



# Microsoft Machine Learning in a Day Workshop

Student Guide

September 2018

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links may be provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

© 2018 Microsoft Corporation. All rights reserved.

Microsoft and the trademarks listed at <https://www.microsoft.com/en-us/legal/intellectualproperty/Trademarks/Usage/General.aspx> are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners.

# Contents

<b>Machine Learning in a Day student guide.....</b>	<b>1</b>
Abstract and learning objectives.....	1
Architecture .....	2
Section 1: Set up a Data Science Virtual Machine (DSVM) on Azure Portal.....	3
Section 2: Download Telco dataset and create Blob Storage Account and Container access .....	8
Section 3: Upload CSV file to Blob Storage Container with Azure Data Factory V2.....	11
Section 4: Build a clustering unsupervised model in Azure Machine Learning Studio .....	32
Section 5: Set up a Spark Cluster on Databricks and connect to Azure Blob Storage Account.....	43
Section 6: Implement Feature Engineering Techniques to Enhance data for Machine Learning .....	50
Section 7: Apply Classification Supervised Model with Logistic Regression on Azure Databricks .....	58
Section 8: Export predicted DataFrame from Spark on Azure Databricks to Power BI for visualizations.....	60



# Machine Learning in a Day student guide

## Abstract and learning objectives

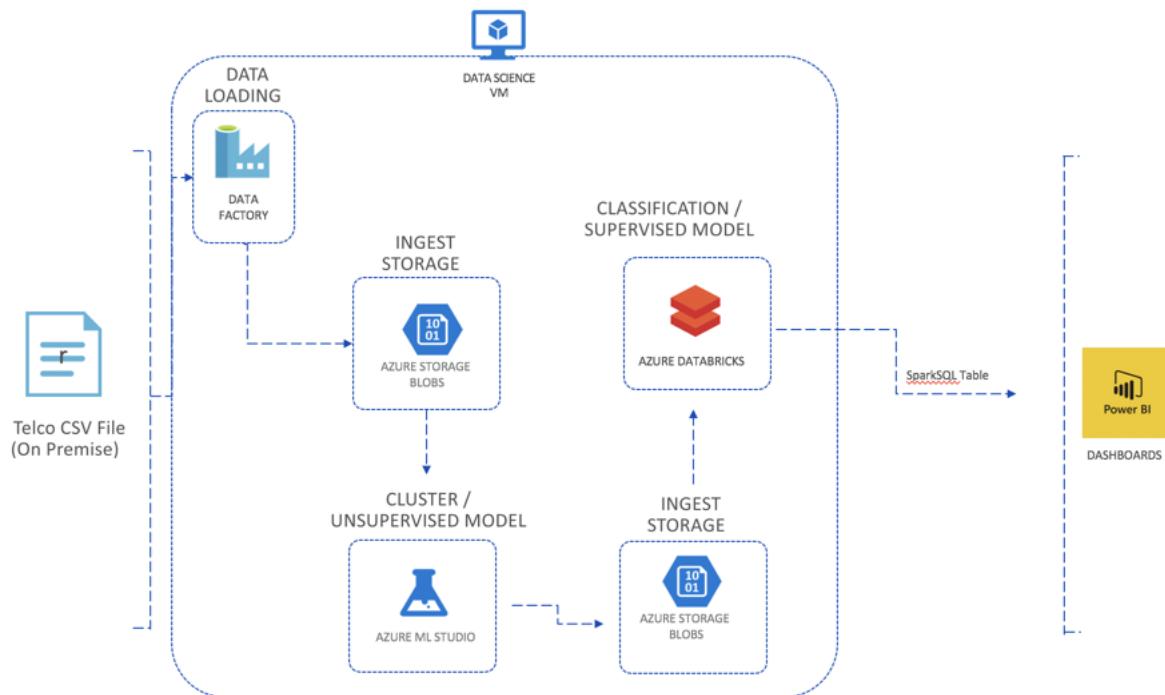
In this workshop, you will complete will build a Machine Learning model that will be used to predict whether customers remain with business or move onto other services based on their behavior, demographics, and spending. You will leverage several Microsoft products and services on Microsoft Azure to deliver the Machine Learning Model.

The dataset that will be used in the workshop contains the following information

1. Customers who left within the last month – the column is called **Churn**
2. Services that each customer has signed up for
  - a. Phone
  - b. Multiple lines
  - c. Internet
  - d. Online security
  - e. Online backup
  - f. Device protection
  - g. Tech support
  - h. Streaming TV and movies
3. Customer account information
  - a. How long they've been a customer
  - b. Contract
  - c. Payment method
  - d. Paperless billing
  - e. Monthly charges
  - f. Total charges
4. Demographic information about customers
  - a. Gender
  - b. Age range
  - c. If they have partners and dependents

