Innovate and improve your business with Azure Conversational AI and Machine Learning

Contents

Create an Azure Machine Learning Workspace

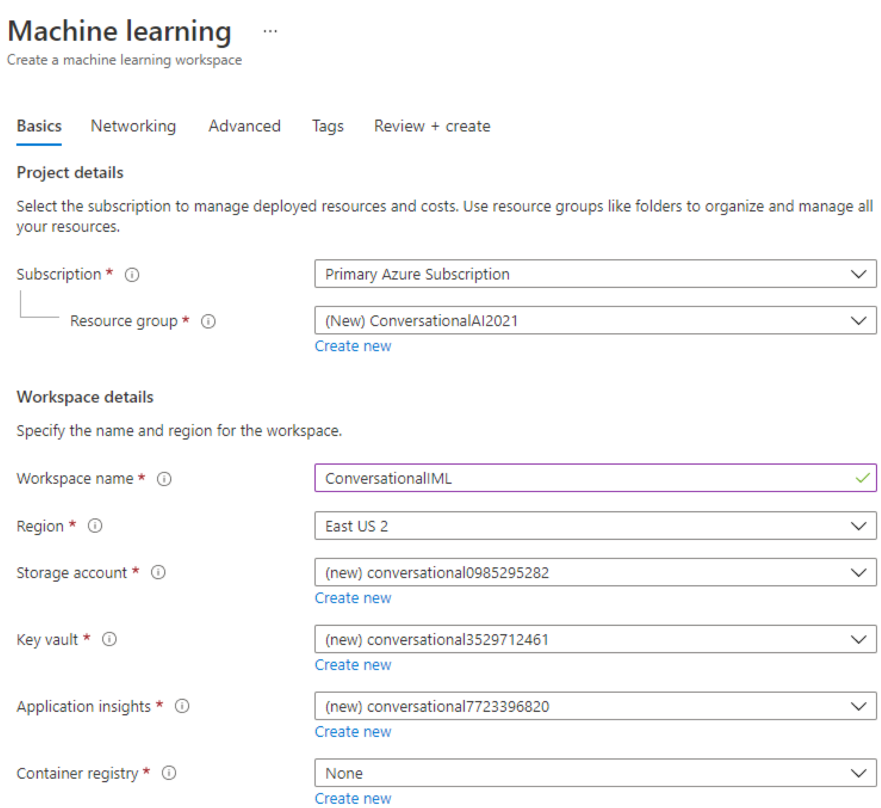
To train machine learning models in AutoML, you first need to set up a machine learning workspace.

1. Sign into the [Azure Portal]()
2. Select "Create a resource" from the list of Azure services.
3. Search for "Machine Learning" in the resource search bar.
4. On the Machine Learning page, click "Create" to start creating your workspace.

From this page, you can configure your workspace:

1. Select your subscription from the subscription menu.
2. For the resource group, select "Create new" and enter a name for the new resource group.
3. For the workspace name, fill in the name for your new workspace. The remaining fields will auto-complete and can be left as-is.
4. If needed, update the region to match your local timezone.
5. The container registry can be left as "None" since this will be created when you generate your workspace.

Your settings should look similar to this:



Now, you can click "Review + create" fand your workspaces settings will be validated. Once validation has passed, click "Create" on the following page to begin deployment of the new resource group. This may take a few minutes to complete.

Once deployment is complete, you can access your workspace through the [Azure Machine Learning Studio](). If you do not already have a workspace established, you will be prompted to select your workspace. Otherwise, you can navigate to https://ml.azure.com/selectWorkspace to select your workspace.

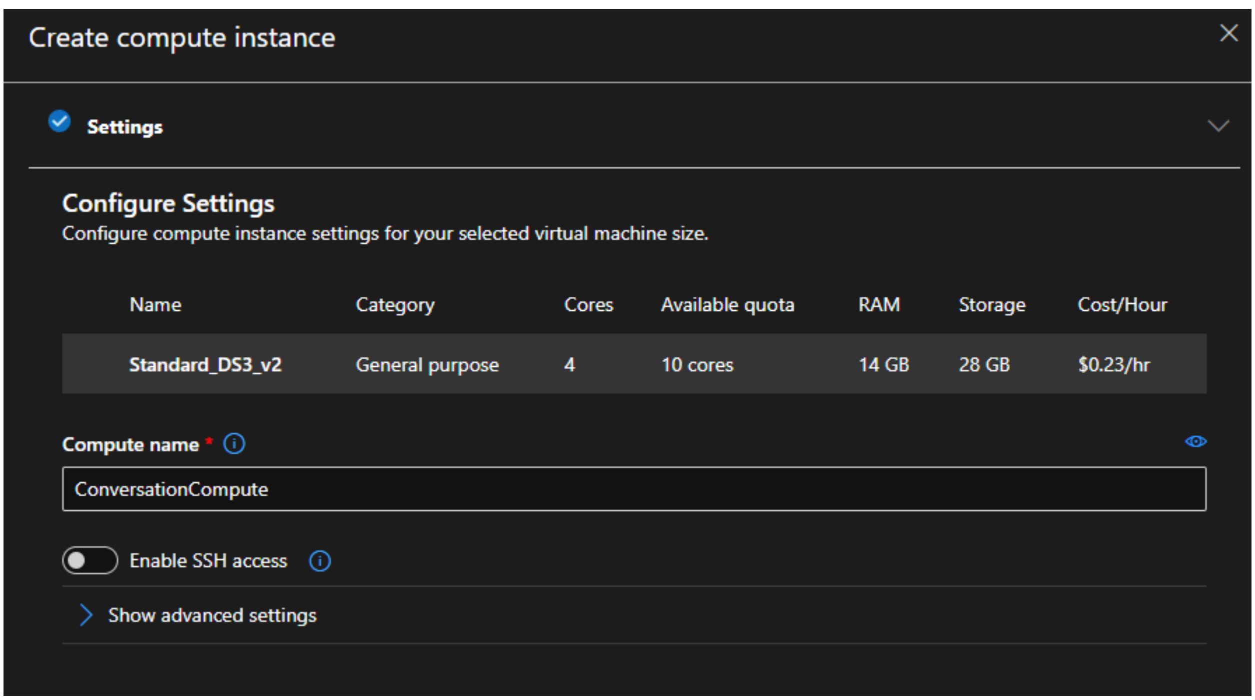
From this prompt, select the directory and subscription used to create your ML workspace. Then, select your workspace and click "Get started"

Creating Compute resources

Now that you've created your workspace, you still need to create compute resources to train your models. These resources provide the computational power to train and validate your models.

1. From the home page of Azure Machine Learning Studio, select "Create New" from the left panel. Then, select "Compute instance."
2. From this page, you can select the type of virtual machine used to process your data. Different options are more appropriate for different types of models, but for this example, you can simply use the default virtual machine, "Standard_DS3_v2."
3. Name your compute instance.

Your settings should look similar to this:



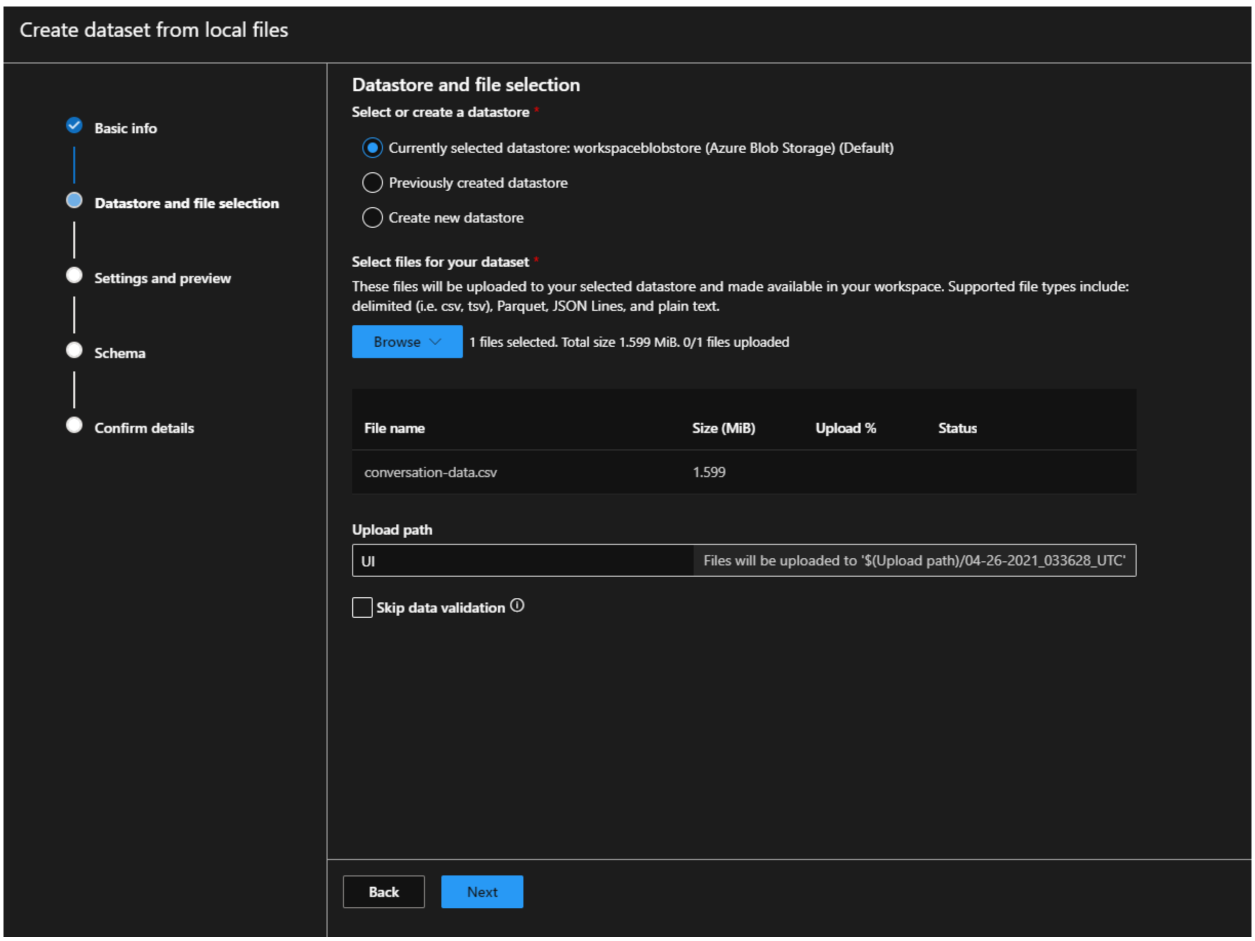
Now, you can click "Create" to generate the resource. This may take a few minutes to run.

