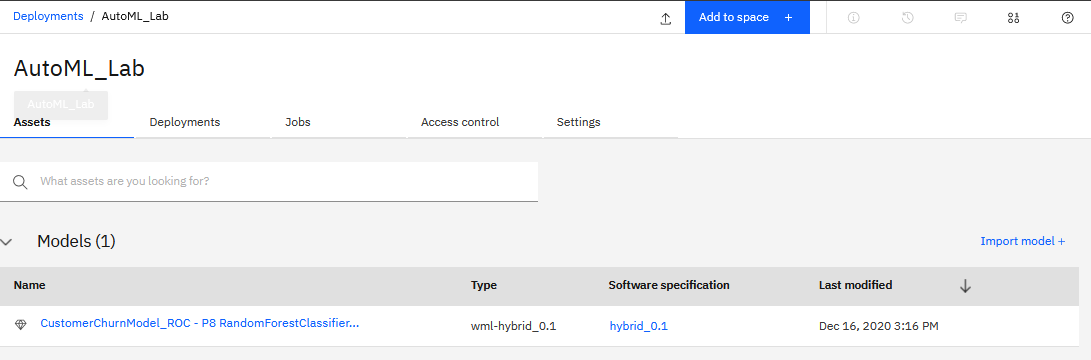


1. In the **Promote to Space** view select the created deployment space from the dropdown and click **Promote**.
2. Navigate to the deployment space by clicking on the link in the confirmation message or accessing from the main CPDaaS menu.

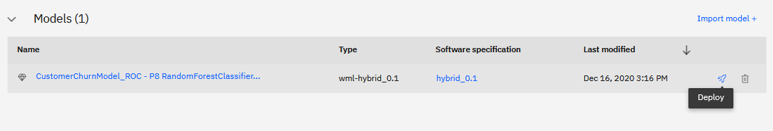




The deployment space looks similar to this screenshot:

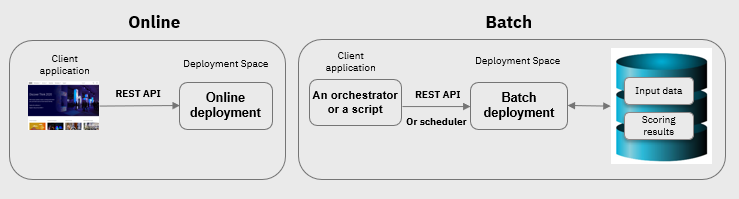


1. Position your mouse over the model until the **Deploy** icon appears. Click **Deploy**.



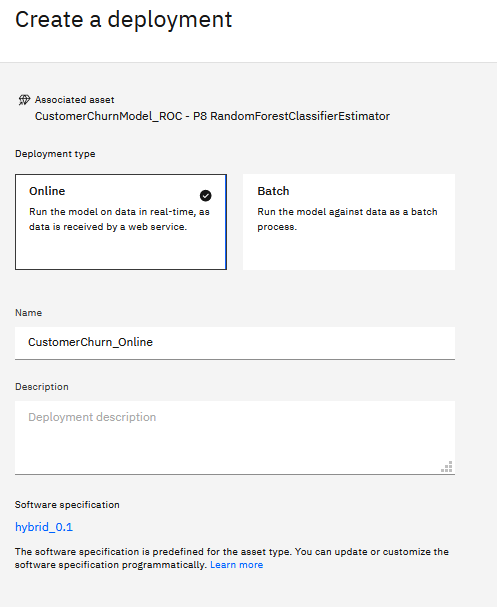
We can deploy the model for online or batch scoring.

* *Online scoring* is used when the use case requires immediate scoring results. For example, a call center agent needs to know if a customer is likely to churn when answering a call. Online scoring is integrated with line of business applications using REST APIs.
* *Batch scoring* is used for a repeatable business process for all customers. For example, creating a weekly marketing campaign for customers who are likely to churn. Batch scoring is integrated via data layer: scoring results are written to a data source that’s used by the line of business application.



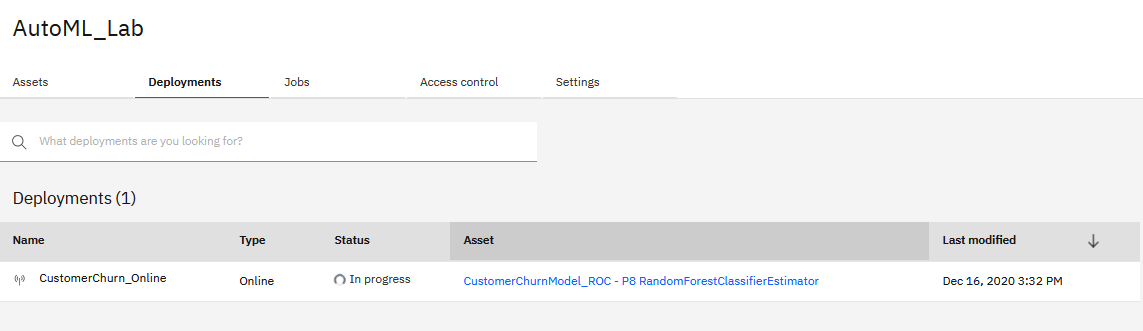
In this lab we will configure online scoring.

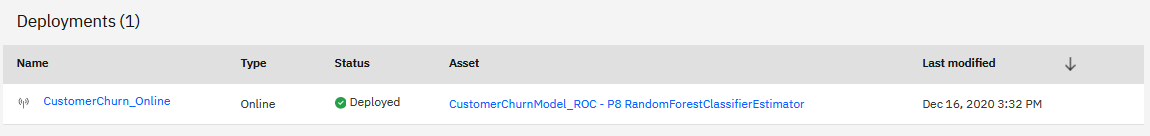
1. On the **Create Deployment** screen select *Online* and provide a model name, for example *CustomerChurn\_online*. Click **Create**.



1. Click the **Deployments** tab. After a few minutes the *Status* will change from *In progress* to *Deployed*.

The model has been deployed into a highly available container in Watson Machine Learning service.

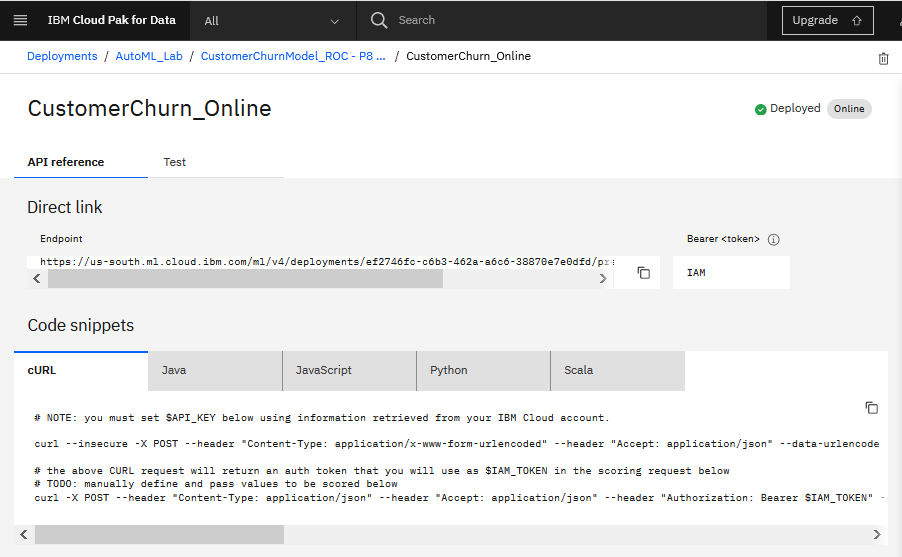




1. Click on the deployment.

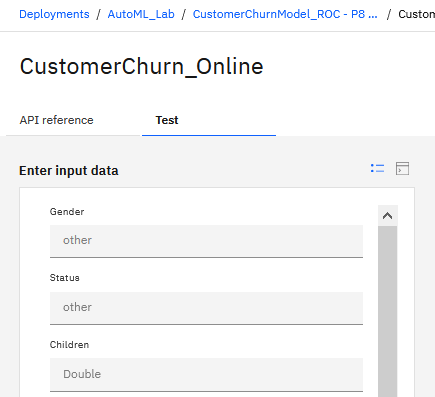
The deployment details screen provides the API reference for integration with different types of applications and the Test UI.

We will test the deployed model in the UI.



1. Click on the **Test** tab.

We can use 2 test approaches – enter values into the form or provide input in JSON format.



If using the form, enter the following values for testing.

**Gender**: F

**Status**: M

**Children**: 2

**Est** **Income**: 50000

**Car Owner:** Y

**Age**: 25

**AvgMonthlySpend**: 10

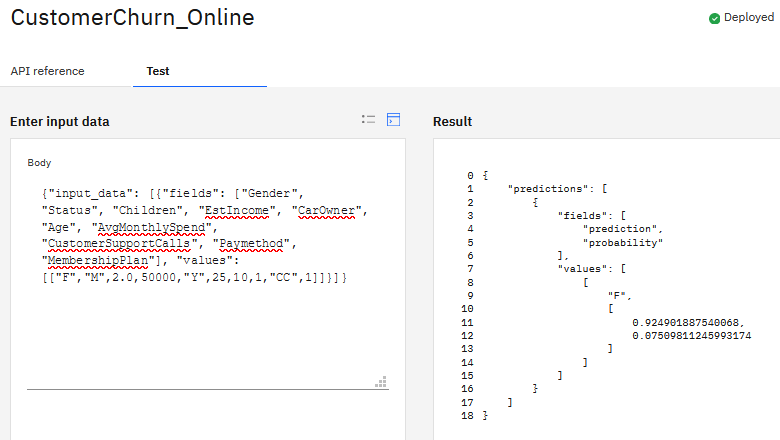
**CustomerSupportCalls**: 1

**Paymethod**: CC

**MembershipPlan**: 1

If using JSON, paste this string into the form:

*{"input\_data": [{"fields": ["Gender", "Status", "Children", "EstIncome", "CarOwner", "Age", "AvgMonthlySpend", "CustomerSupportCalls", "Paymethod", "MembershipPlan"], "values": [["F","M",2.0,50000,"Y",25,10,1,"CC",1]]}]}*



The model returns *F* (false) – the customer is not likely to churn with confidence of *92* percent.

**You have finished developing, deploying, and testing an AutoAI model. The model can be integrated with other applications using REST API.**

# Part 2: Guided ML with Modeler Flows

Similar to AutoAI capabilities covered in the last section, **Modeler Flows** automates the end-to-end process of developing and deploying models. Using Modeler flows provides a *guided ML* approach:

* Modeler implements common best practices or default values for various data science tasks
* Modeler takes actions or makes recommendations based on data types
* Modeler includes automation for several data science tasks: data understanding, data preparation, feature selection, and modeling.

