Lab 4: Introduction to Machine Learning with Watson Studio AutoAI

Get experience with **IBM Watson Studio**'s no-code ***AutoAI*** tool by creating a binary classification Machine-Learning model to evaluate the risk that a customer might leave your service.

Duration: 30 minutes

In this tutorial, you will use **IBM Watson Studio**'s **AutoAI** to train and deploy a machine-learning model with a no-code/low-code paradigm.

For those more inclined to tackle code, a stretch part of this hands-on lab will show how to use the notebook export feature of AutoAI

The data files and notebooks required to run this lab are located in the same box folder

Introduction

Objectives

In this hands-on lab, you will learn how to create, train and deploy a Machine Learning model with a no-code/low-code paradigm, leveraging Watson Studio's AutoAI tool.

Applying supervised Machine Learning

In supervised Machine Learning, a model is trained from historical data. The training phase is fed with records representing past observations of a dataset's behavior, and a model representing the behavior of this dataset is produced.

Telco *Customer Churn* use-case

The use case that we will tackle here is to predict whether a customer is likely to switch telephony operator or not. We have at our disposal a dataset, `customer_churn.csv`, which represents past observations of customers' `CHURN` behavior, alongside defining characteristics of each customer, such as demographics data on age, gender, number of children, and business domain data pertaining to its characteristics as a customer, such as plan, payment method, average usage, …

Once the model will have been trained, it can be used to predict the `CHURN` indicator for new customer records. This can then be used for example to orient actions to be taken when the customer is in contact with a call center, or to drive a marketing customer retention campaign.

Predictive Model setup

The value that we want to predict is represented by a string which can take two values or classes, `T` or `F`. This means that we are facing a *Binary Classification* type of ML problem.
There are many possible implementations of algorithms to solve Classification problems, and Watson AutoAI will help us determine the implementation which has the best accuracy for the training set.
Without going into details, each type of algorithm has several accuracy measurement indicators which can be chosen depending on the overall 'shape' and constitution of the dataset, as well as the intended use of the prediction.
For binary classification, a standard metric is called AUC-ROC (for Area Under Curve-Receiver Operator Characteristics), and measures how well a model is able to predict both Positives and Negatives. Other indicators sur as AUC-PR (AUC-Precision Recall) measures how well a model is able to identify Positives, even if Negatives prediction is less accurate.

Hands-on Lab overview

The instruction below will guide you in achieving the following tasks:

- Load a data set into the project
- Use IBM Watson Studio AutoAI to train, test, and evaluate a machine-learning model
- Deploy the trained model
- Use the deployed model to generate predictions on a new dataset

[A] Building a Predictive Model using AutoAI

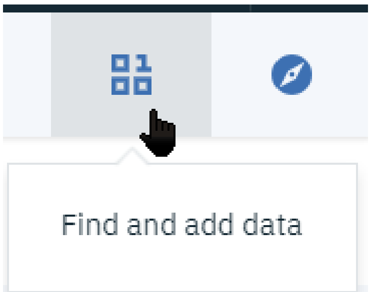
[A.1] Project setup

Switch to your `Workshop2020` project on Watson Studio in IBM Cloud.

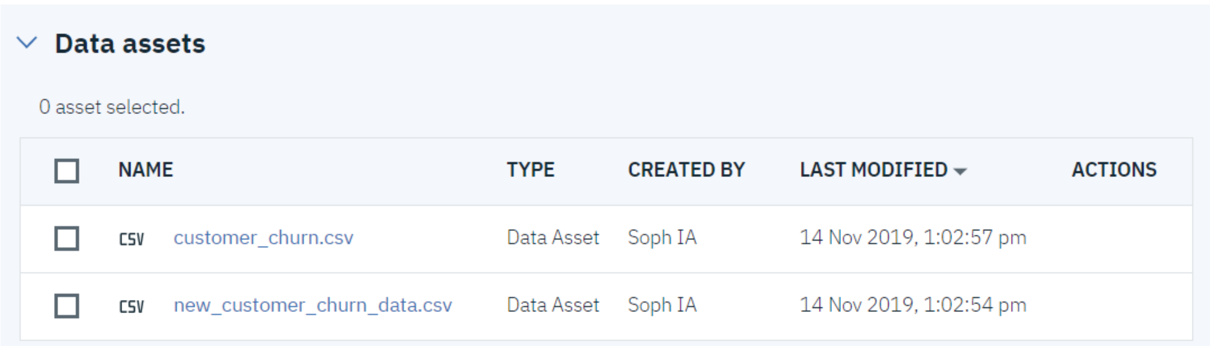
[A.2] Data Preparation for the Training phase

In this section, we will prepare the data to be used for model training and verification.
The data files have been stored in the Box folder, and should be downloaded to your laptop first.

1. In your project, switch to the `Assets` tab.
2. In this tutorial, you work with a data sets stored as project artifacts. **Click the Find and add Data** icon which looks like a `10`



   `01` button . It will open the file management sidebar.
3. From the **Load** tab, Click **Browse** to select from your local file system.
   Navigate to the lab files folder and select both `customer_churn.csv` and `new_customer_churn_data.csv` files and click **Open**.
4. The two files will now be listed in the Data assets section:



Data assets

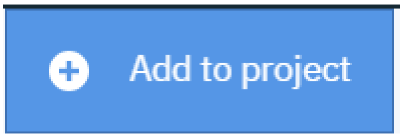
0 asset selected.

	NAME	TYPE	CREATED BY	LAST MODIFIED	ACTIONS
	CSV customer_churn.csv	Data Asset	Soph IA	14 Nov 2019, 1:02:57 pm	
	CSV new_customer_churn_data.csv	Data Asset	Soph IA	14 Nov 2019, 1:02:54 pm	

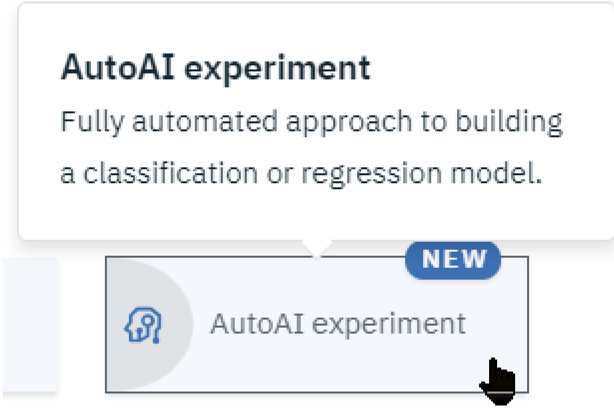
   Alternatively, you can drag and drop a file directly into the sidebar.
   The file is added to your local data sets in your project.

[A.3] Predictive Model training with AutoAI

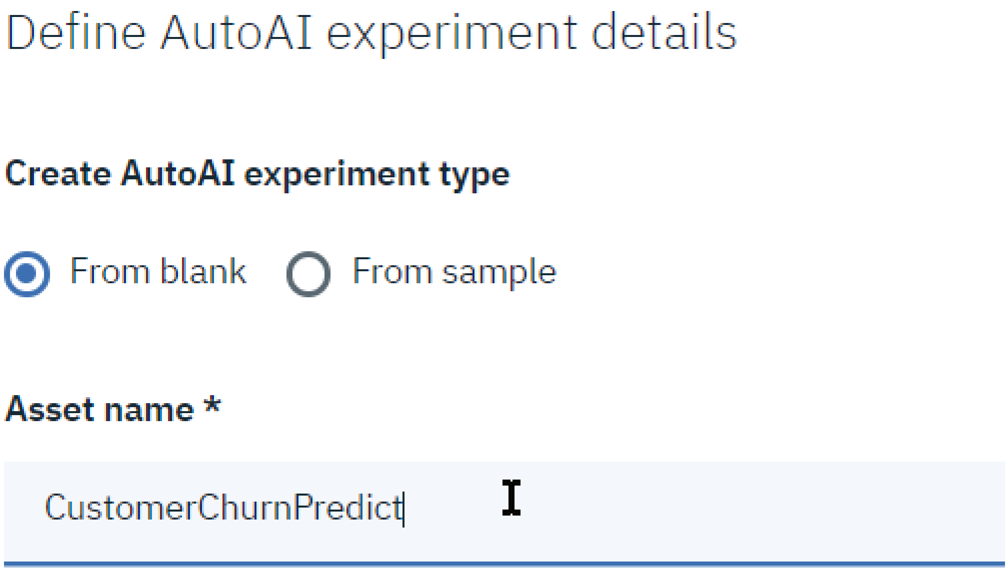
We will now create a model by using the IBM Watson Studio's AutoAI low-code model builder.



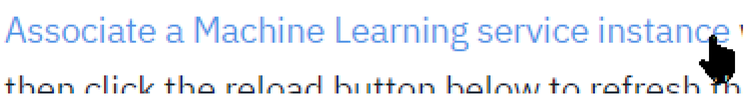
1. Use the `[(*) Add to Project]` button to bring up the artefact creation panel.



