

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

Contents

Create an Azure Machine Learning Workspace.....	2
Creating Compute resources	3
Data Preparation.....	4
Creating a Regression Model in AutoML	6
Training the Regression Model	6
Interpreting the Regression Model.....	7
Creating a Classification Model in AutoML.....	10
Preparing Data for the Classification Model.....	11
Training the Classification Model.....	12
Interpreting the Classification Model	14
Creating a Time Series Model in AutoML	17
Preparing Data for Time Series Model.....	17
Training the Time Series Model	21
Interpreting the Time Series Model.....	23

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:

Machine learning ...
Create a machine learning workspace

Basics Networking Advanced Tags Review + create

Project details
Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ Primary Azure Subscription

Resource group * ⓘ (New) ConversationalAI2021
[Create new](#)

Workspace details
Specify the name and region for the workspace.

Workspace name * ⓘ ConversationalIML ✓

Region * ⓘ East US 2

Storage account * ⓘ (new) conversational0985295282
[Create new](#)

Key vault * ⓘ (new) conversational3529712461
[Create new](#)

Application insights * ⓘ (new) conversational7723396820
[Create new](#)

Container registry * ⓘ None
[Create new](#)

Now, you can click “Review + create” and 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.

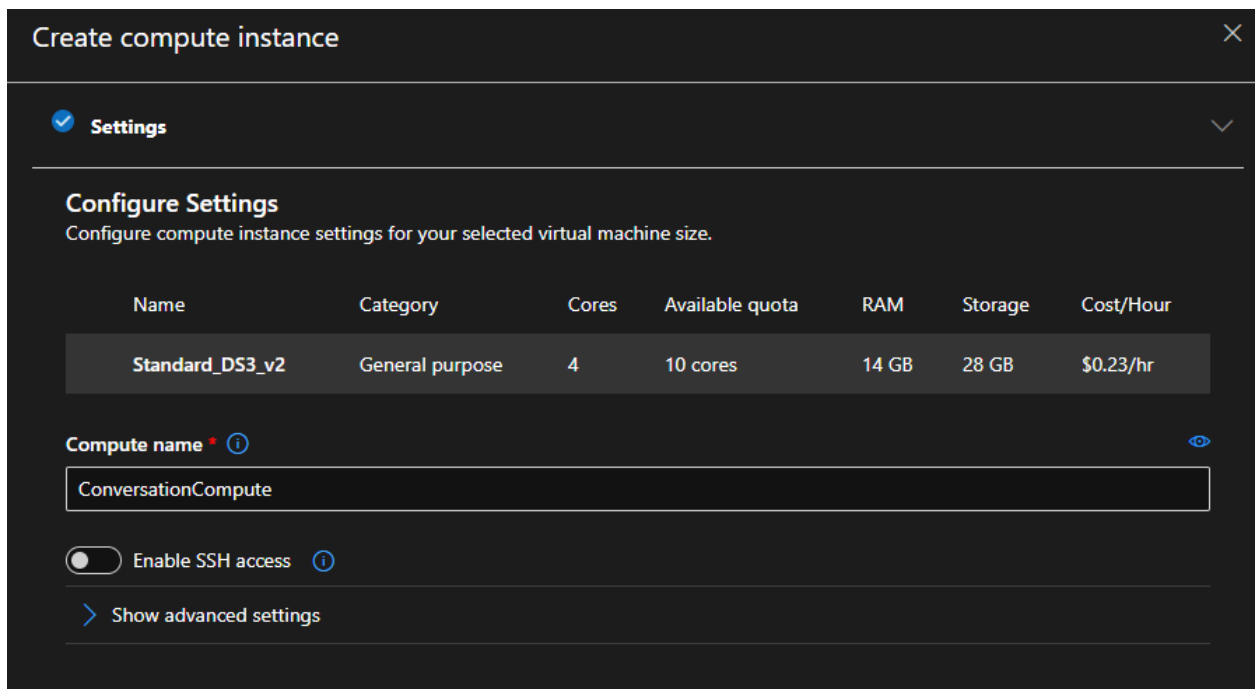
From this prompt, select the directory and subscription used to create your ML workspace. Then, select your workspace and click “Get started”

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:



Create compute instance

Settings

Configure Settings
Configure compute instance settings for your selected virtual machine size.

Name	Category	Cores	Available quota	RAM	Storage	Cost/Hour
Standard_DS3_v2	General purpose	4	10 cores	14 GB	28 GB	\$0.23/hr

Compute name ⓘ

ConversationCompute

☐ Enable SSH access ⓘ

[Show advanced settings](#)

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:

Create dataset from local files

Datastore and file selection

Select or create a datastore *

☒ Currently selected datastore: workspaceblobstore (Azure Blob Storage) (Default)

☐ Previously created datastore

☐ Create new datastore

Select files for your dataset *

These files will be uploaded to your selected datastore and made available in your workspace. Supported file types include: delimited (i.e. csv, tsv), Parquet, JSON Lines, and plain text.

1 files selected. Total size 1.599 MiB. 0/1 files uploaded

File name	Size (MiB)	Upload %	Status
conversation-data.csv	1.599		

Upload path

UI Files will be uploaded to '\$(Upload path)/04-26-2021_033628.UTC'

☐ Skip data validation ⓘ

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

Create dataset from local files

Basic info

Datastore and file selection

Settings and preview

Schema

Confirm details

Settings and preview

These settings were automatically detected. Please verify that the selections were made correctly or update.