While it may seem that Modeler and AutoAI provide the same capabilities, they should be considered complementary tools. Let’s review high-level differences and integration patterns for AutoAI and Modeler.

|  |  |  |
| --- | --- | --- |
| **Task** | **Modeler** | **AutoAI** |
| Data Understanding | Graphs, aggregations, tabular output | Not used for data understanding |
| Data mining | Various data mining tasks to find insights in data | Not used for data mining |
| Feature engineering | *Data scientist* derives new fields using *business knowledge* or statistical techniques | *AutoAI* derives new features using statistical techniques |
| Model building | Classification, regression, clustering, anomaly detection, association, time series, and text analytics | Classification and regression |
| Deployment | Pipeline (all steps required for scoring) | Pipeline |
| Implementation | * IBM code * No code generation for pipelines. * Supports PMML export and SQL generation. * Implementation details provided in documentation. | * IBM code and open source * Code generation option * Implementation details provided in documentation. |

In summary,

* Modeler provides a more granular approach to building models compared to AutoAI.
* AutoAI is used for building classification and regression models, while Modeler can be used for several other tasks and types of data science use cases.

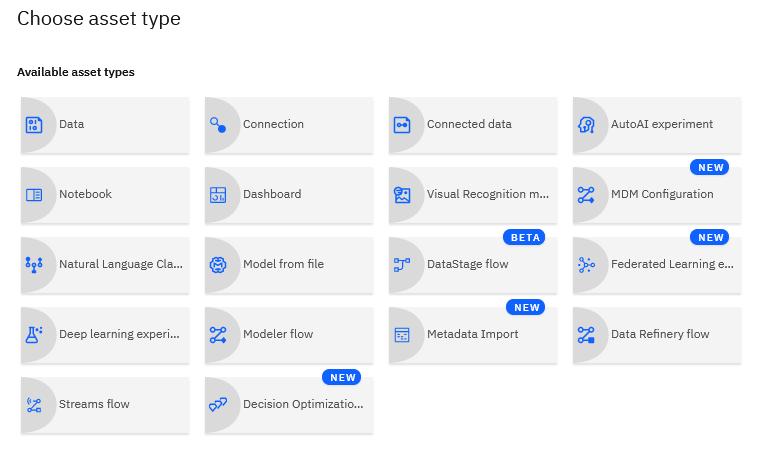
It’s possible to use both Modeler and AutoAI as stand-alone tools, but they can also be integrated. For example, Modeler can be used for data understanding, data preparation, and feature engineering (based on *business knowledge*). The dataset generated by Modeler can then be used by AutoAI to create classification models.

In the next section we will implement the same use case as we did with AutoAI. You will then review flows that perform additional steps – such as data preparation, building models for different use cases, and using unstructured data.

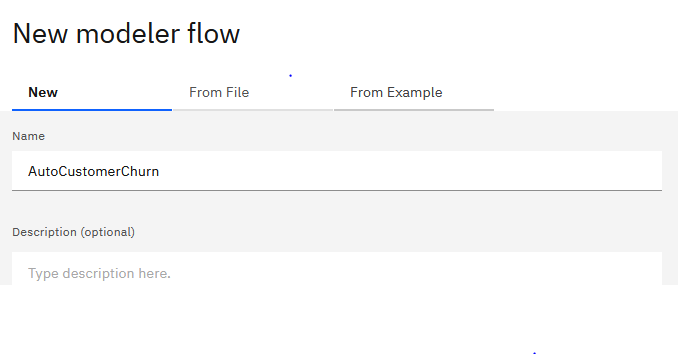
In **Part 3** of the lab we explain how Modeler Flows and AutoAI can be integrated in CPDaaS.

## Auto Modeling in Modeler Flows

1. In your project select **Add to Project -> Modeler Flow**.

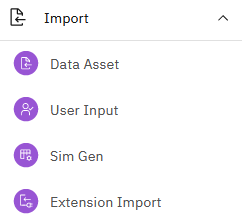


1. Enter flow name *AutoCustomerChurn* and click **Create**.

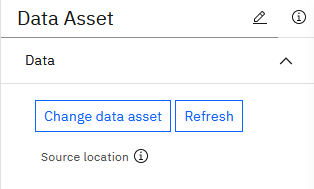


For the classification use case, we will use the same dataset as we did in the previous section.

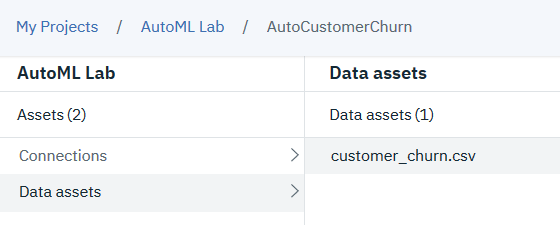
1. Expand the **Import** tab. Select **Data Asset** and drag it onto the canvas.

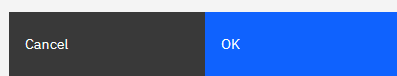


1. Double click on the **Data Asset** icon. In the **Properties** view, click **Change data asset**.

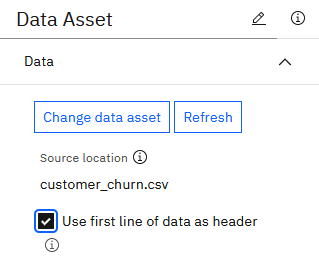


1. Expand **Data assets** and select *customer\_churn.csv* and click **OK.**



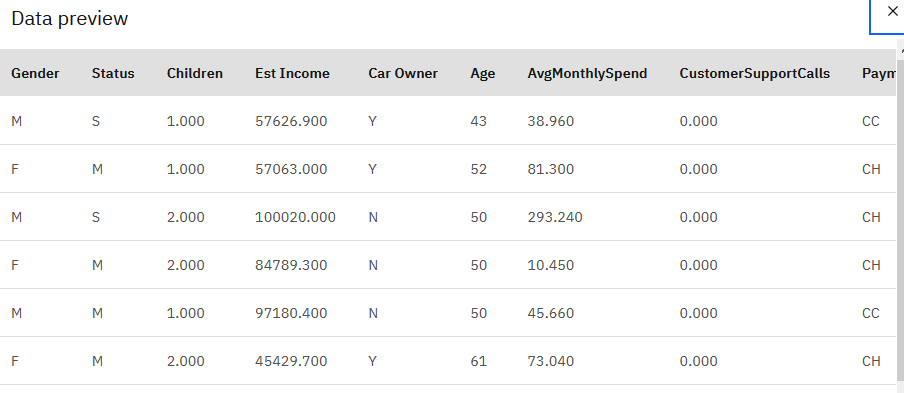


1. Check the **Use first line of data as header** option, then click **Save.**

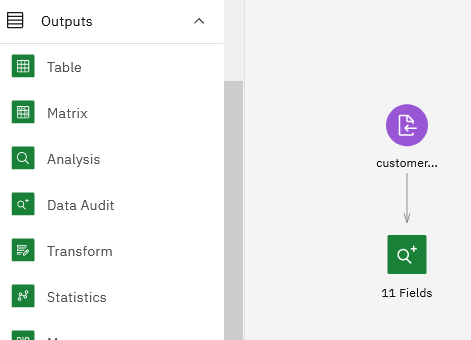


1. Right click on the **Data Asset** node and select **Preview**. If you get an error, make sure you selected the correct data asset and clicked **Save** in the previous step.

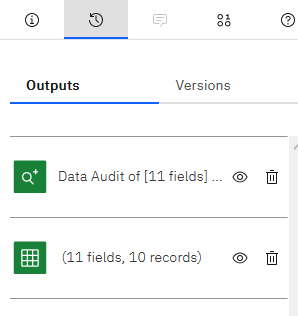
Close the preview.



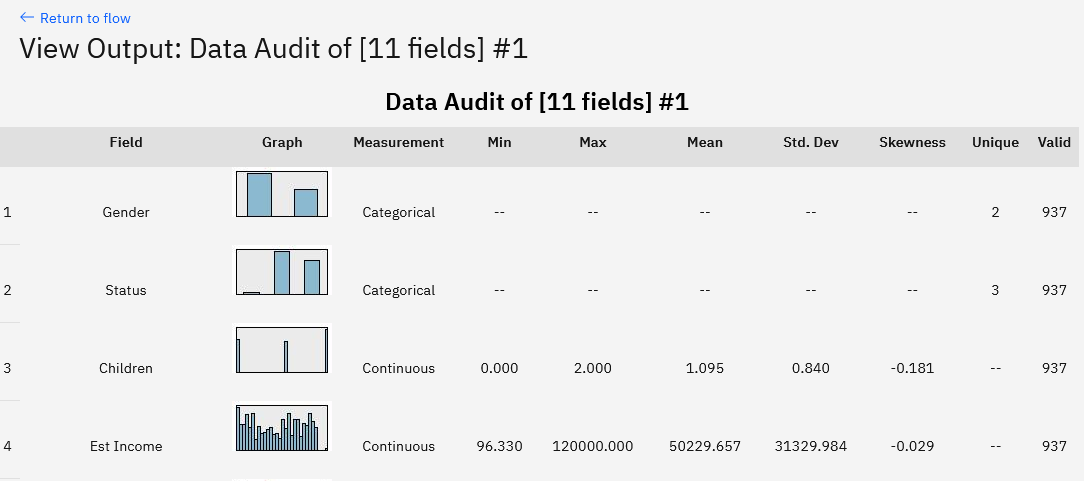
1. From the **Outputs** tab add the **Data Audit** node and connect it to the **Data Asset** node.

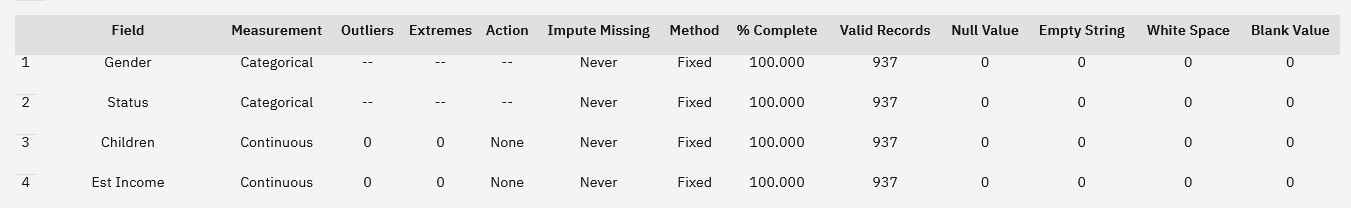


1. Right click on the **Data Audit** node and select **Run**.
2. Navigate to the **Outputs** view and open the output of data audit.



Data audit shows statistical and quality information about each input field.





If a data scientist finds issues in the dataset, such as invalid records or distribution of values that may affect the model, they can use nodes in the **Field** and **Record** tabs to fix the issues.

Next, we will add the **Type**, the **Partition**, and the **AutoClassifier** nodes.