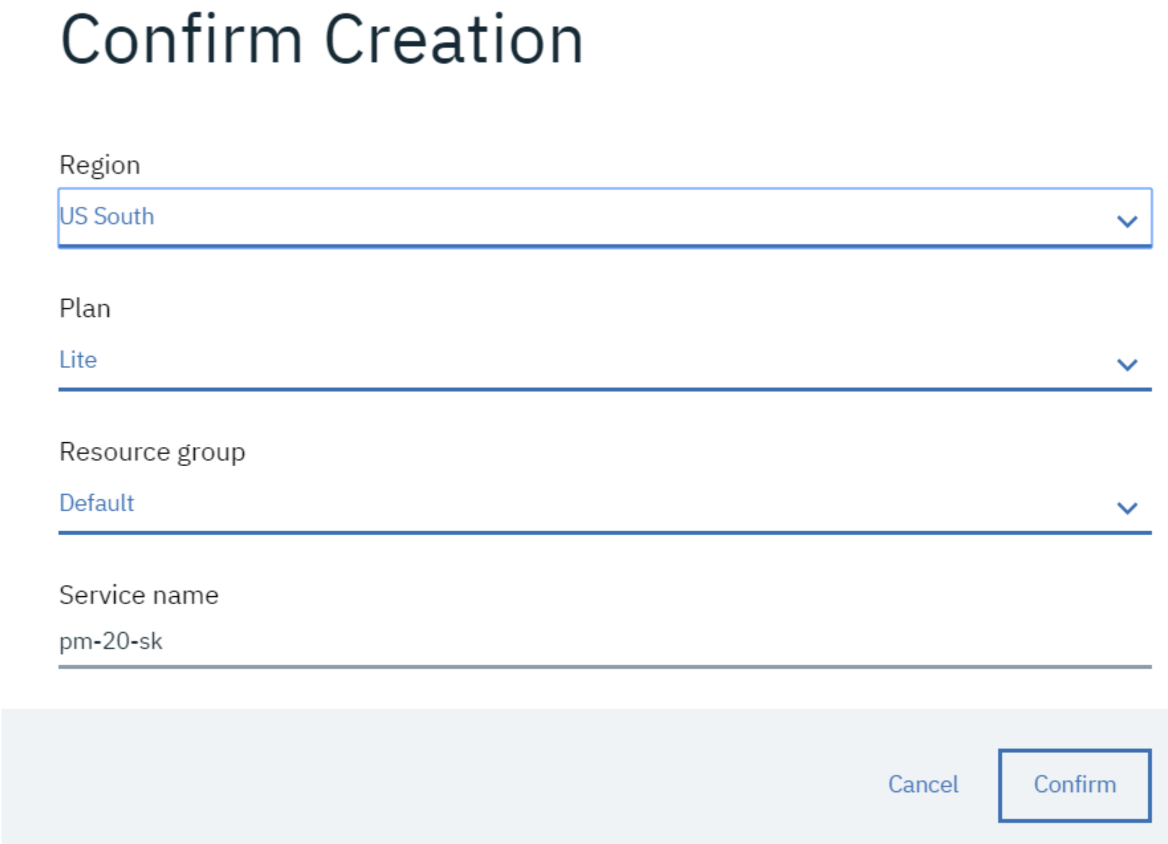
2. Create an **Auto AI Experiment** through the button
3. On the `Create an AutoAI experiment` page, enter a name for the model, `CustomerChurnPredict` for example



4. An **IBM Watson Machine Learning Service** is required:
   i. Click on `Associate a Machine Learning service instance`

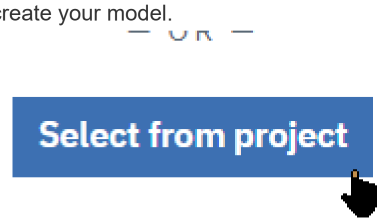


ii. Create a new Lite/Free plan instance:



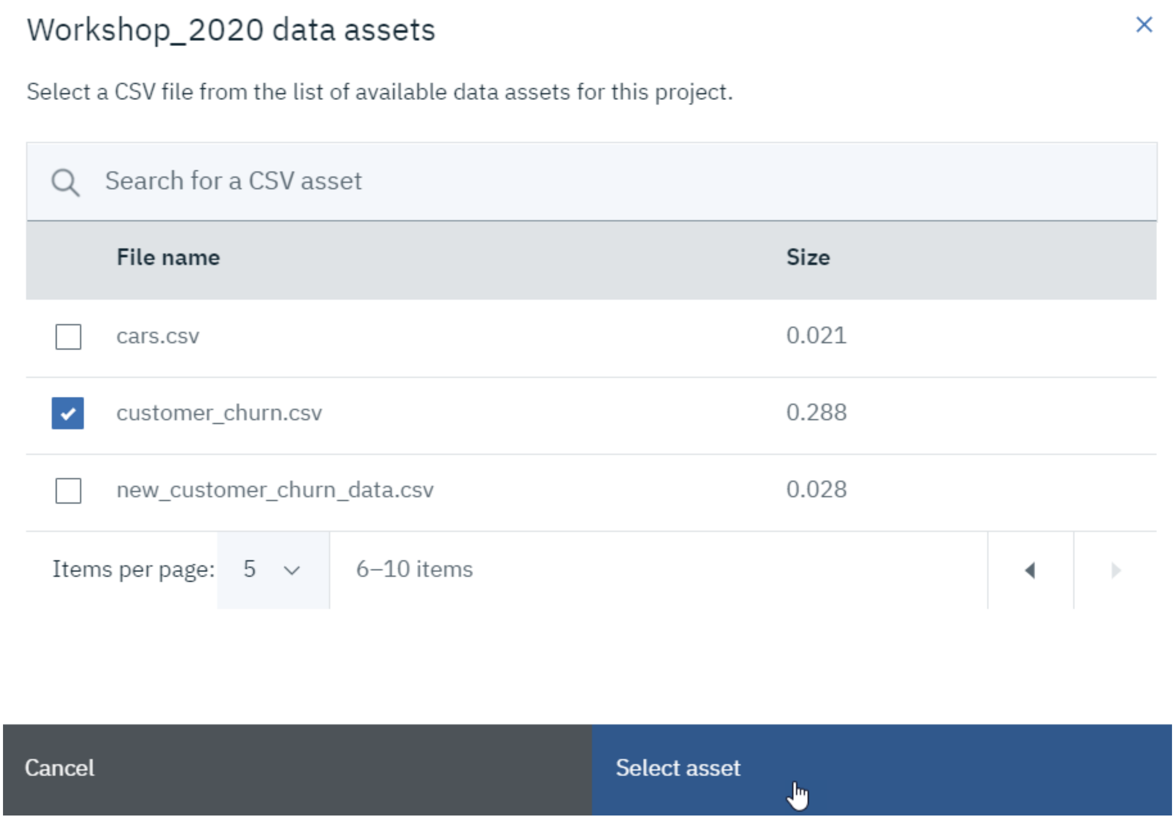
iii. Once created, switch back to the AutoAI creating tab and click `[Reload]` to select your newly created instance.

5. Click `[Create]` button at the bottom right
6. On the "Add data source" page, you can select the data asset to use to create your model.



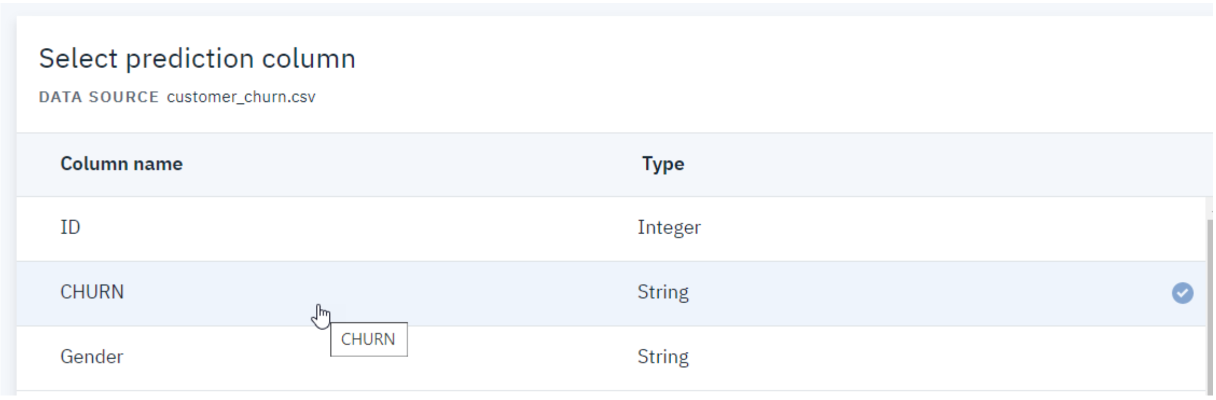
- Since we have uploaded the file as a data asset already, we'll use

- Select `customer_churn.csv` and `[Select asset]` button



- You could have used the browse button to upload `customer_churn.csv` if it had not been done earlier in this lab.

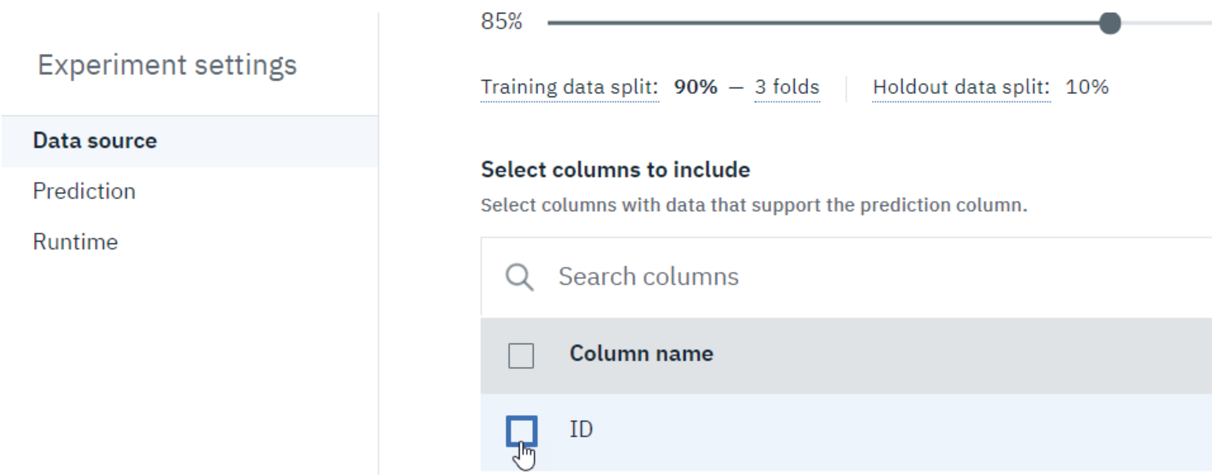
7. We will now configure the AutoAI input to drive the machine-learning predictive model construction.
From the **Select prediction column** list, select **CHURN**. This is the column that contains the historical observations and thus the outcome to predict



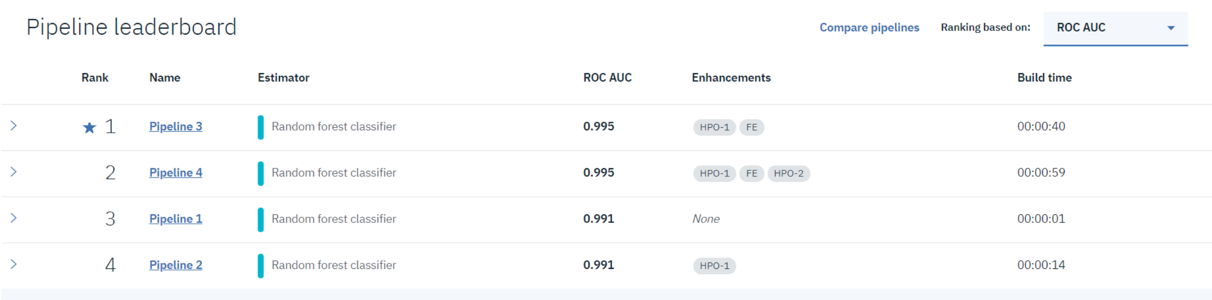
The columns contain the attributes on which the machine learning model will base predictions. All columns (features) that are not part of the prediction will be possible canditates for the prediction. We will see later on that AutoAI can help determine which ones are more pertinent than others.