## Architecture

### MACHINE LEARNING IN A DAY WORKSHOP



## Section 1: Set up a Data Science Virtual Machine (DSVM) on Azure Portal

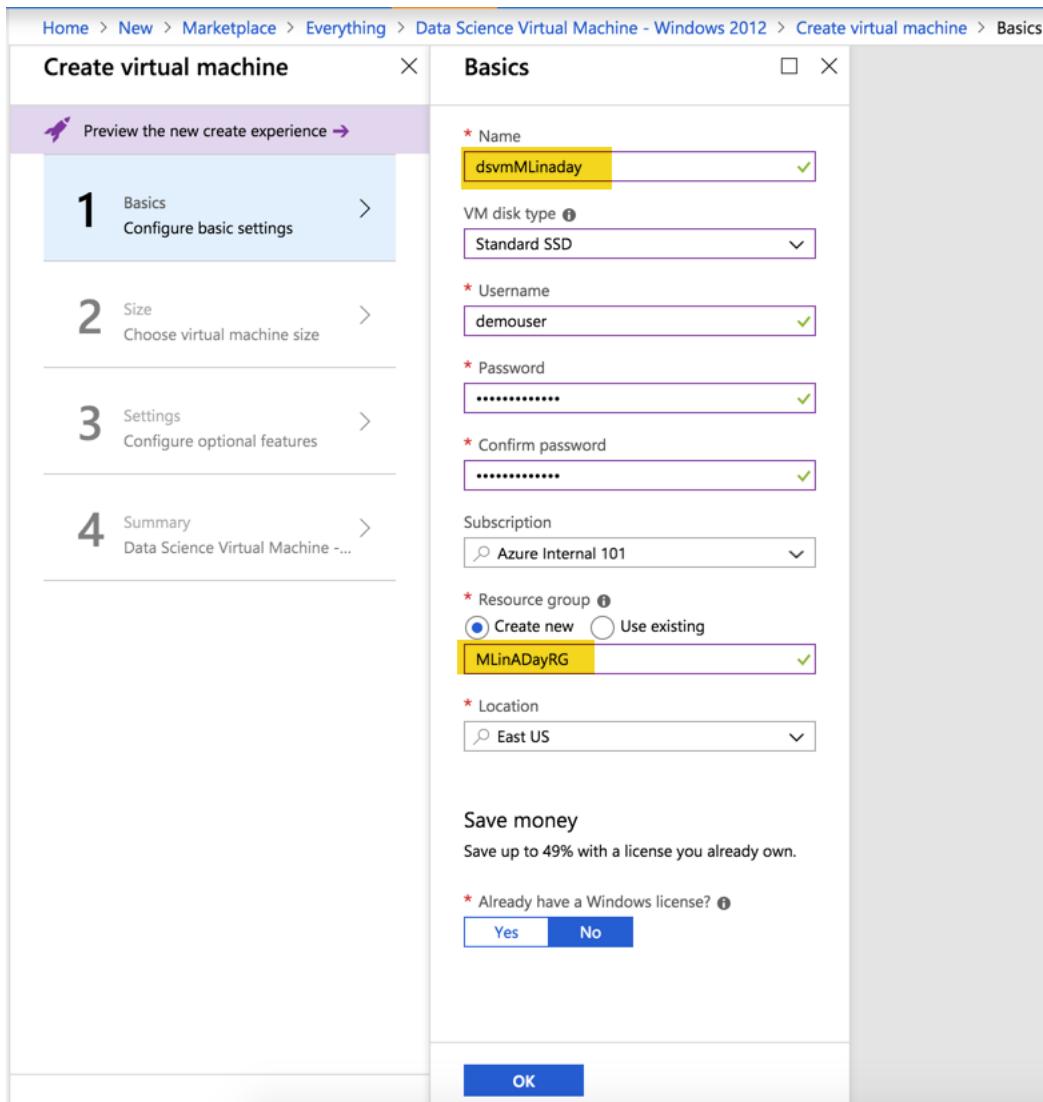
1. The use of this documentation requires the subscription of a Microsoft Azure account either from a personal email account or a work email account.
2. Visit [www.portal.azure.com](http://www.portal.azure.com) and click on **Create a resource** on the left-hand side and type in **Data Science Virtual Machine – Windows 2012** and select the **Create** button as seen in the following screenshot:

The screenshot shows the Azure portal interface for creating a new virtual machine. On the left, the sidebar has 'Create a resource' highlighted. The main area shows the 'Compute' category with a search bar containing 'data science virtual machine'. Below it is a table of results:

NAME	PUBLISHER	CATEGORY
Data Science Virtual Machine - Windows 2012	Microsoft	Virtual Machine Images
(CSP)Data Science Virtual Machine - Windows 2012	Microsoft	Virtual Machine Images
Data Science Virtual Machine for Linux (Ubuntu)	Microsoft	Virtual Machine Images
Data Science Virtual Machine - Windows 2016	Microsoft	Virtual Machine Images
Data Science Virtual Machine for Linux (CentOS)	Microsoft	Virtual Machine Images
Data Science Virtual Machine for Linux (Ubuntu)	Microsoft	Virtual Machine Images
Data Science Virtual Machine - Windows 2016	Microsoft	Virtual Machine Images
Geo AI Data Science VM with ArcGIS	Microsoft	Solution Templates
Signal Sciences -BYOL	Signal Sciences	Virtual Machine Images
Deep Learning Virtual Machine	Microsoft	Solution Templates
Microsoft Machine Learning Server 9.3.0 on Ubuntu 16.04	Microsoft	Recommended
Microsoft Machine Learning Server 9.3.0 on Red Hat Enterprise Linux 7.2	Microsoft	Recommended
Microsoft Machine Learning Server 9.3.0 on Windows Server 2016	Microsoft	Recommended
Microsoft Machine Learning Server 9.3.0 on CentOS Linux 7.2	Microsoft	Recommended

To the right, a detailed description of the selected 'Data Science Virtual Machine - Windows 2012' is shown, along with a 'Save for later' button. At the bottom, there's a 'Select a deployment model' dropdown set to 'Resource Manager' and a large green 'Create' button.

3. Assign the following basic configuration settings for your Data Science Virtual Machine and the select the **OK** button as seen in the following screenshot:



Please note that you can keep your Names and passwords different from this document as long as you remember what you used for future use. In this section, we will assign a username of **demouser** and a password of **#MLinaDay2018** (Please keep track of both the Username and Password as you will need both of them several times as we go through future sections of this workshop).

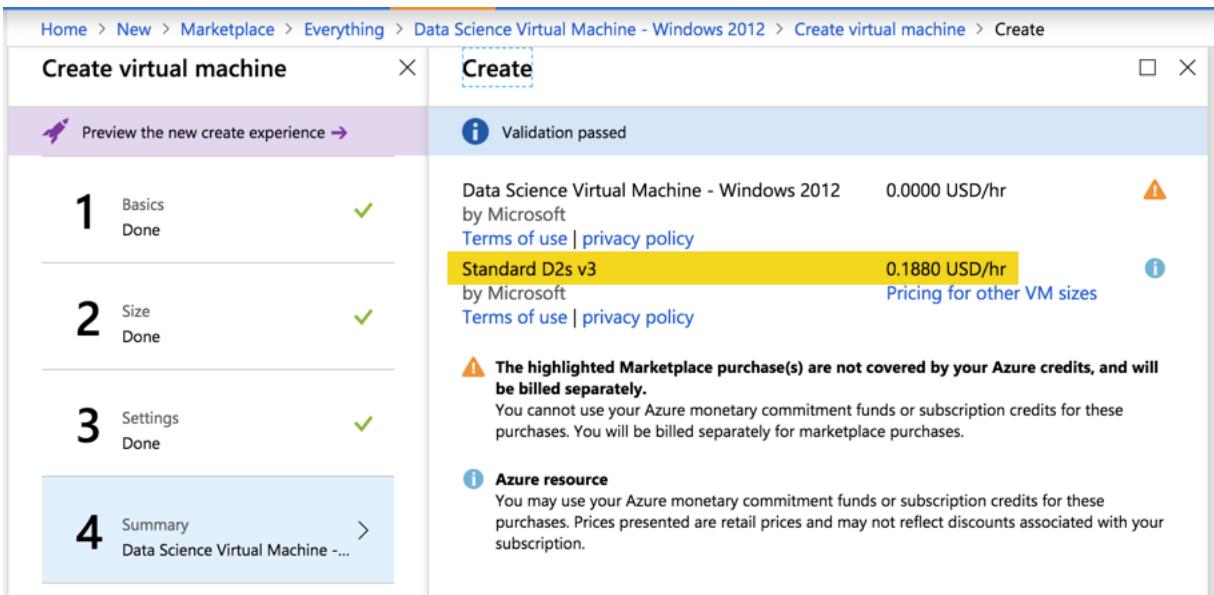
4. Choose the following size VM (**D2S\_v3**) for the purposes of this workshop as seen in the following screenshot:

The screenshot shows the 'Choose a size' step in the Azure portal. On the left, a navigation pane lists steps 1 through 4: Basics (Done), Size (Choose virtual machine size), Settings (Configure optional features), and Summary (Data Science Virtual Machine - ...). Step 2 is currently selected. On the right, a table titled 'Available' lists various virtual machine sizes. The columns include SKU, Compute type, Disk type, vCPUs, RAM, Data Disks, Max IOPS, Local SSD, Premium SSD, Additional Options, and USD/MON. The D2s\_v3 row is highlighted with a yellow border. A note at the bottom states: 'Prices presented are estimates in your local currency that include Azure infrastructure applicable software costs, as well as any discounts for the subscription and location. Recommended sizes are determined by the publisher of the selected image based on hardware and software requirements.' A 'Select' button is at the bottom.