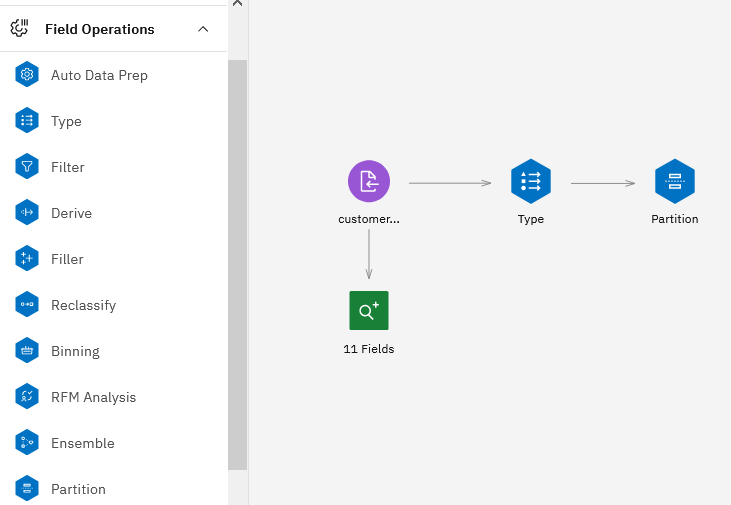
* In the **Type** node we will specify the input fields and the target variable. We will also make sure that Modeler correctly determined data types.
* In the **Partition** node we will specify the split between training and test data.
* The **AutoClassifier** node contains a set of algorithms that will be used to train the model.

1. Return to the canvas by selecting the flow in the navigation menu.
2. From the **Field Operations** tab drag the **Type** and **Partition** nodes onto the canvas.

Connect all nodes from left to right: the data will “flow” through the nodes and the operations will be applied in the direction of the flow.

Connections are made by dragging the mouse from/to the connection points. When done, your flow should look similar to this screenshot.

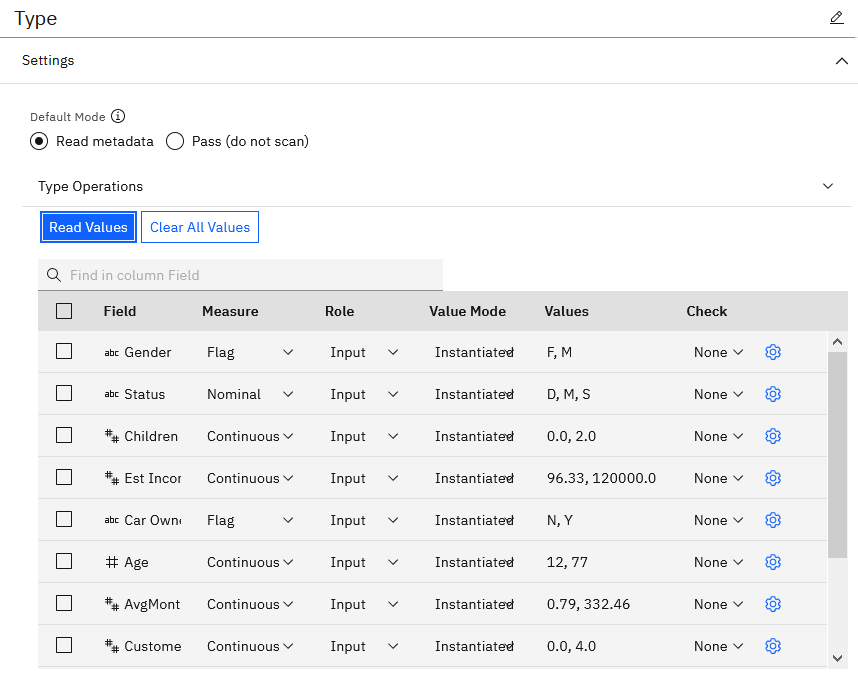
Alternatively, if the purple **Data Asset** node is selected on the canvas, you can just double-click the **Type** node from the **Field Operations** palette and it will be added to the canvas and automatically connected to the selected node. You can then double-click the **Partition** node to have it automatically added to the canvas and connected to the **Type** node.



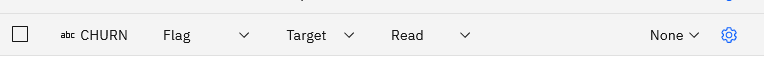
Next, we need to configure the nodes.

1. Double click on the **Type** node. In the **Type** node properties, click **Read Values**.

Reading the values initializes the **Type** node. Notice that it now displays values for each field. The type node has also determined the variable type (*Flag, Nominal, Continuous*, etc.)

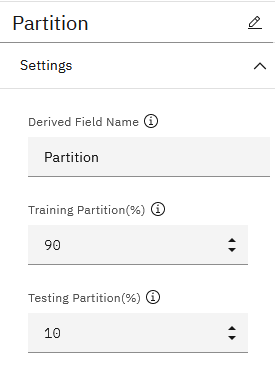


We need to make 2 changes in the **Type** node: set the **Role** of *CHURN* to *Target*, and change **Measure** to *Flag*.



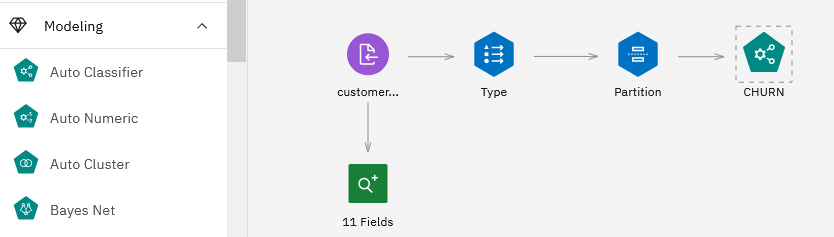
Click **Save** on the **Type** node.

1. Double click on the **Partition** node. Change the split between training and testing data to *90* and *10*. Click **Save**.



1. From the **Modeling** tab add the **AutoClassifer** node and connect it to the **Partition** node.

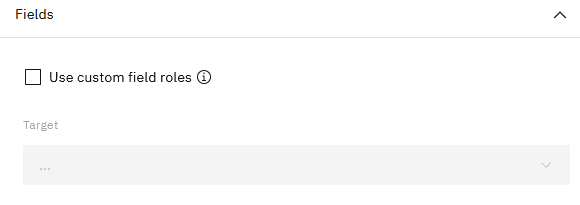
Similar to AutoML, **AutoClassifier** will build several models and select the best ones. In Modeler **AutoClassifier** selects the top models that meet performance metrics defined in the node configuration.



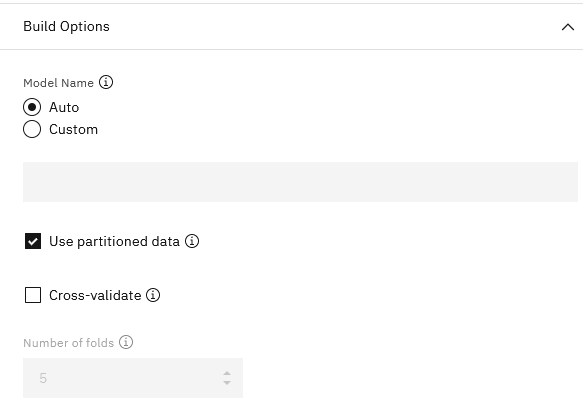
1. **This review step is optional.**

We don’t need to make any changes to the **AutoClassifer** node settings, but if you wish to review it – double click on the node. An experienced data scientist can modify several options in the node settings.

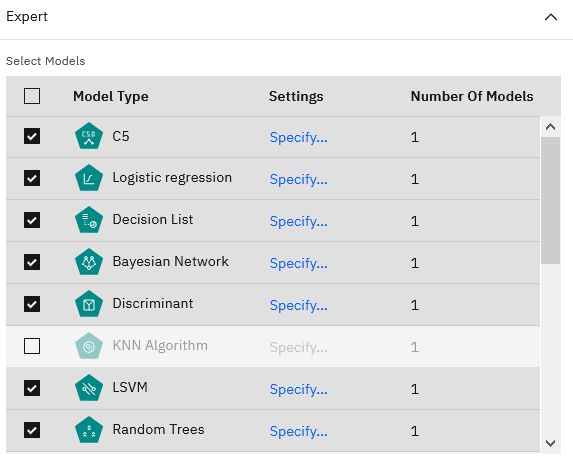
Notice that you don’t have to select input and target field. In Modeler all nodes are aware of settings of previous nodes. In the **Type** node we specified *CHURN* as the target field, and all other nodes as input fields. The **AutoClassifer** node will use these settings.



Click on **Build Options** tab. On this tab you can specify number of folds for cross validation.

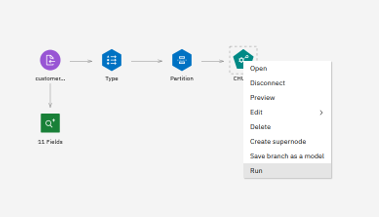


Click on the **Expert** tab. Here we can select which algorithms will be considered when building the model.



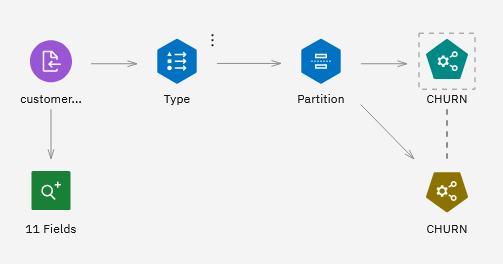
Since we do not need to make any changes in this node, click **Cancel** to return to canvas.

1. Now we are ready to build the model. Right mouse click on the **AutoClassifier** node and select **Run**.



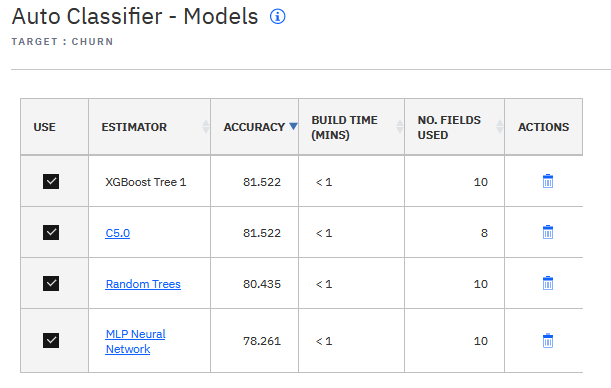
1. When the model has been built, a gold icon (*CHURN*) will appear on the canvas. The dotted line between the *CHURN* algorithm (green icon) and *CHURN* model means that the model was building using the settings in the algorithm node and if the algorithm node is re-executed then that gold icon will be where the model results are updated to (as you may have multiple models on the canvas at any one time).

*Note: In Modeler nodes are color-coded by function. As you can see in the flow that we have built, the purple icons are data sources, blue – data manipulation, green – visual output, teal – algorithms, and gold – models.*



1. Right click on the generated model and select **View Model**.

Modeler displays the top 4 models.