Data Preparation

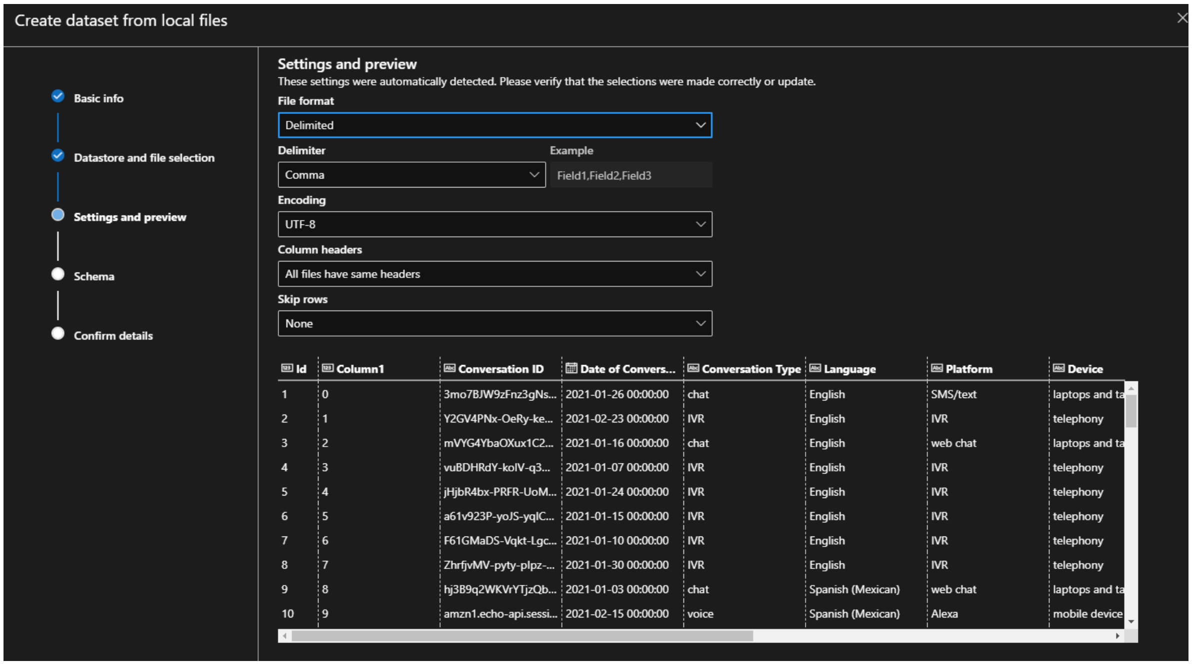
Now that you've created your workspace and compute resources, you're ready to upload your data. From the left-hand panel, click the "+ New" option and then select "Dataset."

1. From this page, click "Create dataset" and then select "From local file."
2. Name your dataset and leave the "Dataset type" as Tabular. Click "Next."
3. In the file browser, select the local file that contains your data. The other settings can be left to their default values.

Your settings should look similar to this:

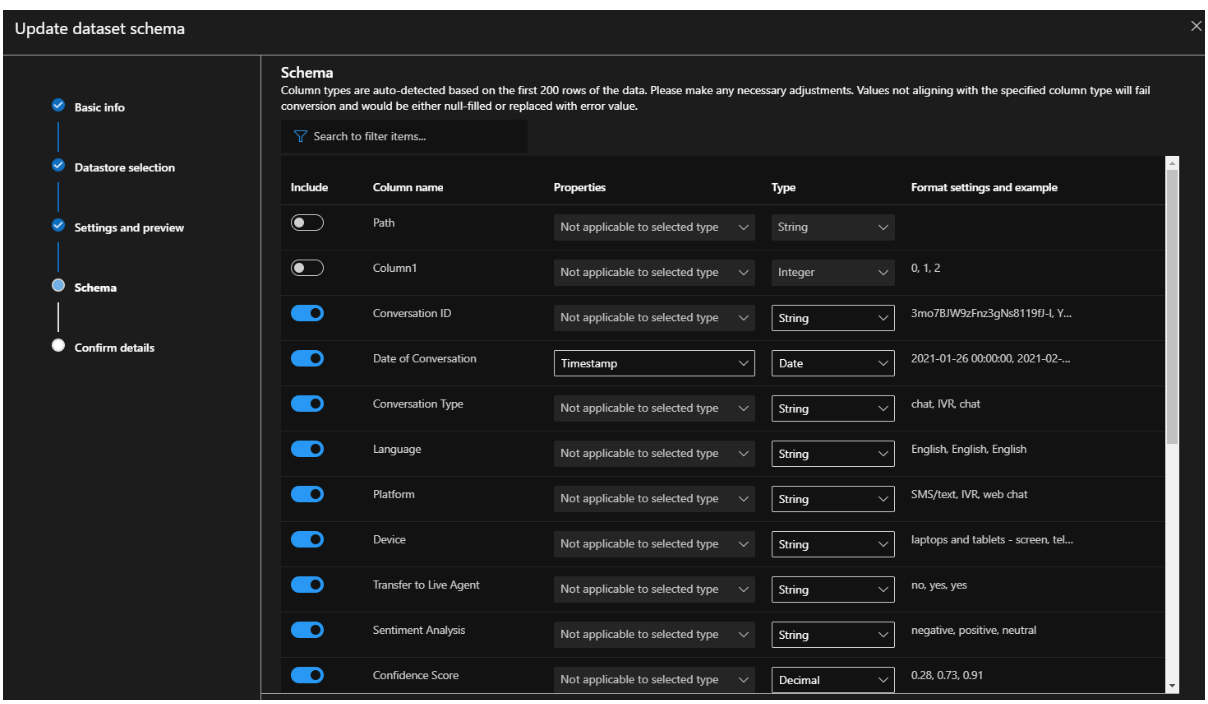


After clicking "Next," confirm the settings that the dataset returns. This should look similar to this:



Click “Next” again to view the schema of the file. From this next page, you can toggle the properties and types of various columns in the dataset and select where they should be included in the final dataset.

Since “Column1” only contains the data index and no useful information, we can exclude the column from the dataset. Toggle the “Include” column to exclude this data. Additionally, we can set the the property of “Date of Conversation” to “Timestamp.” All other variables can be left as is:



Click “Next” and then “Create.” You now have a dataset for training your model!

Creating a Regression Model in AutoML

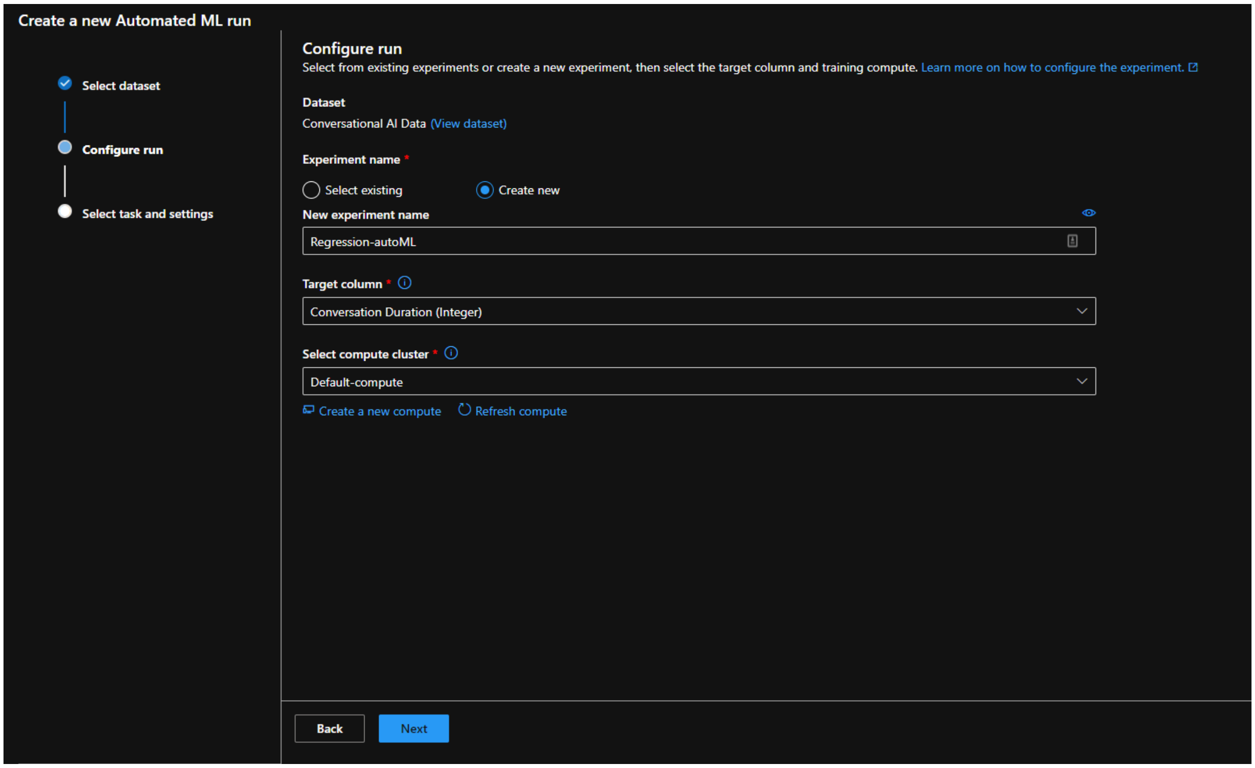
Key Question: How long will a conversation last and what factors have the strongest impact on conversation duration?

Training the Regression Model

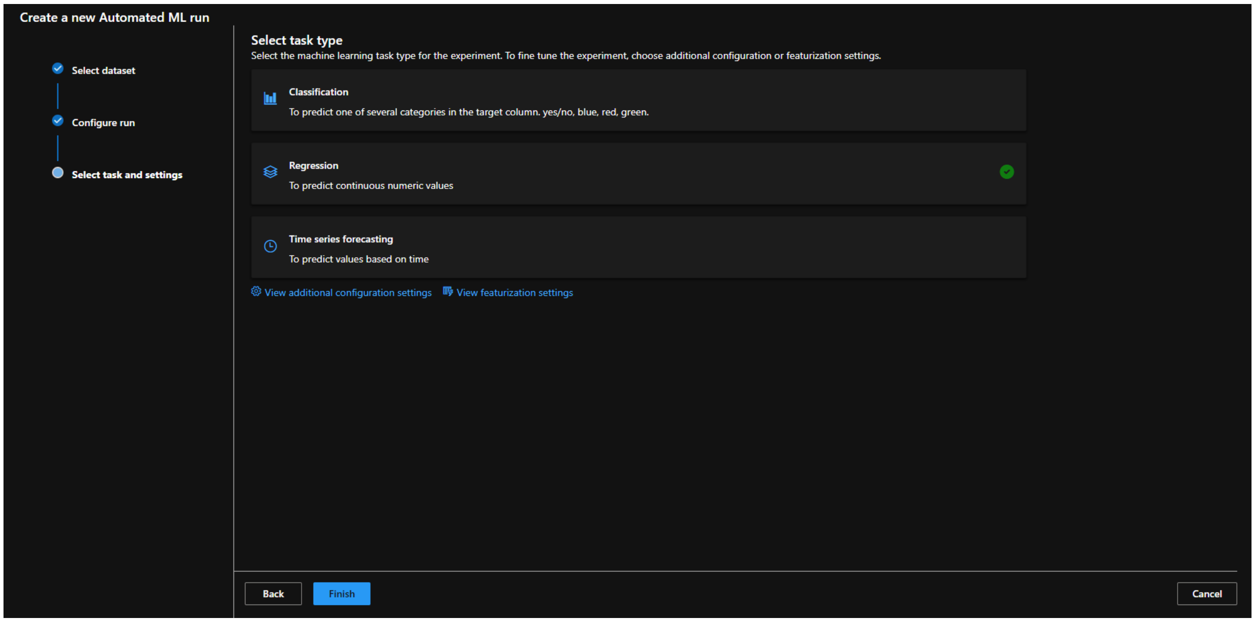
Now that you have a workspace, compute resources, and a dataset, you're ready to start training your data in AutoML. From the left-hand panel, click "Automated ML" under "Author." Click "New Automated ML run." For the dataset, select the Conversation AI dataset you previously created.

After clicking "Next," you will be brought to a page to configure your run:

1. Under the experiment name, click "Create new," and then type an experiment name.
2. For the regression model, we want to predict conversation duration so, for the "Target column," select conversation duration.
3. Under "Select compute cluster", select the compute cluster you created previously.