5. For settings, the only configuration that is truly needed is to enable on **auto-shutdown** every evening at **7pm EST** (your manager will thank you later on !!!!) and then select **OK** as seen in the following screenshot:

The screenshot shows the 'Settings' step in the Azure portal. On the left, a navigation pane lists steps 1 through 4: Basics (Done), Size (Done), Settings (Configure optional features), and Summary (Data Science Virtual Machine - ...). Step 3 is currently selected. On the right, the 'Settings' tab is open, showing configuration options. The 'Auto-shutdown' section is highlighted with a yellow border. It includes 'Enable auto-shutdown' (set to 'On'), 'Shutdown time' (set to '7:00:00 PM'), 'Time zone' (set to '(UTC-05:00) Eastern Time (US & Canada)'), and 'Notification before shutdown' (set to 'Off'). Other sections like 'Network Security Group' and 'Extensions' are also visible.

6. Click **OK** on the summary section and confirm everything that was selected is accurate and let the provisioning process of the DSVM begin by clicking on **Create** as seen in the following screenshot:



- Once the provisioning is complete, you can go to your resource and view your DSVM overview and obtain your **Public IP address** as seen in the following screenshot:

Home > Resource groups > MLinADayRG > dsvmMLinADay

**Virtual machine**

**Connect**

**Overview**

Resource group (change)  
MLinADayRG

Status  
Creating

Location  
East US

Subscription (change)  
Azure Internal 101

Subscription ID

Tags (change)  
Click here to add tags

Computer name  
delete

Operating system  
Windows

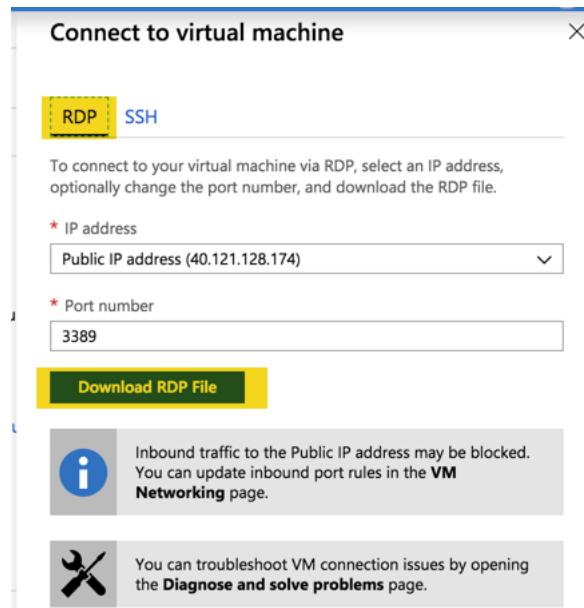
Size  
Standard D2s v3 (2 vcpus, 8 GB memory)

Public IP address  
**40.121.128.174**

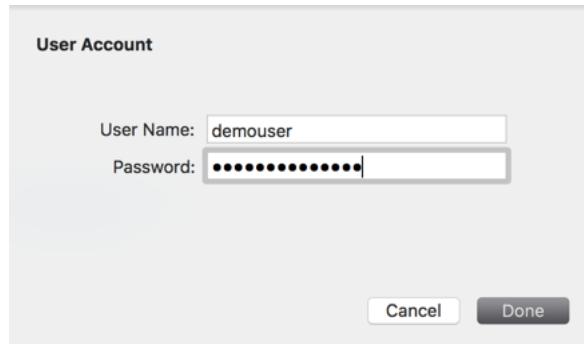
Virtual network/subnet  
MLinADayRG-vnet/default

DNS name  
Configure

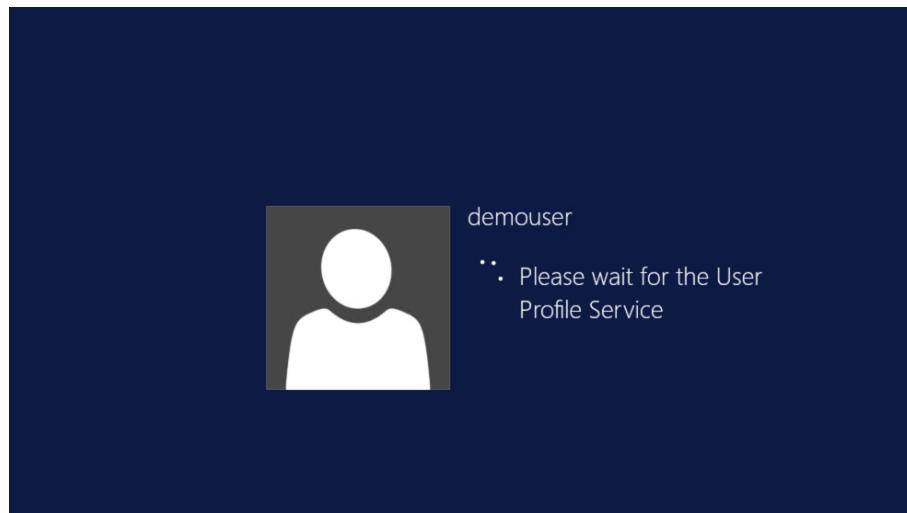
- Make sure that your DSVM is currently running and click on the **Connect** button to enter your virtual machine using **RDP** as seen in the following screenshot:



9. Enter your login credentials that were created back in Step # 3 as seen in the following screenshot:



10. It may take a few extra moments while your profile is being created and you will see the following window:

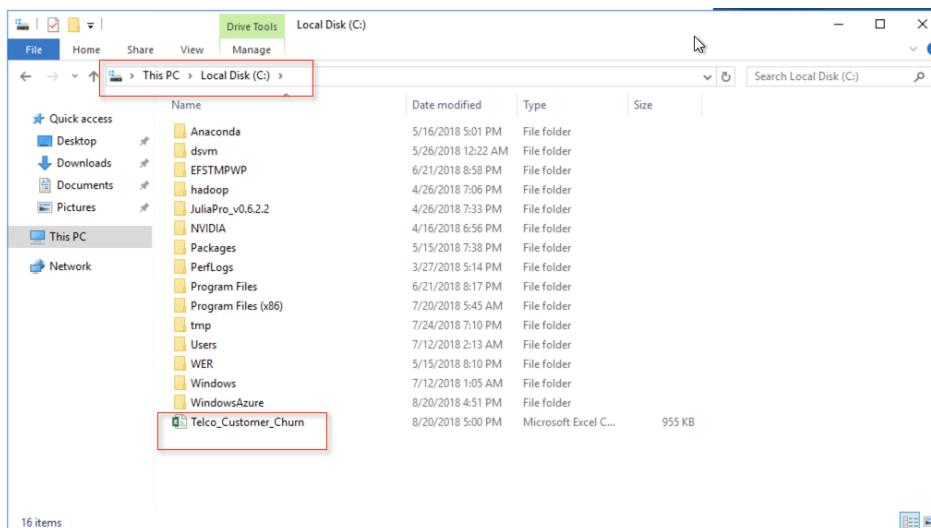


11. Once you log in using your credentials using the RDP approach, you should see the following remote server for the data science virtual machine as seen in the following screenshot:



## Section 2: Download Telco dataset and create Blob Storage Account and Container access

- The Telco Churn Dataset ([TELCO\\_CUSTOMER\\_CHURN.CSV](https://blobmlinadayahmed.blob.core.windows.net/datasources/Telco_Customer_Churn.csv)) can be downloaded from the following link:  
[https://blobmlinadayahmed.blob.core.windows.net/datasources/Telco\\_Customer\\_Churn.csv](https://blobmlinadayahmed.blob.core.windows.net/datasources/Telco_Customer_Churn.csv)
- For now, the dataset can be saved in the C:\ drive as seen in the following screenshot:



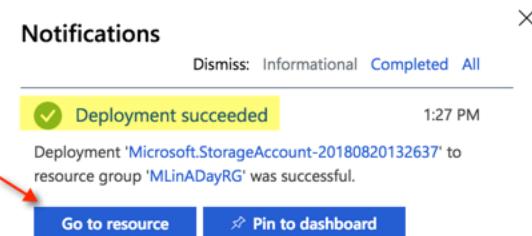
3. Next we will return to our Azure portal (<https://portal.azure.com>) and create a Blob storage account to house our data. We can do this by selecting **Create A Resource**, typing **Blob** in the search, and then selecting **Storage Account – blob, file, table, queue** as seen in the following screenshot:

The screenshot shows the Azure Marketplace search interface. On the left, there's a sidebar with a 'Create a resource' button highlighted with a red box. Below it are sections for 'All services', 'FAVORITES' (which includes 'Dashboard', 'All resources', 'Resource groups', 'App Services', 'Function Apps', and 'Web'), and a 'Marketplace' section with 'My Saved List' and 'Everything'. In the main area, the search bar contains 'blob' and has a red arrow pointing to it. The results list shows a single item: 'Storage account - blob, file, table, queue', which is also highlighted with a red box.

4. Once you create a **Blob** account, you can name and customize the properties of the account as the following, for optimal performance in this workshop (Please note than any existing resource that is created from this point forward in the workshop should ideally use the same existing **Resource Group** that we created in Section 1):

The screenshot shows the 'Create storage account' dialog box. It includes fields for Name (set to 'blobmlinaday'), Deployment model (set to 'Resource manager'), Account kind (set to 'Storage (general purpose v1)'), Location (set to 'East US'), Replication (set to 'Locally-redundant storage (LRS)'), Performance (set to 'Standard'), Secure transfer required (set to 'Enabled'), Subscription (set to 'Azure Internal 101'), Resource group (set to 'MLinADayRG'), and Virtual networks (set to 'Disabled'). At the bottom are 'Create' and 'Automation options' buttons.

5. Once the Blob account has been successfully deployed, we can go to the storage account by clicking on **Go to Resource**, as seen in the following screenshot:



6. Once inside the resource, select **Blobs** under the **Services** section as seen in the following screenshot:

The screenshot shows the 'blobmlinaday' storage account overview page. On the left, there's a sidebar with links like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Events, and Storage Explorer (preview). The main area shows details such as Resource group (MLinADayRG), Status (Primary: Available), Location (East US), Subscription (Azure Internal 101), Subscription ID (ba3edd26-e2b1-4cc5-a19e-5bd21d7e9f5d), and Tags (Click here to add tags). Below this is a 'Services' section with a 'Blobs' item, which is highlighted with a red arrow. The 'Blobs' item description is: 'REST-based object storage for unstructured data'.

7. We should now be in the Containers section; however, if this is the first time we are entering the Container, we should see a message that says **You don't have any containers yet. Click '+ Container' to get started.**  
 8. Go ahead and create a new Container called **Churn** and set **Public access level** to **Container (anonymous read access for containers and blobs)**, as seen in the following screenshot:

The screenshot shows the 'blobmlinaday - Blobs' storage account page. On the left, there's a sidebar with links like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Events, and Storage Explorer (preview). The main area shows a 'New container' dialog. In the dialog, the 'Name' field is filled with 'churn'. Under 'Public access level', a dropdown menu is open, showing 'Container (anonymous read access for containers and blobs)' as the selected option. At the bottom of the dialog are 'OK' and 'Cancel' buttons. A red box highlights the 'Public access level' dropdown.

9. The only thing that we will need right now is an **Access Keys** that will be used in future sections of this workshop. We can copy **key1** and paste it somewhere handy in a notepad editor as seen in the following screenshot:

**blobmlinaday - Access keys**

Storage account

Search (Ctrl+ /)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Storage Explorer (preview)

Settings

Access keys

CORS

Configuration

Encryption

Shared access signature

Firewalls and virtual networks

Properties

Locks

Automation script

Use access keys to authenticate your applications when making requests to this Azure storage account. Store your access keys securely - for example, using Azure Key Vault - and don't share them. We recommend regenerating your access keys regularly. You are provided two access keys so that you can maintain connections using one key while regenerating the other.