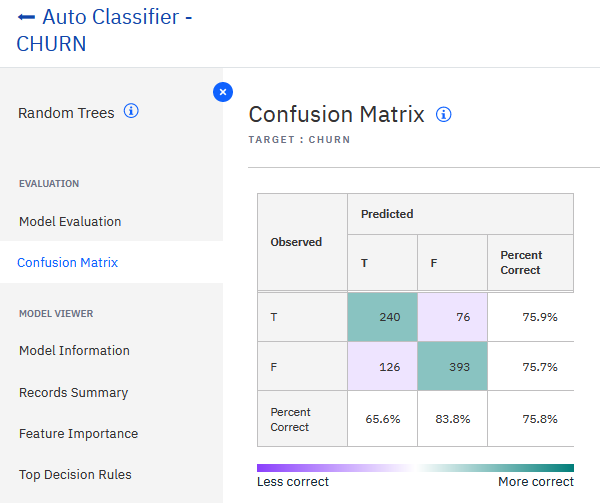


1. Next, we will review model details.

We can drill down into most models that are generated by Modeler. The level of detail available for each model depends on the model type and implementation in Modeler.

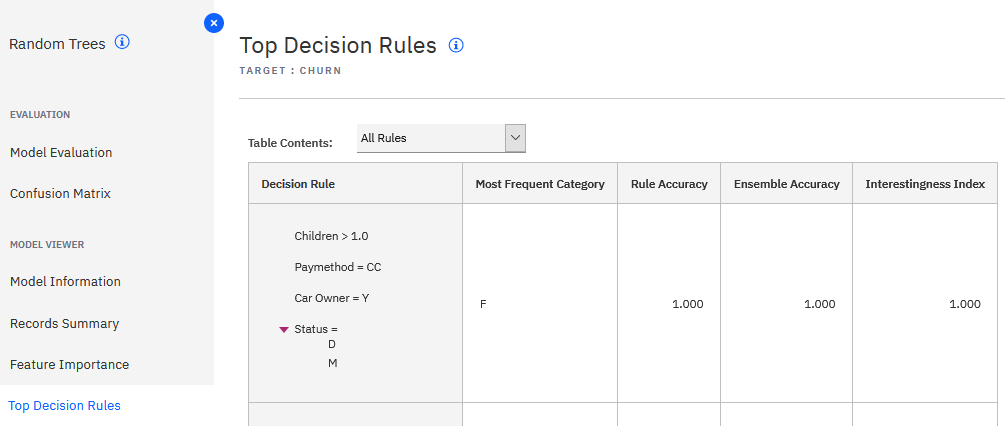
Click on **Random Trees**. Random trees model details provides similar details as AutoAI. Click through various tabs – **Model Evaluation**, **Confusion Matrix**, etc.

A different number of records are evaluated in confusion matrix compared to AutoAI, and it’s one of the reasons why accuracy is different. AutoAI evaluates the *test* partition (10 percent of records), while Modeler shows confusion matrix for the *training* partition (90 percent of records).



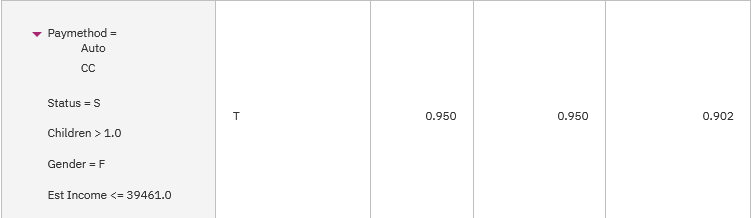
Model accuracy is also slightly different because implementation of AutoAI and Modeler are different. This is the expected behavior when using any data science software or API – results depend on implementation. IBM documents the implementation approach of Modeler flows (including statistical formulas) in the ***Algorithms Guide,*** which is referenced in **Additional Information** section.

The *Random Forest* model has the **Top Decision Rules** tab. Click on the tab to review the rules.



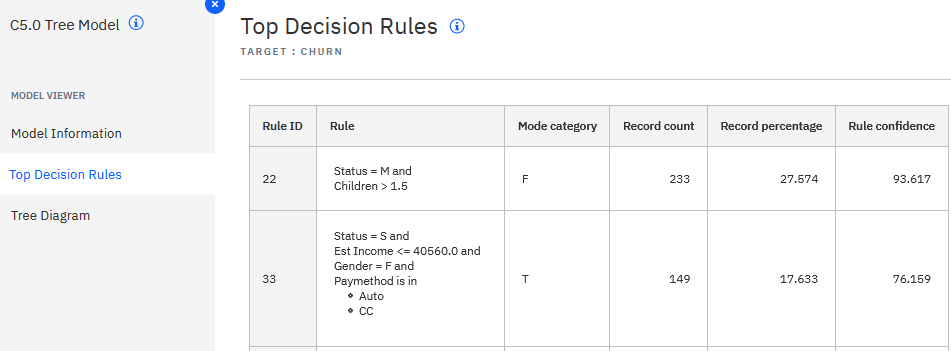
The goal of many data science projects is not just to build a model, but to gain understanding of patterns in data. Decision rules can be helpful in understanding factors that drive a specific outcome.

For example, this rule tells us that based on the current dataset, *single women (F) with 1+ children and income less than $39461 who pay by credit card or by autopay* have *95 percent* chance of churning (prediction – T).

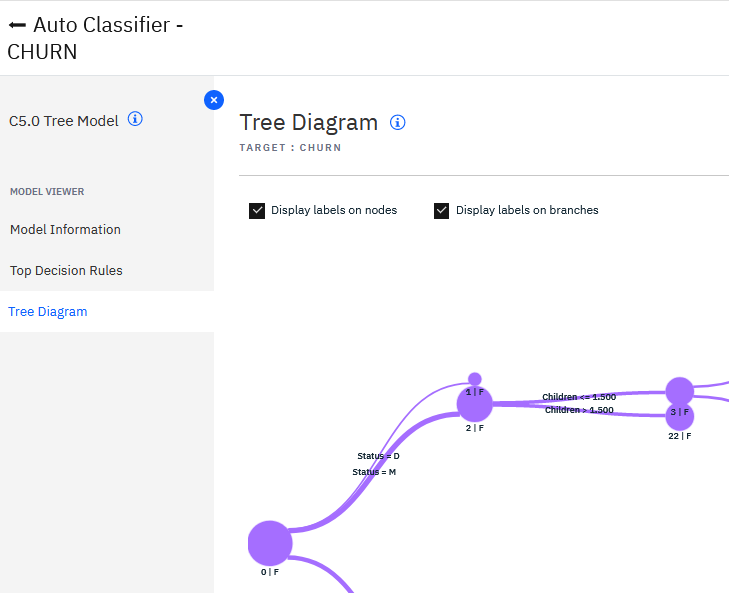


Another model that shows top decision rules is **C5**. Navigate back to the model summary view and click on **C5**.

Here we have a similar rule for churn. This model also displays the percentage of records that were used to derive the rule.



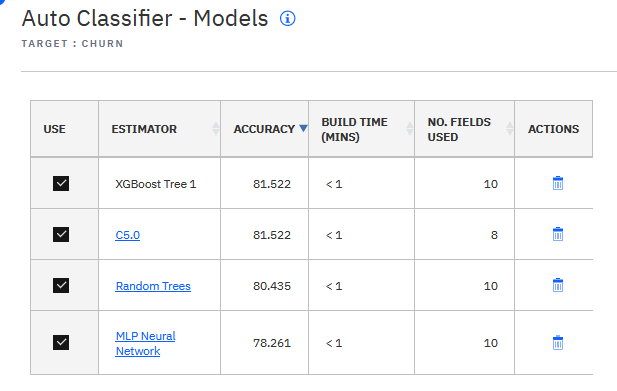
**C5** also provides a tree diagram for the model. Check the **Display labels on branches** checkbox to view the values that lead to a specific outcome.



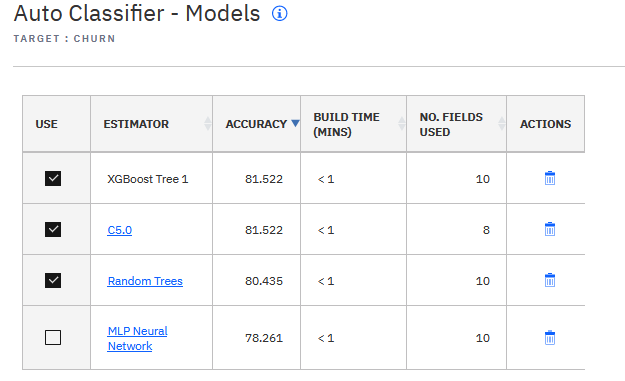
1. Return to the **Model summary** view.

At this point a data scientist can take a few actions:

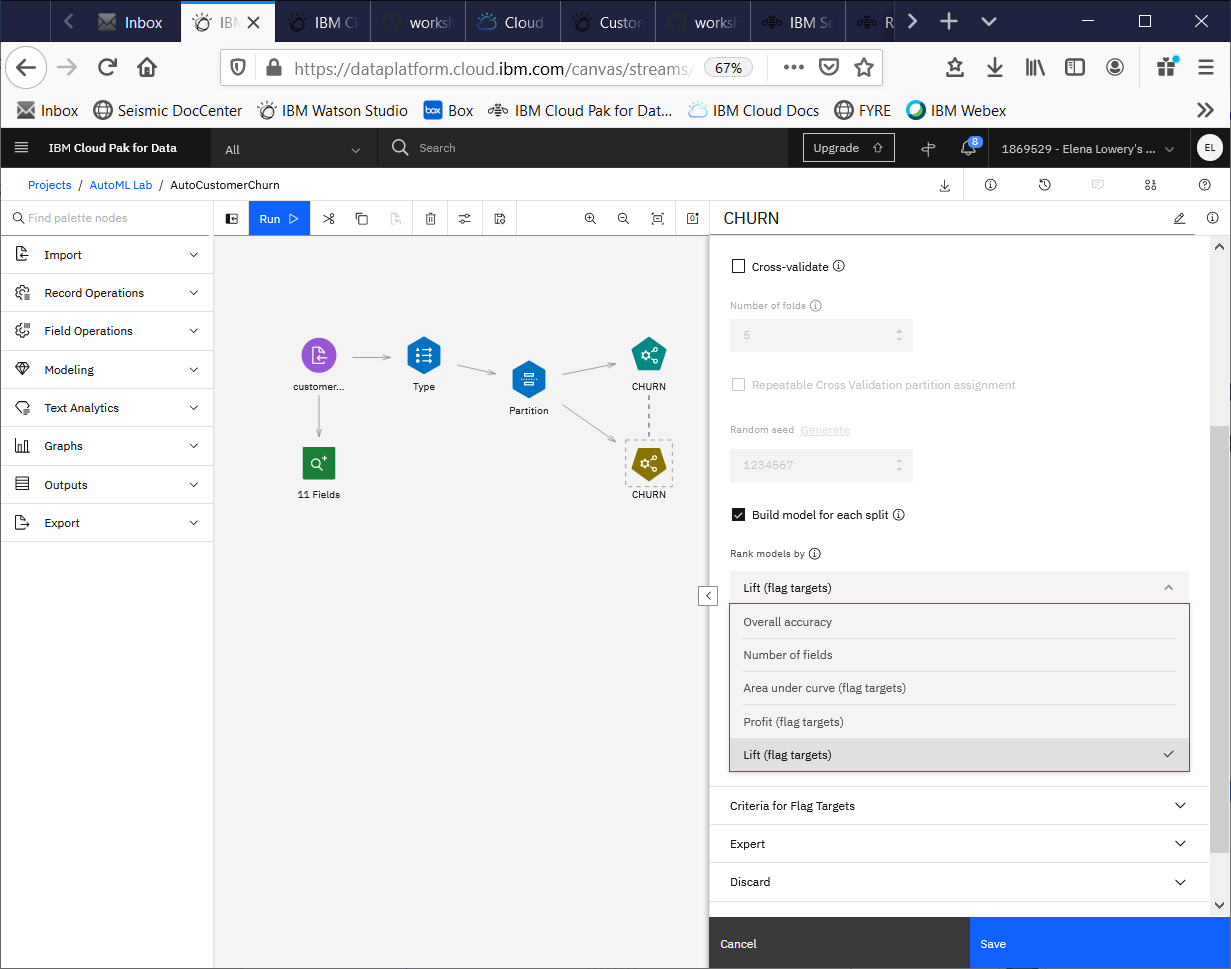
* Keep the default setting – the checkboxes for all models are checked. In this scenario Modeler will perform ensemble scoring (all models will be used when scoring). The settings for ensemble scoring can be modified in the AutoClassifier node settings.
* Delete or uncheck the models from the ensemble (they will not be used for scoring).
* Use auto modeling just as a quick step to determine which model should be used. A data scientist can then build an individual model for the top performer - all algorithms that are used in Auto nodes are available as individual algorithms on the *Modeling* tab.