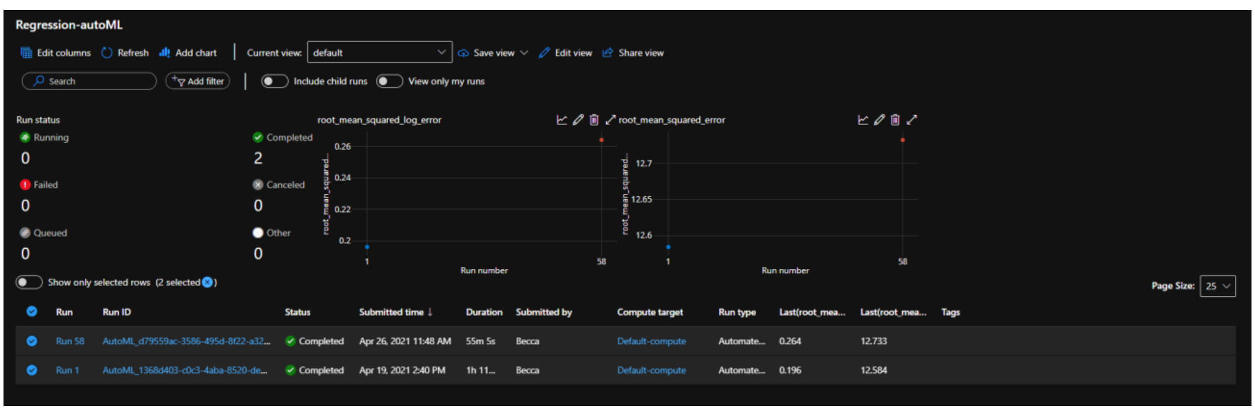
After clicking "Next," select Regression as the model type:



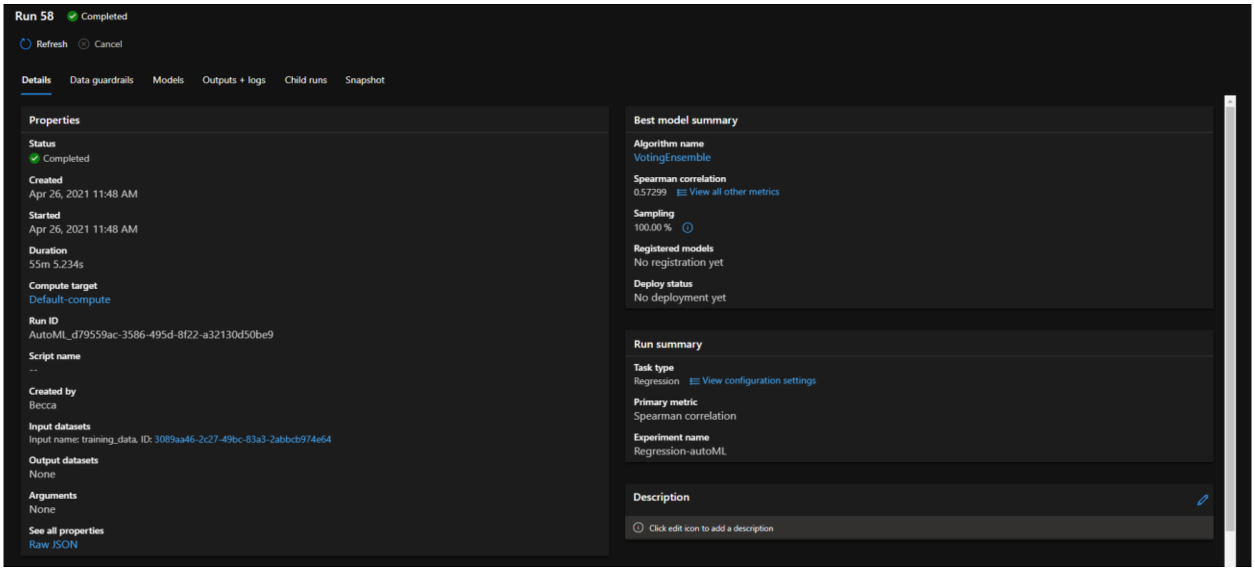
Click “Finish” to begin the run.

Interpreting the Regression Model

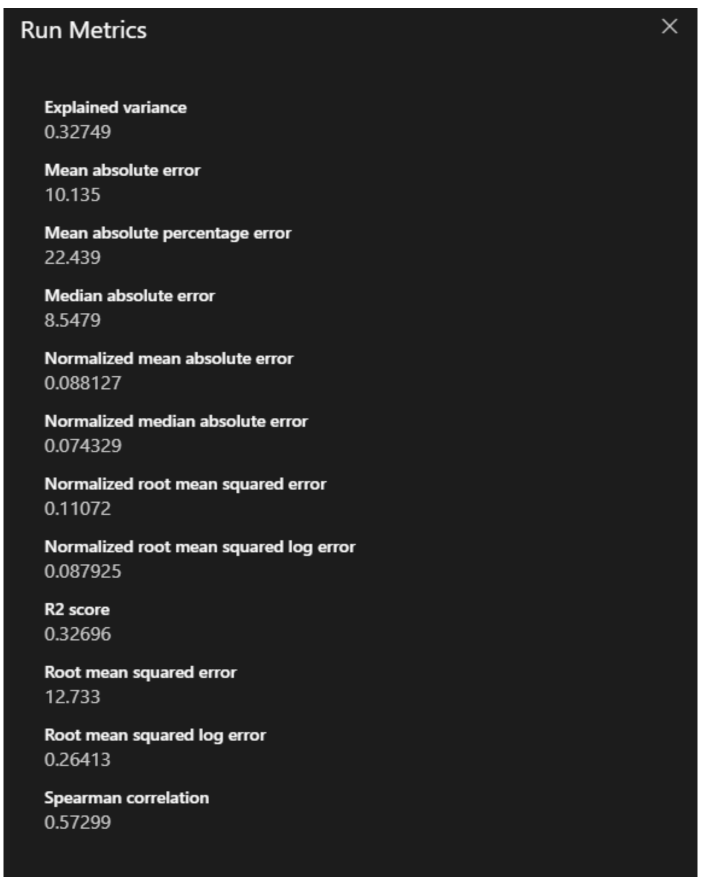
Once the model has finished running, navigate to the “Experiment” tab in the left panel. Locate the experiment that you just ran and click on the experiment name to navigate to the experiment page. This page will show the run history of your current experiment:



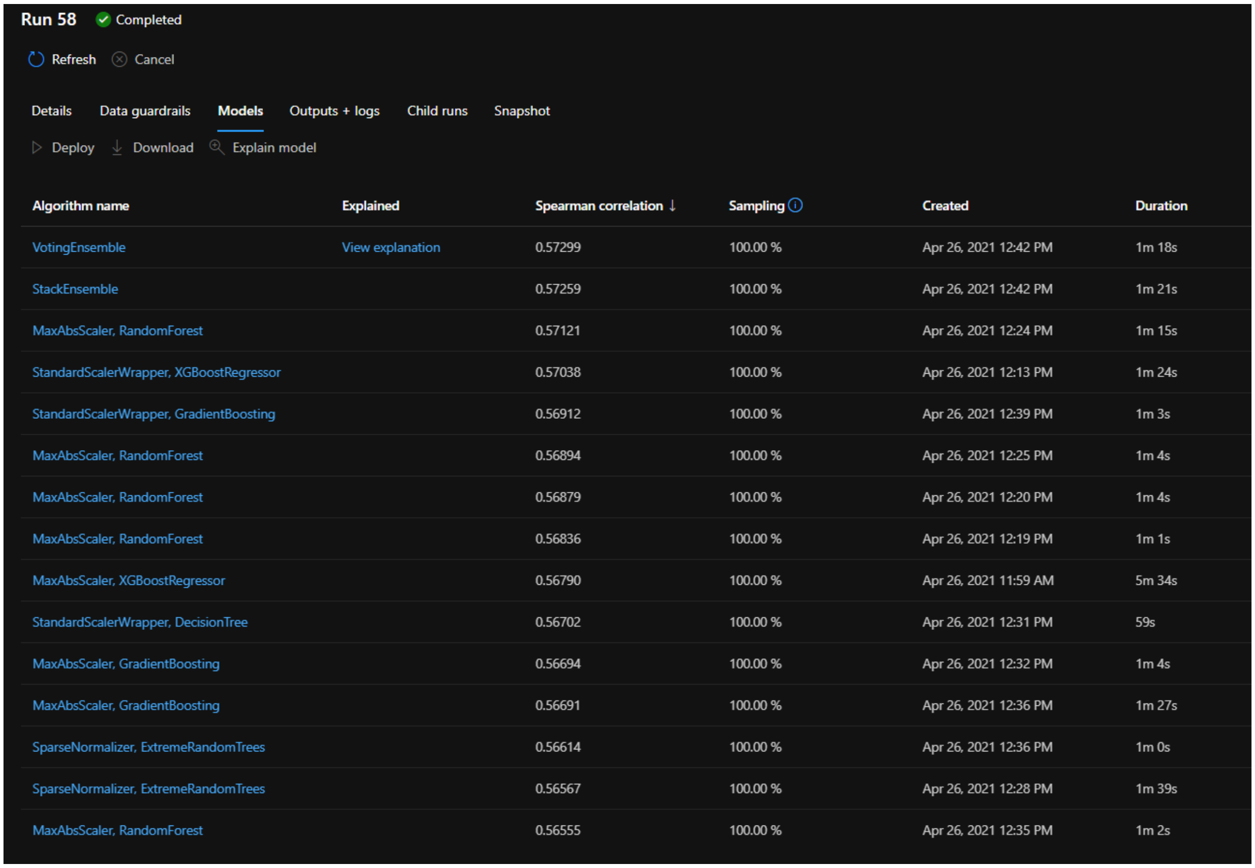
To view the results from your most recent experiment, click on the most recent run name in the run column (in this case “Run 58”). This will bring you to a page that describes the details of the run:



In our example, we can see that the VotingEnsemble was the best performing model with a Spearman correlation of 0.57. By clicking “View all other metrics” under the best model summary, we can see all the metrics for this model:

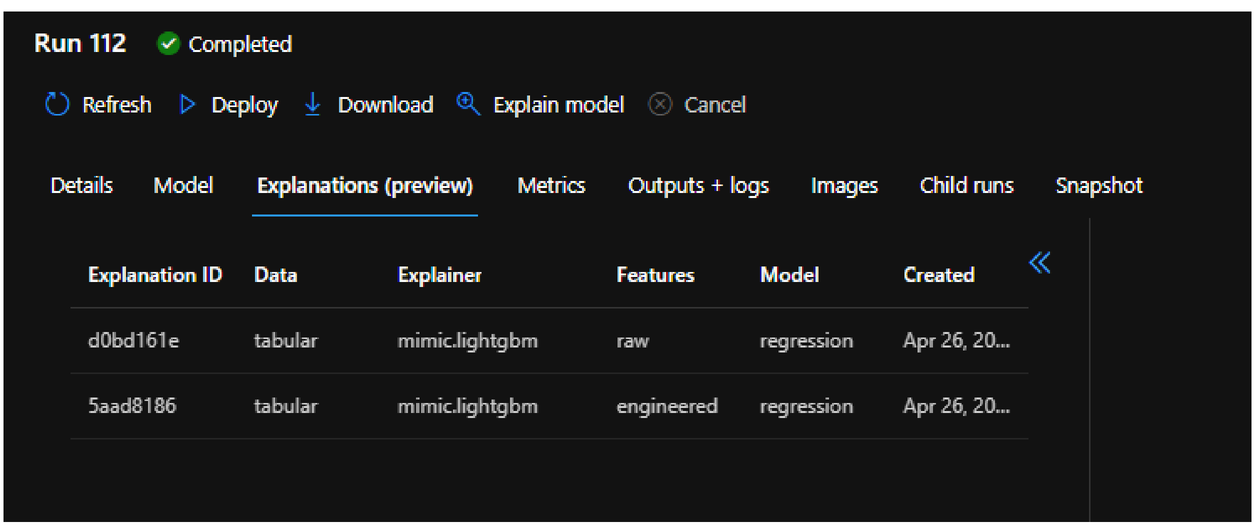


We can also view the statistics for all other tested algorithms by navigating to the “Models” tab:



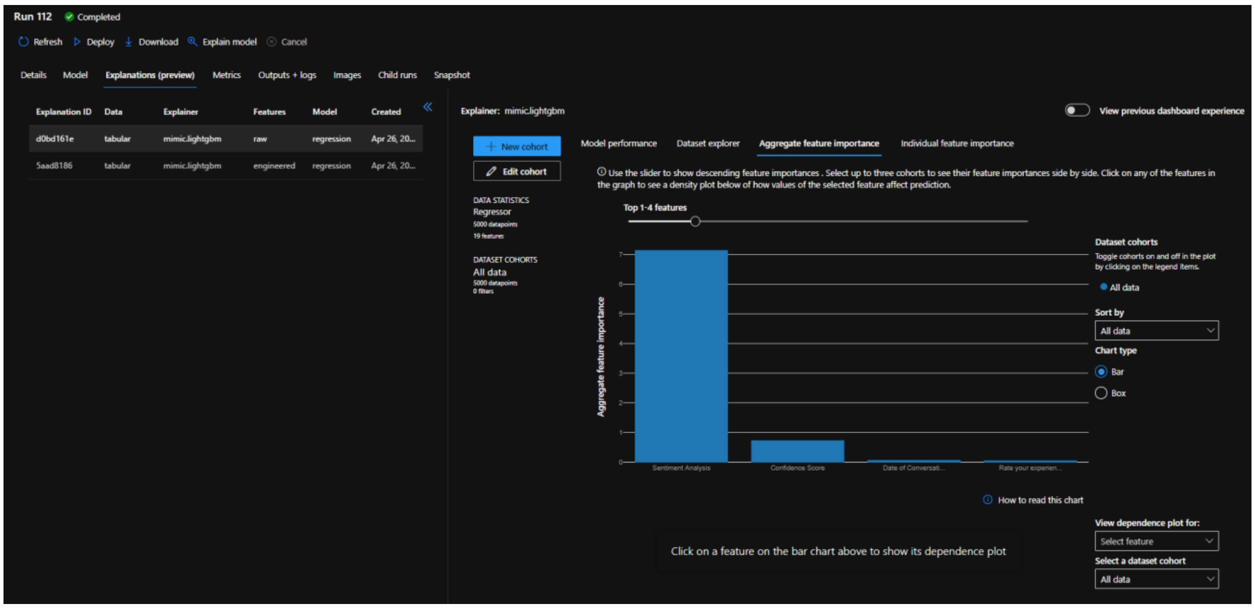
By default, AutoML provides an explanation for the best-performing model. Here, we'll focus on the VotingEnsemble. By clicking "View explanation," we can view information about this model.

Here, we'll see there are two provided explanations, one based on the raw features and one based on the engineered features:



For our explanation, we'll focus on the raw features. After clicking on the explanation ID, the window to the right populates with information about our dataset. This window can be used to explore the relationships between variables in the dataset.

By navigating to the aggregate feature importance tab, we can explore which features are most important for determining the output of our model. For this model, sentiment analysis (positive, negative, or neutral) was the most important indicator of conversation duration as well as its associated confidence score:



By defining a new cohort, this window can also be used to explore relationships between subsets of the data.

Business impact: The data from the model identifies factors associated with increased conversation duration. It also highlights the relationship between positive sentiment and longer conversations.

Creating a Classification Model in AutoML

Key Question: Which conversations will be successful and what factors have the strongest impact on a conversation's success?

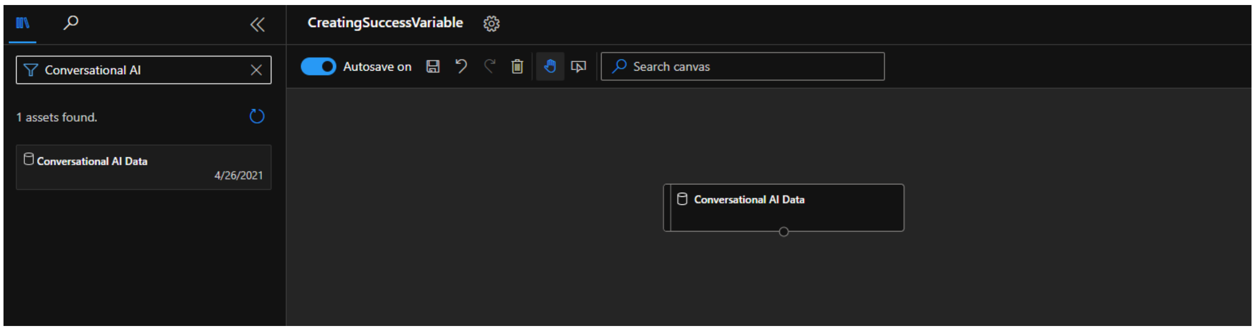
Now we're ready to create a second model in AutoML. For this model, we want to predict whether a conversation is successful. We're defining success as a conversation with a positive sentiment analysis and a confidence score > 0.50. Since this target variable is not included in our raw data, we need to edit the dataset to include this field.

We can add this field directly in Azure using either ML Azure Designer or a Jupyter notebook. Here, we'll cover using ML Azure Designer.

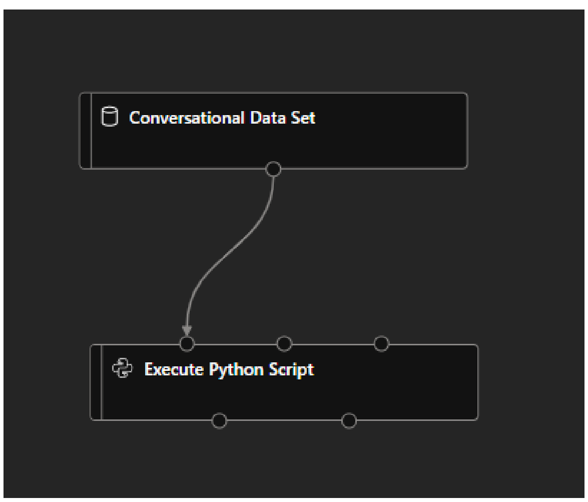
Preparing Data for the Classification Model

From the left-hand panel, select “Designer.” Then, click the “+” under New pipeline to create a new pipeline in Designer view.

From Designer view, you can begin creating your pipeline. In the left panel, search for the name of your dataset. Drag the dataset into the view.



Now search for “Execute Python Script” in the left panel, and drag this block into the view. Connect your dataset to the left side of the “Execute Python Script” block:



We now need to add a script to the “Execute Python Script” block. This script creates a new composite “success” field where success is defined as a conversation with positive sentiment and a confidence score greater than 0.50. Because AutoML doesn’t permit data to be excluded from training, this script also removes the data used to calculate that outcome.

Click on the “Execute Python Script” block and copy and paste the following code into the Python script box:

```
import numpy as np

def azureml_main(dataframe1 = None, dataframe2 = None):
```

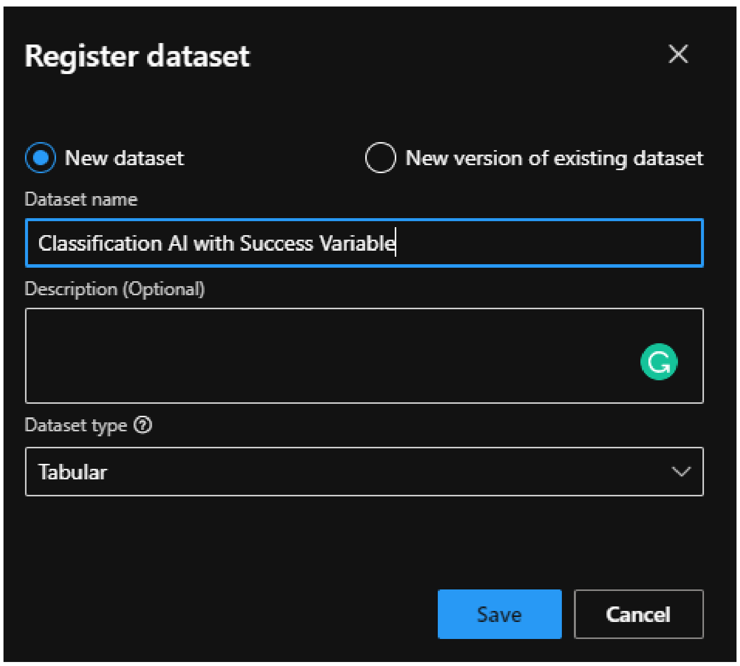
```
dataframe1['Success'] = np.logical_and(dataframe1['Sentiment Analysis'] == "positive", dataframe1["Confidence Score"] > 0.5)
del dataframe1['Sentiment Analysis']
del dataframe1['Confidence Score']
return dataframe1,
```

After clicking "Submit" on the pipeline, you will be prompted to select a compute instance. Select the compute instance that you previously created, and then run the pipeline. This may take a few minutes.

Once the pipeline has finished running, click on the "Execute Python Script" block. Then navigate to the "Outputs + logs" option. From here, you can visualize the dataset, "Result dataset," by clicking the bar graph icon next to it. You should see a table containing the raw data. Scroll all the way to the right, and you will see that a "Success" variable has been added.

Click the save button to the right of "Result dataset" to register the dataset. This will be the dataset used to train the classification model.

Name the dataset, and leave the dataset type as "Tabular."



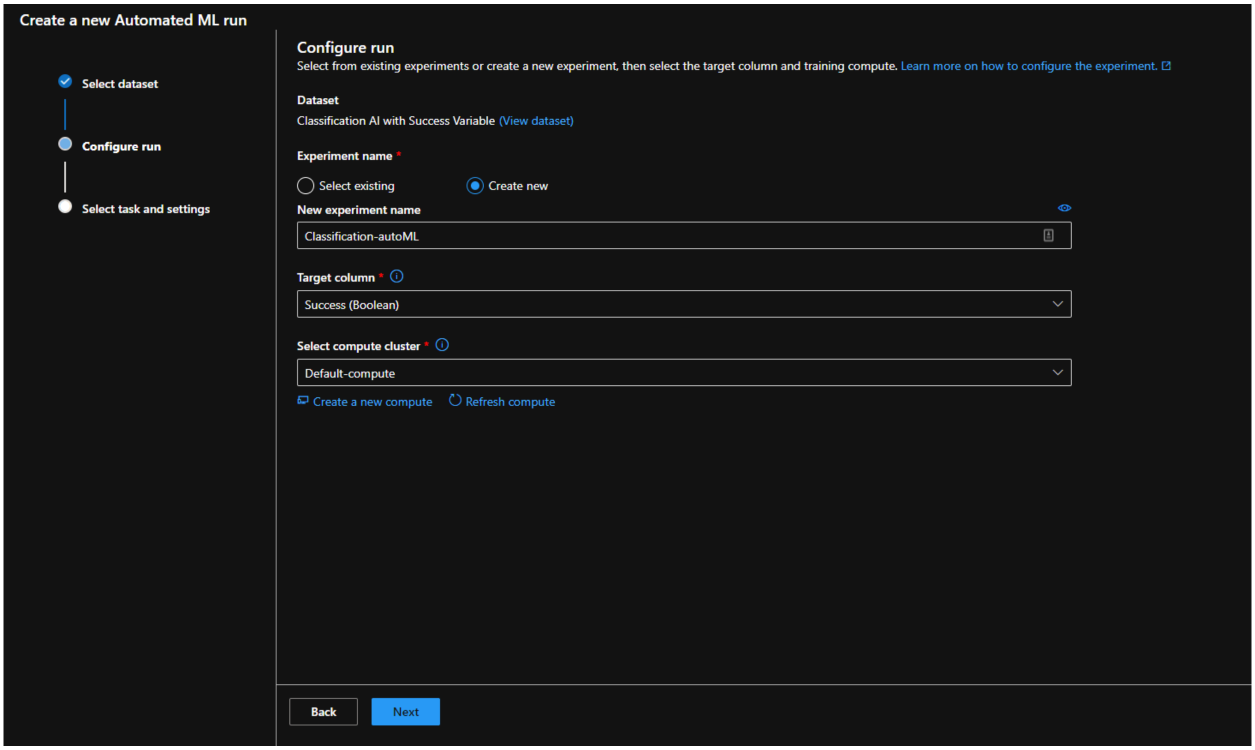
Click "Save," and now we're ready to train the model.