8. As you can see, the AutoAI model builder selects **Binary Classification** by default as the type of model to build, because the `CHURN` column has been introspected and found to contain only two values, `T` and `F`. The model also selects `Accuracy` as the metric for model evaluation. You could change those defaults under the `[Experiment settings]` button.
9. We will review and modify the settings for the AutoAI training. Open the `[Experiment settings]` tab:

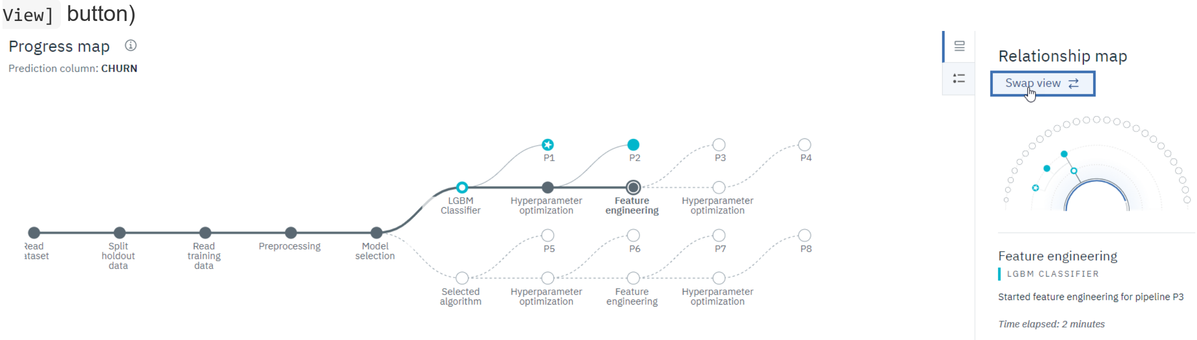
10. In the *Data source* tab, review the data split value, set at 90% by default.
11. The `ID` column is arbitrary, so we will want to remove it from the dependant variables to consider for prediction, unselect it in the list:



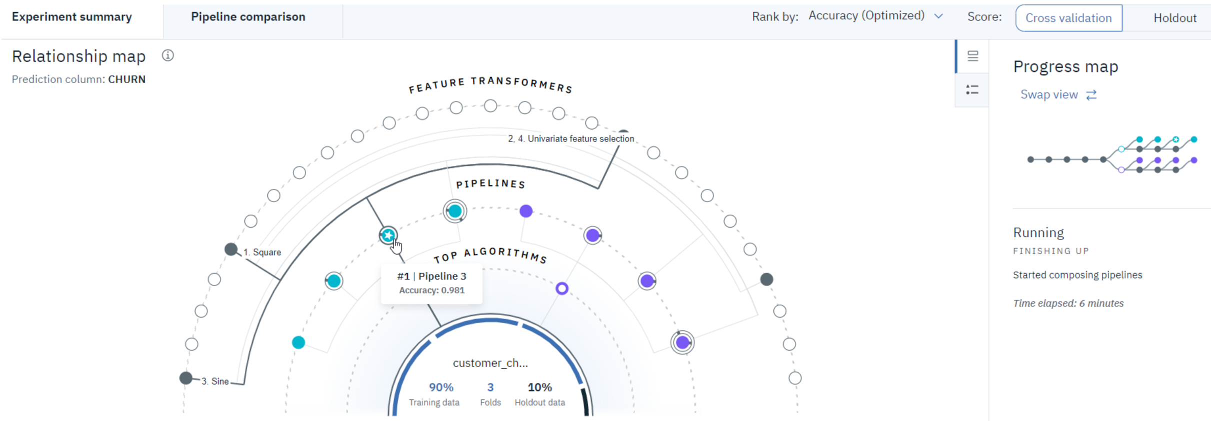
12. Switch to the `Prediction` tab. This is where the *Prediction type*, *Positive class*, *Metric* are selected. Scrolling down reveals the list of algorithm implementations that AutoAI will test, as well as the number of top algos (according to the selected metric) to push further through Hyper Parameter Optimization (HPO) and Feature Engineering (FE). The default is to retain 2.
13. Click `[Save settings]` to go back
14. Now click on the `[Run experiment]` button 

.
15. The canditate models will display in the pipeline leaderboard as they are evaluated 
Pipeline leaderboard — Compare pipelines — Ranking based on: ROC AUC

Rank	Name	Estimator	ROC AUC	Enhancements	Build time
★ 1	Pipeline 3	Random forest classifier	0.995	HPO-1 FE	00:00:40
2	Pipeline 4	Random forest classifier	0.995	HPO-1 FE HPO-2	00:00:59
3	Pipeline 1	Random forest classifier	0.991	None	00:00:01
4	Pipeline 2	Random forest classifier	0.991	HPO-1	00:00:14

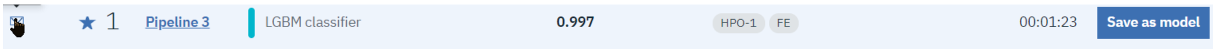
, and ranked according to the selected metric (ROC AUC here by default). The ROC (Receiver Operating Characteristic) and PR (Precision Recall) Area Under Curve (AUC) are metrics used to evaluate the accuracy of a model's true positive and true negative predictions, evaluated on the test subset. The closer they are to 1.0, the better the sensitivity of the model.
16. This will trigger **AutoAI**'s evaluation of the possible algorithms implementations and their configurations in order to select the best fitting one. The model evaluation and selection process is displayed as it executes (you may want to use the `[Swap View]` button)


17. An animation shows the various steps while evaluating the different model algorithms and configurations. Each attempt is a different pipeline as illustrated in the diagram.
18. Once completed, the relationship map shows the various paths that have been explored, highlighting the most accurate one as **Top performer**, hovering over the corresponding icon will reveal the selected algorithm (here, *LGBM CLassifier* here), and which *Feature Engineering* has been applied



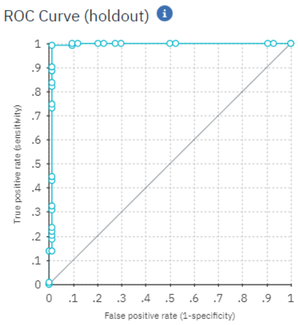
. You can hover over the diagram's elements to get a summary.

19. We can get more details on a given candidate model by expanding it:



Model evaluation measures

	Cross validation score	Holdout score
Accuracy	0.980	0.981
Avg Precision	0.994	0.992
F_1	0.976	0.976
Log Loss	0.065	0.077
Normalized Gini Coefficient	0.962	0.971
Precision	0.973	0.965
Recall	0.979	0.988
ROC AUC	0.997	0.989

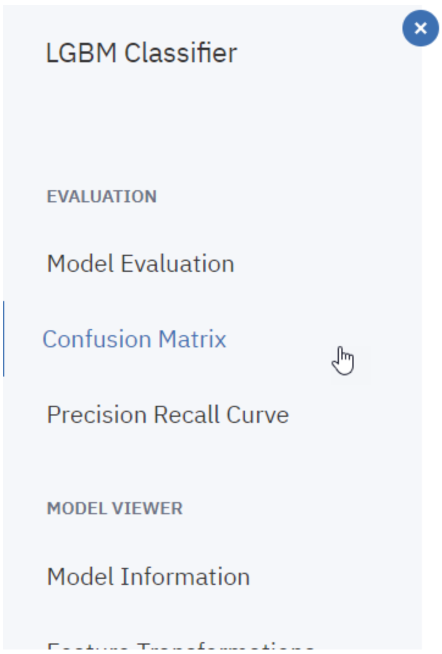


. You will notice the AUC ROC plotout, and the metric computations on the cross-validation data set, which has been used for training, as well as the holdout, which has never been seen by the training and gives an idea of the actual accuracy of the model.

20. Looking deeper into the selected model yields more insights on your data set. Select the top pipeline

.