In our example we will uncheck the last model, the *MLP Neural Network* model.



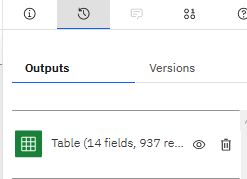
*Note: By default in AutoClassifier models are ranked based on Lift (a variation of ROC curve), but the Build settings of the node also provide other options.*



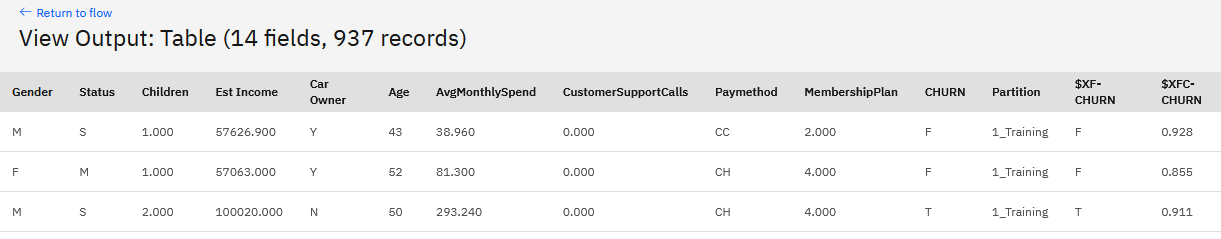
1. Return to the canvas. Add the **Table** and the **Analysis** nodes from the **Output** tab

We don’t need to configure any of these nodes. We will run each node individually by selecting **Run** from the right click menu of the node.

1. Run the **Table** node and access it through the **Outputs** tab.



When we run the **Table** node, it scores the data and displays it in a table format. The last two columns, *$XF-CHURN*, and *$XFC-CHURN* are the scoring results – the prediction and the confidence in the prediction.

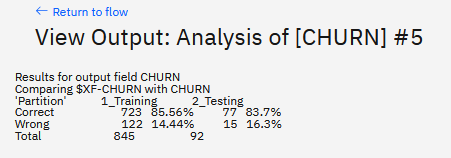


1. Run the **Analysis** node and access it through the **Outputs** tab.

In this view we can see mode accuracy for training and testing partitions.

*Note: IBM is working on improving the output of this node.*

Model accuracy is 85.56 percent on the training partition, and 83.7 percent on the test partition.



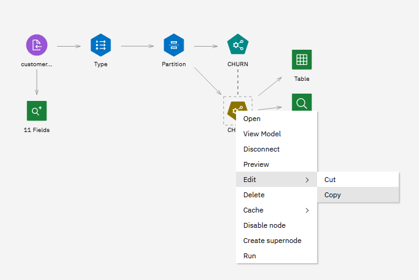
Next, we will deploy the model for scoring. You can choose the *quick* approach or the *recommended* approach.

The *quick* approach does not require any more modifications to the flow. However, when you test the model, the deployment will require the CHURN field because when a flow is deployed, it always reads the input/output schema from the *Source* and *Export* nodes. The *CHURN* field will not be used for scoring, but it will still have to be passed in. This implementation is by design because Modeler flows can implement more than scoring of classification models, and therefore the target variable should not be automatically dropped from the input schema.

The quick approach is often used for demonstrations, but for a production deployment, we recommend that you add the required modifications to the flow.

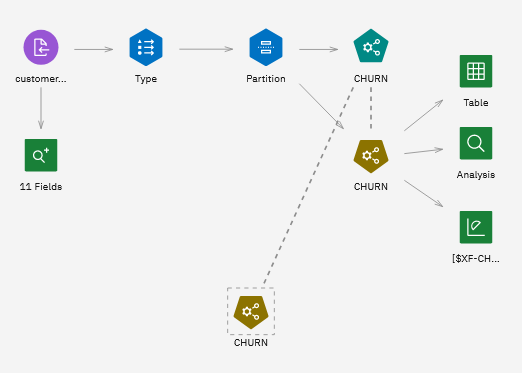
If you want to use the quick approach, continue to **step 31**.

1. In the flow, copy the *model* by selecting **Edit -> Copy** from the right click menu.



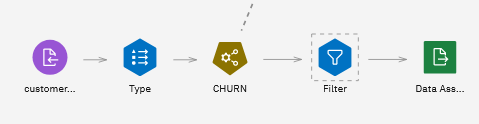
Paste the model below the flow you built.

The dotted line between the algorithm and the model means that the model was built using the settings in the algorithm node.

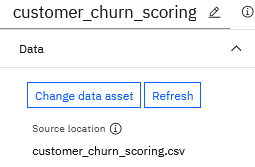


Next, we will add the **Data Asset Import**, the **Type,** the **Filter** and the **Data Asset Export** nodes.

1. Drag the **Data Asset Import**, the **Type** (**Field Operations**tab), the **Filter** *(***Field Operations**tab*)*, and the **Data Asset Export** nodes to the canvas and connect them.



1. Change the **Data Asset Import** node to point to *customer\_churn\_scoring.csv* and ensure that you check the option to “**Use first line of data as header**”



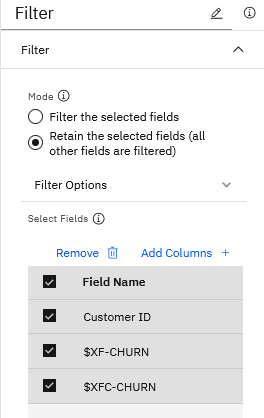
This file is an example of “new data” or “operational data” – it contains the list of customer records that should be scored.

Preview the file. It has all the same fields as training data with 2 exceptions:

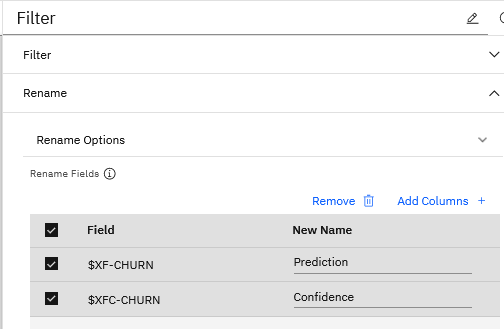
* It includes a *customer id*
* It does not include the *CHURN* field.

1. We don’t need to make any changes to the **Type** node but it still needs to be in the flow because it instantiates data types.
2. Double click on the **Filter** node and edit it.

Select the **Retain the selected fields** radio button, click **Add Columns** and add 3 fields – *Customer ID*, *$XF-CHURN* (predictions), and *$XFC-CHURN* (confidence in the prediction)

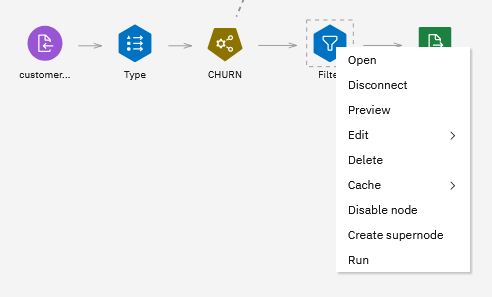


You can also rename the fields to make them more readable.

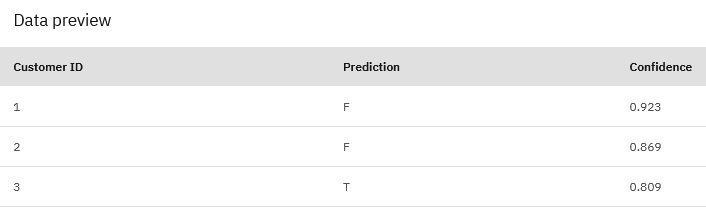


Save the node settings.

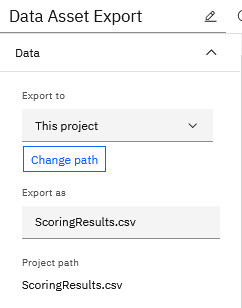
If you want to preview data after the **Filter** has been applied, you can use the **Preview** option by right clicking on the node.



Since this node is the last data manipulation node before the terminal (Export node), this schema (3 fields) will be used as the output schema for the model.



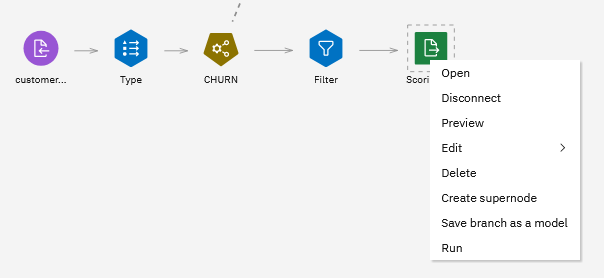
1. Change the **Data Asset Export** to write to *ScoringResults.csv*.



Save the node.

When we run this node, the flow will score new customer data and write it to *ScoringResults.csv* to the project.

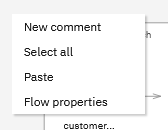
If you wish, select **Run**. You can review scoring results after we finish saving the model.



This task is called “interactive batch scoring”. We can also schedule batch scoring or deploy the flow for real time scoring.

1. Optionally, add comments to the flow to make it easier to understand for other users.

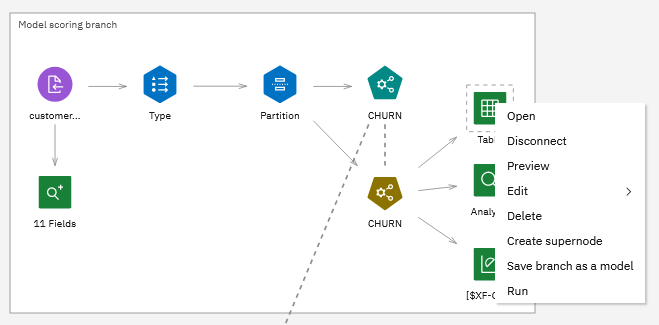
You can add a comment by selecting **New Comment** the right click menu in the canvas (not over a node). You can then move and resize the comment and add text.