File format

Delimited

Delimiter

Comma

Example

Field1,Field2,Field3

Encoding

UTF-8

Column headers

All files have same headers

Skip rows

None

	Id	Column1	Conversation ID	Date of Convers...	Conversation Type	Language	Platform	Device
1	0		3mo7BjW9zFmz3gNs...	2021-01-26 00:00:00	chat	English	SMS/text	laptops and ta
2	1		Y2GV4PNx-OeRy-ke...	2021-02-23 00:00:00	IVR	English	IVR	telephony
3	2		mVYG4YbaOXux1C2...	2021-01-16 00:00:00	chat	English	web chat	laptops and ta
4	3		vu8DHRdY-koIV-q3...	2021-01-07 00:00:00	IVR	English	IVR	telephony
5	4		jHjbR4bx-PRFR-UoM...	2021-01-24 00:00:00	IVR	English	IVR	telephony
6	5		a61v923P-yoIS-yqIC...	2021-01-15 00:00:00	IVR	English	IVR	telephony
7	6		F61GMaDS-Vqkt-Lgc...	2021-01-10 00:00:00	IVR	English	IVR	telephony
8	7		ZhrfjvMV-pyly-plpz...	2021-01-30 00:00:00	IVR	English	IVR	telephony
9	8		hj3B9q2WKvYJzQb...	2021-01-03 00:00:00	chat	Spanish (Mexican)	web chat	laptops and ta
10	9		amzn1.echo-api.sessi...	2021-02-15 00:00:00	voice	Spanish (Mexican)	Alexa	mobile device

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:

Update dataset schema

Basic info

Datastore selection

Settings and preview

Schema

Confirm details

Schema

Column types are auto-detected based on the first 200 rows of the data. Please make any necessary adjustments. Values not aligning with the specified column type will fail conversion and would be either null-filled or replaced with error value.

Search to filter items...

Include	Column name	Properties	Type	Format settings and example
<input type="checkbox"/>	Path	Not applicable to selected type	String	
<input type="checkbox"/>	Column1	Not applicable to selected type	Integer	0, 1, 2
<input checked="" type="checkbox"/>	Conversation ID	Not applicable to selected type	String	3mo7BjW9zFmz3gNs81190-I, Y...
<input checked="" type="checkbox"/>	Date of Conversation	Timestamp	Date	2021-01-26 00:00:00, 2021-02-...
<input checked="" type="checkbox"/>	Conversation Type	Not applicable to selected type	String	chat, IVR, chat
<input checked="" type="checkbox"/>	Language	Not applicable to selected type	String	English, English, English
<input checked="" type="checkbox"/>	Platform	Not applicable to selected type	String	SMS/text, IVR, web chat
<input checked="" type="checkbox"/>	Device	Not applicable to selected type	String	laptops and tablets - screen, tel...
<input checked="" type="checkbox"/>	Transfer to Live Agent	Not applicable to selected type	String	no, yes, yes
<input checked="" type="checkbox"/>	Sentiment Analysis	Not applicable to selected type	String	negative, positive, neutral
<input checked="" type="checkbox"/>	Confidence Score	Not applicable to selected type	Decimal	0.28, 0.73, 0.91

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.

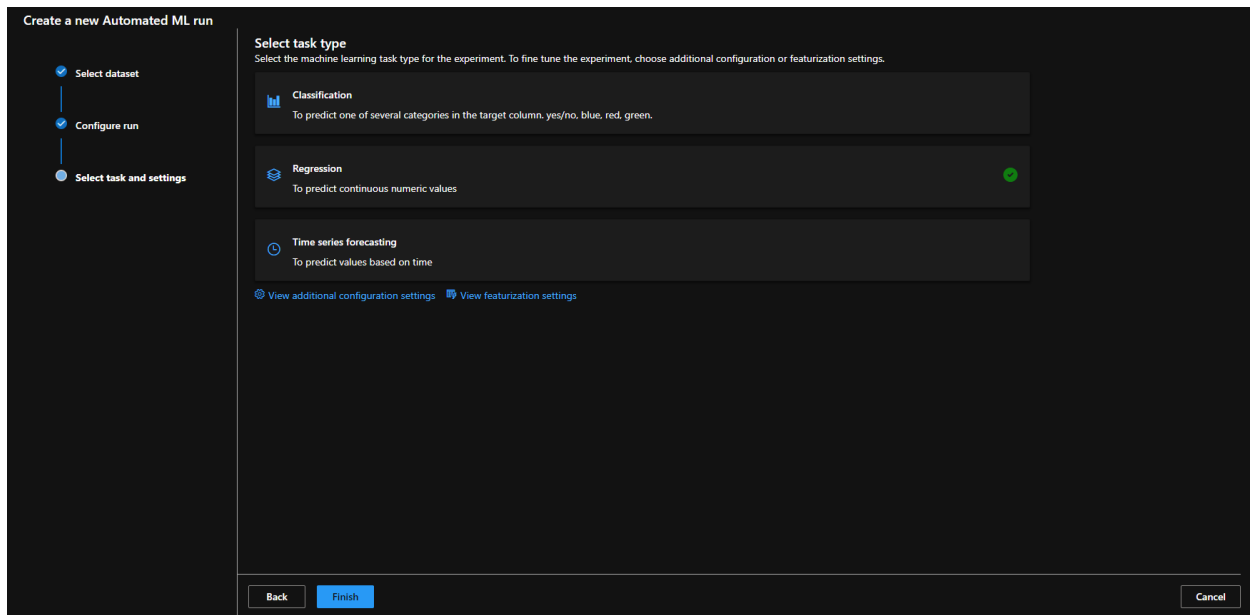
The screenshot shows the 'Create a new Automated ML run' interface. On the left, a vertical sidebar contains three steps: 'Select dataset' (completed with a blue checkmark), 'Configure run' (active with a blue circle), and 'Select task and settings' (inactive with a grey circle). The main area is titled 'Configure run' and includes a subtitle: 'Select from existing experiments or create a new experiment, then select the target column and training compute. [Learn more on how to configure the experiment.](#)'

The configuration section includes:

- Dataset:** 'Conversational AI Data (View dataset)'
- Experiment name:** Radio buttons for 'Select existing' and 'Create new' (selected). Below is a text input field for 'New experiment name' containing 'Regression-autoML'.
- Target column:** A dropdown menu showing 'Conversation Duration (Integer)'.
- Select compute cluster:** A dropdown menu showing 'Default-compute'.
- At the bottom of the configuration section are links: 'Create a new compute' and 'Refresh compute'.