21. This opens the model details. Several tabs are of particular interest:
   - The *Confusion Matrix*



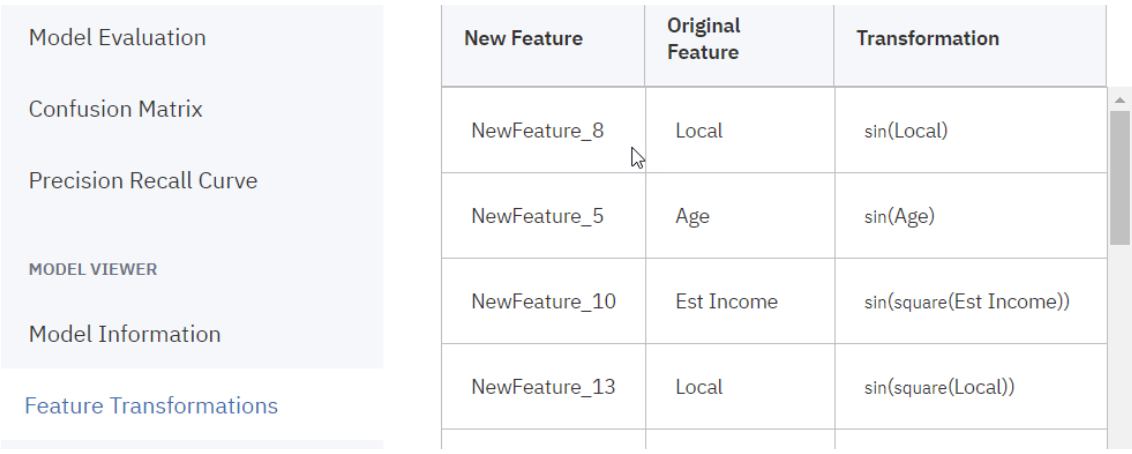
Confusion Matrix

TARGET : CHURN

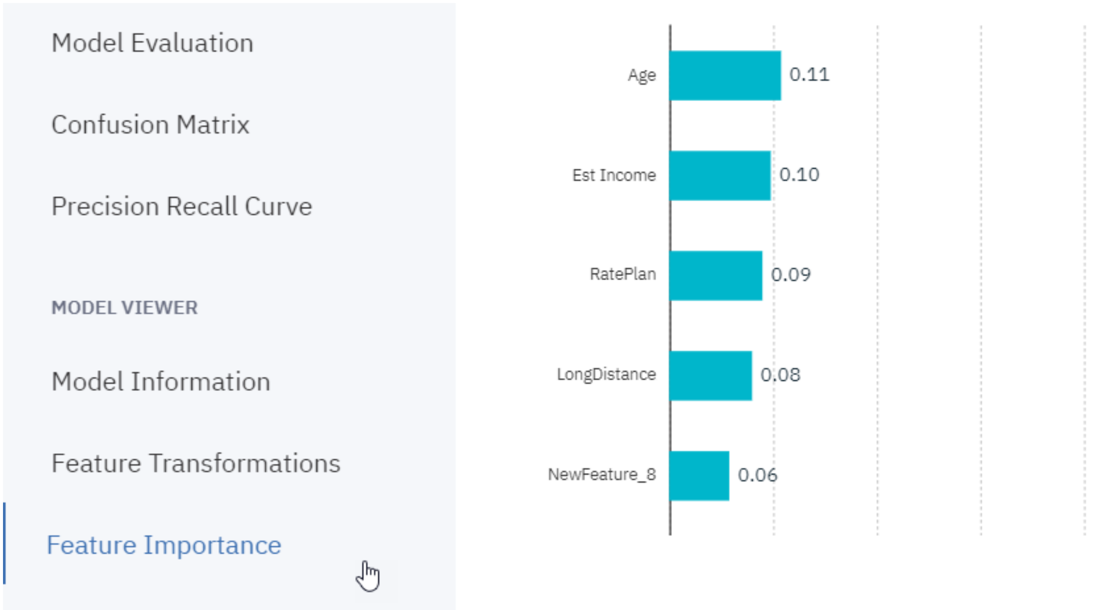
Observed	Predicted F	Predicted T	Percent Correct
F	122	1	99.2%
T	1	83	98.8%
Percent Correct	99.2%	98.8%	99.0%

shows more details on the model evaluation on the Holdout set. Here we can see that out of the 10% of rows set aside for the holdout set, in this case 206 rows, only 2 have been predicted wrong, for an overall accuracy of 99.0%.

- The *Features Transformations*

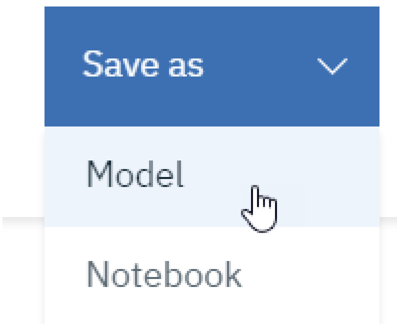
  

  tab shows which features **AutoAI** has generated, ranked by order of importance.
- The *Feature Importance*

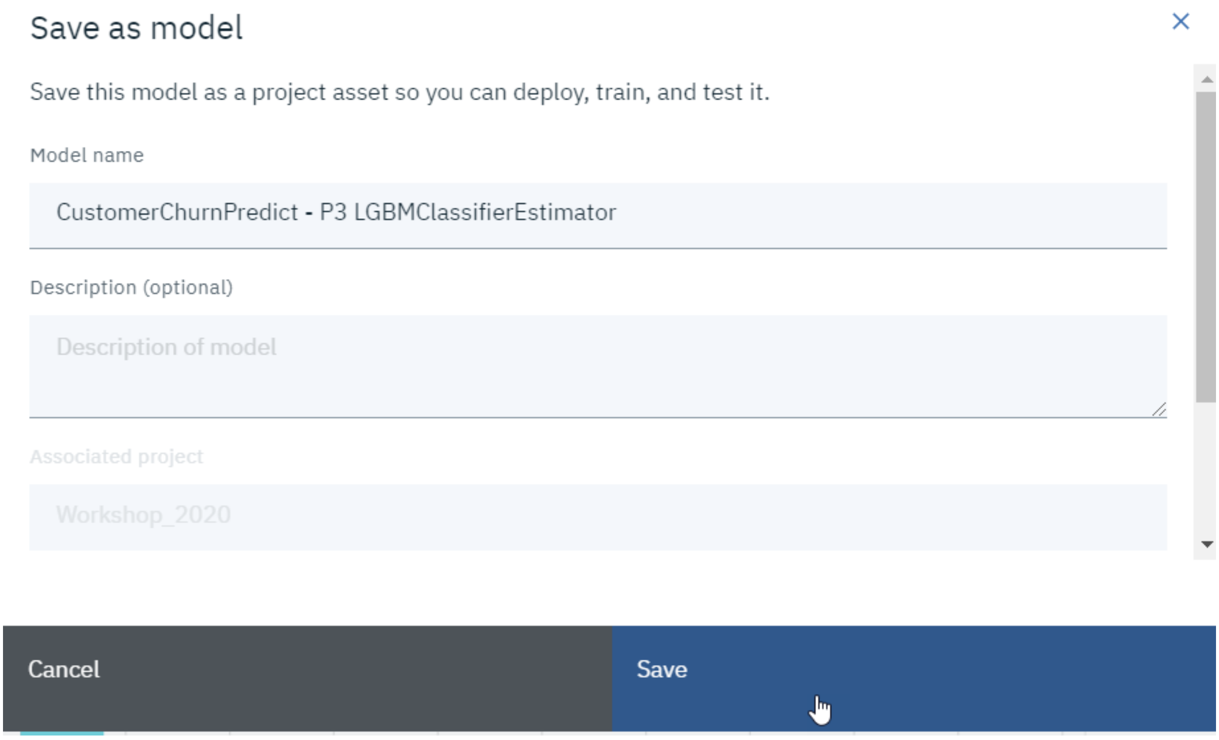
  shows which factors influence the most the `CHURN` target, in this case `Age`, `Est Income`, `RatePlan` account for 11%, 10% and 9% of the predictive power. `Newfeature_8` is a synthetic feature which accounts for 6% in the prediction.

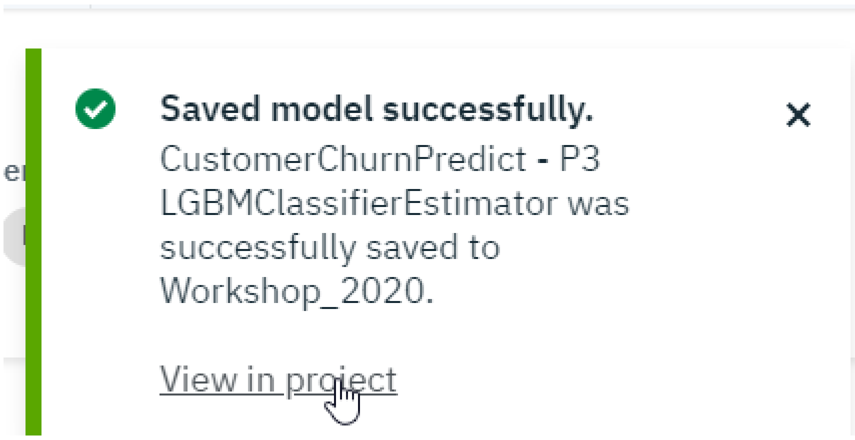
22. Lastly, we will select the best performing model according to the selected metric, here *Accuracy*, and use the `[Save`



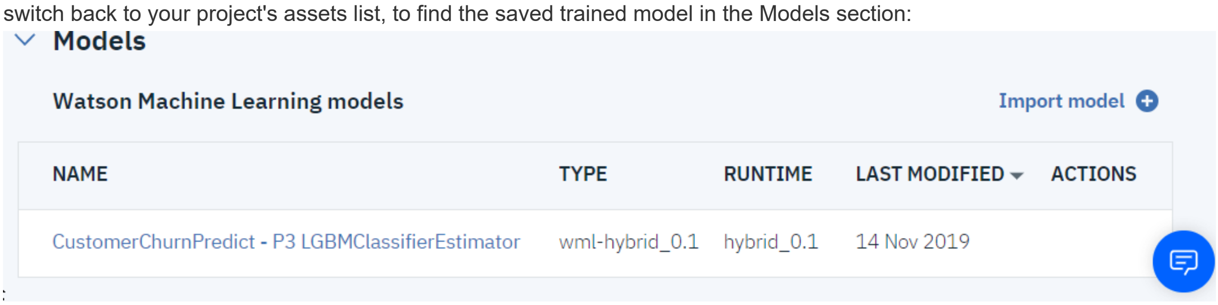
`as>model]` button to retain it

23. When you're prompted to confirm, click **Save** again





24. Use the *View in project* link or switch back to your project's assets list, to find the saved trained model in the Models section:



:

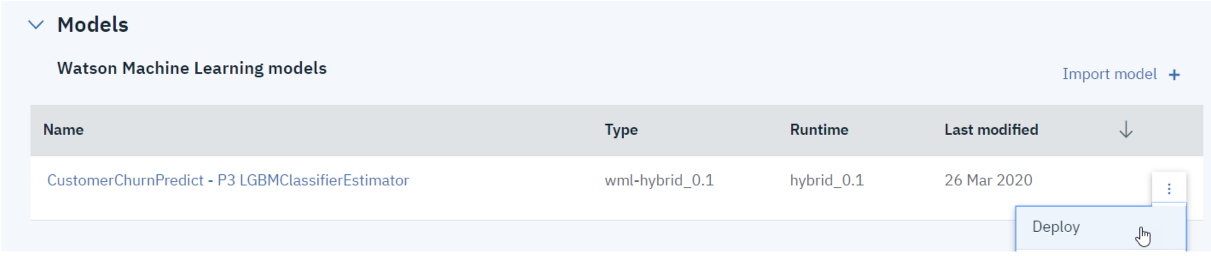
> **You now have a trained model, next you will deploy the model to test on out-of-sample data.**

[B] Deploy and test the trained model

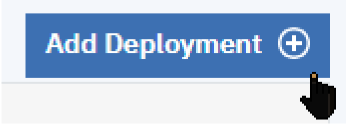
Before you can use your trained model to make predictions on new data, it must be made accessible from an Application Programming Interface (API).
This step is called **Deployment**, and it hinges between development and operations.