Training the Classification Model

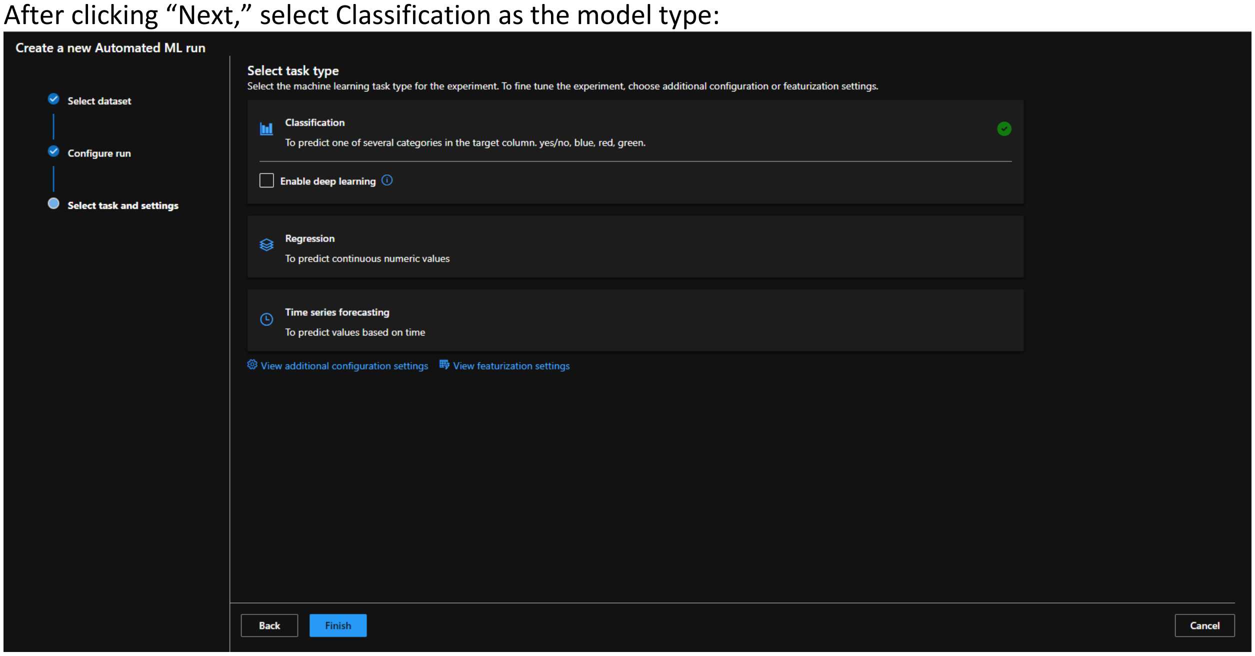
Now that we're ready to train the classification model, navigate back to the AutoML view by clicking "Automated ML" in the left-hand panel. Then, click "New Automated ML run." Select the dataset to be included in the experiment. This should be the conversation AI dataset that you just created that contains a "Success" variable.

After clicking "Next," you will be brought to a page to configure your run:

1. Under the experiment name, click “Create new,” and then type an experiment name.
2. For the classification model, we want to predict whether a conversation is successful so, for the “Target column,” select conversation duration.
3. Under “Select compute cluster”, select the compute cluster you created previously.



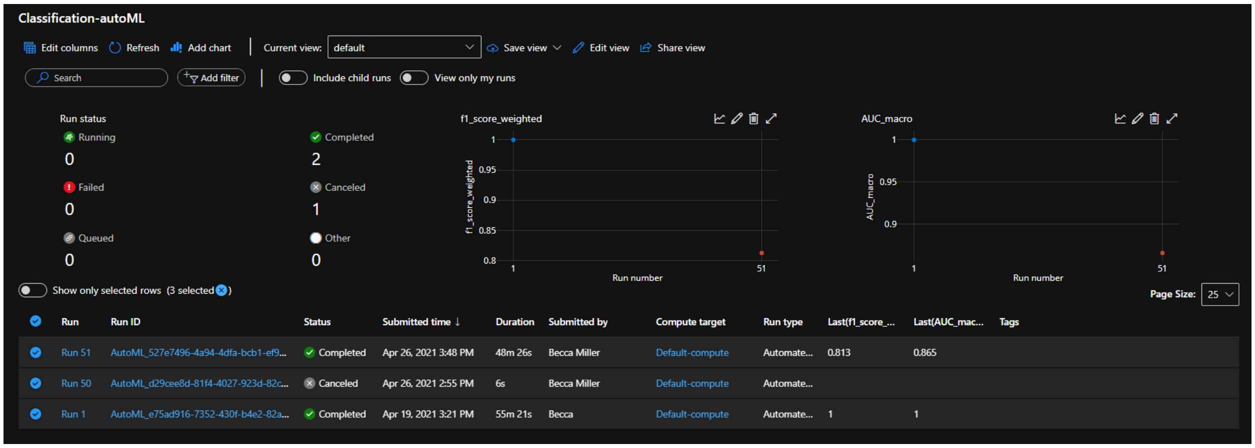
After clicking “Next,” select Classification as the model type:



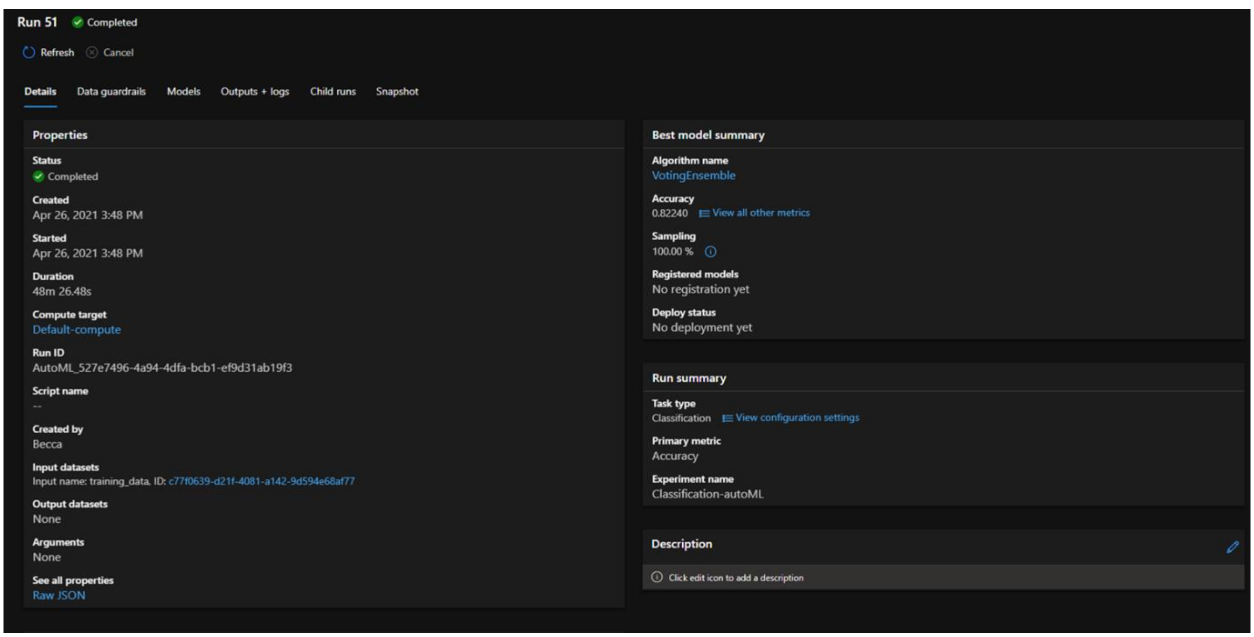
Click “Finish” to begin the run.

Interpreting the Classification Model

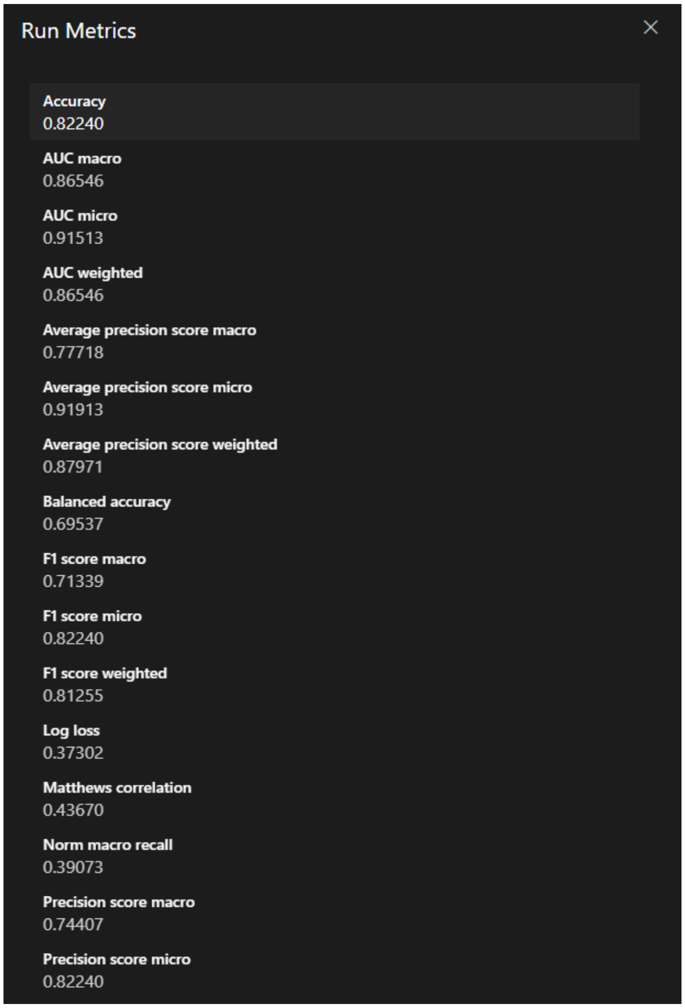
Once the model has finished running, navigate to the "Experiment" tab in the left panel. Locate the experiment that you just ran and click on the experiment name to navigate to the experiment page. This page will show the run history of your current experiment:



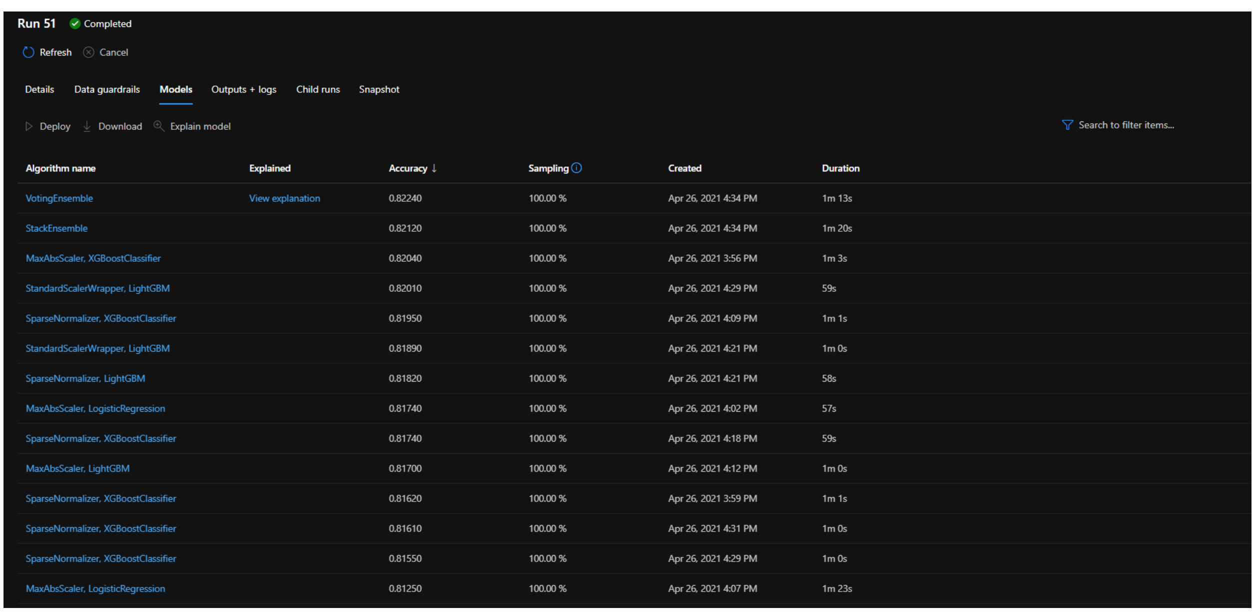
To view the results from your most recent experiment, click on the most recent run name in the run column (in this case "Run 51"). This will bring you to a page that describes the details of the run:



In our example, we can see that the VotingEnsemble was the best performing model with an accuracy of 0.82. By clicking "View all other metrics" under the best model summary, we can see all the metrics for this model:

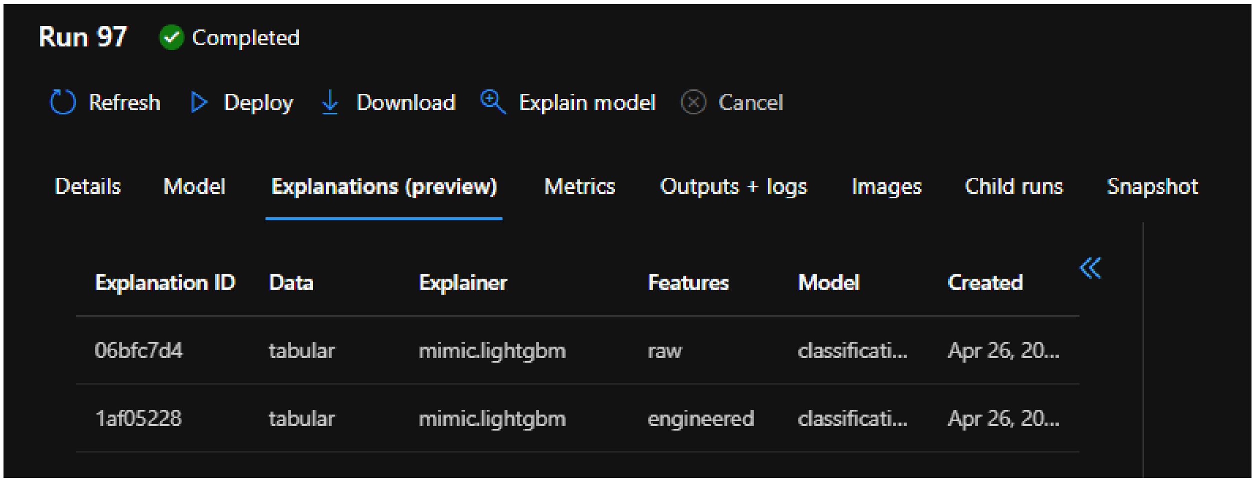


We can also view the statistics for all other tested algorithms by navigating to the “Models” tab:



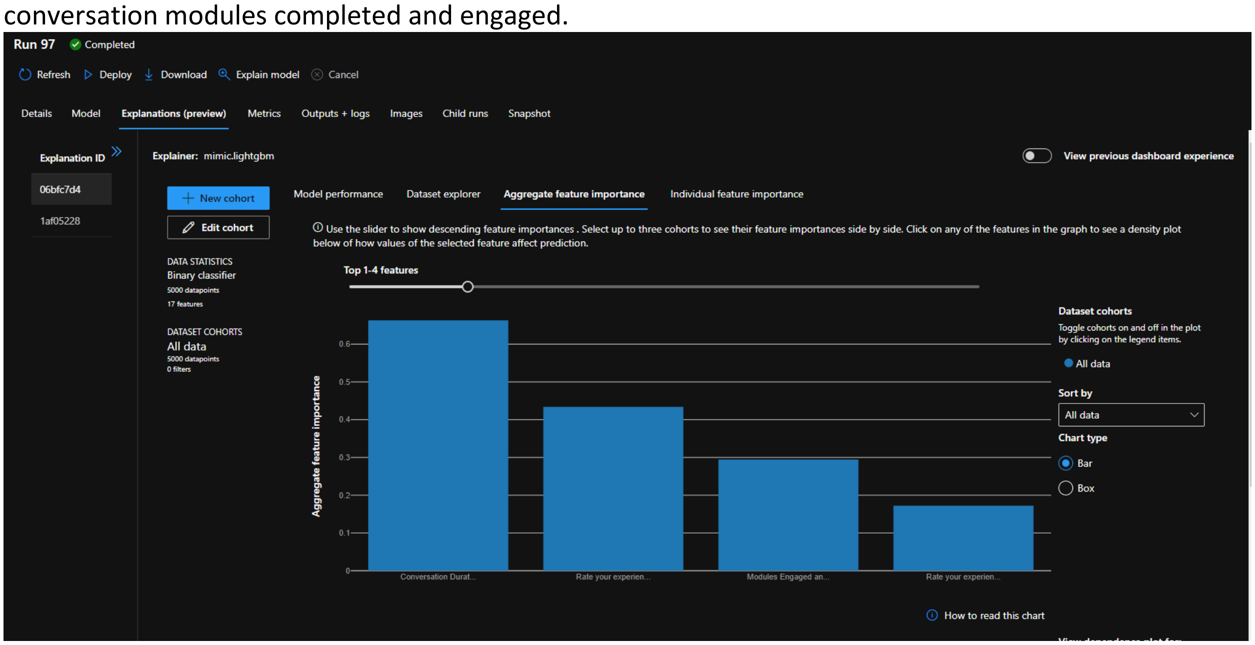
Algorithm name	Explained	Accuracy ↓	Sampling	Created	Duration
VotingEnsemble	View explanation	0.82240	100.00 %	Apr 26, 2021 4:34 PM	1m 13s
StackEnsemble		0.82120	100.00 %	Apr 26, 2021 4:34 PM	1m 20s
MaxAbsScaler, XGBoostClassifier		0.82040	100.00 %	Apr 26, 2021 3:56 PM	1m 3s
StandardScalerWrapper, LightGBM		0.82010	100.00 %	Apr 26, 2021 4:29 PM	59s
SparseNormalizer, XGBoostClassifier		0.81950	100.00 %	Apr 26, 2021 4:09 PM	1m 1s
StandardScalerWrapper, LightGBM		0.81890	100.00 %	Apr 26, 2021 4:21 PM	1m 0s
SparseNormalizer, LightGBM		0.81820	100.00 %	Apr 26, 2021 4:21 PM	58s
MaxAbsScaler, LogisticRegression		0.81740	100.00 %	Apr 26, 2021 4:02 PM	57s
SparseNormalizer, XGBoostClassifier		0.81740	100.00 %	Apr 26, 2021 4:18 PM	59s
MaxAbsScaler, LightGBM		0.81700	100.00 %	Apr 26, 2021 4:12 PM	1m 0s
SparseNormalizer, XGBoostClassifier		0.81620	100.00 %	Apr 26, 2021 3:59 PM	1m 1s
SparseNormalizer, XGBoostClassifier		0.81610	100.00 %	Apr 26, 2021 4:31 PM	1m 0s
SparseNormalizer, XGBoostClassifier		0.81550	100.00 %	Apr 26, 2021 4:29 PM	1m 0s
MaxAbsScaler, LogisticRegression		0.81250	100.00 %	Apr 26, 2021 4:07 PM	1m 23s

By default, AutoML provides an explanation for the best-performing model. Here, we'll focus on the VotingEnsemble. By clicking "View explanation," we can view information about this model.



For our explanation, we'll focus on the raw features. By clicking on the explanation ID, the window to the right populates with information about our dataset. This window can be used to explore the relationships between variables in the dataset.

By navigating to the aggregate feature importance tab, we can explore which features are most important for determining the output of our model. For this model, the strongest predictors of conversation success included conversation duration, self-reported user ratings, and the number of conversation modules completed and engaged.



By defining a new cohort, this window can also be used to explore relationships between subsets of the data.

Business impact: The data from the model can be used to identify stronger indicators of conversation success and suggests that customer engagement is a strong indicator of success.

Creating a Time Series Model in AutoML

Key Question: How many conversations are expected to occur on future dates for each platform type, and how do these trends over time vary between platforms?

Compared to the previous two models, creating a time series model takes a little more data preparation. Luckily, we can perform this preparation directly in Azure using either ML Azure Designer or a Jupyter notebook. Here, we'll cover using ML Azure Designer.