At the bottom of the page are two buttons: 'Back' and 'Next'.

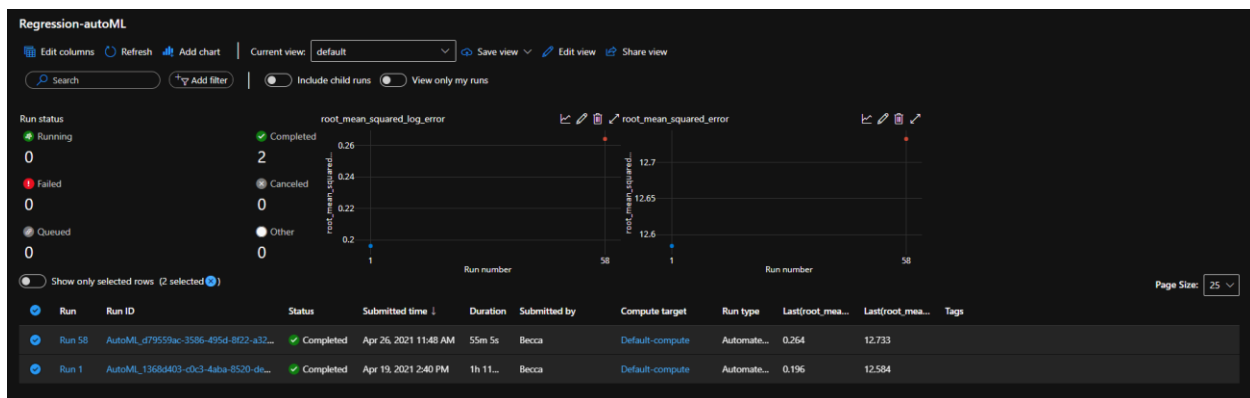
After clicking "Next," select Regression as the model type:



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:

Run 58 Completed

[Refresh](#) [Cancel](#)

[Details](#) [Data guardrails](#) [Models](#) [Outputs + logs](#) [Child runs](#) [Snapshot](#)

Properties

Status
Completed

Created
Apr 26, 2021 11:48 AM

Started
Apr 26, 2021 11:48 AM

Duration
55m 5.234s

Compute target
Default-compute

Run ID
AutoML_d79559ac-3586-495d-8f22-a32130d50be9

Script name
--

Created by
Becca

Input datasets
Input name: training_data, ID: 3089aa46-2c27-49bc-83a3-2abbd974e64

Output datasets
None

Arguments
None

[See all properties](#)
[Raw JSON](#)

Best model summary

Algorithm name
VotingEnsemble

Spearman correlation
0.57299 [View all other metrics](#)

Sampling
100.00 %

Registered models
No registration yet

Deploy status
No deployment yet

Run summary

Task type
Regression [View configuration settings](#)

Primary metric
Spearman correlation

Experiment name
Regression-autoML

Description

[Click edit icon to add a description](#)

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:

Run Metrics

Explained variance
0.32749

Mean absolute error
10.135

Mean absolute percentage error
22.439

Median absolute error
8.5479

Normalized mean absolute error
0.088127

Normalized median absolute error
0.074329

Normalized root mean squared error
0.11072

Normalized root mean squared log error
0.087925

R2 score
0.32696

Root mean squared error
12.733

Root mean squared log error
0.26413

Spearman correlation
0.57299

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

Run 58 ✔ Completed

[Refresh](#)
[Cancel](#)

[Details](#)
[Data guardrails](#)
[Models](#)
[Outputs + logs](#)
[Child runs](#)
[Snapshot](#)

[Deploy](#)
[Download](#)
[Explain model](#)

Algorithm name	Explained	Spearman correlation ↓	Sampling ⓘ	Created	Duration
VotingEnsemble	View explanation	0.57299	100.00 %	Apr 26, 2021 12:42 PM	1m 18s
StackEnsemble		0.57259	100.00 %	Apr 26, 2021 12:42 PM	1m 21s
MaxAbsScaler, RandomForest		0.57121	100.00 %	Apr 26, 2021 12:24 PM	1m 15s
StandardScalerWrapper, XGBoostRegressor		0.57038	100.00 %	Apr 26, 2021 12:13 PM	1m 24s
StandardScalerWrapper, GradientBoosting		0.56912	100.00 %	Apr 26, 2021 12:39 PM	1m 3s
MaxAbsScaler, RandomForest		0.56894	100.00 %	Apr 26, 2021 12:25 PM	1m 4s
MaxAbsScaler, RandomForest		0.56879	100.00 %	Apr 26, 2021 12:20 PM	1m 4s
MaxAbsScaler, RandomForest		0.56836	100.00 %	Apr 26, 2021 12:19 PM	1m 1s
MaxAbsScaler, XGBoostRegressor		0.56790	100.00 %	Apr 26, 2021 11:59 AM	5m 34s
StandardScalerWrapper, DecisionTree		0.56702	100.00 %	Apr 26, 2021 12:31 PM	59s
MaxAbsScaler, GradientBoosting		0.56694	100.00 %	Apr 26, 2021 12:32 PM	1m 4s
MaxAbsScaler, GradientBoosting		0.56691	100.00 %	Apr 26, 2021 12:36 PM	1m 27s
SparseNormalizer, ExtremeRandomTrees		0.56614	100.00 %	Apr 26, 2021 12:36 PM	1m 0s
SparseNormalizer, ExtremeRandomTrees		0.56567	100.00 %	Apr 26, 2021 12:28 PM	1m 39s
MaxAbsScaler, RandomForest		0.56555	100.00 %	Apr 26, 2021 12:35 PM	1m 2s

