Data & AI – Personalized Customer Experience with AI

End to End AI Hands-On Lab
Telco Customer Churn Aware Virtual Assistant

Table of Contents

Disclaimer

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.
The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

The development, release, and timing of any future features or functionality described for our products remains at our sole discretion I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results like those stated here.
Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.
IBM products are manufactured from new parts or new and used parts.
In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."
Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.
Performance data contained herein was generally obtained in controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.
References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.
Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.
It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Introduction

In this lab you will learn to develop a complete end-to-end solution for Telco Customer Churn prediction using IBM's data science portfolio on IBM Public Cloud. This lab would cover

- Training and testing the machine learning (ML) model for predicting customer churn.
- Deploying the ML model as a web service
- Building a virtual agent for customer support.
- Integrating the virtual agent with churn prediction model for a personalized end user experience.
- Creating visualization dashboards using Cognos Analytics for better insights.

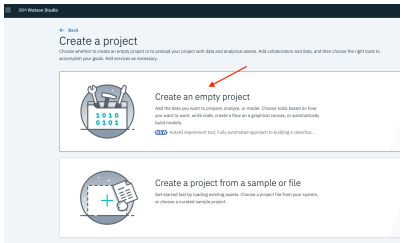
1. Pre-Requisites

This hands-on lab illustrates how to train, test, and deploy machine learning models for customer churn prediction using IBM Public Cloud environment. You will need to setup and access an IBM Public Cloud account to be able to complete this lab.
Please make sure to run through all the steps in the pre-requisites document before starting the lab (https://ibm.biz/dataaiprereqs)
Additionally, please download theses files from box:
customers.csv - https://ibm.box.com/s/ibuqc09ips4ye67tgpt017ande192khf
cloudfunctions_checkChurn.py - https://ibm.box.com/s/0iz8x4cfdhb5eg6qftn8wn34b1ewmwuv

2. Train & Deploy Churn Prediction Model

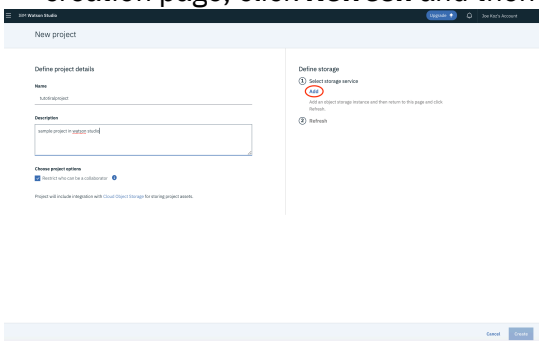
In Watson Studio, a project is how you organize your resources to achieve a particular goal. A project allows for high-level isolation, enabling users to package their project assets independently for different use cases or departments. Your project resources can include data, collaborators, scripts, and analytic assets like notebooks and models.

1- Select **Create an empty** project (or re-use a project you had created earlier)

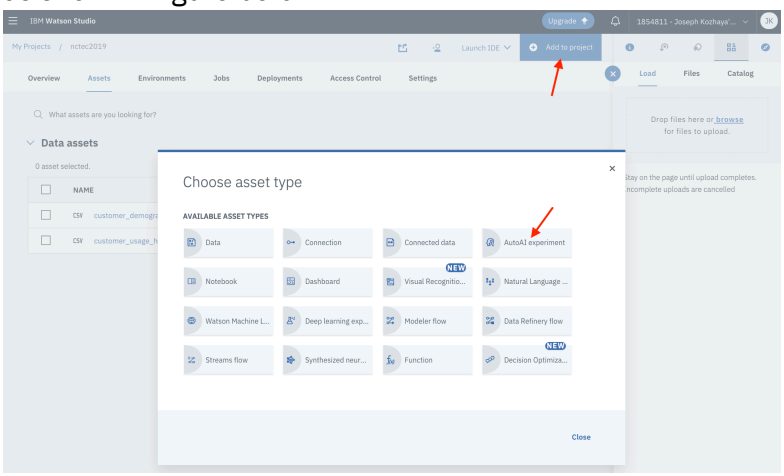


2- Provide a name for the project, a description if you wish (optional), and select the cloud object storage to associate with your project; then press Create.
Cloud Object Storage is needed for storing project assets. If you don't have a Cloud Object Storage (COS) instance created in your IBM Cloud account, you will need to create a COS instance by clicking **Add** (annotated with red oval in Figure below) on Watson Studio project creation page. This will redirect you to your IBM Cloud account where you can select the Lite plan for Cloud Object Storage and click **Create**.

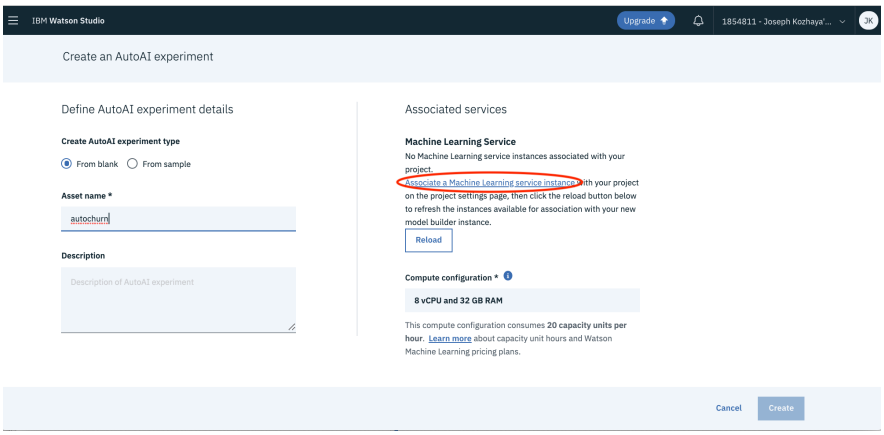
Once the Cloud Object Storage instance is created, go back to Watson Studio project creation page, click **Refresh** and then click **Create**.



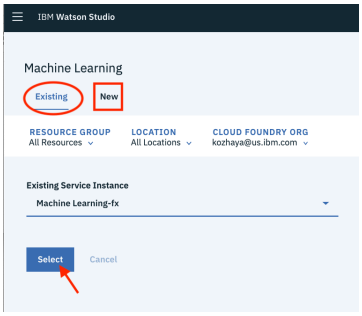
3- On the project page, click **Assets** tab (annotated with red arrow), click **Add to project** (annotated with red arrow) and select **AutoAI experiment** (annotated with red arrow) as show in Figure below.



4- Select “From blank”, provide an “Asset name”, a description (optional) and the Watson Machine Learning service. If you don’t have a Watson Machine Learning service instance associated with your project, you will need to associate one before you can proceed. To do so, click the **Associate a Machine Learning service instance** to select which WML instance to use.

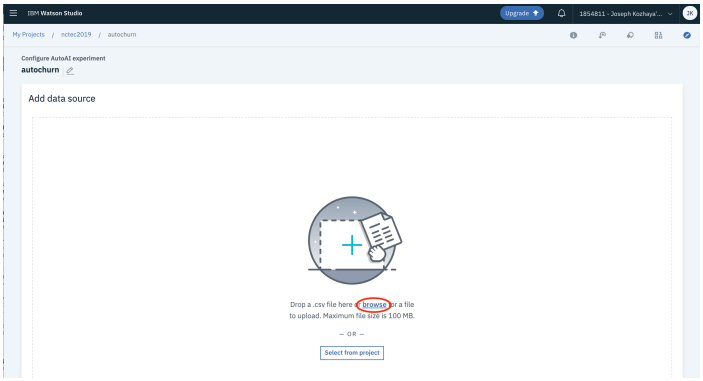


5- If you have already created a WML service instance, select the **Existing** tab (annotated with red oval in figure below) and choose the WML instance you'd like to use. If not, select the **New** tab (annotated with red rectangle in figure below) and create a new WML using Lite plan. Press **Select** (annotated with red arrow in figure below) when complete and this takes you back to your Watson Studio project.