When you regenerate your access keys, you must update any Azure resources and applications that access this storage account to use the new keys. This action will not interrupt access to disks from your virtual machines. [Learn more](#)

Storage account name: blobmlinaday

**key1**

Key: 2Le22d38roB6lFqAeds/dFXRfc6VoKUDBXqcsjhzRv8r3YDaB1xI7FdmyzLcaH04NAFrL+OQmXsNlo4NzICg==

**key2**

Key: e3rNW07ji+s37Xy7/v84m8mCYSYDQSH+p9lxUmnfpgrwta02/YyaOEpp9JhYcReHuqZLv0s/qYGbm6CPxaE8yA==

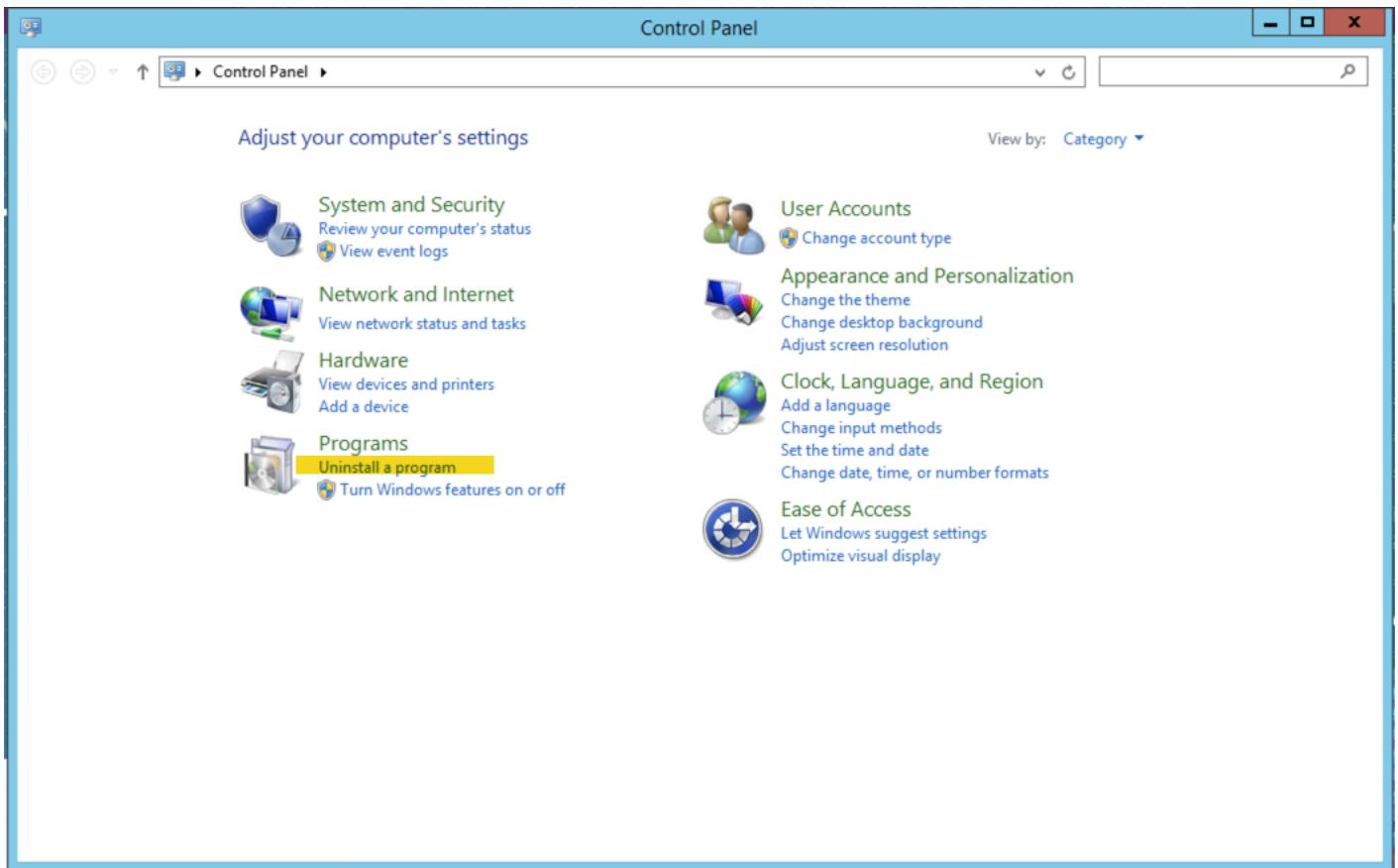
Connection string: DefaultEndpointsProtocol=https;AccountName=blobmlinaday;AccountKey=2Le22d38roB6lFqAeds/dFXRfc6VoKUDBXqcsjhzRv8r3YDaB1xI7FdmyzLcaH04NAFrL+OQmXsNlo4NzICg==

Connection string: DefaultEndpointsProtocol=https;AccountName=blobmlinaday;AccountKey=e3rNW07ji+s37Xy7/v84m8mCYSYDQSH+p9lxUmnfpgrwta02/YyaOE...

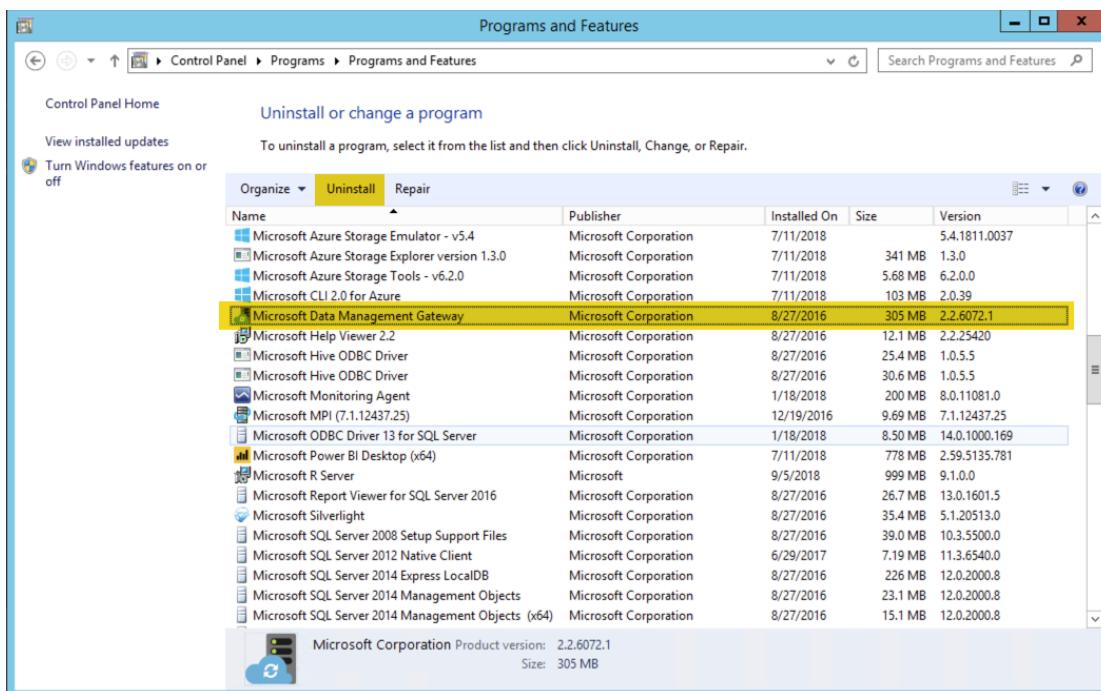
- Once the container has been successfully created, we can move onto the next section of adding data to the container using an orchestration/ETL tool called Azure Data Factory.