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:

Run 112 ✔ Completed

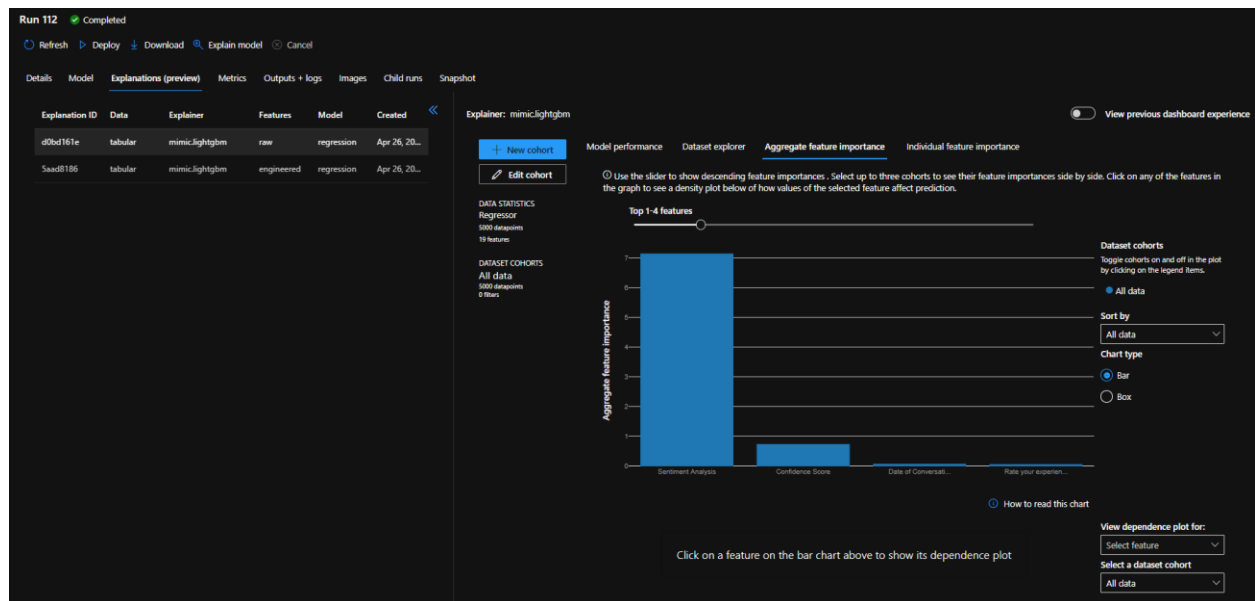
[Refresh](#)
[Deploy](#)
[Download](#)
[Explain model](#)
[Cancel](#)

[Details](#)
[Model](#)
[Explanations \(preview\)](#)
[Metrics](#)
[Outputs + logs](#)
[Images](#)
[Child runs](#)
[Snapshot](#)

Explanation ID	Data	Explainer	Features	Model	Created
d0bd161e	tabular	mimic.lightgbm	raw	regression	Apr 26, 20...
5aad8186	tabular	mimic.lightgbm	engineered	regression	Apr 26, 20...

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?

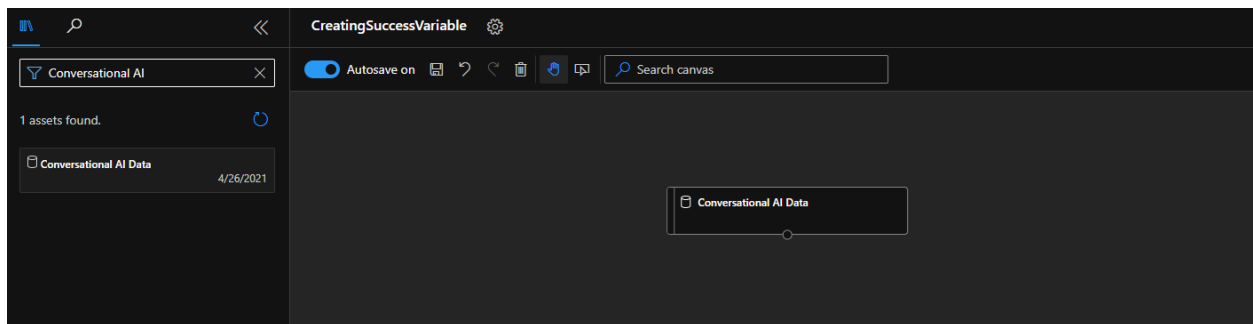
Now we're ready to create a second model in AutoML. For this model, we want to predict whether a conversation is successful. We're defining success as a conversation with a positive sentiment analysis and a confidence score > 0.50. Since this target variable is not included in our raw data, we need to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