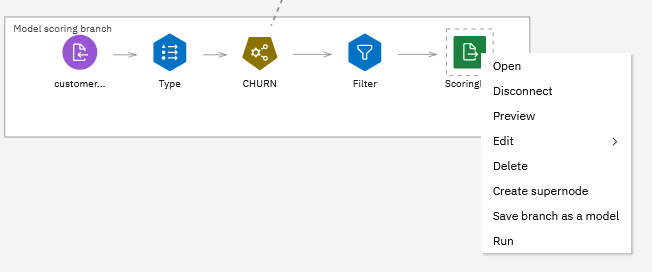
We are now ready for deployment.

1. Deployment steps are the same for *quick* and *recommended* deployment, the only difference is the branch that you select for deployment.

*Quick deployment:* right mouse click on **Table** node and select **Save branch as a model.**

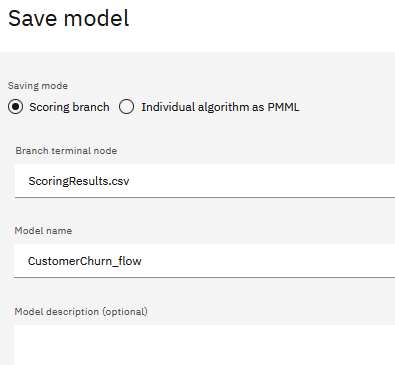


*Recommended deployment:* right mouse click on **Table** node and select **Save branch as a model.**



1. Change model name to *CustomerChurn\_flow*.

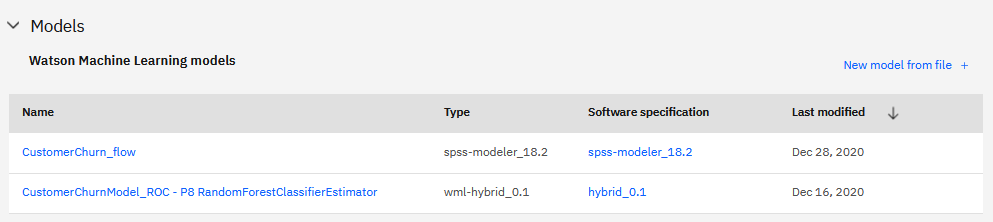
The **Branch Terminal node** setting does not need to be changed – it will be either *Table* or ScoringResults.csv, depending on whether you are following the quick or the recommended approach.



Click **Save**, then **Close** on the confirmation message.

1. Navigate back to the project.

The flow model is now saved in the project.

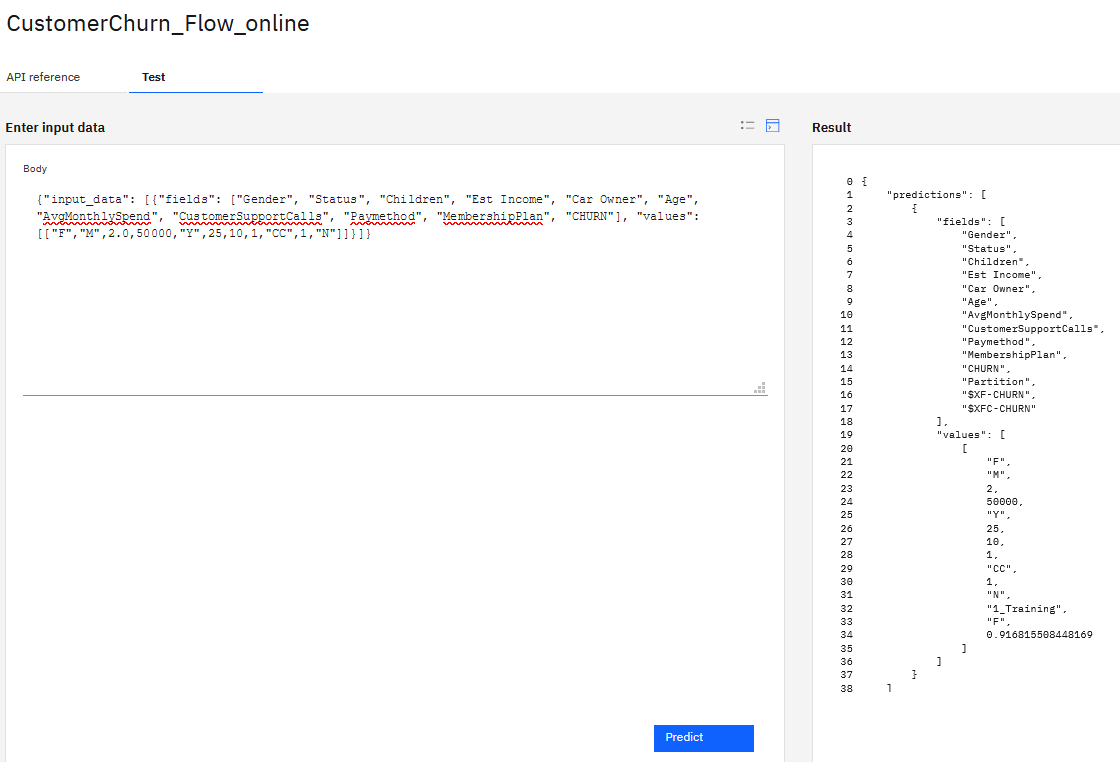


1. Follow the same steps as you did for the AutoAI model to deploy the flow model for online scoring:
   * Promote the model to previously created deployment space
   * Configure online deployment
   * Test the deployment.

You can use the following values for testing.

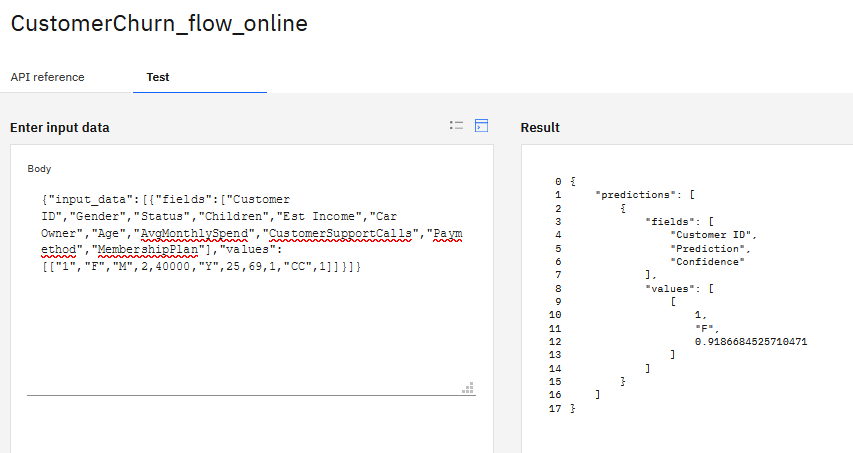
**Quick deployment** (reminder, we are passing a “dummy” value for CHURN):

*{"input\_data": [{"fields": ["Gender", "Status", "Children", "Est Income", "Car Owner", "Age", "AvgMonthlySpend", "CustomerSupportCalls", "Paymethod", "MembershipPlan", "CHURN"], "values": [["F","M",2.0,50000,"Y",25,10,1,"CC",1,"N"]]}]}*



Recommended deployment:

*{"input\_data":[{"fields":["Customer ID","Gender","Status","Children","Est Income","Car Owner","Age","AvgMonthlySpend","CustomerSupportCalls","Paymethod","MembershipPlan"],"values":[["1","F","M",2,40000,"Y",25,69,1,"CC",1]]}]}*



You have finished deploying a Modeler flow for online scoring.

**You have finished the Guided ML in Watson Studio lab.**

# AutoML and Guided ML Summary

In **Part 1** and **Part 2** you learned how to create a classification model in *Auto ML* and *Guided ML*. We implemented the same use case so that you can understand the differences in implementation approach.

When we use AutoML, we need to make sure that training data contains all features (input columns) that we want use for training. While AutoML will perform feature engineering (deriving new input columns), it will be “statistical”, and not a “business use case” feature engineering. For example, AutoML may derive a new feature by applying a *log* function to a numeric column. The log function reduces skewness of data, which can improve accuracy of a model.

Prior to building a model, a data scientist can consult with a domain expert who will recommend features that should be considered for modeling. For example, in a credit card fraud use case, the domain expert may know that *time between transactions* may be an important predictor. Most credit card transaction records do not contain this field by default, and this feature should be derived prior to building a model. The “business use case” features can be created in Modeler flows by a domain expert or a data scientist.

Another common issue with fraud (and several other use cases) is unbalanced data – the number of known fraudulent use cases is often much smaller than the number of non-fraudulent use cases. While data scientists can use sampling techniques or generate synthetic data, a Modeler flows provides another useful approach for this issue – creating a model that flags anomalies in transactions. Once the anomaly field is then added to the dataset and used to build a classification model.

AutoML and Guided ML can be used as stand-alone or complimentary tools. In the following table we describe a few use cases that will help you understand how to use each tool.

|  |  |  |
| --- | --- | --- |
| **Use Case** | **Guided ML** | **AutoML** |
| *Stand-alone* | Complete all steps for building a model: data understanding, data preparation, modeling. Build several model types – classification, regression, clustering, association, forecasting, text analytics | Build classification and regression models |
| *Integrated:* data preparation in Guided ML followed by model building in AutoML | Perform data understanding steps and derive new features. Export the data set to use in AutoML | Build classification and regression models using the dataset created by Guided ML |
| *Integrated:* prototyping in Guided ML and model building/source code generation in AutoML | Use Guided ML to get a deeper understanding of feature importance and best performing models | Use AutoML to build model and generate source code for further modification in AutoML. |

We encourage you to continue implementing various use cases in both tools – it will give you a better understanding of each tool’s strengths.

# Part 3: Advanced Guided ML and Integration with AutoML

In this section we will review a few examples of different Guided ML flows that can be used stand-alone or in combination with AutoML.

We do not include step-by-step instructions for building the flows but explain how to run them. Modeler flows is a mature and comprehensive tool and learning how to use it requires an investment of time. See **Additional Resources** for recommended classes and self-study materials.