[B.1] Deploying the AutoAI predictive model

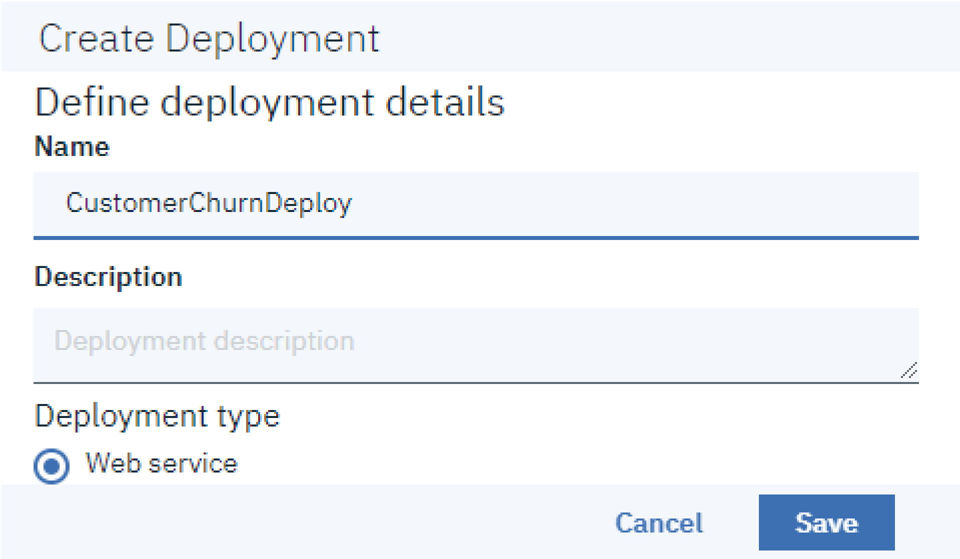
1. From the models list, locate your model and select the `Deploy` action



2. You are taken to your model's *Deployments* tab. Click **Add Deployment (+)** link on the upper-right section of the pane

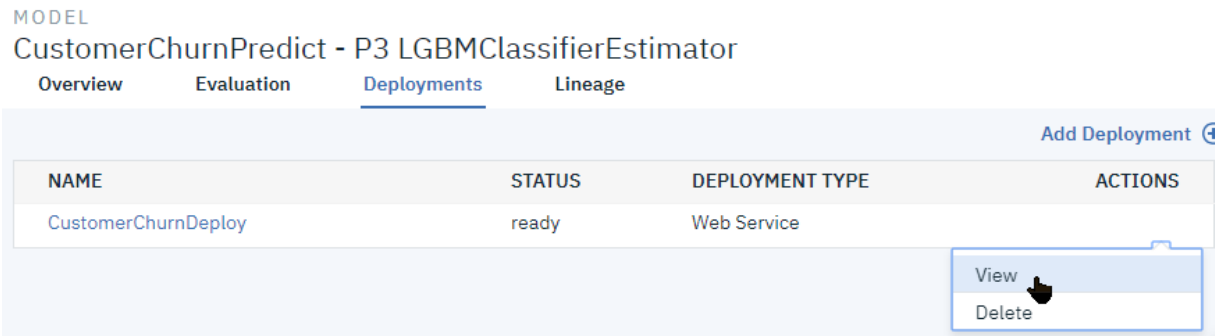


3. On the **Create Deployment** page, give a name and a description to your deployment, e.g. `CustomerChurnDeploy` .
4. Clicking `[Save]` will deployed the model as a REST endpoint, as defined by **Web service** selection:



.

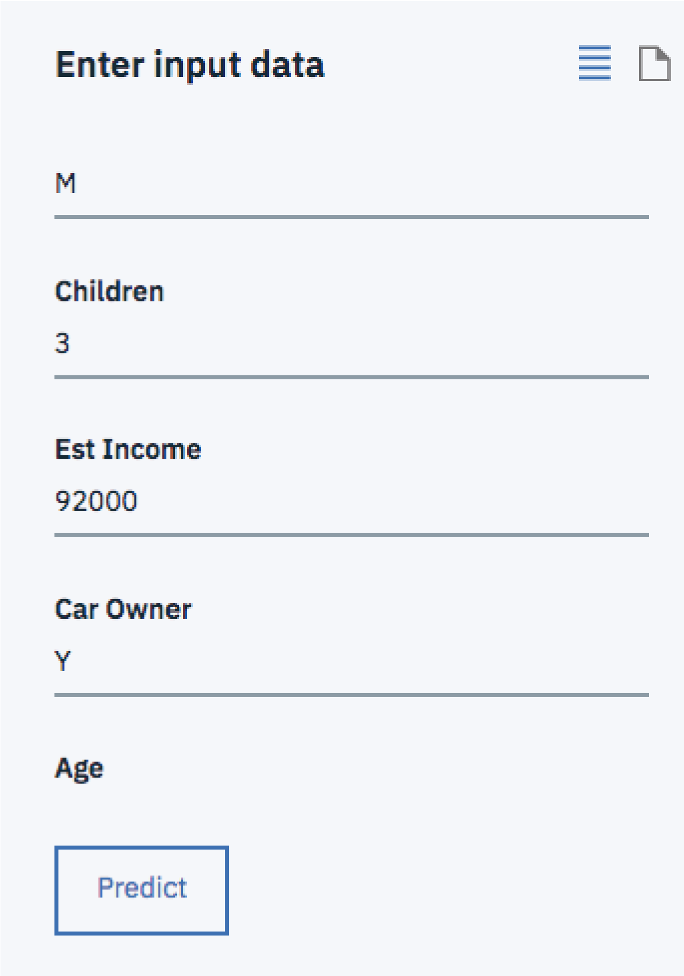
5. When model deployment is complete, the STATUS turns to `ready` , and from the **Actions menu**, click **View**:



If you are a developer, you may want to review the information in the **Implementation** tab.

> The **Code Snippets** shows examples of how to use the REST endpoints from several environments, and can be passed on to you application developer to integrate the deployed model into a business application. We'll come back to it in the last section of this lab.
>
> 1. We are now able to the model prediction: go to **Test** tab.

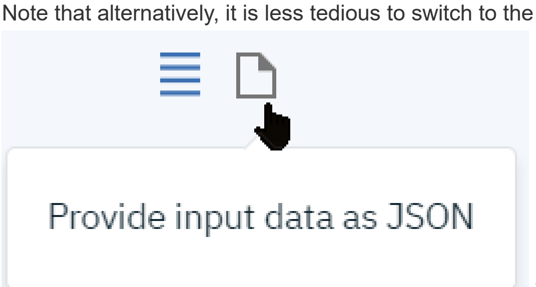
2. Enter data in all the fields for a sample record from the data set.



For example, get data values from one line of the `new_customer_churn_data.csv` file, e.g:

```
8
F
M
0
19732.80
N
50.67
24.81
0
22.44
0
CC
FreeLocal
Standard
47.25
3
```

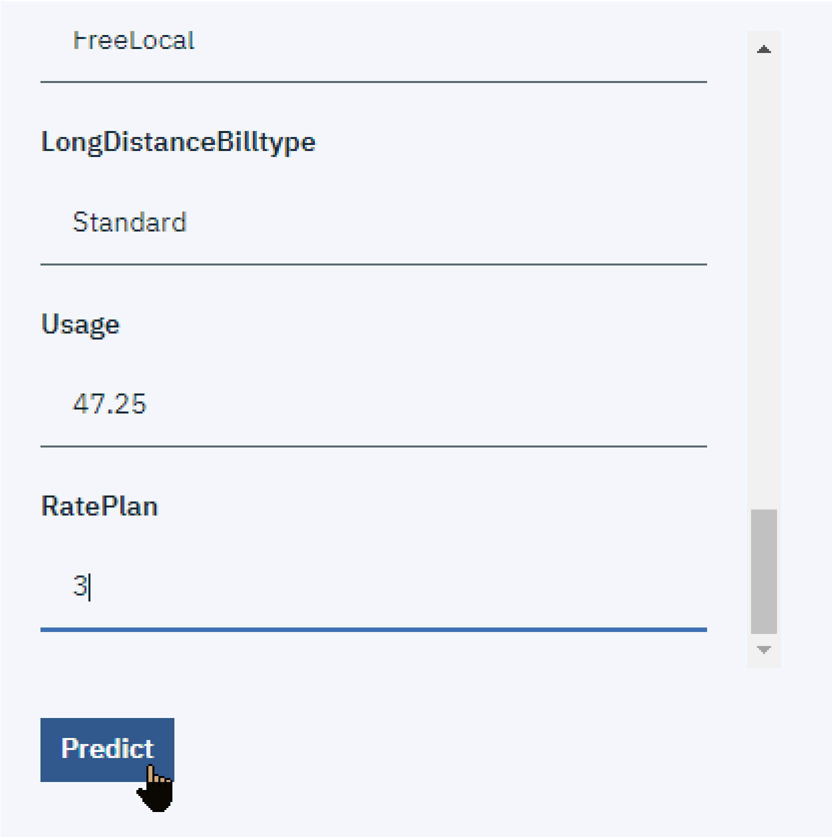
1. Note that alternatively, it is less tedious to switch to the raw JSON input using



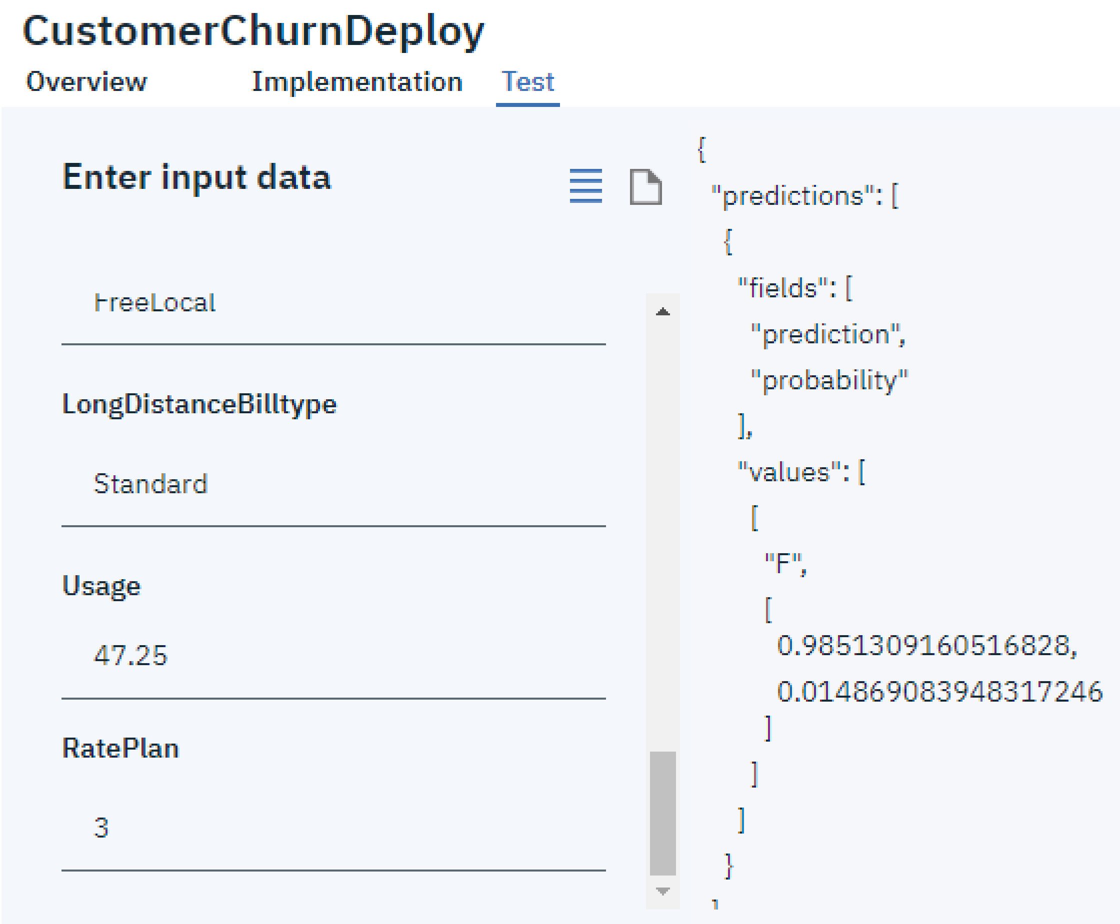
and copy the data from here and paste into the `Input JSON Payload data`:

```
{"input_data":[{"fields": [ "ID", "Gender", "Status", "Children", "Est Income", "Car Owner","Age", "LongDistance", "Int
```