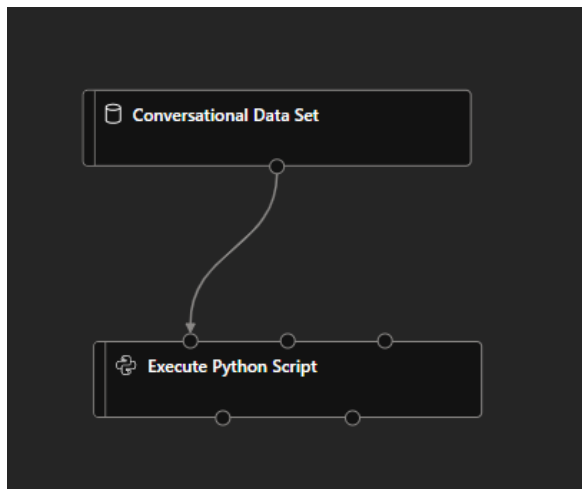
Preparing Data for the Classification Model

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. 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):
```

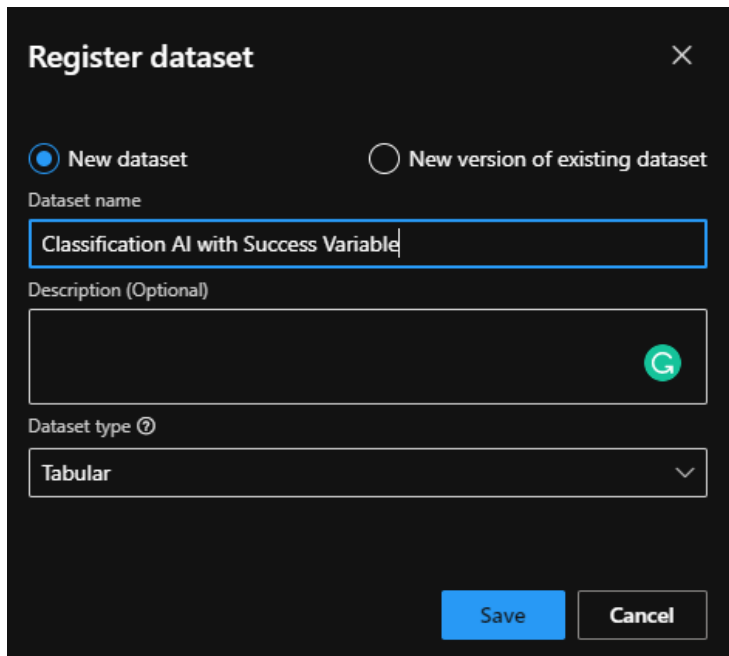
```
dataframe1['Success'] = np.logical_and(dataframe1['Sentiment Analysis'] == "positive", dataframe1['Confidence Score'] > 0.5)
del dataframe1['Sentiment Analysis']
del dataframe1['Confidence Score']
return dataframe1,
```

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.

The screenshot shows the 'Configure run' step of the 'Create a new Automated ML run' wizard. On the left, a vertical sidebar contains three steps: 'Select dataset' (checked), 'Configure run' (active), and 'Select task and settings'. The main area is titled 'Configure run' and includes a subtitle: 'Select from existing experiments or create a new experiment, then select the target column and training compute. [Learn more on how to configure the experiment.](#)'

The configuration options are as follows:

- Dataset:** Classification AI with Success Variable ([View dataset](#))
- Experiment name:** Radio buttons for 'Select existing' and 'Create new' (selected). Below is a text input field for 'New experiment name' containing 'Classification-autoML'.
- Target column:** A dropdown menu showing 'Success (Boolean)'.
- Select compute cluster:** A dropdown menu showing 'Default-compute'. Below the dropdown are links for 'Create a new compute' and 'Refresh compute'.

At the bottom of the main area are 'Back' and 'Next' buttons.

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

The screenshot shows the 'Select task type' step of the 'Create a new Automated ML run' wizard. The sidebar on the left is the same as in the previous step, with 'Configure run' checked and 'Select task and settings' active. The main area is titled 'Select task type' with a subtitle: 'Select the machine learning task type for the experiment. To fine tune the experiment, choose additional configuration or featurization settings.'