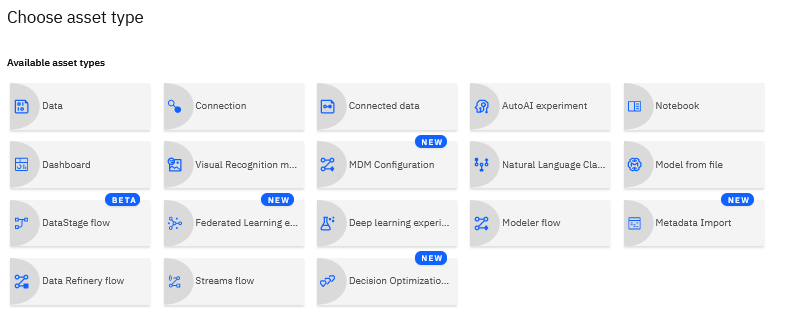
## Flow Import

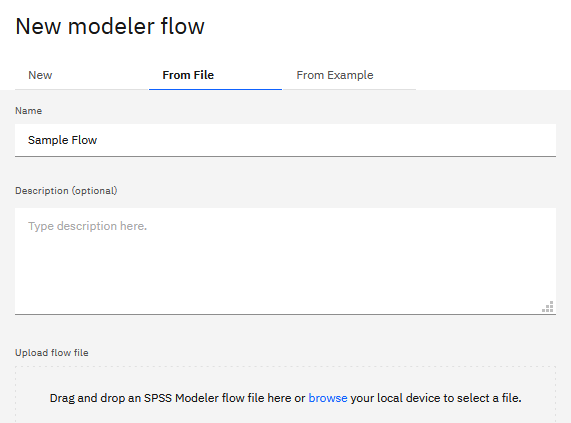
This section provides general instructions for all flows that you will import in the subsequent steps.

Currently flows are not imported as a part of the Cloud Pak for Data project. This feature will be added in the first half of 2021.

1. To import each flow, use the **Add to Project -> Modeler Flow** option, then select **Import from File**.

All sample flows are in the *git repo/Flows* folder (downloaded in the beginning of the lab).



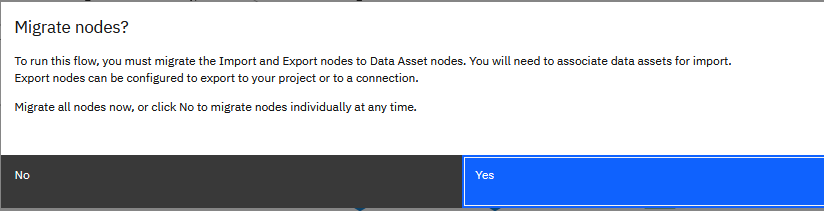


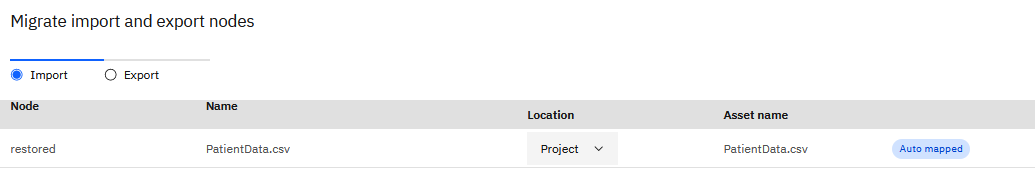
1. Next, we need to modify *import* and *export* data sources.

Always import *csv files* prior to importing the flow. If the csv file name matches the name of the file used in the flow, the csv data source will be auto-mapped.

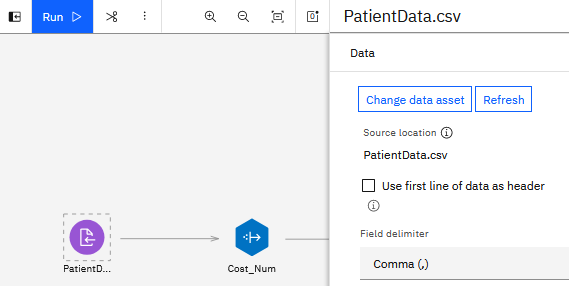
In this lab we imported all required csv files as a part of the project import.

When the flow opens, you will see the migrate nodes dialog. Select **Yes**.





If the csv file was not auto-mapped or if you’re using a different type of data source, you can manually change the data source in the settings of the Data Asset node (both import and export).

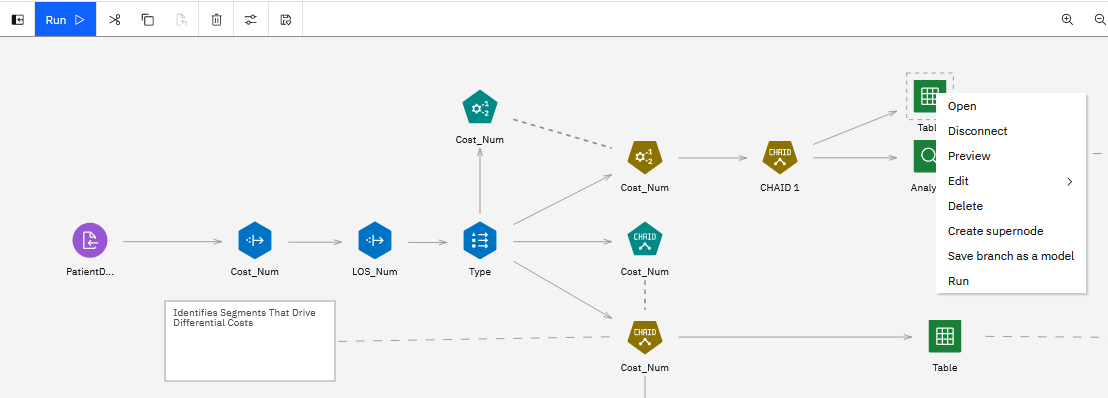


Always do a **Preview** (right click) on the data source to make sure that data is read in correctly.

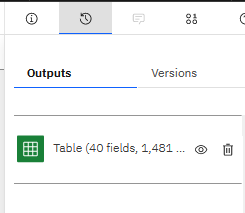
## Running Flows

When running flows, using the **Run** icon on the top menu bar will run the entire flow, including model building, which can take some time.

You can run an individual branch by right-clicking a terminal node and selecting **Run**. This action will run only the selected branch.



Output of all nodes that provide output is accessible through the **Output** panel.



## Sample flows

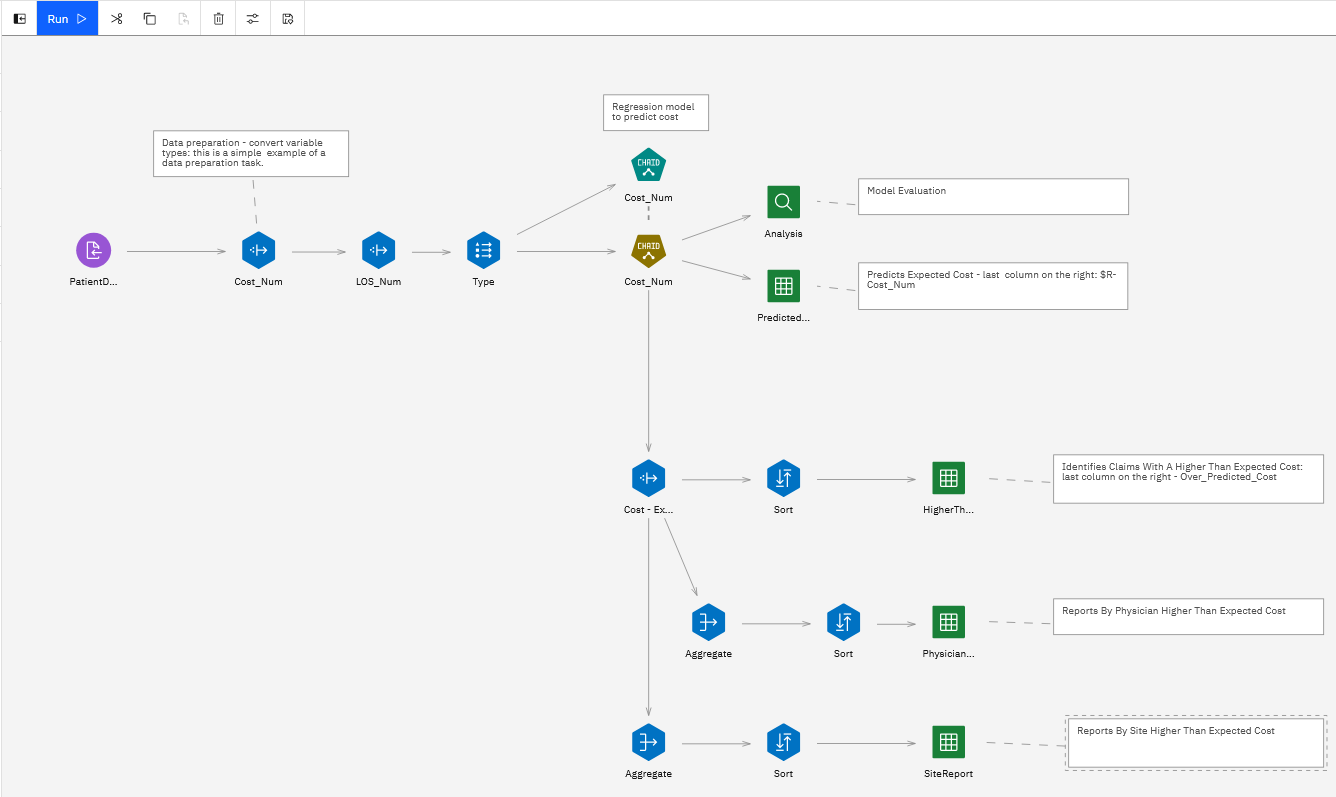
Import and review each flow. Optionally, follow instructions to integrate Guided ML output with AutoML.

### Healthcare Cost Prediction

*Flow name: HealthcareCostPrediction.str*

This flow predicts cost for a medical procedure using a *CHAID* algorithm. The flow then creates several reports to flag claims, physicians, and sites with cost higher than expected.

This flow uses **Derive** and **Aggregate** nodes, which are some of the most frequently used nodes in flows.



The **Derive** node is be used to implement many tasks related to data manipulation. In this flow we have 3 derive nodes: 2 Derive nodes in the beginning of the flow convert a variable to a number type, and 1 calculates the difference between the predicted cost and actual cost.

The syntax of the **Derive** node uses the *CLEM* language – a scripting format that’s used in flows. You can find more details about CLEM syntax in documentation.

|  |  |
| --- | --- |
|  |  |

The **Aggregate** node is similar to *group by* in SQL, but it provides more capabilities, such as aggregating by mean, sum, min, max, etc.

|  |  |
| --- | --- |
|  |  |

Run each individual terminal node and review the output. If you run the entire flow using the **Run** button, it will rebuild the model, and the flow will take longer to run.