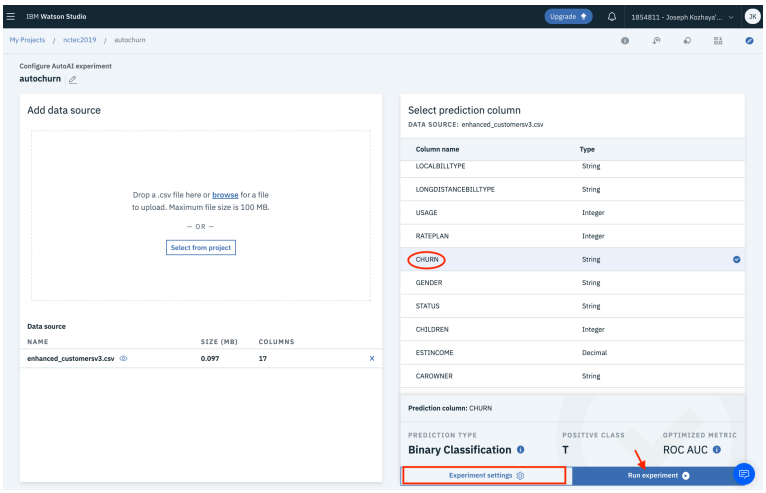


6- Back in Watson Studio, click Reload to associate the WML instance with the project. The Create button should now be active. Click **Create** to kick off the AutoAI experiment. Please note the "Compute configuration" which is set to 8 vCPU and 32 GB RAM. This consumes 20 capacity unit hours which is within the lite plan (50 capacity unit hours per month) but if you need more compute, you'll need to upgrade to a paid plan.

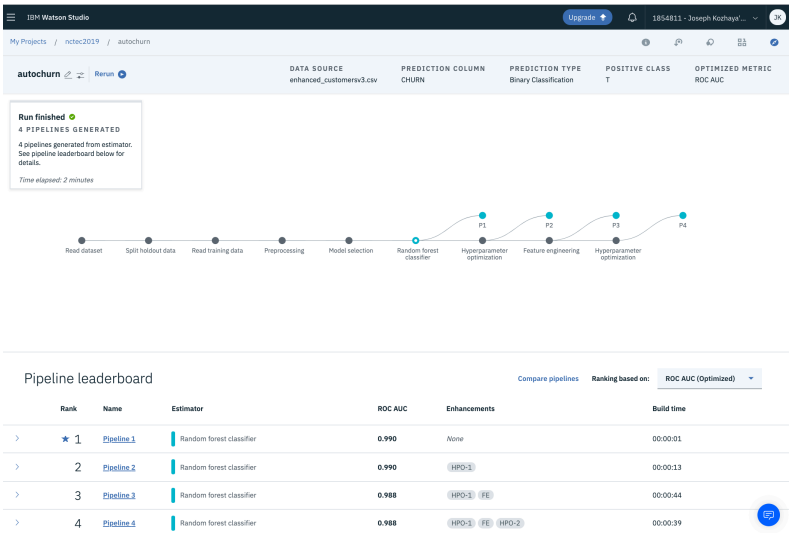
7- Click browse (annotated with red oval in figure below) and select the file "cusotmers.csv" which you had downloaded from box earlier (under data directory).



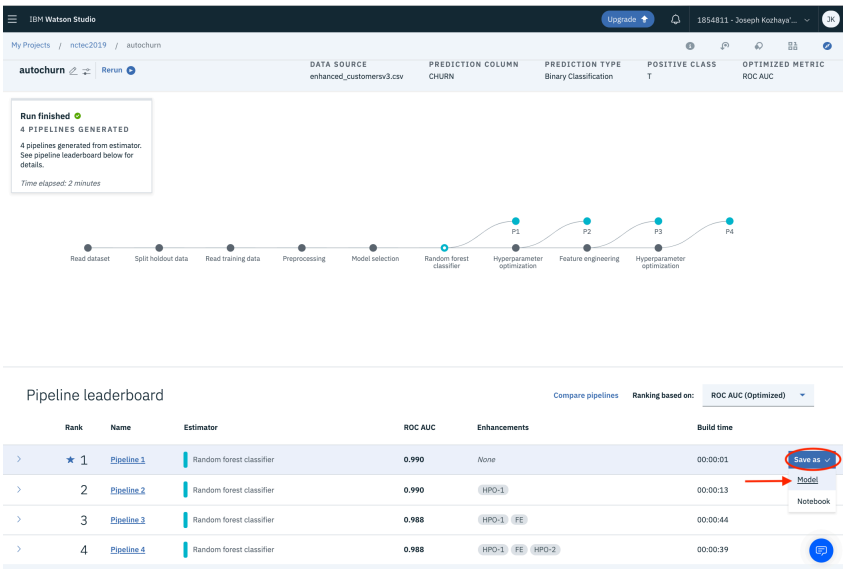
8- Once the "cusomters.csv" file is read in, you are prompted to select the prediction column. This is where you instruct AutoAI which field in the input data set is the target prediction column. Select **CHURN** field (as annotated with red oval in Figure below). AutoAI will automatically set the Prediction Type to "Binary Classification", indicate the "Positive Class" to be "T" and set the "Optimize Metric" to ROC AUC which is a popular metric for binary classification.
Please note that you can change these settings or specify additional settings by clicking the Experiment settings button (annotated with red rectangle in figure below). For purposes of this lab, the default settings are good so go ahead and click **Run experiment** (annotated with red arrow in figure below).



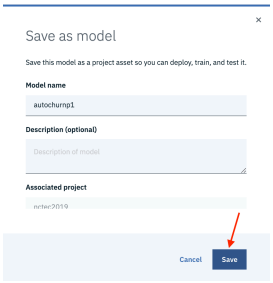
9- AutoAI runs for a few minutes on this dataset and produces a number of pipelines as shown in figure below including training/test data split, data preprocessing, feature engineering, model selection, and hyperparameter optimization. You can dig deeper into any of the pipelines to better understand feature importance, the resulting metrics, the selected model, and any applied feature transformation.



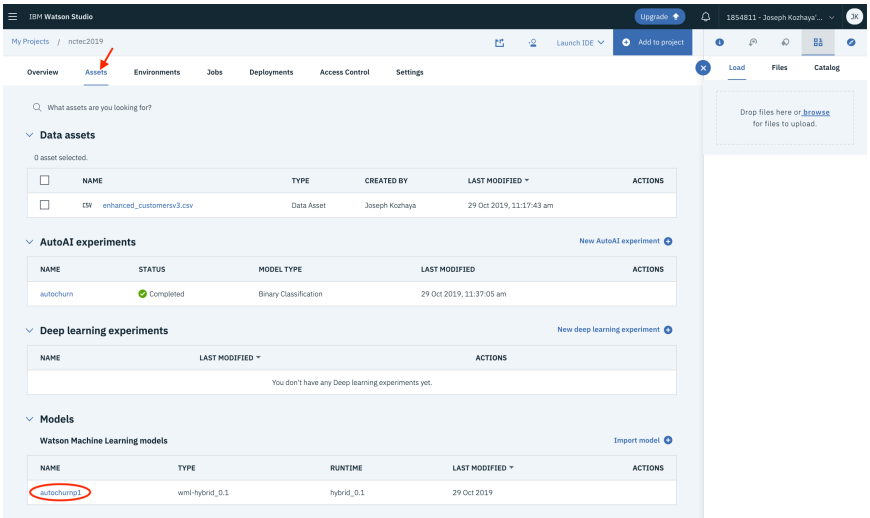
10- Reviewing the trained pipelines, we will select to save the model trained using Pipeline 1. Click **Save as** (annotated with red oval in figure below) and select **Model** (annotated with red arrow in figure below). You can also save the pipeline as a Notebook which you can customize further.



11- Provide a name for your model and click **Save** as shown in figure below.