The task type selection options are:

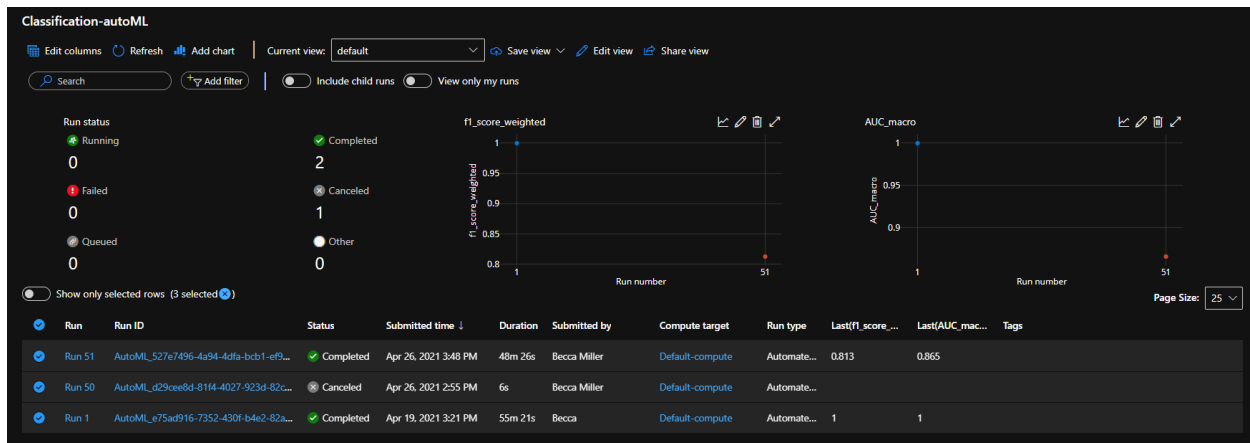
- Classification:** To predict one of several categories in the target column, yes/no, blue, red, green. This option is selected, indicated by a green checkmark.
- Enable deep learning:** A checkbox that is currently unchecked.
- Regression:** To predict continuous numeric values.
- Time series forecasting:** To predict values based on time.

At the bottom of the main area are links for 'View additional configuration settings' and 'View featurization settings'. At the bottom of the entire wizard are 'Back', 'Finish', and 'Cancel' buttons.

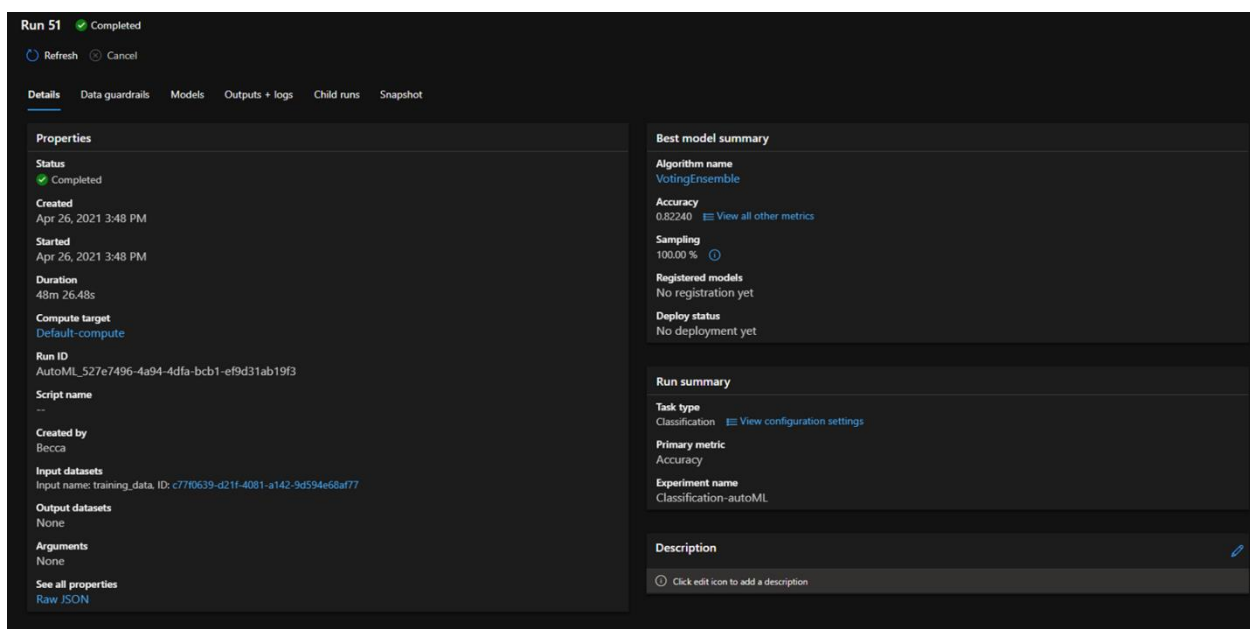
Click “Finish” to begin the run.

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:



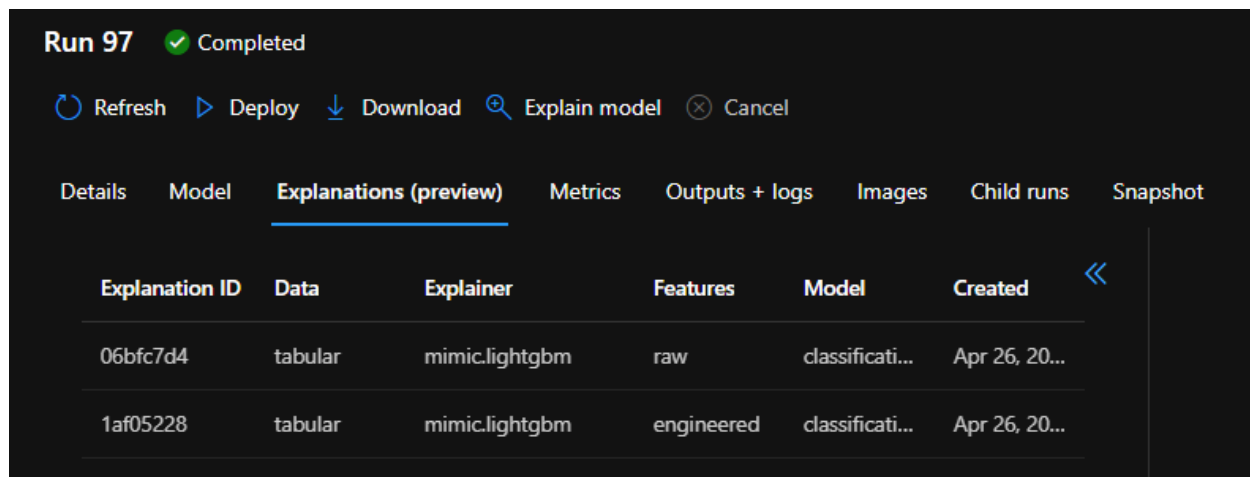
In our example, we can see that the VotingEnsemble was the best performing model with an accuracy of 0.82. By clicking “View all other metrics” under the best model summary, we can see all the metrics for this model:

Accuracy	0.82240
AUC macro	0.86546
AUC micro	0.91513
AUC weighted	0.86546
Average precision score macro	0.77718
Average precision score micro	0.91913
Average precision score weighted	0.87971
Balanced accuracy	0.69537
F1 score macro	0.71339
F1 score micro	0.82240
F1 score weighted	0.81255
Log loss	0.37302
Matthews correlation	0.43670
Norm macro recall	0.39073
Precision score macro	0.74407
Precision score micro	0.82240

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

Refresh Cancel					
Details Data guardrails Models Outputs + logs Child runs Snapshot					
Deploy Download Explain model Search to filter items...					