## Section 3: Upload CSV file to Blob Storage Container with Azure Data Factory V2.

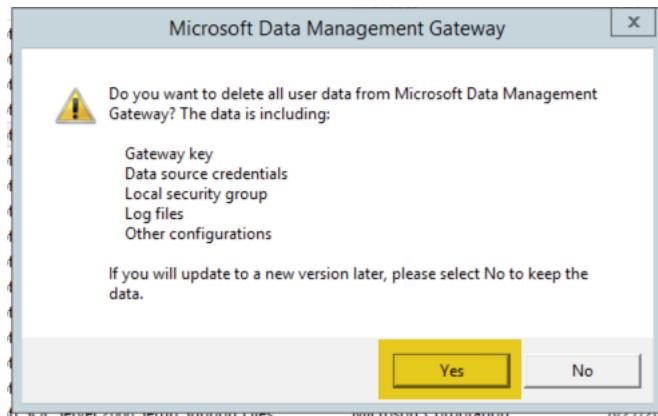
- You have now completed the provisioning of a data science virtual machine on Windows Server 2012. One note to factor for this virtual machine is we will uninstall a product that is outdated and download later on in this section. First, we will go to the **Control Panel** in the DSVM and select to **Uninstall A Program** as seen in the following screenshot:



- Next we will go ahead and **Uninstall Microsoft Data Management Gateway** as seen in the following screenshot:



- Select **Yes** to get rid of all data as seen in the following screenshot:



4. As we move forward in this section, we will require accessing the Azure Portal (<https://portal.azure.com>) inside of a browser within the data science virtual machine to set up **Azure Data Factory V2**.
5. Once again, as we did with the Blob Storage as well as with the Data Science Virtual Machine, we are ready to create a new resource. This time we will build an orchestrator that will create a job to move our CSV file from on-premise on our Virtual Machine to our storage in Blob on Azure. The next step is to create a resource for **Azure Data Factory** as seen in the following screenshot:

NAME	PUBLISHER	CATEGORY
Azure Data Factory Analytics (Preview)	Microsoft	Management Tools
OutSystems on Microsoft Azure	OutSystems	Compute
Data Factory	Microsoft	Analytics

6. Once the resource is created, we will want to specify the following options and then click on **Create**:

**New data factory**

\* Name  ✓

\* Subscription

\* Resource Group  Create new  Use existing  
MLinADayRG

Version

\* Location

**Create**   [Automation options](#)

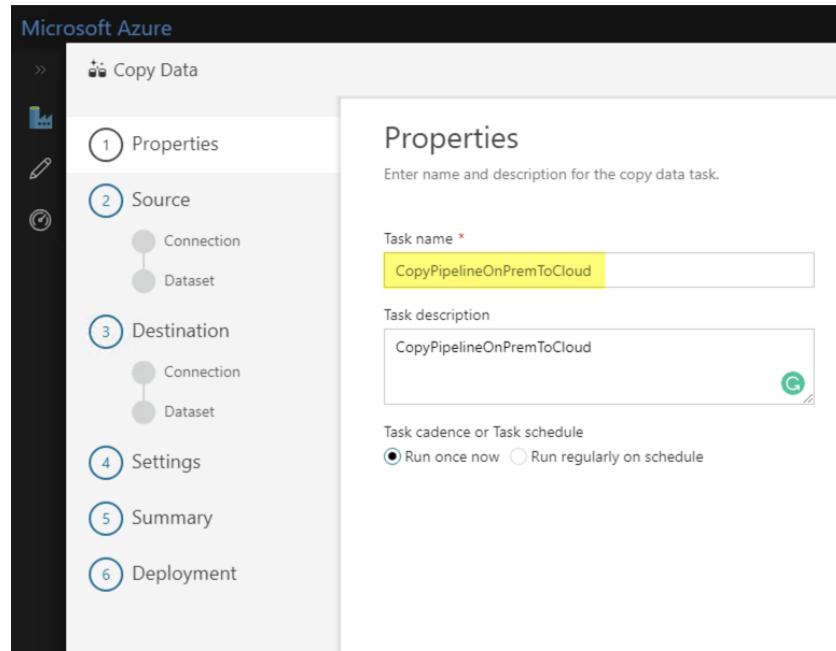
7. Once the resource is completed, we can go to the resource and click on the icon **Author & Monitor** as seen in the following screenshot:

The screenshot shows the Azure Data Factory landing page for a resource named 'adfmmlinaday'. The left sidebar includes links for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings (Locks), General, and Properties. The main area displays 'Essentials' information: Resource group 'MLinADayRG', Location 'EastUS', Provisioning state 'Succeeded', Getting started 'Quick start', and a 'Quick links' section. The 'Author & Monitor' link is highlighted with a red box.

8. We are now in the landing page for Azure Data Factory. The next step is to click on the second icon, **Copy Data**:

The screenshot shows the Azure Data Factory landing page with the title 'Let's get started'. It features several icons: 'Create pipeline' (blue circle with a valve), 'Copy Data' (yellow circle with two green cylinders), 'Configure SSIS Integration Runtime' (white circle with a blue cylinder), and 'Set up Code Repository' (white circle with a red GitHub icon). Below these are sections for 'Videos' with three video thumbnails: 'Overview of Azure Data Factory', 'Azure Data Factory visual tools now integrated with GitHub', and 'Execute Jars and Python scripts on Azure Databricks using Data Factory'. The 'Copy Data' icon is highlighted with a green box.