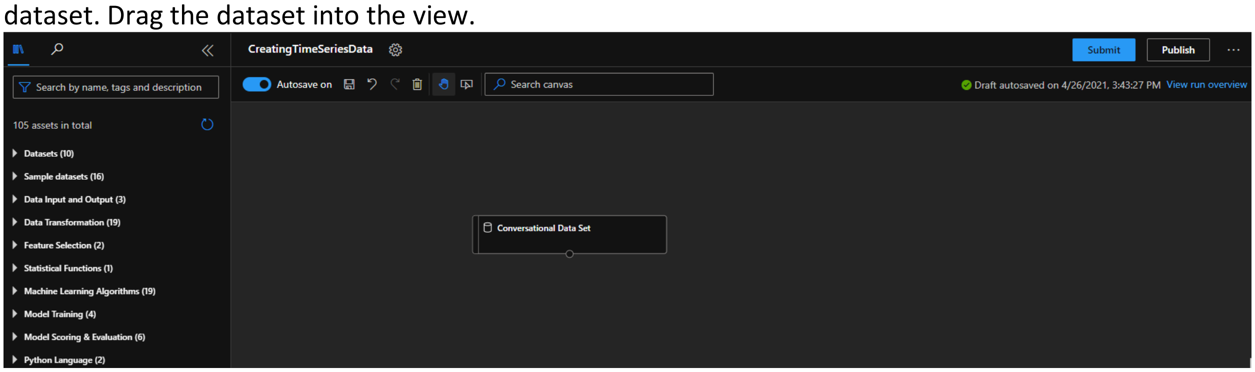
Preparing Data for Time Series Model

To train a time series model, the expected data format is as follows:

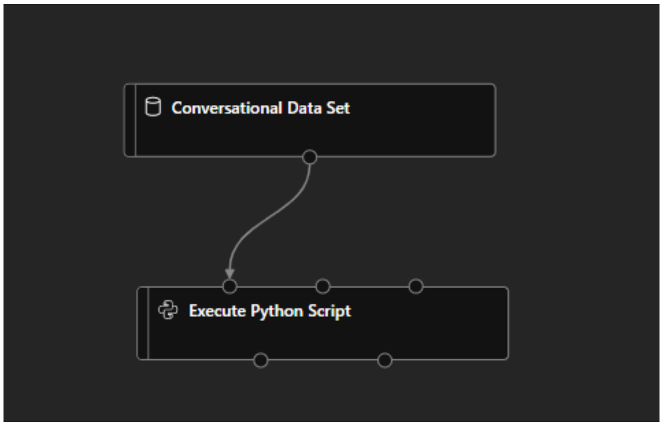
Date	Conversation Count	Conversation Type
2021-01-01	53	IVR
2021-01-01	48	Voice
2021-01-01	55	Chat
2021-01-02	42	IVR
2021-01-02	45	Voice
2021-01-02	53	Chat
…	…	…

We can reshape our raw data to fit this format using Azure Designer. From the left-hand panel, select "Designer." Then, click the "+" under New pipeline to create a new pipeline in Designer view.

From Designer view, you can begin creating your pipeline. In the left panel, search for the name of your dataset. Drag the dataset into the view.



Now search for "Execute Python Script" in the left panel, and drag this block into the view. As with the classification model, connect your dataset to the left side of the "Execute Python Script" block:



We now need to add a script to the “Execute Python Script” block. This script transforms the data to match the format described above.

Click on the “Execute Python Script” block and copy and paste the following code into the Python script box:

```
import pandas as pd
import numpy as np

CONVERSATION_TYPE = 'Conversation Type'
CONVERSATION_DATE = 'Date of Conversation'
CONVERSATION_COUNT = 'Conversation Count'

"""
This function takes the full conversation model dataset and converts it into a dataframe
that has a column for the date and a column for each conversation type containing the
associated conversation count for that date
"""
def azureml_main(dataframe1 = None, dataframe2 = None):
    # Convert conversation date into right format
    dataframe1[CONVERSATION_DATE] = pd.to_datetime(dataframe1[CONVERSATION_DATE])

    # Get sorted list of all possible dates
    dates = list(set(dataframe1[CONVERSATION_DATE]))
    dates.sort()


    # The only info we really care about is conversation type, date of conversation, and number of conversations, so
    # let's create a new dataframe1 fitting this format
    conversation_types = list(set(dataframe1[CONVERSATION_TYPE].values))

    # Populate a dictionary of the format {str: list()}, where the key is the conversation type and the
    # the value is the sequential list of conversation counts by date
```

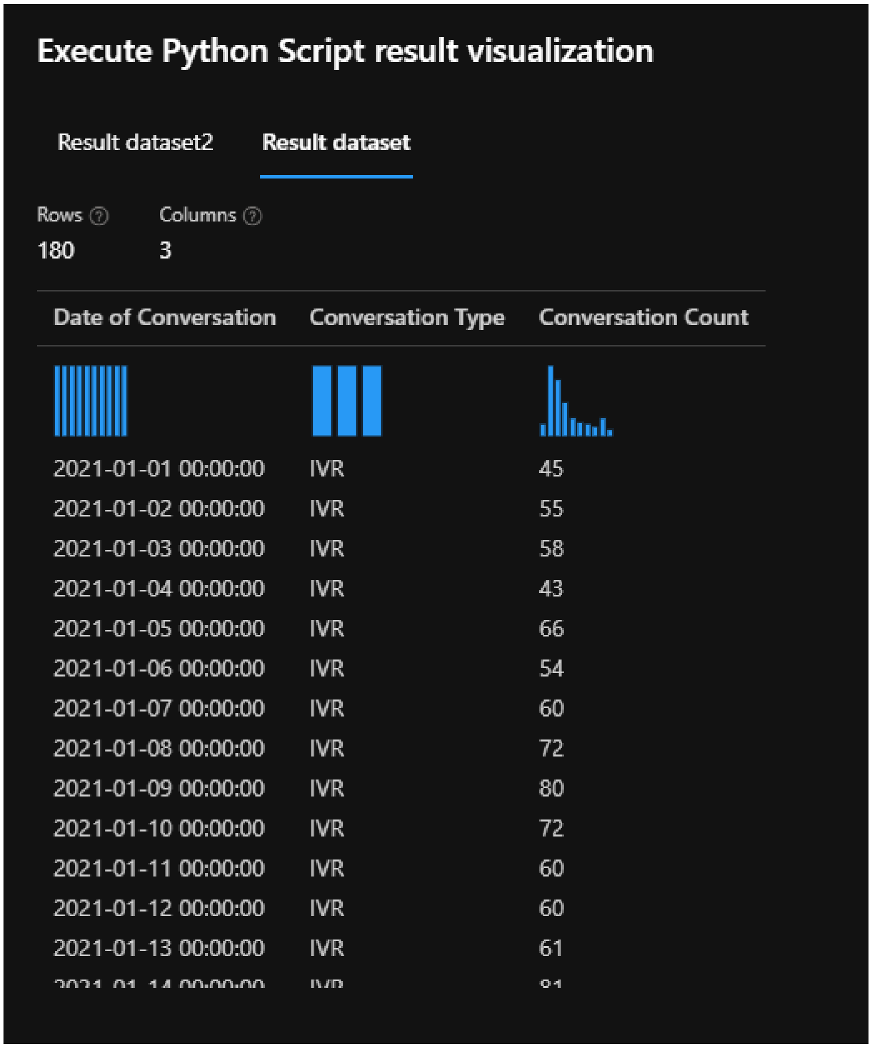
```
  conv_to_count = {}
  for conv in conversation_types:
     conv_to_count[conv] = []
     for date in dates:
        conv_count = sum(np.logical_and(dataframe1[CONVERSATION_TYPE] == conv, dataframe1[CONVERSATION_
DATE] == date))
         conv_to_count[conv].append(conv_count)

   # Convert the above dictionary into a dataframe that has a column for the date and a column for each
  # conversation type containing the associated conversation count for that date
  time_series_df = pd.DataFrame()
  for conv in conversation_types:
     platform_df = pd.DataFrame()
     platform_df[CONVERSATION_DATE] = dates
     platform_df[CONVERSATION_TYPE] = conv
     platform_df[CONVERSATION_COUNT] = conv_to_count[conv]
     time_series_df = time_series_df.append(platform_df)
  return time_series_df,
```

After clicking “Submit” on the pipeline, you will be prompted to select a compute instance. Select the compute instance that you previously created, and then run the pipeline. This may take a few minutes.

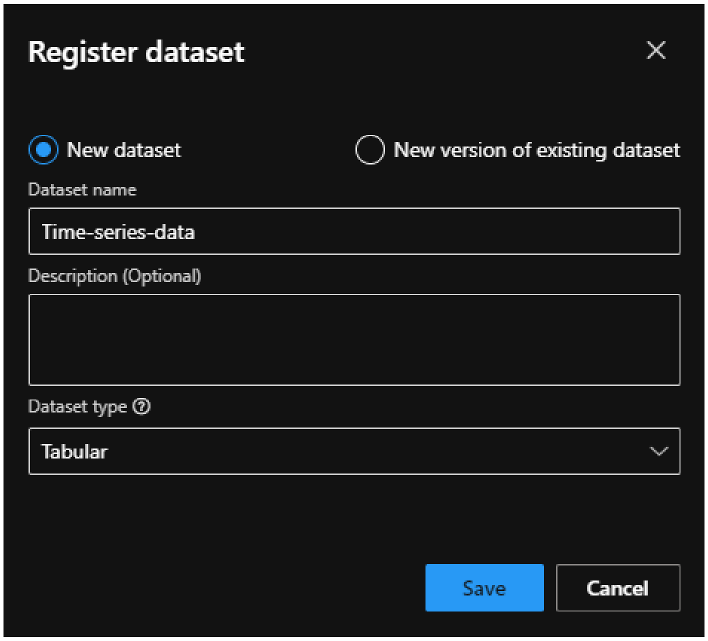
Once the pipeline has finished running, click on the “Execute Python Script” block. Then navigate to the “Outputs + logs” option. From here, you can visualize the dataset, “Result dataset,” by clicking the bar graph icon next to it.

You’ll see a table containing the transformed data:



Date of Conversation	Conversation Type	Conversation Count
2021-01-01 00:00:00	IVR	45
2021-01-02 00:00:00	IVR	55
2021-01-03 00:00:00	IVR	58
2021-01-04 00:00:00	IVR	43
2021-01-05 00:00:00	IVR	66
2021-01-06 00:00:00	IVR	54
2021-01-07 00:00:00	IVR	60
2021-01-08 00:00:00	IVR	72
2021-01-09 00:00:00	IVR	80
2021-01-10 00:00:00	IVR	72
2021-01-11 00:00:00	IVR	60
2021-01-12 00:00:00	IVR	60
2021-01-13 00:00:00	IVR	61

Click the save button to the right of “Result dataset” to register the dataset. This dataset will be used to train the time series model:



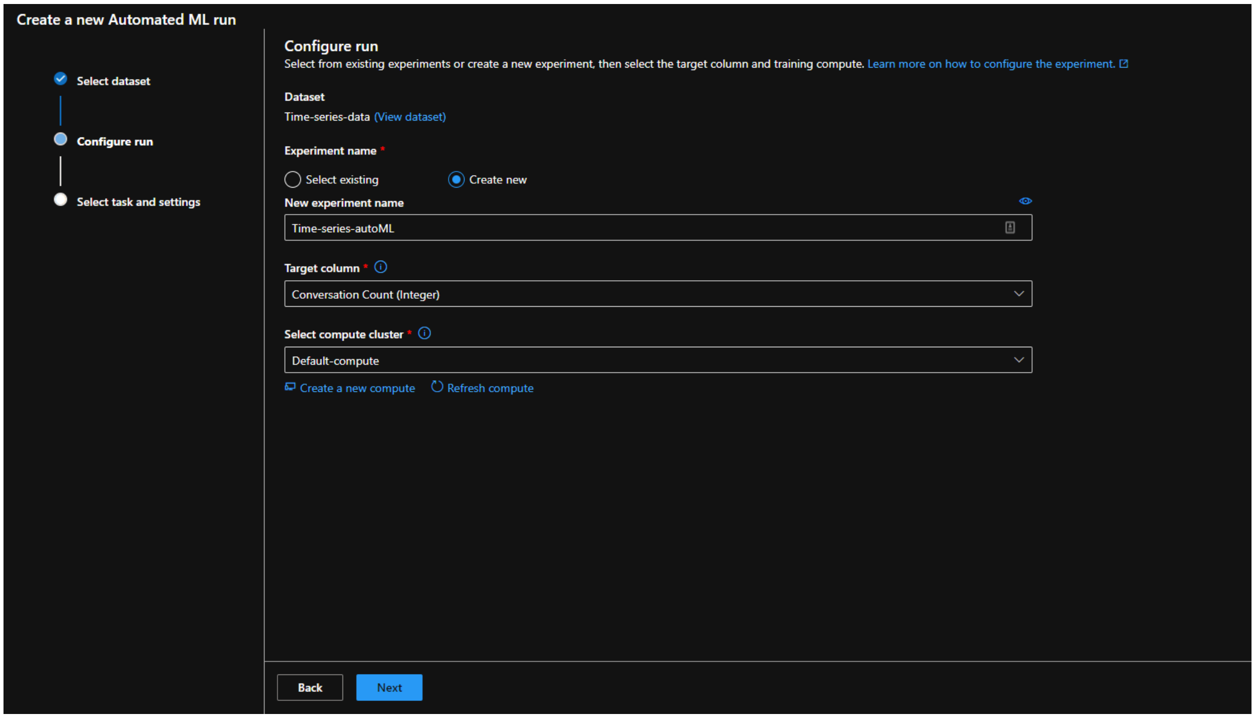
Hit "Save," and now we're ready to train the model.