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.



Explanation ID	Data	Explainer	Features	Model	Created
06bfc7d4	tabular	mimiv.lightgbm	raw	classificati...	Apr 26, 20...
1af05228	tabular	mimiv.lightgbm	engineered	classificati...	Apr 26, 20...

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.

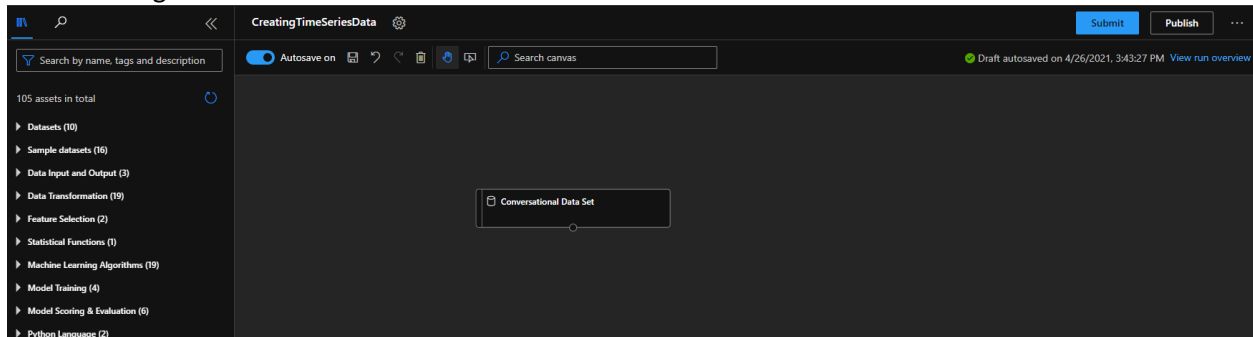
Preparing Data for Time Series Model

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

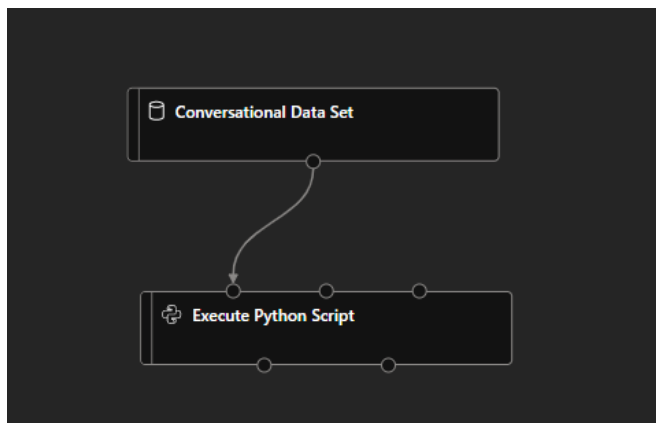
Date	Conversation Count	Conversation Type
2021-01-01	53	IVR
2021-01-01	48	Voice
2021-01-01	55	Chat
2021-01-02	42	IVR
2021-01-02	45	Voice
2021-01-02	53	Chat
...

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:

Execute Python Script result visualization

Result dataset2




Result dataset

Rows ?

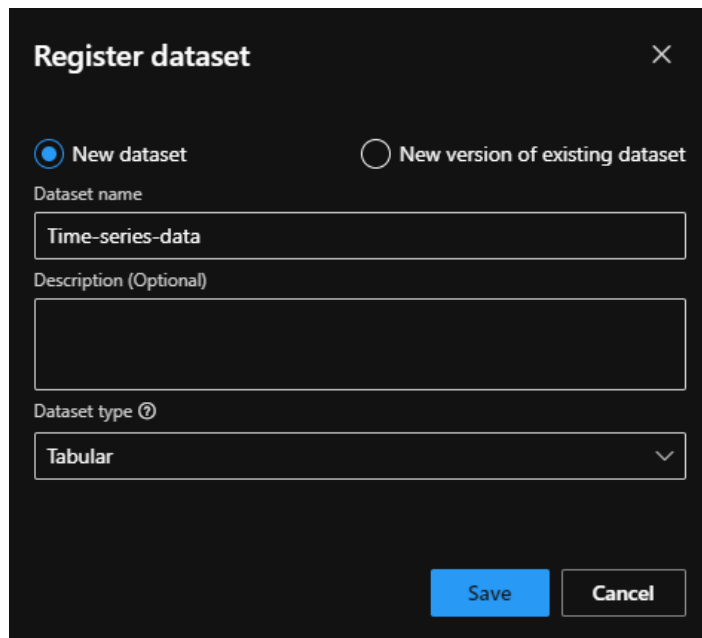
180

Columns ?

3

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
2021-01-14 00:00:00	IVR	91

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:

A dark-themed dialog box titled "Register dataset" with a close button (X) in the top right corner. It contains two radio buttons: "New dataset" (selected) and "New version of existing dataset". Below the radio buttons are three input fields: "Dataset name" with the text "Time-series-data", "Description (Optional)" which is empty, and "Dataset type" with a dropdown menu showing "Tabular". At the bottom right are two buttons: "Save" (highlighted in blue) and "Cancel".

Register dataset ✕

☒ New dataset ☐ New version of existing dataset

Dataset name
Time-series-data

Description (Optional)

Dataset type ⓘ
Tabular ▼

Save Cancel

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.

Create a new Automated ML run

- Select dataset
- Configure run**
- Select task and settings

Configure run