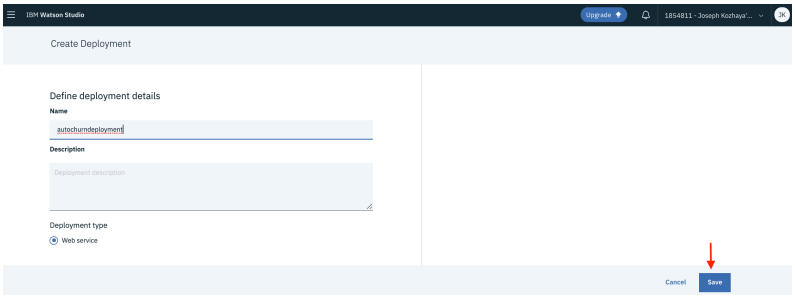


12- Navigate back to the project assets by clicking the **Assets** tab (annotated with red arrow in figure below) and notice the new model you created shows up under the Watson Machine Learning models section. Click on the saved model, **autochurnp1,** (annotated with red oval in figure below). Note that the name of your model may be different.

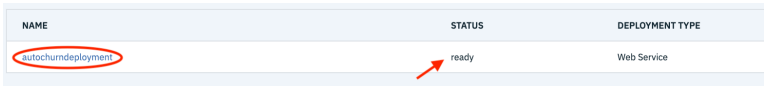


13- On the MODEL page, click on the **Deployments** tab (annotated with red arrow in figure below) and click **Add Deployment** (annotated with red oval in figure below).

14- Provide a name for your model deployment, keep the "Web service" selected, and click **Save** (annotated with red arrow in figure below). At this time, the only option to deploy a model trained with AutoAI is a web service.



15- Wait until the deployment status changed to "ready" (annotated with red arrow in figure below). Then click on the deployed model name **"autochurndeployment"** (annotated with red oval in figure below).

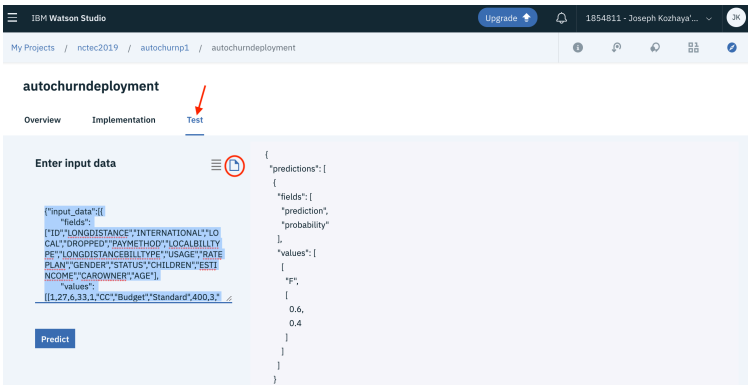


16- On the model page, select Test tab (annotated with red arrow in figure below), click on the "Provide input data as JSON" icon (annotated with red oval in figure below), paste the following JSON sample in the window and click **Predict**.

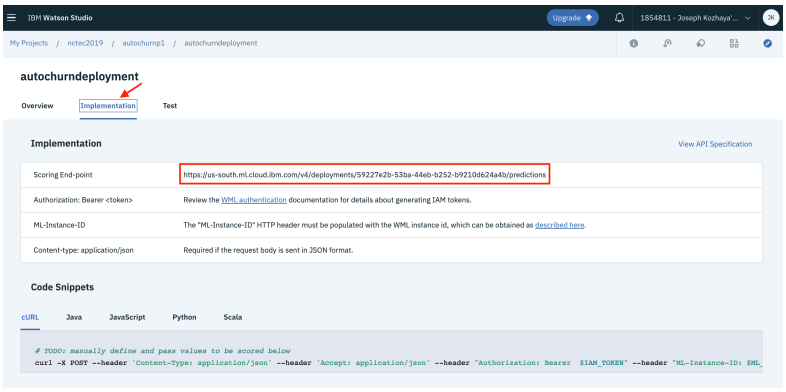
JSON test sample (be careful with copy/paste so that JSON format is preserved):

```
{
"input_data": [
        {
        "fields":["ID","LONGDISTANCE","INTERNATIONAL","LOCAL","DROPPED","PAYMETHOD
        ","LOCALBILLTYPE","LONGDISTANCEBILLTYPE","USAGE","RATEPLAN","GENDER","STA
        TUS","CHILDREN","ESTINCOME","CAROWNER","AGE"],
        "values":[[1,27,6,33,1,"CC","Budget","Standard",400,3,"M","S",0,46000,"Y",38]]
        }

]
}
```



17- Now that we have confirmed the trained model is deployed to production via a web service, click the **Implementation** tab (annotated with red arrow in figure below) and copy the "Scoring End-point" (annotated with red rectangle in figure below) for your deployed model as you'll need it in subsequent steps in this lab.



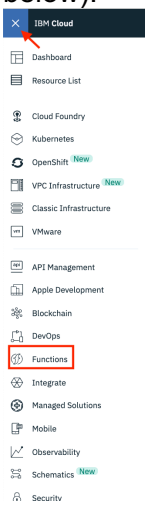
Before you proceed to next steps, make sure you have the following information which you will need in later steps.

- ***apikey*** ➔ your Watson Machine Learning service apikey
- ***ml_instance_id*** ➔ your Watson Machine Learning instance id
- ***scoring_url*** ➔ REST endpoint for your deployed machine learning model
- **fields** ➔ list of features needed for churn prediction. A couple of examples are sufficient.
- **values** ➔ list of values corresponding to the required features. A couple of examples are sufficient

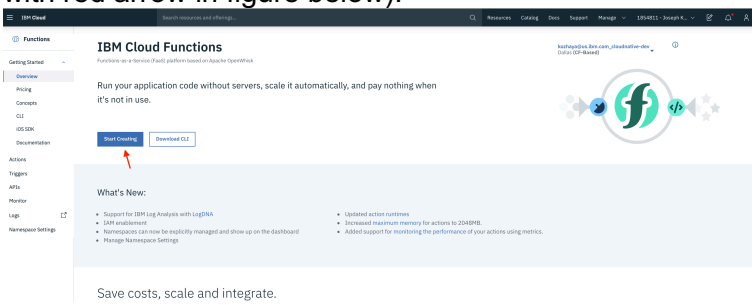
3. Cloud Functions Wrapper

In this section, we will create an IBM Cloud Functions wrapper to the model we trained and deployed in the previous section. IBM Cloud Functions is IBM's Function as a Service (FaaS) cloud computing service that allows you to execute code in response to events without dealing with the complexity of setting up the required infrastructure.

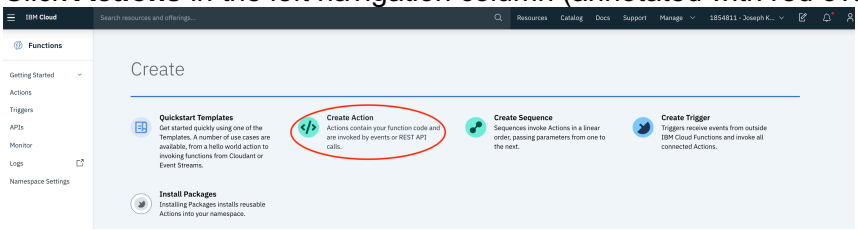
1- Logged into your IBM Cloud account, navigate to the IBM Cloud Menu (annotated with red arrow in figure below) and select Functions (annotated with red rectangle in figure below).



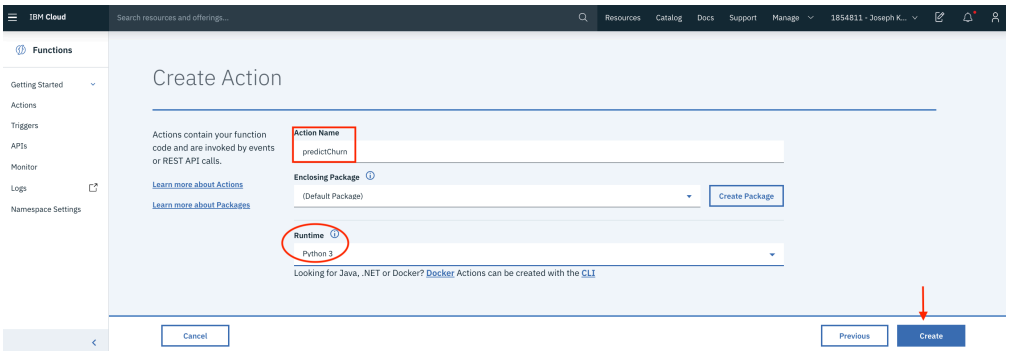
2- On the page that loads, IBM Cloud Functions page, click on **Start Creating** (annotated with red arrow in figure below).