Training the Time Series Model

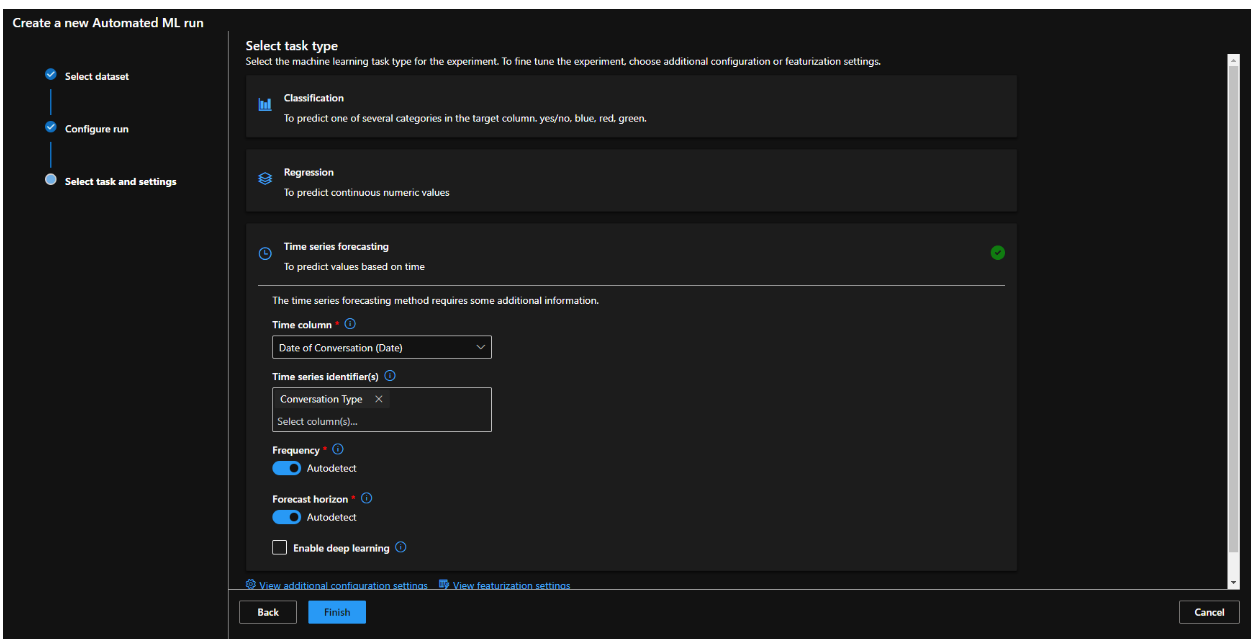
Now that we're ready to train the time series model, navigate back to the AutoML view by clicking "Automated ML" in the left-hand panel. Then, click "New Automated ML run." Select the dataset to be included in the experiment. This should be the new time series dataset you just created.

After clicking "Next," you will be brought to a page to configure your run:

1. Under the experiment name, click "Create new," and then type an experiment name.
2. For the time series model, we want to predict how many conversations will occur on a given day, so for the "Target column," select conversation count.
3. Under "Select compute cluster", select the compute cluster you created previously.



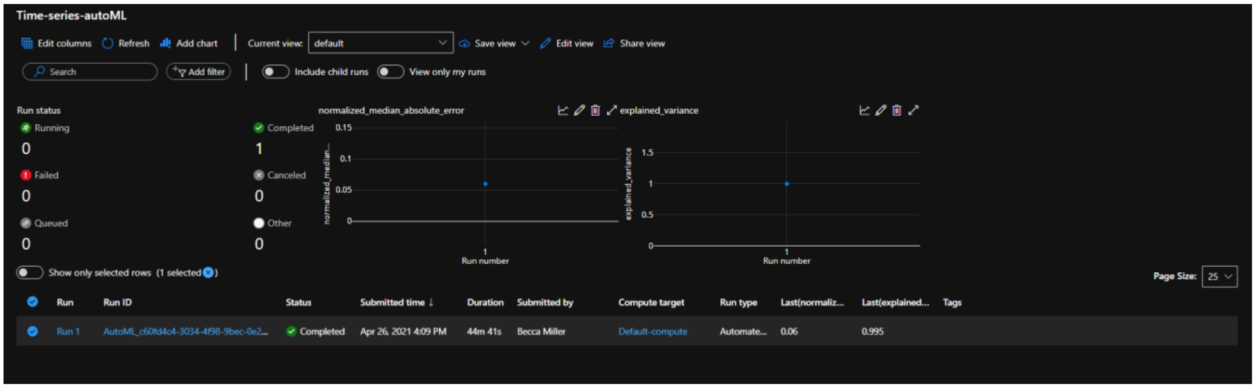
After clicking “Next,” select Time Series as the model type. The Time column should show Date of Conversation by default. In the Time series identifier column, add “Conversation type” to indicate that we’re predicting the counts for different types of conversations across dates:



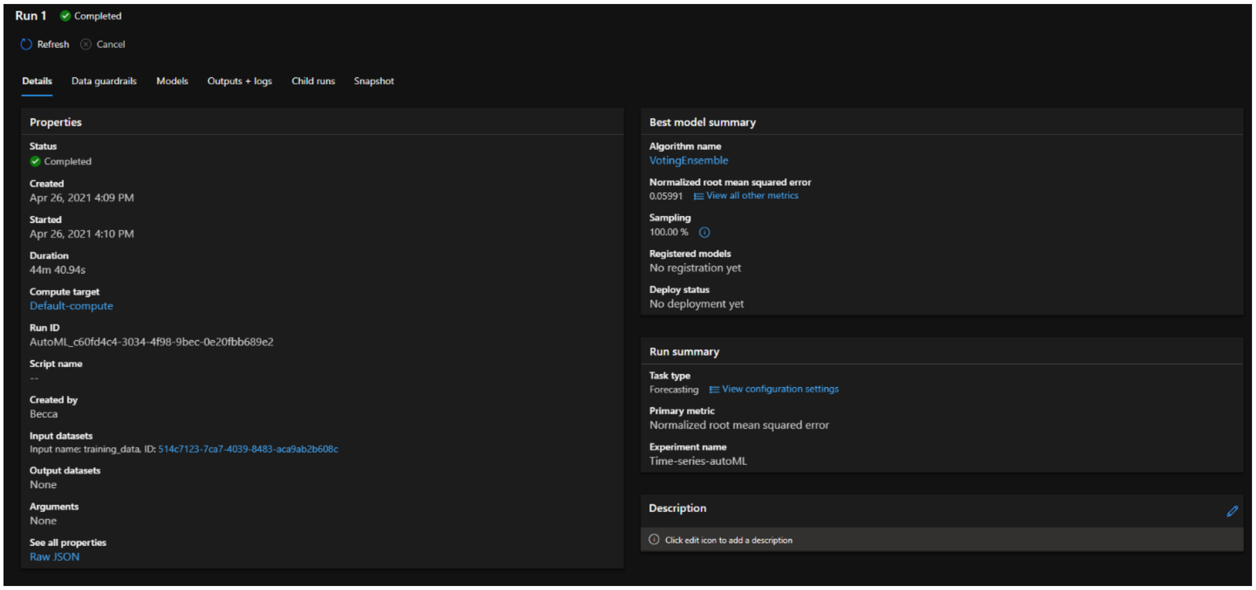
Click “Finish” to begin the run.

Interpreting the Time Series Model

Once the model has finished running, navigate to the "Experiment" tab in the left panel. Locate the experiment that you just ran and click on the experiment name to navigate to the experiment page. This page will show the run history of your current experiment:



To view the results from your most recent experiment, click on the most recent run name in the run column (in this case "Run 1"). This will bring you to a page that describes the details of the run:



In our example, we can see that the VotingEnsemble was the best performing model with an accuracy of 0.82. By clicking "View all other metrics" under the best model summary, we can see all the metrics for this model:

Run Metrics

Explained variance
0.99508

Mean absolute error
3.0094

Mean absolute percentage error
13.790

Median absolute error
2.8984

Normalized mean absolute error
0.059905

Normalized median absolute error
0.059905

Normalized root mean squared error
0.059905

Normalized root mean squared log error
0.083631

R2 score
0.99406

Root mean squared error
3.3367

Root mean squared log error
0.14617

Spearman correlation
1.0000

We can also view the statistics for all other tested algorithms by navigating to the "Models" tab:

Run 1 Completed

Refresh Cancel

Details Data guardrails **Models** Outputs + logs Child runs Snapshot

Deploy Download Explain model Search to filter items...

Algorithm name	Explained	Normalized root mean s... ↑	Sampling	Created	Duration
VotingEnsemble	Not supported	0.05991	100.00 %	Apr 26, 2021 4:53 PM	1m 16s
ExponentialSmoothing	Not supported	0.06314	100.00 %	Apr 26, 2021 4:20 PM	1m 19s
MinMaxScaler, RandomForest		0.06755	100.00 %	Apr 26, 2021 4:45 PM	1m 4s
StandardScalerWrapper, DecisionTree		0.06755	100.00 %	Apr 26, 2021 4:46 PM	58s
StandardScalerWrapper, DecisionTree		0.06755	100.00 %	Apr 26, 2021 4:44 PM	1m 5s
MinMaxScaler, DecisionTree		0.06755	100.00 %	Apr 26, 2021 4:44 PM	57s
MinMaxScaler, DecisionTree		0.06755	100.00 %	Apr 26, 2021 4:39 PM	1m 1s
RobustScaler, DecisionTree		0.06755	100.00 %	Apr 26, 2021 4:35 PM	58s
RobustScaler, DecisionTree		0.06755	100.00 %	Apr 26, 2021 4:32 PM	57s
StandardScalerWrapper, DecisionTree		0.06755	100.00 %	Apr 26, 2021 4:30 PM	1m 16s
RobustScaler, DecisionTree		0.06755	100.00 %	Apr 26, 2021 4:25 PM	56s
RobustScaler, DecisionTree		0.06755	100.00 %	Apr 26, 2021 4:22 PM	56s
StandardScalerWrapper, GradientBoosting		0.06854	100.00 %	Apr 26, 2021 4:47 PM	1m 3s
ProphetModel		0.07109	100.00 %	Apr 26, 2021 4:11 PM	1m 32s

Unlike for the classification and regression models, AutoML does not currently provide explanations for the time series models. However, it does still provide the ability to deploy or download a selected model.

By deploying or downloading the model, the model can be used to predict the number of conversations of various types that are expected to occur on future dates. We would find that this model predicts conversation counts that follow the current trends: The number of IVR conversations increasing over time and the number of voice and chat conversations decreasing.

Business impact: The data from the model can be used to recognize trends in the types of conversations that are occurring over time and to identify future deviations from those trends.