1. To test the model and make a `CHURN` prediction on this data, click the `[Predict]` button:



2. The system will invoke the Watson Machine Learning REST endpoint for the AutoAI model, you will get the resulting prediction buffer, looking like:



Here the prediction is that customer `CHURN` is `F` with a 98.5% probability.

3. Additionally, you can test several other records taken

[S-A] Optional Stretch Lab A: invoke the WML model's REST API from Python

> **Note**: This section assumes that you have some proficiency in application development and will be comfortable dealing with some code!

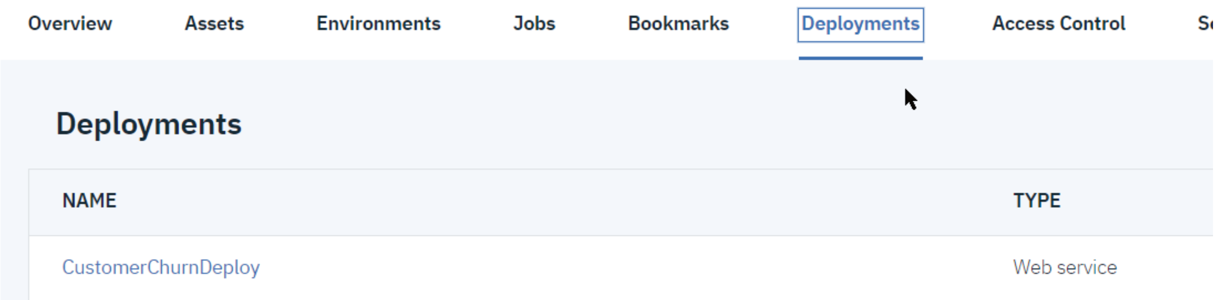
The model just deployed can be invoked from any development environment that supports invoking REST endpoint, in the world of Data Science, the first intent will often be a Jupyter notebook coded in the Python language.

This is implemented in the `Lab-Stretch-RunModelFromNotebook.ipynb` notebook. This can be a first step towards integrating a ML model into an application.

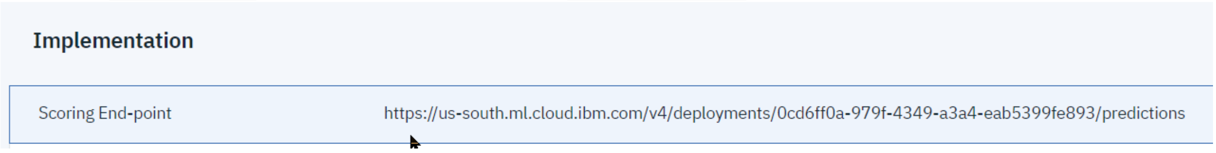
Before creating the notebook, you will need to take note of the WML scoring REST endpoint for your model, and the Watson Machine Learning Service credentials:

[S-A.1] REST Scoring endpoint

1. Select the **Deployments** tab in your project



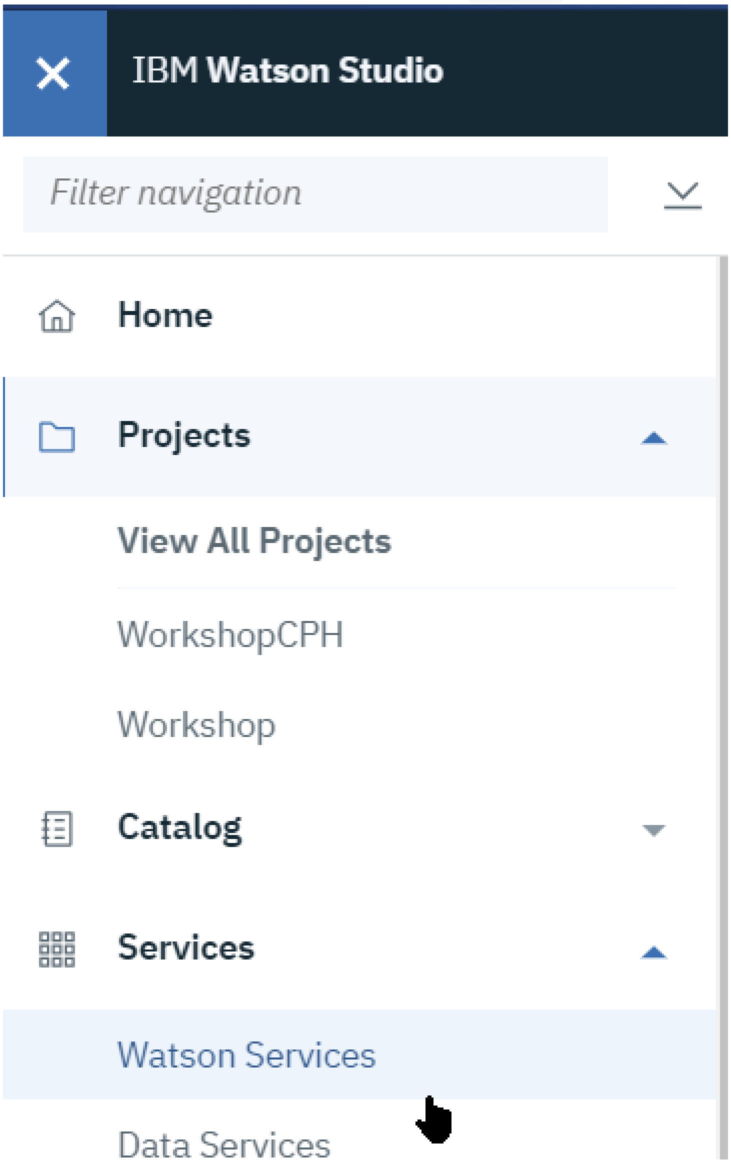
2. Select the `CustomerChurnDeploy` created earlier, and switch to the
3. Select the `Implementation` tab, and copy the value of the `Scoring End-point` :



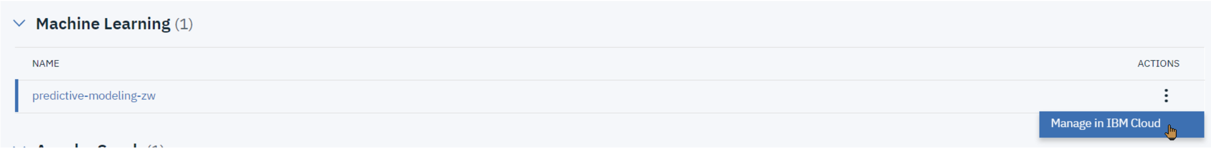
4. Paste this value in e.g. a notepad file on your laptop.

[S-A.2] Watson Machine Learning credentials

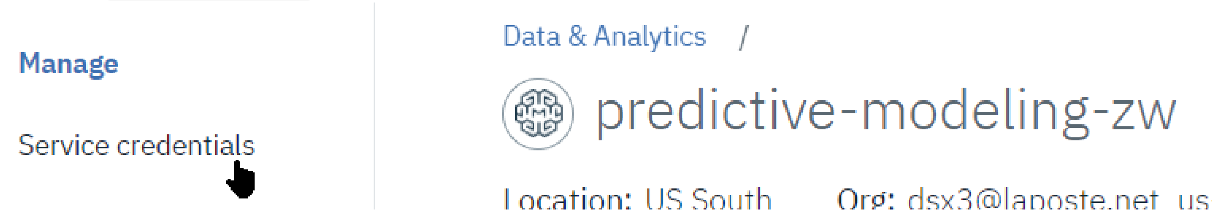
1. From the Hamburger menu (top left), select the `Services` menu and then the `Watson Services`



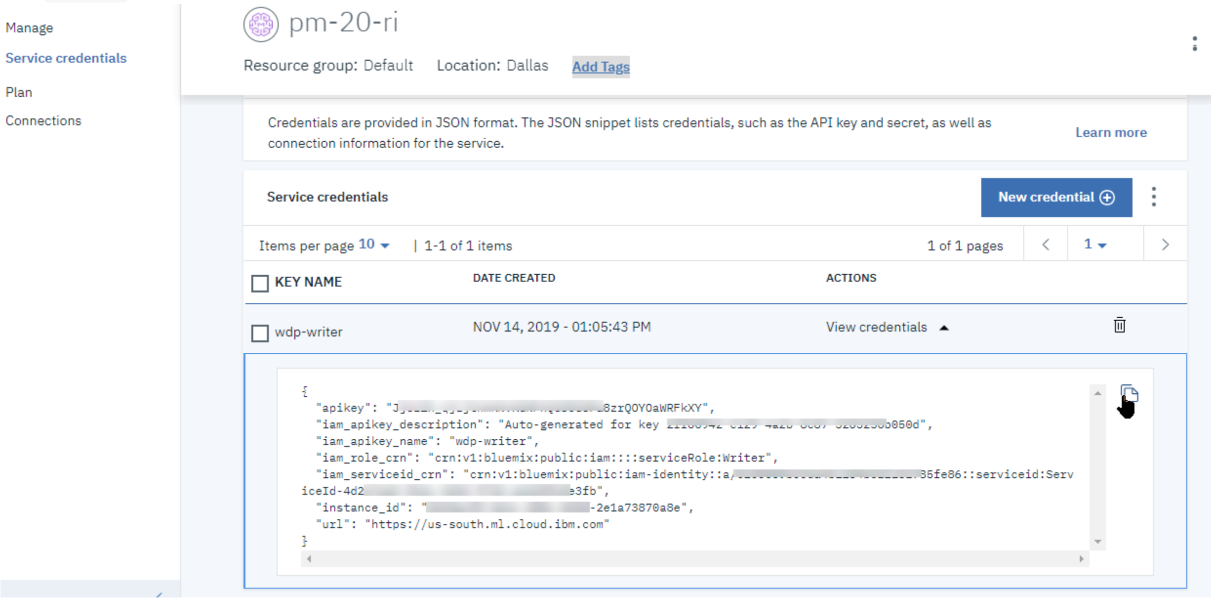
2. In the `Machine Learning` section, locate your service, and select `Manage in IBM Cloud` from its menu