3- Click **Actions** in the left navigation column (annotated with red oval in figure below).



4- On the Create Action page, provide a name for your Cloud Functions action (annotated with red rectangle in figure below), select “Python 3” for the Runtime (annotated with red oval in figure below) and click **Create** (annotated with red arrow in figure below).
Note that you can create the action using a different Runtime but in this lab we’re using Python 3.



5- Replace the contents of the Code section of the action with the code in the `cloudfunctions_checkChurn.py` script you downloaded from box. The code is listed here for reference as well (please be careful with copy/paste as Python is peculiar about indentation of code).

```
#### Begin Code ####
import sys
import requests, json
import ast

def getToken(apikey):
    # Get an IAM token from IBM Cloud
    url     = "https://iam.bluemix.net/oidc/token"
    headers = { "Content-Type" : "application/x-www-form-urlencoded" }
    data    = "apikey=" + apikey + "&grant_type=urn:ibm:params:oauth:grant-type:apikey"
    IBM_cloud_IAM_uid = "bx"
    IBM_cloud_IAM_pwd = "bx"
    response  = requests.post( url, headers=headers, data=data, auth=( IBM_cloud_IAM_uid, IBM_cloud_IAM_pwd ) )
    iam_token = response.json()["access_token"]
    return iam_token

def predictChurn(iam_token,ml_instance_id,scoring_url,flds,vals):
    header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + iam_token, 'ML-Instance-ID': ml_instance_id}

    payload_scoring = {"input_data": [{"fields": flds, "values": vals}]}
    response_scoring = requests.post(scoring_url, json=payload_scoring, headers=header)
    churn = json.loads(response_scoring.text)
    return churn

def main(dict):
    apikey = dict['apikey']
    ml_instance_id = dict['ml_instance_id']
    try:
        scoring_url = dict['scoring_url']
        flds_array = ast.literal_eval(dict['fields'])
        vals_array = ast.literal_eval(dict['values'])
    except:
        print("error")
        response = {"error": "not all required parameters are provided. Please make sure you pass the scoring_url, fields,
and values parameters"}
        return response

    iam_token = getToken(dict['apikey'])
    churn_prediction = predictChurn(iam_token,ml_instance_id,scoring_url,flds_array,vals_array)
    predlabel = churn_prediction['predictions'][0]['values'][0][0]
    predprob = churn_prediction['predictions'][0]['values'][0][1][0]
    response = {"label": predlabel, "prob":predprop}
    #return churn_prediction
    return response
#### End Code ####
```

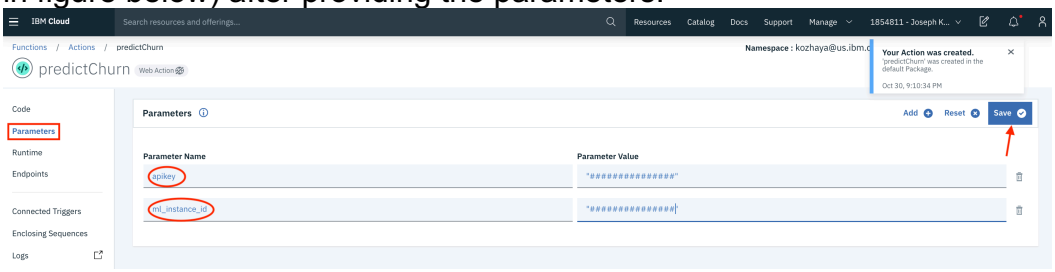
The code section is annotated with red rectange in figure below. After you copy the code, click **Save** (annotated with red arrow in figure below).

Feel free to review the simple code to understand what it does. Effectively, it gets an iam_token from IBM Cloud and then makes a call to the deployed machine learning model to predict churn.

6- Next, click on **Parameters** in the left column (annotated with red rectangle in figure below) to provide required parameters to pass to the code. Specifically, this is where you can specify authentication information like apikey. For this action, we need two parameters:

apikey
ml_instance_id

You should have these values from the previous step in this lab. If not, log into your IBM Cloud account, find your Watson Machine Learning service and click on Service Credentials. Thos would include the apikey and the ml_instance_id (annotated with red ovals in figure below). Click **Save** (annotated with red arrow in figure below) after providing the parameters.



7- Next, test the action by clicking on **Change Input** (annotated with red oval in figure below) and provide the following values in the Change Action Input window.

scoring_url ➔ this would be the REST endpoint from the deployed machine learning model you trained in the previous section.

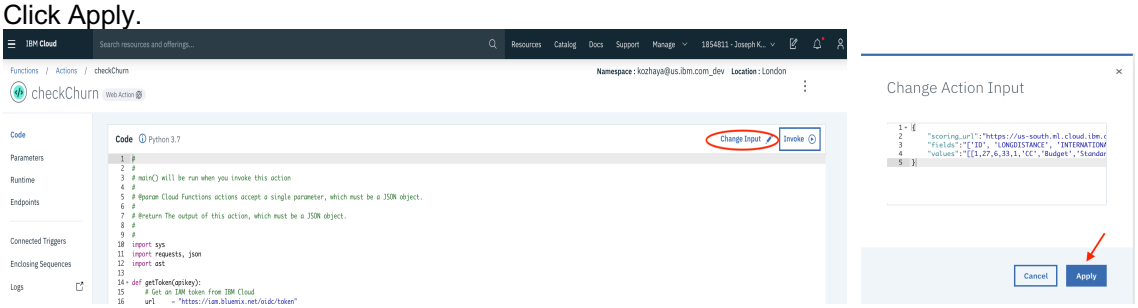
fields ➔ the array of fields used needed for scoring the model to obtain likelihood of a user to churn.

values ➔ the array of values corresponding to the field

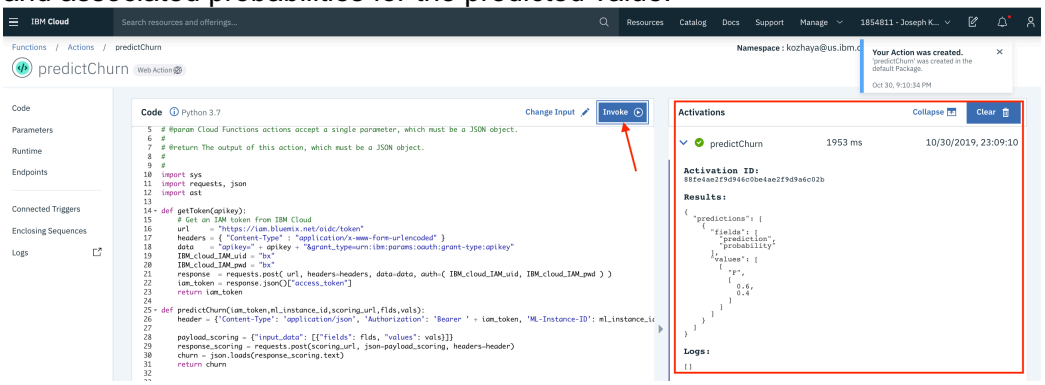
```
{
  "scoring_url":"############",
```

"fields":"['ID', 'LONGDISTANCE', 'INTERNATIONAL', 'LOCAL', 'DROPPED', 'PAYMETHOD', 'LOCALBILLTYPE', 'LONGDISTANCEBILLTYPE', 'USAGE', 'RATEPLAN', 'GENDER', 'STATUS', 'CHILDREN', 'ESTINCOME', 'CAROWNER', 'AGE']",
"values":"[[1,27,6,33,1,'CC','Budget','Standard',400,3,'M','S',0,46000,'Y',38]]"
}

Click Apply.