Select from existing experiments or create a new experiment, then select the target column and training compute. [Learn more on how to configure the experiment.](#)

Dataset
Time-series-data ([View dataset](#))

Experiment name *

☐ Select existing ☒ Create new

New experiment name [🔗](#)

Time-series-autoML

Target column * ⓘ

Conversation Count (Integer) ▼

Select compute cluster * ⓘ

Default-compute ▼

[Create a new compute](#) [Refresh compute](#)

[Back](#) [Next](#)

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:

Create a new Automated ML run

- Select dataset
- Configure run
- Select task and settings**

Select task type

Select the machine learning task type for the experiment. To fine tune the experiment, choose additional configuration or featurization settings.

Classification
To predict one of several categories in the target column. yes/no, blue, red, green.

Regression
To predict continuous numeric values

Time series forecasting ✔
To predict values based on time

The time series forecasting method requires some additional information.

Time column ⓘ

Date of Conversation (Date) ▼

Time series identifier(s) ⓘ

Conversation Type ✕

Select column(s)...

Frequency ⓘ

☒ Autodetect

Forecast horizon ⓘ

☒ Autodetect

☐ Enable deep learning ⓘ

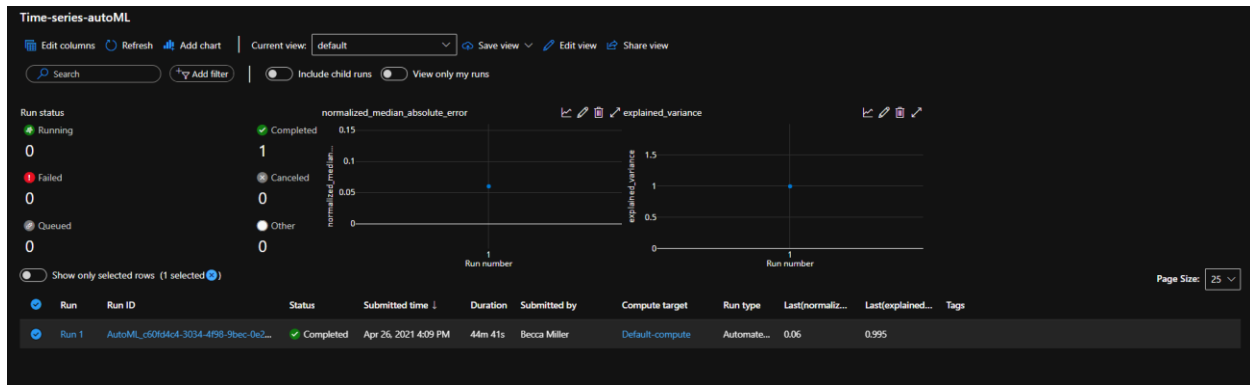
[View additional configuration settings](#) [View featurization settings](#)

[Back](#) [Finish](#) [Cancel](#)

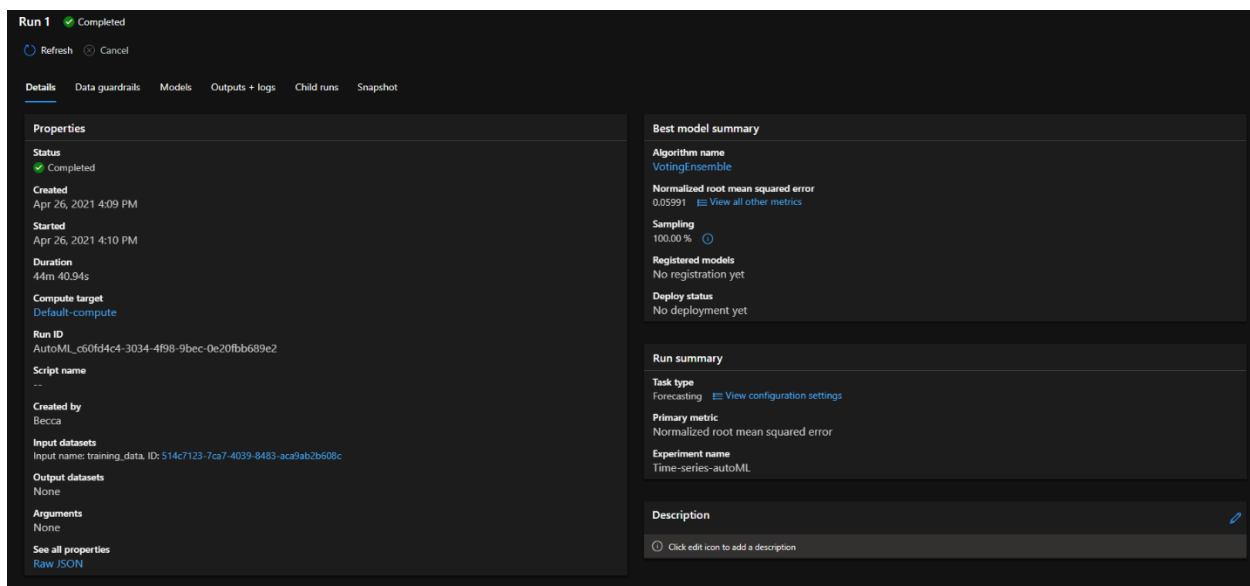
Click “Finish” to begin the run.

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:

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					
<a>Refresh <a>Cancel					
<a>Details <a>Data guardrails <a>Models <a>Outputs + logs <a>Child runs <a>Snapshot					
<a>Deploy <a>Download <a>Explain model					
<a>Search to filter items...					
Algorithm name	Explained	Normalized root mean s... ↑	Sampling <a>	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.