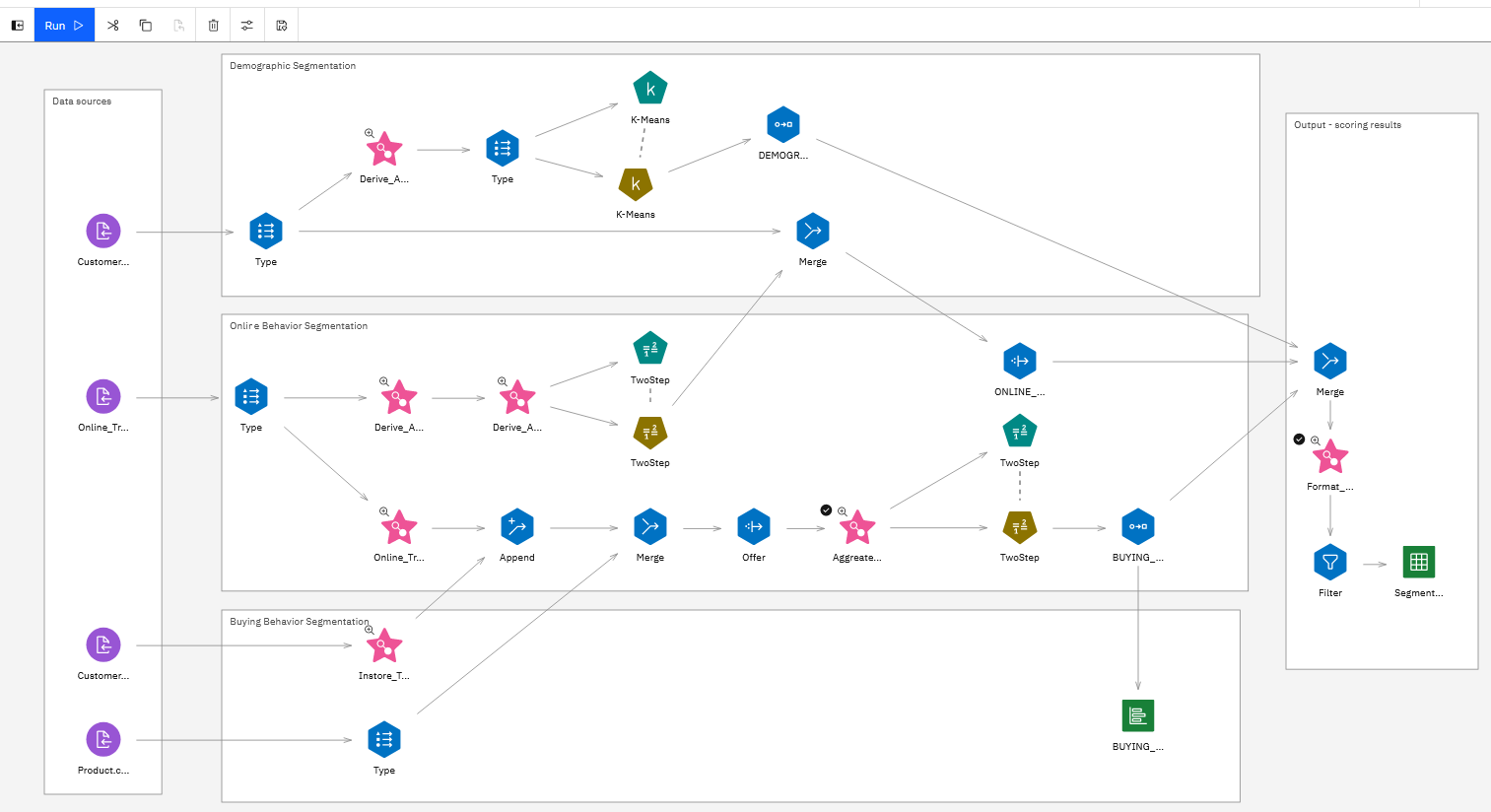
**If you would like to use AutoML for this use case:**

1. Build a model in AutoML using the same input and target variables as the flow – you can look up the input and target variable in the **Type** node.
2. Save the AutoML model as a notebook.
3. Add code to the notebook to do batch scoring of a dataset.
4. Build a flow that
   * Calculates the difference between predicted cost (generated by AutoML) and actual cost – the **Derive** node in the sample flow
   * Performs the same aggregations as the sample flow

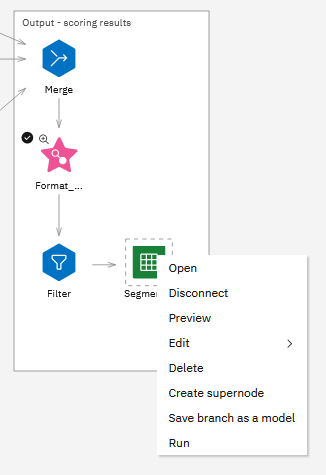
### Customer Segmentation

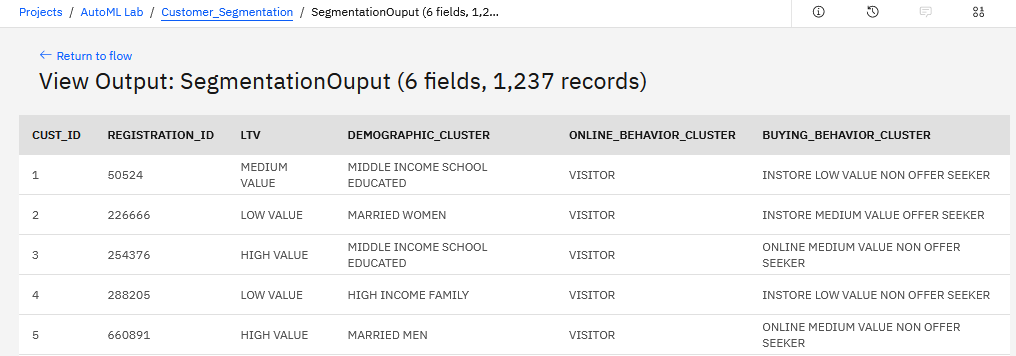
*Flow name: Customer\_segmentation.str.*

The customer segmentation flow uses *clustering* algorithms (*K-means* and *TwoStep*) and data preparation to derive customer lifetime value (LTV), demographic cluster, online behavior cluster, and buying behavior cluster.



Run the terminal node in the **Output** box to view segmentation results.

g



This flow includes *super nodes* marked by pink stars. Super nodes are used for visually organizing the flow. Right click on any super node and select **View Supernode.**

|  |  |
| --- | --- |
|  |  |

The segmentation flow uses several **Record** and **Field** nodes for data preparation prior to building nodes and for formatting output:

|  |  |
| --- | --- |
|  | Create bins of data, for example, *age group* or *income group* |
|  | Use formulas (business rules or transformations) to derive new fields |
|  | Change value of an existing field using a formula or a transformation (for example, replace data type) |
|  | Remove columns that are not needed in the final output |
|  | Change valuesin fields to specified values |
|  | Remove or keep rows based on specified criteria (similar to *where* clause in SQL) |

Look for these nodes in the flow. You can review the configuration of each node by double clicking on it.

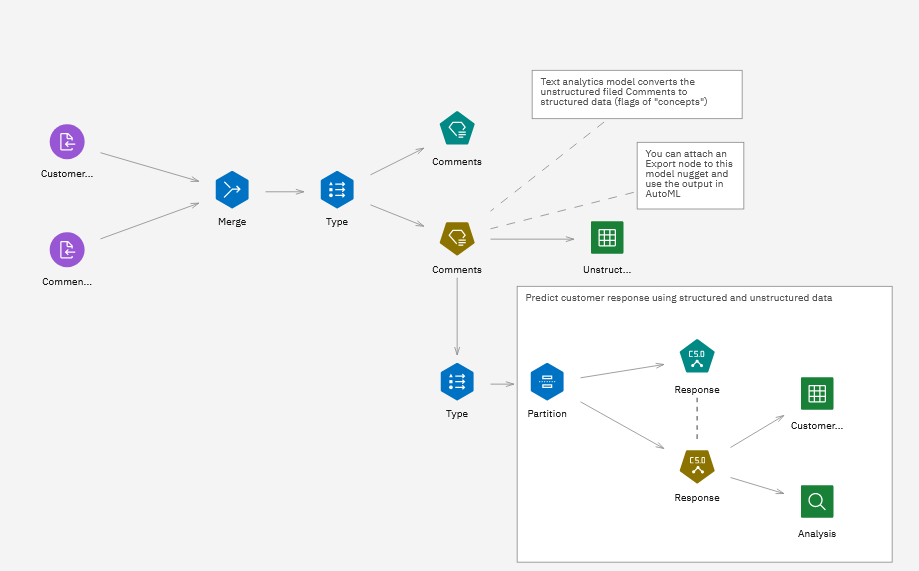
I**f you would like to use AutoML for this use case:**

1. Use the flow “as is” to derive segments. Run it interactively or schedule it to run as a batch job.
2. Merge customer segmentation data with the dataset that you would like to use in AutoML for building a classification model.
3. Build a classification model in AutoML.

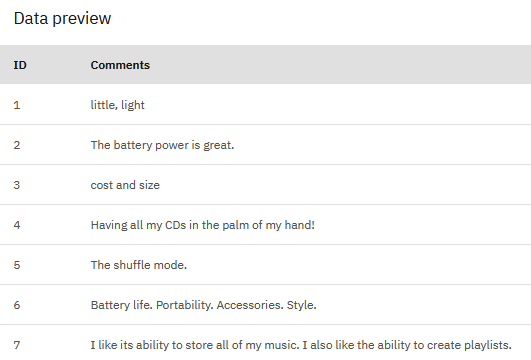
### Predicting Campaign Response with Structured and Unstructured Data

*Flow name: Campaign Response with Unstructured.str.*

Modeler flows includes comprehensive text analytics capabilities. In the sample flow we use just one capability – converting unstructured data to structured in order to use it as input for a classification model.



Preview the **Comments** data source and notice that it includes an unstructured field.



The **Comments** model is the *Text Analytics* model that “extracts categories” from this field. A “category” is a container for similar terms. The text analytics model is able to extract these concepts because Modeler includes pre-trained model for various domains. We used the *Customer Satisfaction Opinions* template to train this model.

Right click on the **Comments** model to view the concepts that will be extracted.

|  |  |
| --- | --- |
|  |  |

Run the **Table** node connected to the model. Categories get appended as flags (*T* = category present, *F* = category not present) to the right of existing fields (scroll all the way to the right to view the categories).

|  |  |
| --- | --- |
|  |  |

In this flow we also build a *C5* (classification) model that uses both structured and unstructured data that has been transformed by the Text Analytics model.

Right mouse click on the *C5* model and select **View Model**. In the details of the model click on **Top Decision Rules**. Notice that categories are used as for predictions.

|  |  |
| --- | --- |
|  |  |

**If you would like to use AutoML for this use case:**

1. Use the Text Analytics model to convert unstructured data to structured.
2. Export converted data and use it to build a classification model in AutoML.

*Note: We recommend that you take a class to get a full understanding of Text Analytics capabilities in Modeler.*

**You have finished reviewing sample use cases that can be implemented in Guided ML and AutoML.**

# Additional Resources

* **AutoAI documentation:** <https://www.ibm.com/support/knowledgecenter/en/SSQNUZ_3.5.0/wsj/analyze-data/autoai-overview.html>
* **Modeler Flows documentation:** <https://www.ibm.com/support/knowledgecenter/en/SSQNUZ_3.5.0/wsd/spss-modeler.html>
* **Modeler Flows Algorithms Guide:** <https://www.ibm.com/support/knowledgecenter/en/SSQNUZ_3.5.0/wsd/spss_algorithms.html>
* **Introduction to Data Science:**
  + IBM Coursera classes: <https://www.coursera.org/professional-certificates/ibm-data-science>