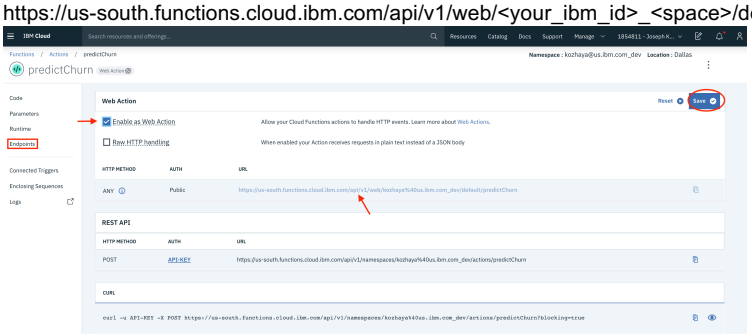


8- Click **Invoke** (annotated with red arrow in figure below) and the action will run and returns the results (annotated with red rectangle in figure below) which is the prediction and associated probabilities for the predicted value.



9- Next enable your cloud functions action as a web action to handles HTTP events. Click on **Endpoints** in the left column (annotated with red rectangle in figure below), then check the Enable as Web Action checkbox (annotated with red arrow in figure below) and click **Save** (annotated with red oval in figure below). Copy the URL (annotated with red arrow in figure below) as that will be needed to call this web action in subsequent steps of this lab. The url will have the following form:

https://us-south.functions.cloud.ibm.com/api/v1/web/<your_ibm_id>_<space>/default/<action_name>



To recap, so far we have done the following steps:

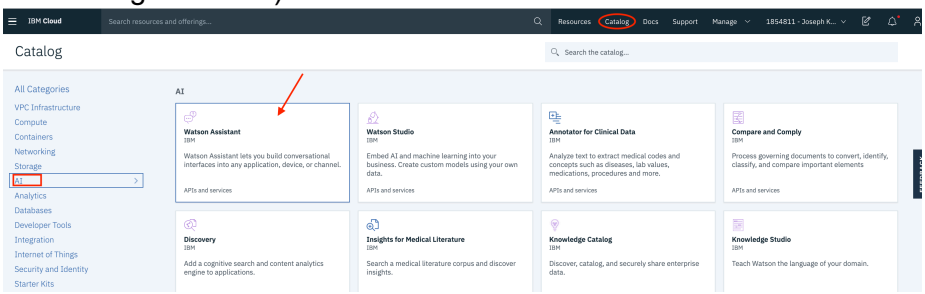
- Used AutoAI to train a machine learning model to predict likelihood of a user to churn.

- Deployed trained model to Watson Machine Learning (WML) as a web service accessible via a REST endpoint.
- Created an action using IBM Cloud Functions to call the WML REST endpoint and return the churn prediction likelihood for a user.

4. Watson Assistant Setup

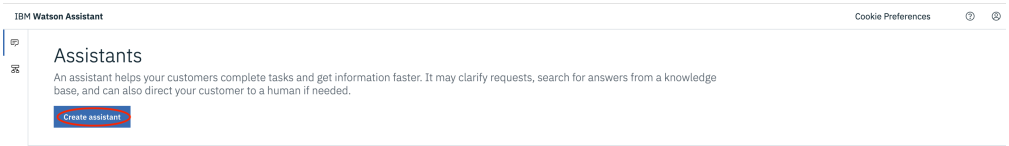
In this section, we will use Watson Assistant service to create an assistant and associate it with a dialog skill. We will integrate the dialog skill with the Cloud Functions action via the Assistant's webhook capability to personalize the user experience and respond to each user differently depending on the likelihood of that user churning.

1- Logged into your IBM Cloud account, navigate to your Watson Assistant service and launch it. If you don't have a Watson Assistant service, click on Catalog (annotated with red oval in figure below) and select AI from left navigation column (annotated with red rectangle in figure below) and then select Watson Assistant tile (annotated with red arrow in figure below).

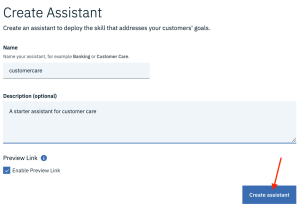


On the Watson Assistant page, select the Lite plan and click Create. You can optionally provide a name for your Watson Assistant service. Please note you can only have one Watson Assistant service with the Lite plan so if you have created one already please use that one. One the service is created, click **Launch Watson Assistant** button.

2- On the IBM Watson Assistant landing page, click **Create assistant** (annotated with red oval in figure below).



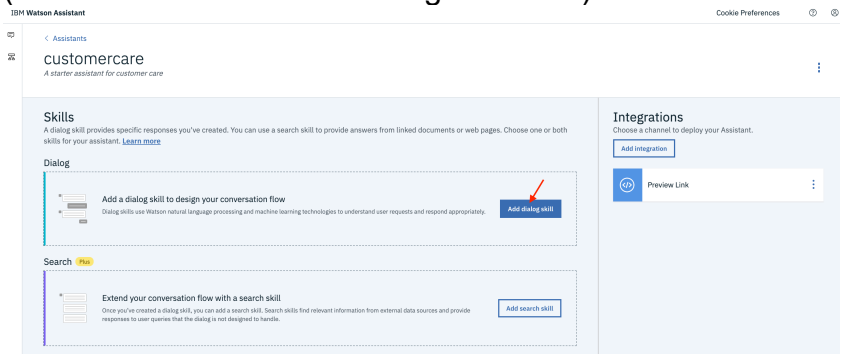
3- Provide a name for your assistant and a description (option) and click **Create assistant** (annotated with red arrow in figure below).



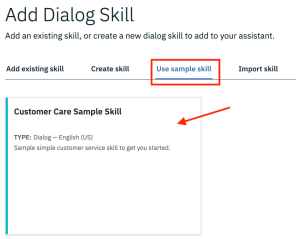
4- An assistant can handle dialog and/or search skills.

- A dialog skill uses natural language processing and machine learning technologies to understand user requests and respond appropriately based on specific responses you've created in the dialog. A dialog skill is ideal for handling FAQ-style requests.
- A search skill helps in answering complex (long-tail) questions by finding relevant information in external data sources. It leverages Watson Discovery for collecting, enriching and indexing the external data sources.

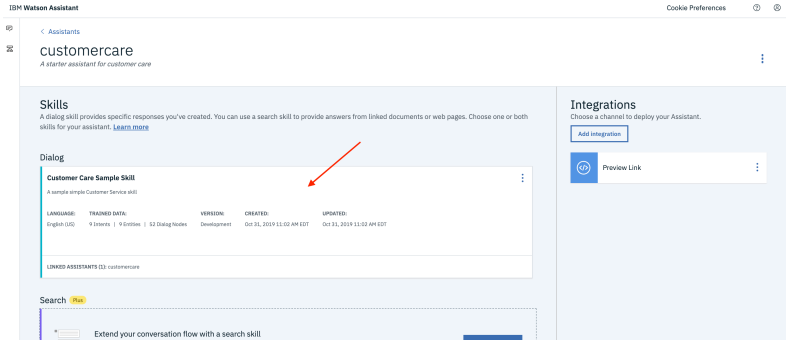
In this lab, we will work with the dialog skill. Click on Add dialog skill button (annotated with red arrow in figure below).



5- On the "Add Dialog Skill" page, select the **Use sample skill** tab (annotated with red rectangle in figure below) and click on the **Customer Care Sample Skill** tile (annotated with red arrow in figure below). This associated the customer care sample skill with the assistant you just created.

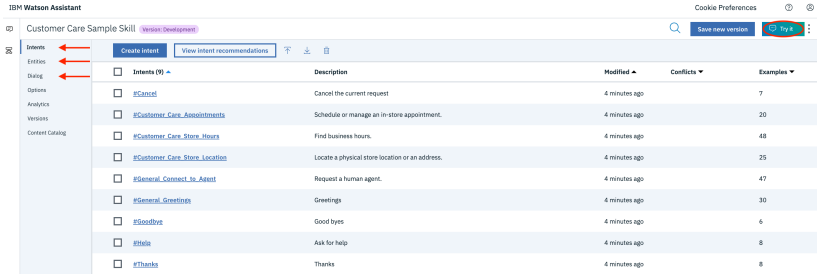


6- Back on your assistant page, click on the Customer Care Sample Skill to load that skill.