3. Select the `Service Credentials` tab



4. Expand `View Credential`, and use the copy button (top right), and then paste them to a text file on your computer (using e.g. `notepad` on windows)



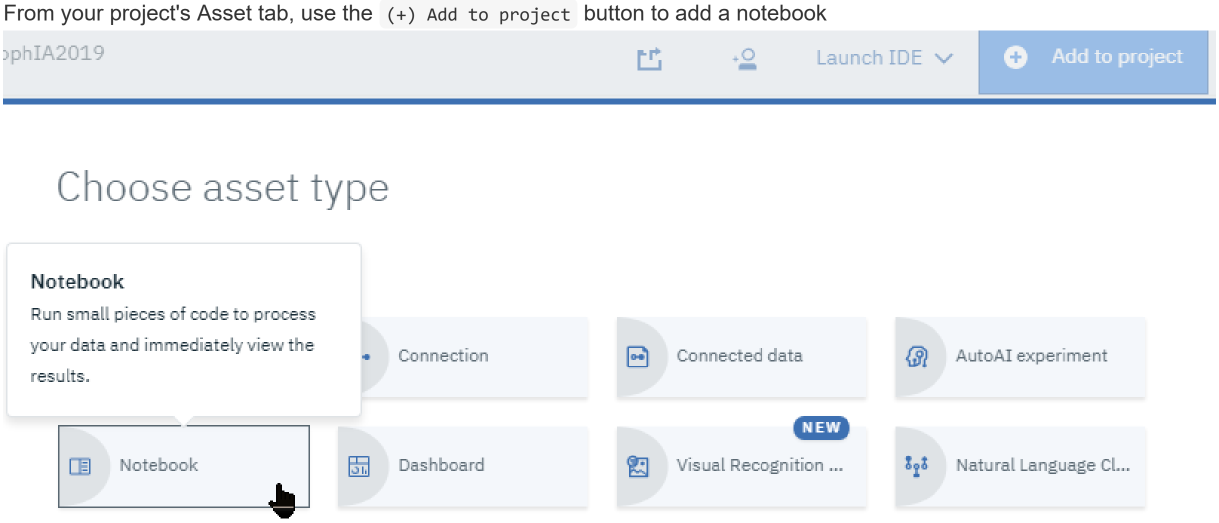
.

We will use these values in a python notebook shortly.

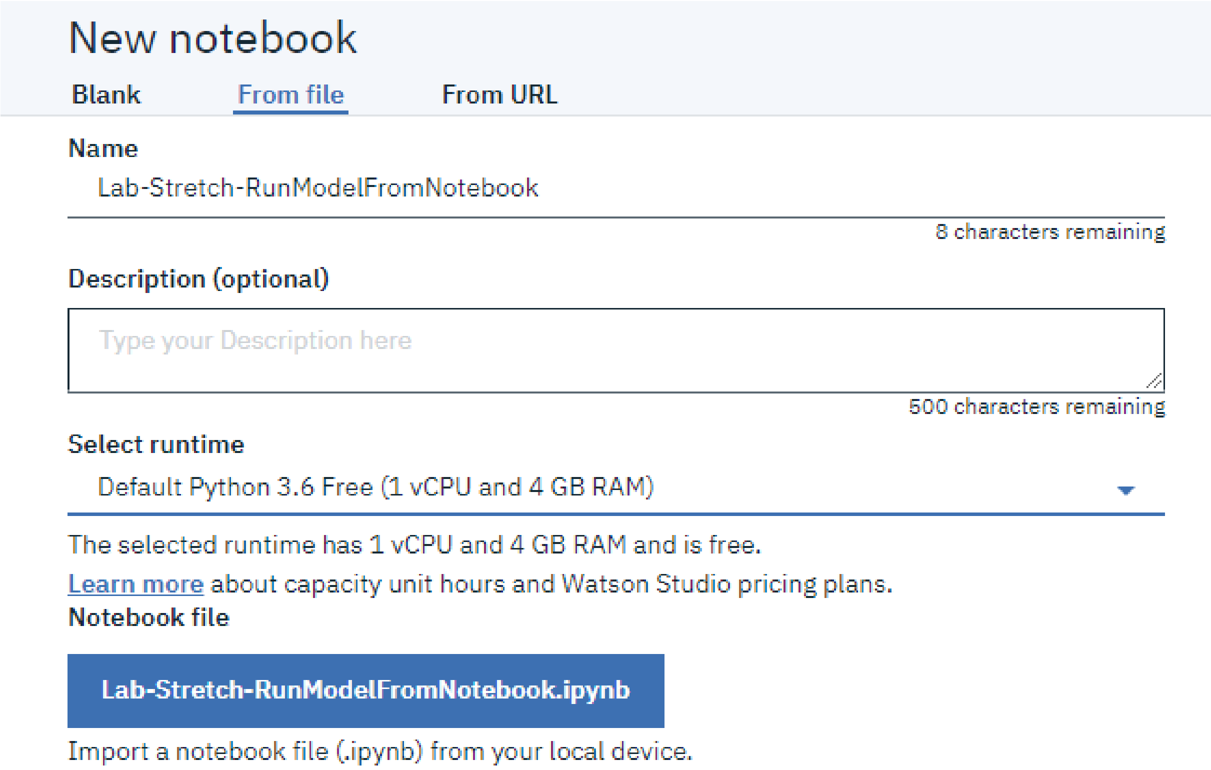
[S-A.3] Creating the notebook

Now you can switch back to Watson Studio and add a notebook from file:

1. From your project's Asset tab, use the `(+) Add to project` button to add a notebook



2. Use the `From file` tab and `[Choose file]` button to load the `Lab-Stretch-ScoreModelFromNotebook.ipynb` notebook file
3. Select a `Default Python 3.6 Free` environment from the



4. Click `[Create Notebook]` , and follow the instructions within the notebook.

[S-A.4] Executing the notebook

Instructions are within the notebook itself, in comment cells.

In essence, you will execute the notebook's code cells using the `[(>) Run]` button.
Minimal changes will be required to specify the proper REST endpoint URL and WML credentials.

[S-B] Optional Stretch Lab B: generate a notebook for the AutoAI training pipeline

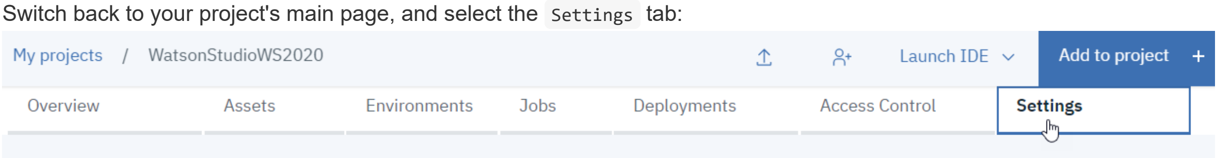
A recently GA-ed feature of AutoAI is the ability to generate a `sklearn` Jupyter notebook which implements the Auto-AI generated training pipeline.

In this lab, we will generate the notebook and add a few code cells to export the model to a file stored as a project data asset, and reload it in another notebook for execution.

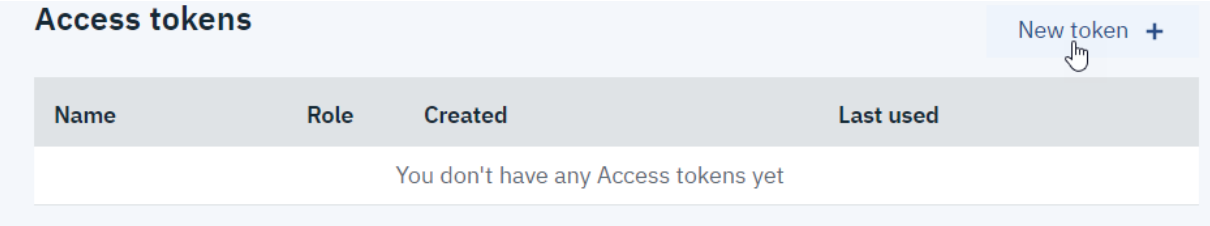
[S-B.0] Generate a project access token

We will use the `project-lib` API to access project storage, and it needs to be enabled first:

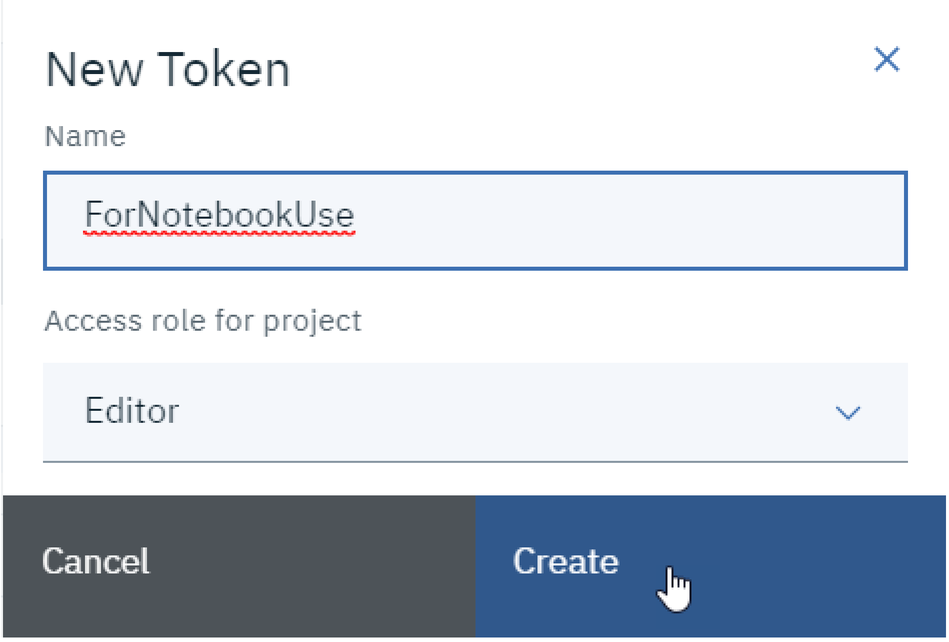
1. Switch back to your project's main page, and select the `Settings` tab:



2. Scroll all the way down to the **Access tokens** section, and select `[New token]` :

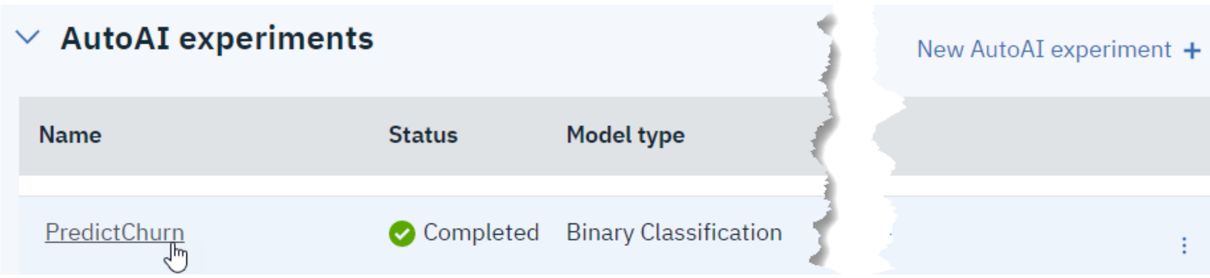


3. Give the token a name, select role as *Editor* and `[Create]`



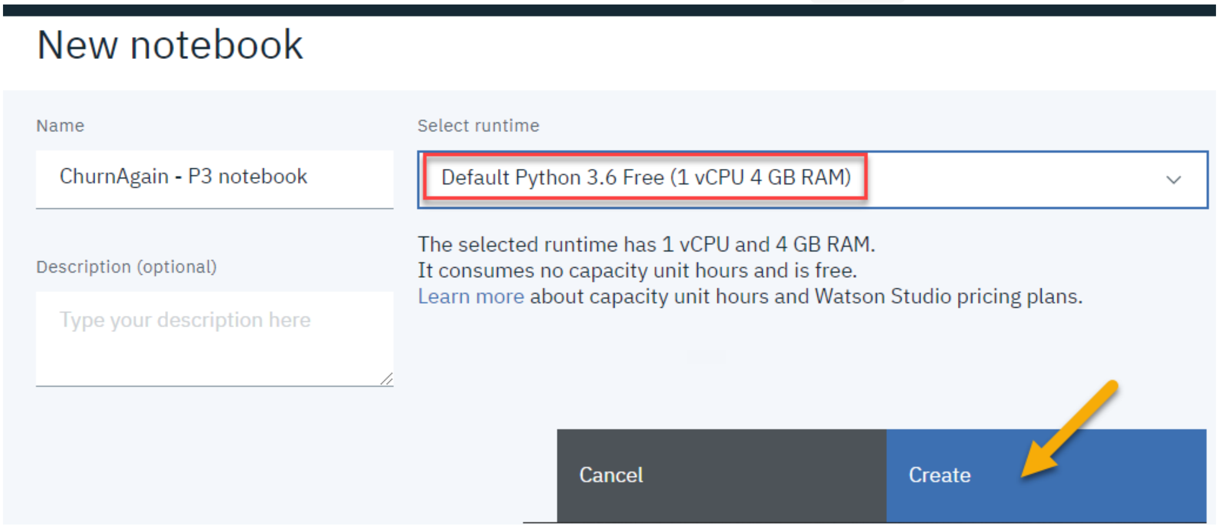
[S-B.1] Generate the `sklearn` pipeline notebook

1. Switch back to your project's assets list, locate your AutoAI Churn experiment and open it:



2. In the Pipeline leaderboard, select the best-performing pipeline's `Save as/Notebook` menu:

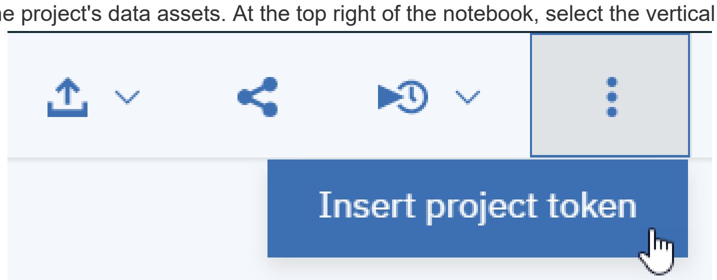
3. Validate the notebook name. You may want to select a Free runtime, then click the `[Create]` button:



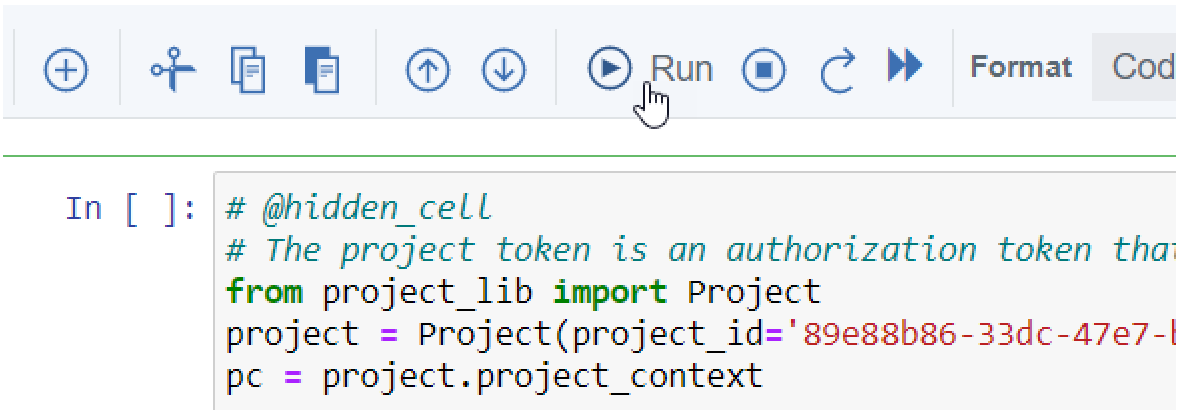
4. A new notebook is created in your project. The code is a transcription in `sklearn` framework of the selected Auto-AI pipeline. You can examine the code, which can feel pretty dense. A cell towards the end trains this pipeline with `pipeline.fit(X,y)` , and then goes about scoring and cross-validating.
5. Run the notebook, up to the last cell that displays the cross-validation results (`cv_results`) At this stage, your kernel contains a `preprocessingpipeline` and a `pipeline` variables which are trained sklearn pipelines. We will export them to a model files in pickle format.

[S-B.2] Add code to save the trained `sklearn` pipelines to file

1. We will use the `project-lib` API to deal with the project's data assets. At the top right of the notebook, select the vertical



ellipsis icon, and then `[Insert project token]` :
2. This inserts a cell at the top of the notebook with code to initialize the project-lib API and create the `project` variable. Execute that cell:



3. Now add a cell all the way down at the end of the notebook. Type the following code:

```
import pickle
project.save_data('Churn_P3_pre.pickle',pickle.dumps(preprocessing_pipeline),overwrite=True)
project.save_data('Churn_P3.pickle',pickle.dumps(pipeline),overwrite=True)
```

1. Execute the cell, it will report the last saved file:

```
In [34]: import pickle
         project.save_data('Churn_P3_pre.pickle',pickle.dumps(preprocessing_pipeline),overwrite=True)
         project.save_data('Churn_P3.pickle',pickle.dumps(pipeline),overwrite=True)

Out[34]: {'file_name': 'Churn_P3.pickle',
          'message': 'File saved to project storage.',
          'bucket_name': 'watsonstudiows2020-donotdelete-pr-gyityavxovtrbp',
          'asset_id': '8f375da8-b30c-4fa2-a26c-8e22ef550a4e'}
```

[S-B.3] Load and use the pipelines for scoring

We will reload the scoring pipelines and use them from another notebook:

1. Create a new notebook, with name e.g. `Score sklearn`
2. Insert project token and execute cell
3. Add and execute a cell with the following code:

```
# Import the transformer and scoring pipelines from their pickle export files
import pickle
preprocessingpipelineNew = pickle.load(project.get_file('Churn_P3_pre.pickle'))
pipelineNew = pickle.load(project.get_file('Churn_P3.pickle'))
```

This loads the two pipelines from their file storage.

1. Add a cell which loads the test data from `new_customer_churn_data.csv` file

```
# Load the new customer data to score
import pandas as pd
dfNew = pd.read_csv(project.get_file('new_customer_churn_data.csv'))
```

1. And finally, use the two pipelines to score this new data:

```
# Transform and score using the loaded pipeline
X_new=preprocessingpipelineNew.transform(dfNew.values)
pipelineNew.predict(X_new)
```

You will get an output array with the model predictions for the new data:

```
In [2]: # Import the transformer and scoring pipelines from their pickle export files
        import pickle
        preprocessingpipelineNew = pickle.load(project.get_file('Churn_P3_pre.pickle'))
        pipelineNew = pickle.load(project.get_file('Churn_P3.pickle'))

In [5]: # Load the new customer data to score
        import pandas as pd
        dfNew = pd.read_csv(project.get_file('new_customer_churn_data.csv'))

In [6]: # Transform and score using the loaded pipeline
        X_new=preprocessingpipelineNew.transform(dfNew.values)
        pipelineNew.predict(X_new)

Out[6]: array(['T', 'T', 'F', 'F', 'T', 'F', 'T', 'F', 'F', 'T', 'T', 'T', 'F',
               'T', 'F', 'T', 'F', 'F', 'F', 'T', 'F', 'F', 'T', 'T', 'T', 'F',
               'F', 'F', 'F', 'F', 'F', 'F', 'T', 'F', 'F', 'T', 'T', 'F', 'T',
               'F', 'T', 'F', 'F', 'T', 'F', 'F', 'T', 'F', 'F', 'F', 'F', 'T',
               'F', 'T', 'F', 'T', 'F', 'F', 'T', 'F', 'F', 'F', 'F', 'F', 'T',
               'T', 'T', 'T', 'F', 'T', 'F', 'F', 'T', 'F', 'F', 'F', 'F', 'F',
               'T', 'F', 'T', 'T', 'T', 'T', 'T', 'T', 'T', 'F', 'T', 'T', 'F'
```

Conclusion

You completed the lab for Training and Deploying a model using the **IBM Watson Studio**'s ***AutoAI*** model builder.