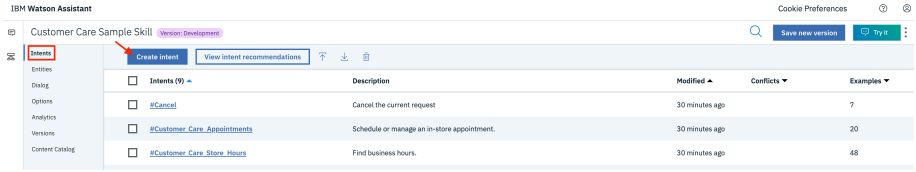
This skill is pre-built to include a number of intents and entities relevant to customer care as well as a dialog flow to handle user interactions. You can review some of the created intents, entities, and dialog by clicking through the links in the left navigation column (annotated with red arrows in figure below). After reviewing some of the prebuilt content,

click on the **Try it** button (top right - annotated with red oval in figure below) to interact with the assistant.

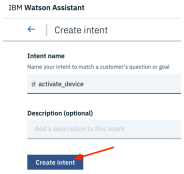


You can ask things like “What are your store hours” or “I’d like to make an appointment”.

7- Next, we’ll add a new intent to the initial set of intents defined in this skill, specifically the “#activate_device” intent. The scenario we’re addressing is for a user interacting with the chatbot to request activation of a new phone. Then depending on the churn likelihood of that user, the dialog will reflect different interactions. For users with high churn prediction probability, the chatbot will connect them directly to an agent for better experience. On the other hand, users with low churn prediction probability, the chatbot will step them through the activation process directly. Note that for your use case, you can include other factors in deciding how to offer best experience for a user but in this lab, we would like to illustrate how to integrate the assistant with a trained machine learning model to offer a more personalized user experience.
Click on **Intents** in the left navigation column (annotated with red rectangle in figure below) and click **Create intent** (annotated with red arrow in figure below).



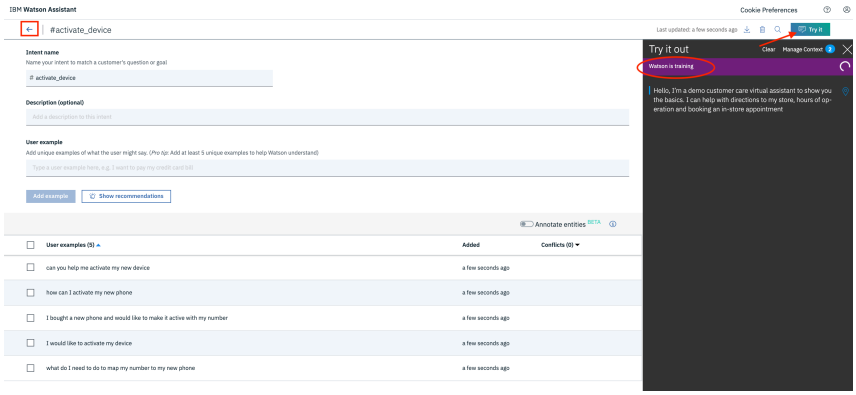
8- Provide a name for your intent, #activate_device, and click **Create intent**.



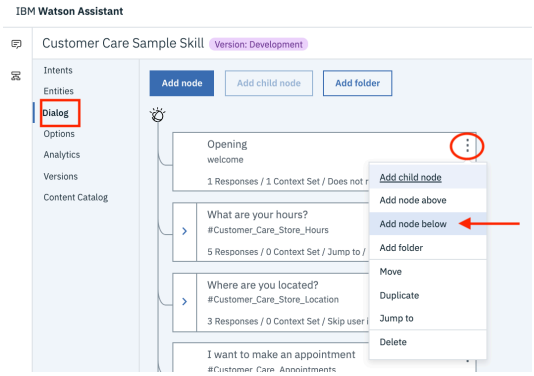
9- Next, we need to provide some sample utterances that indicate this new intent. Add the following examples:

can you help me activate my new device
how can I activate my new phone
I bought a new phone and would like to make it active with my number
I would like to activate my device
what do I need to do to map my number to my new phone

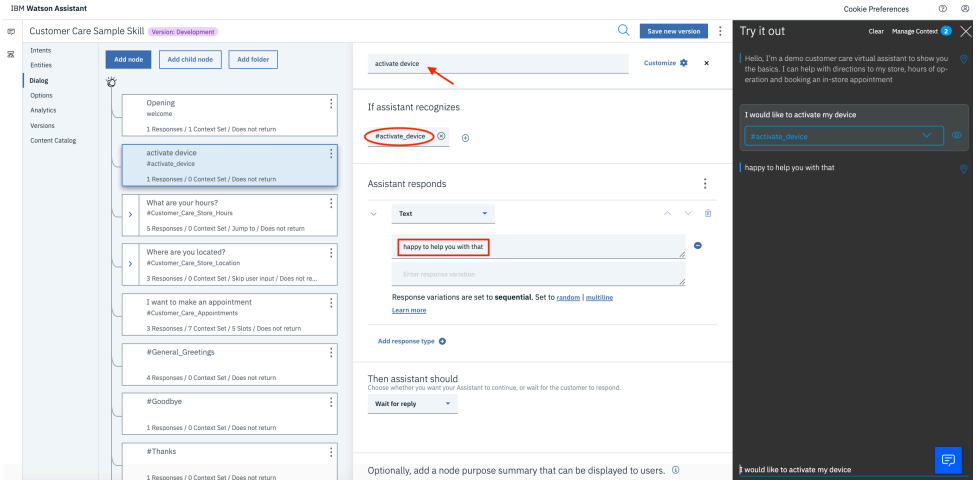
Note that as you add more examples, Watson Assistant starts training. You can view that by clicking the **Try it** button (annotated with red arrow in figure below) to see status of Watson Assistant training (annotated with red oval in the figure below)



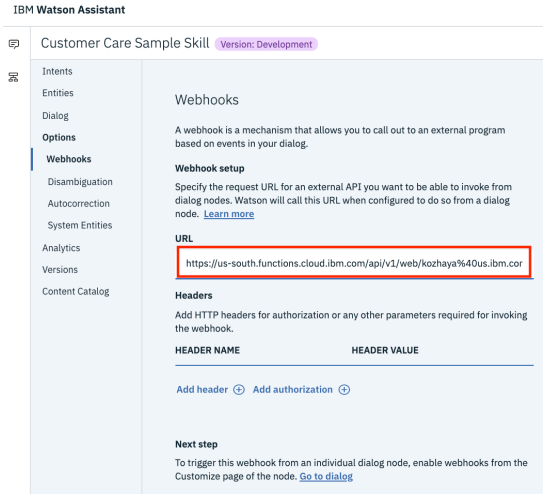
10- Next we need to update the dialog flow to handle device activation requests. Click on the back arrow (annotated with red rectangle in figure above), then select Dialog in the left navigation column (annotated with red rectangle in figure below). Click the Menu on the "Opening" node (three vertical dots annotated with red oval in figure below) and select **Add node below** (annotated with red arrow in figure below).



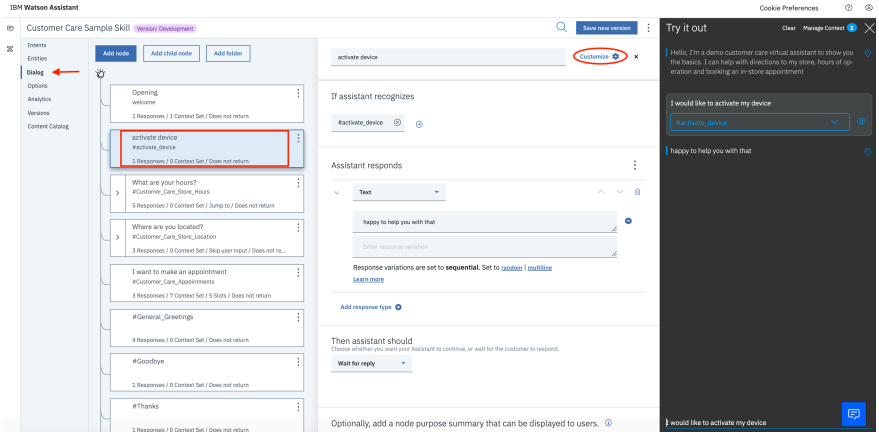
11- On the node window, provide a name for the node (for example, "activate device", annoteated with red arrow in figure below), specify the condition under "If assistant recognizes" as #activate_device intent and add a response "happy to help you with that". Once you've added these, click the **Try it** button (if not open already) and test it by typing "I would like to activate my device".



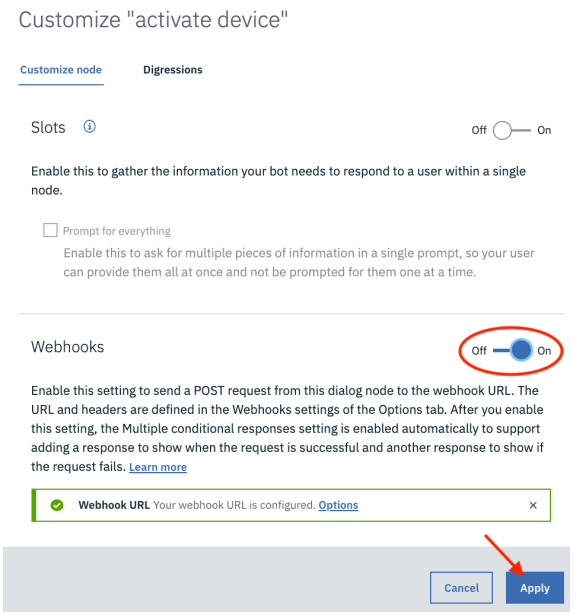
12- Next, we enable webhooks and connect the dialog node we just created to the web action we created in the previous section using IBM Cloud Functions. Click on Options in the left navigation column and then select Webhooks. In the URL field, provide the url you copied from the action Endpoint in IBM Cloud Functions (annotated with red rectangle in figure below). It will have the following :
https://us-south.functions.cloud.ibm.com/api/v1/web/<your_ibm_id>_<space>/default/<action_name>
PLEASE NOTE that you should append a **.json** to the end of that url or else you will get an error in assistant. For example, the url could be:
https://us-south.functions.cloud.ibm.com/api/v1/web/kozhaya%40us.ibm.com_dev/default/predictChurn.json



13- After adding the URL in Webhook setup page, click back on the **Dialog** tab in left navigation column (annotated with red arrow in figure below) then select the “activate device” node (annotated with red rectangle in figure below) and click **Customize** (annotated with red oval in figure below).



14- On the Customize “activate device” page, turn Webhooks on by moving the slider from Off to On (annotated with red oval in figure below) and click **Apply** (annotated with red arrow in figure below).



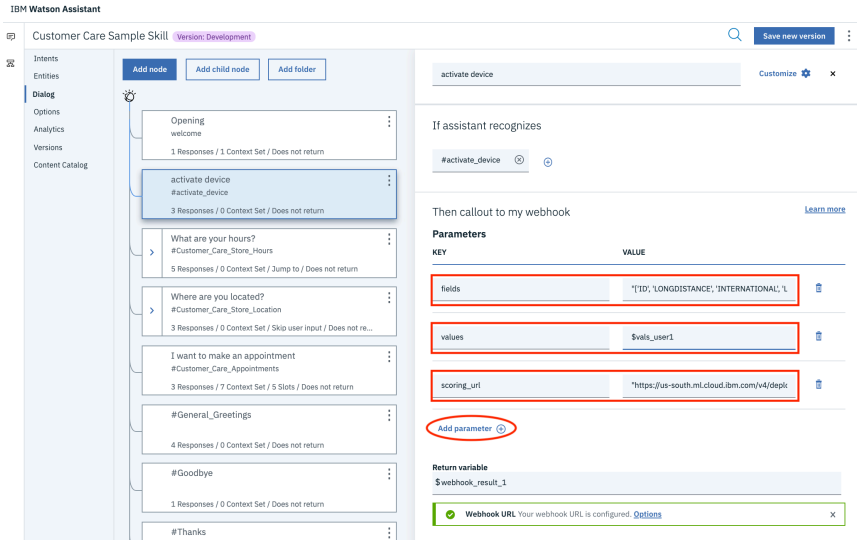
15- Once the Webhooks are turn On for this node, the “activate device” node gets updated to reflect the Parameters that can be provided to the webhook url. If you remember in the previous section, the predictChurn action required the following 3 parameters:

scoring_url
fields
values

Specify these parameters in the “activate device” node. The scoring_url is the endpoint for the deployed machine learning model. The fields is the list of features to pass to the model for predicting likelihood of churn. The values is the array of values for the fields specific to the user.

scoring_url: https://us-south.ml.cloud.ibm.com/v4/deployments/59227e2b-53ba-44eb-b252-b9210d624a4b/predictions
fields:"['ID', 'LONGDISTANCE', 'INTERNATIONAL', 'LOCAL', 'DROPPED', 'PAYMETHOD', 'LOCALBILLTYPE', 'LONGDISTANCEBILLTYPE', 'USAGE', 'RATEPLAN', 'GENDER', 'STATUS', 'CHILDREN', 'ESTINCOME', 'CAROWNER', 'AGE']"
values: $user_vals

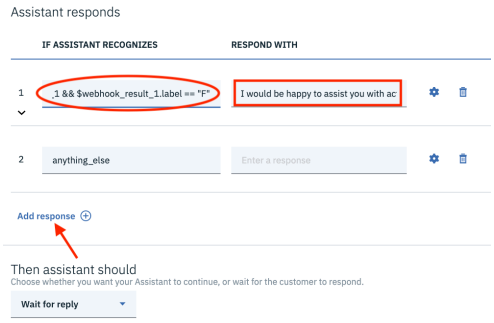
The figure below shows how the parameters should be defined (annotated with red rectangles). To add new parameters, you click the **Add parameter** (annotated with red oval in figure below). Note that the values parameter is defined as a context variable which would be different for different end users. The actual field values can be obtained from a backend system when a user authenticates into the system and initiates the chatbot interaction. For purposes of this lab, we emulate this behavior by setting the context variable ($vals_user1) directly using the “Manage Context” functionality in the “Try it” panel. In a production environment, this context variable would be set by a back end system once the user is authenticated.



16- Now that we've integrated the webhook in the "activate device" node, we edit the response to handle the results from the cloud function indicating likelihood of user to churn. If the prediction likelihood returns False ('F'), the virtual assistant will help the user with device activation process. If the prediction likelihood return True ('T'), on the other hand, the virtual assistant will transfer the user to a human agent to guarantee best experience for the user.

- Scroll down in the "activate device" node to the "Assistant responds" section and add the following condition and response:

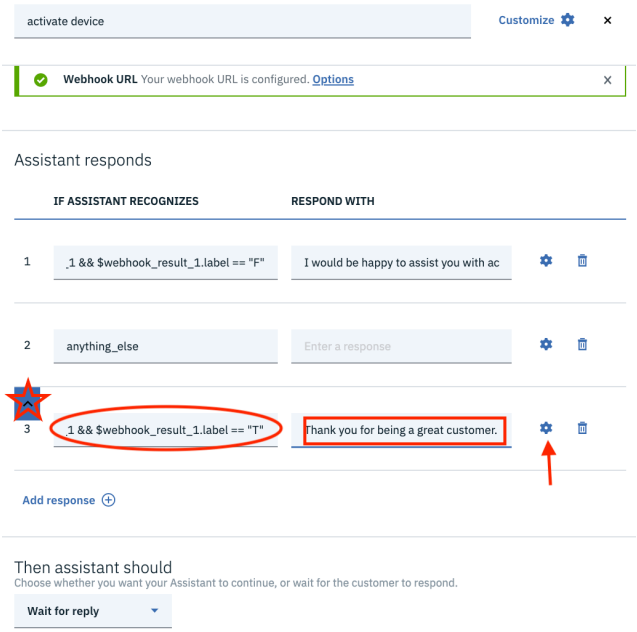
"IF ASSISTANT RECOGNIZES" => $webhook_result_1 && $webhook_result_1.label == "F"
"RESPOND WITH" => I would be happy to assist you with activating your device.



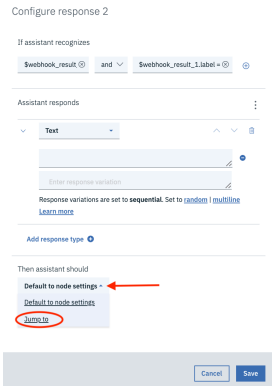
This effectively checks if the returned prediction is false (no churn) and if so, respond to assist the user with device activation.

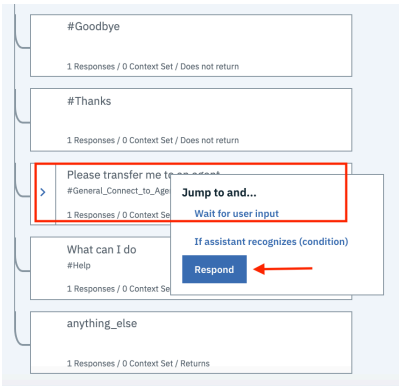
- Then click **Add response** (annotated with red arrow in figure above) and add the following condition.

"IF ASSISTANT RECOGNIZES" => $webhook_result_1 && $webhook_result_1.label == "T"
"RESPOND WITH" => Thank you for being a great customer.



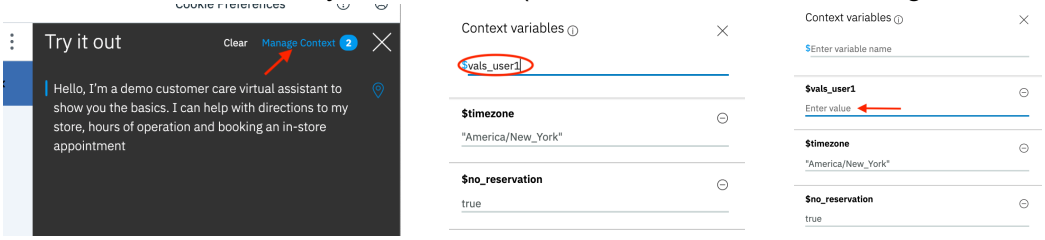
- Click on the arrow (annotated with red star in figure above) to move this response to be the second (above the anything_else response).
- Click the Customize response (gear icon annotated with red arrow in figure above) to customize the response for the scenario when the churn prediction is true. In the "Configure response 2" window, click on **Default to node settings** (annotated with red arrow in figure below) and then select **Jump to** (annotated with red oval in figure below). This will take you back to the overall dialog flow so you can select which node to "Jump to".
- Select the "Please transfer me to an agent" node (annotated with red rectangle in figure below) and click **Respond**. Click **Save** to to close the Configure Response window.





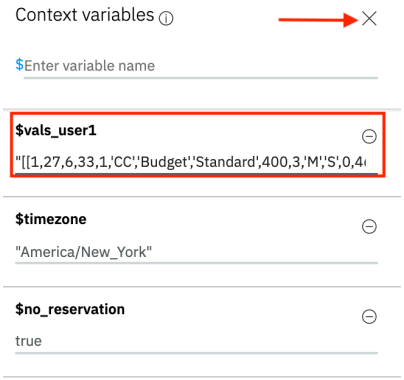
17- Next, we test the dialog. To emulate different users, click the Try it button (if the Try it out panel is not open already) and click on Manage Context (annotated with red arrow in figure below). Then add a context variable and call it ***vals_user1*** (annotated with red

oval in figure below) and hit **Enter**. Then enter the following value for the ***$vals_user1*** context value where it says Enter value (annotated with red arrow in figure below).



"[[1,27,6,33,1,'CC','Budget','Standard',400,3,'M','S',0,46000,'Y',38]]"

The final context variables window should look as shown in figure below with the $vals_user1 context variable defined and assigned the value above. Click the X (annotated with red arrow in figure below) to go back to the Try it out window.

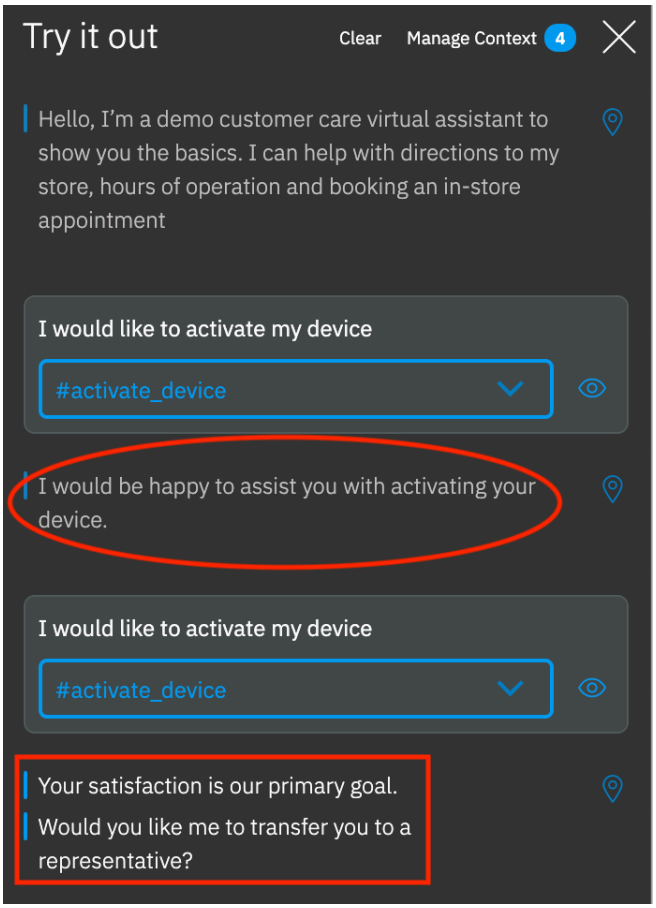


18- Test the dialog flow in the Try it window by entering the phrase “I would like to activate my device”

19- Edit the $vals_user1 context variable by clicking on Manage Context, updating the values of $vals_user1 to the following values and then existing back to the Try it out window.
"[[1,27,6,33,7,'CC','Budget','Standard',400,3,'M','S',0,96000,'Y',38]]"

20- Test the dialog flow again by entering the same phrase “I would like to activate my device”. The figure below shows the first response for the first set of values associated with user 1 (annotated with red oval) where the churn prediction is False as well as the second response for the second set of values associated with user 2 (annotated with red rectange) where the churn prediction is True.
User 1 => values: "[[1,27,6,33,1,'CC','Budget','Standard',400,3,'M','S',0,46000,'Y',38]]"
- ⇨ Churn Prediction: “False”
- ⇨ Chatbot offers to assist with activating the user device

User 2 => values: "[[1,27,6,33,7,'CC','Budget','Standard',400,3,'M','S',0,96000,'Y',38]]"
- ⇨ Churn Prediction: “True”
- ⇨ Chatbot offers to transfer to a representative for a better end user experience.



5. Summary

In this lab, we illustrated how to infuse AI in a virtual assistant to personalize end user experience. We leveraged Watson Studio AutoAI capabilities to quickly explore, evaluate, and train the best model for predicting churn based on the given dataset. Then we deployed that trained model to a REST endpoint. Lastly, we infused that trained model in a virtual assistant with the help of cloud functions to call the REST endpoint for the ML model and to personalize the end user experience based on their likelihood to churn. The same approach can be leveraged for a variety of use cases across industries with even deeper personalization by including more end user features in the model and customizing the dialog interaction accordingly.
For example, using a recommendation model, different users would see different products or offers being recommended based on their specific data.