9. Our first task is to enter name and description for the copy data task as seen in the following screenshot:



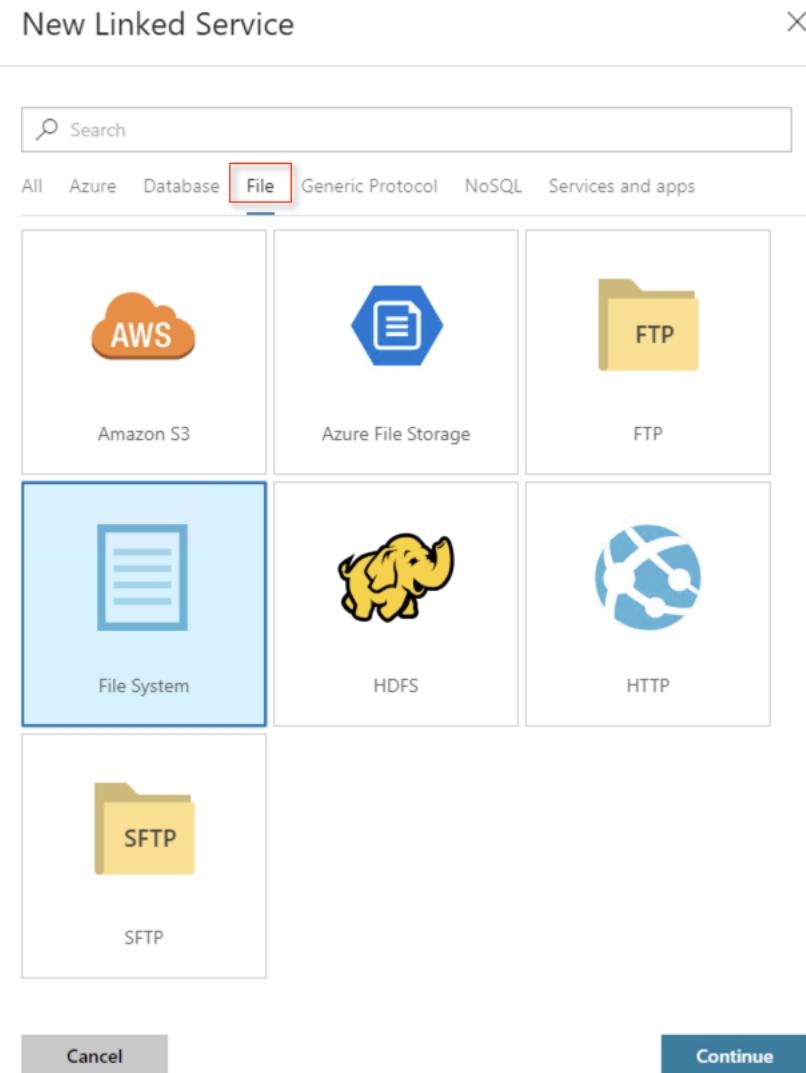
10. We named the task **CopyPipelineOnPremToCloud**. The next step is to specify a data source. Since this source will be on-premise on our virtual machine, we will need to set up + **Create a new connection** as seen in the following screenshot:

### Source data store

Specify the source data store for the copy task. You can use an existing data store connection or specify a new data store.

A screenshot of the 'Source data store' selection screen. At the top, there is a horizontal navigation bar with tabs: All (selected), Azure, Database, File, Generic Protocol, NoSQL, and Services and apps. Below the navigation bar is a search bar with a dropdown menu and a 'Filter by name' input field. To the right of the search bar is a yellow button labeled '+ Create new connection'. A green callout box highlights the '+ Create new connection' button.

11. Select **File** and then **File System** for a **new Linked Service** between on-premise and in Azure. Then select **Continue** as seen in the following screenshot:



12. Specify a name for the Linked Service such as **linkedservicemlinaday** as well as clicking on a **+New integration runtime** as seen in the following screenshot:

← New Linked Service (File System) X

Name \*  
linkedservicemlinaday

Description

Connect via integration runtime \*  
AutoResolveIntegrationRuntime

+ New 

AutoResolveIntegrationRuntime

Password Azure Key Vault

Password \*

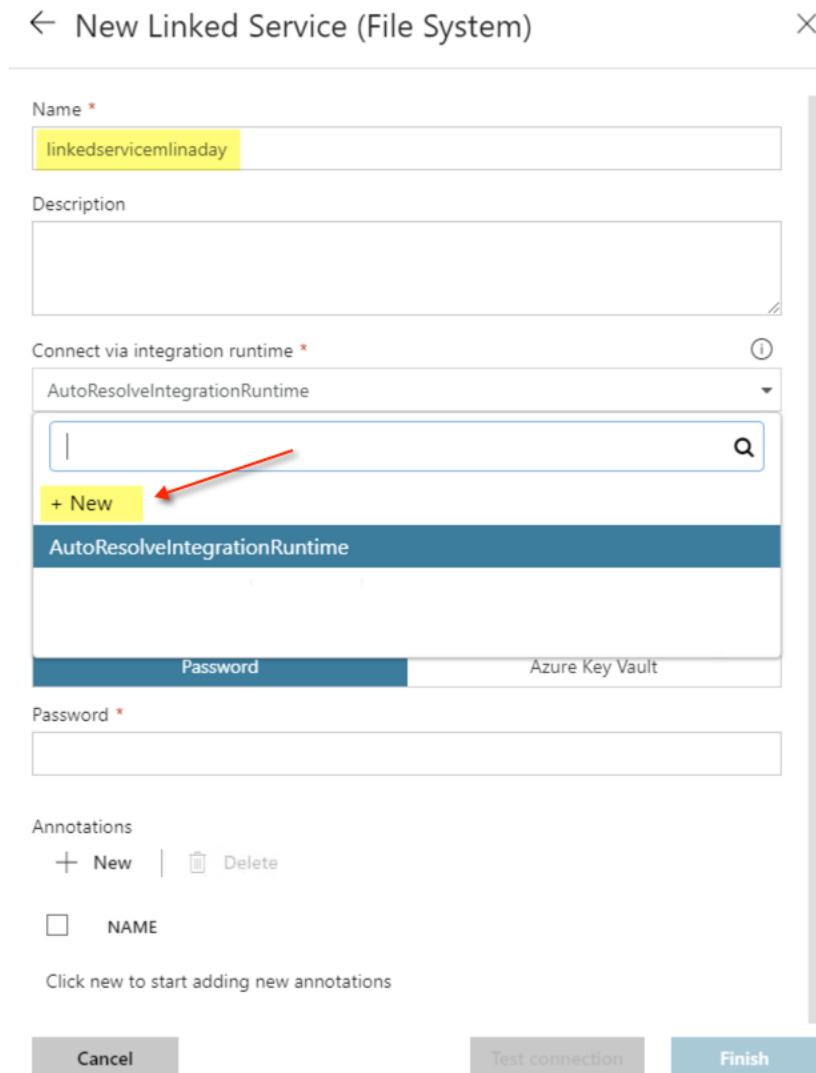
Annotations

+ New | Delete

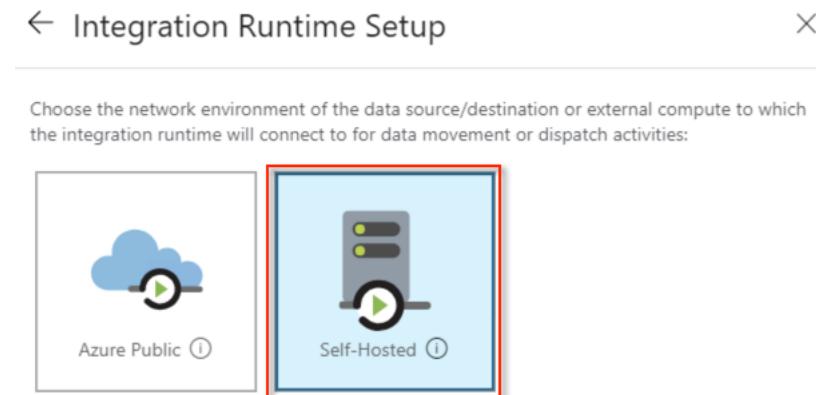
NAME

Click new to start adding new annotations

Cancel Test connection Finish

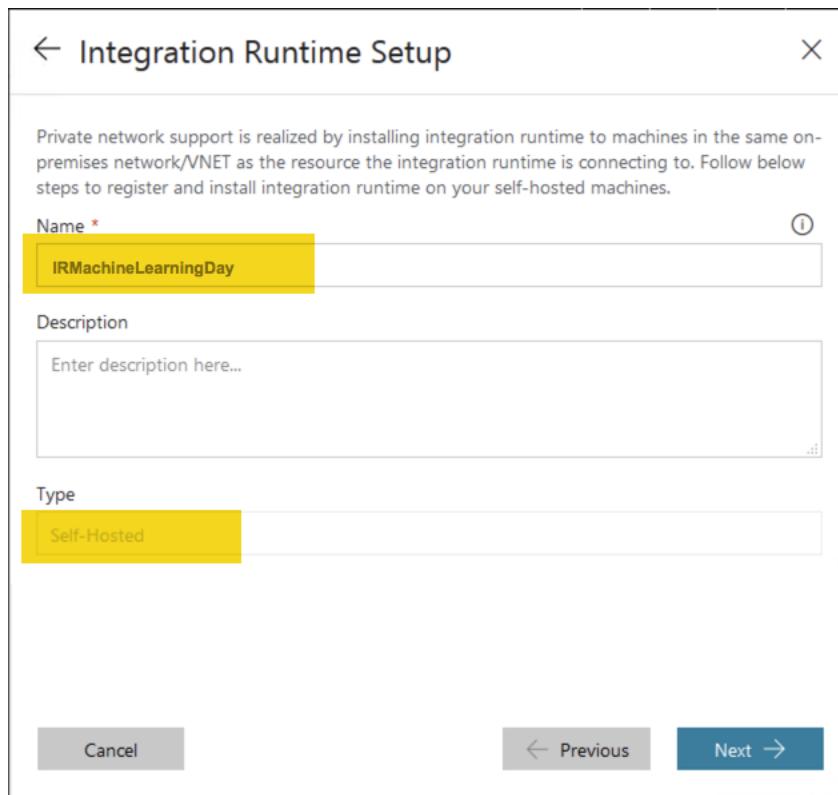


13. Select a **Self-Hosted** setup for the integration runtime as seen in the following screenshot:

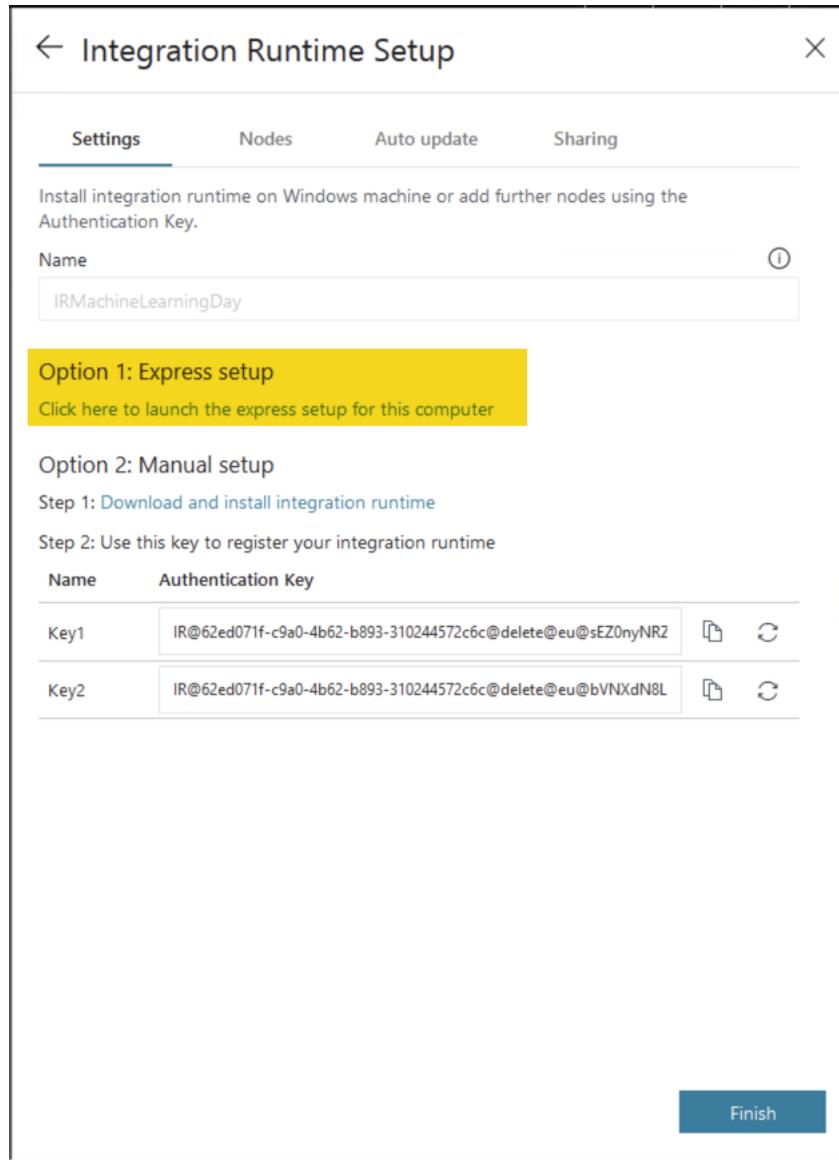


Choose an **Azure Public** environment if you are accessing services with public accessible endpoints. Choose **Self-Hosted** if you are accessing services in a private network like your on-premises environment, or in Virtual Network Environments like Azure Virtual Network. For our purposes, we will go with **Self-Hosted**.

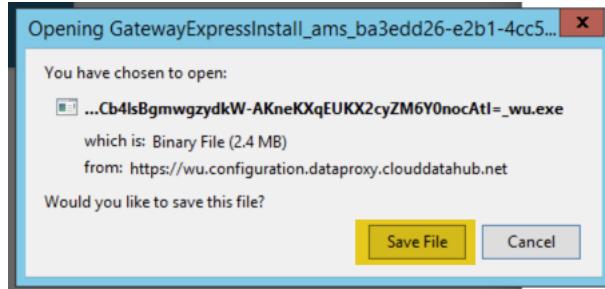
14. Private network support is realized by installing integration runtime to machines in the same on-premises network/VNET as the resource the integration runtime is connecting to. Follow below steps to register and install integration runtime on your self-hosted machines with a name called **IRMachineLearningDay** and then select **Next** as seen in the following screenshot:



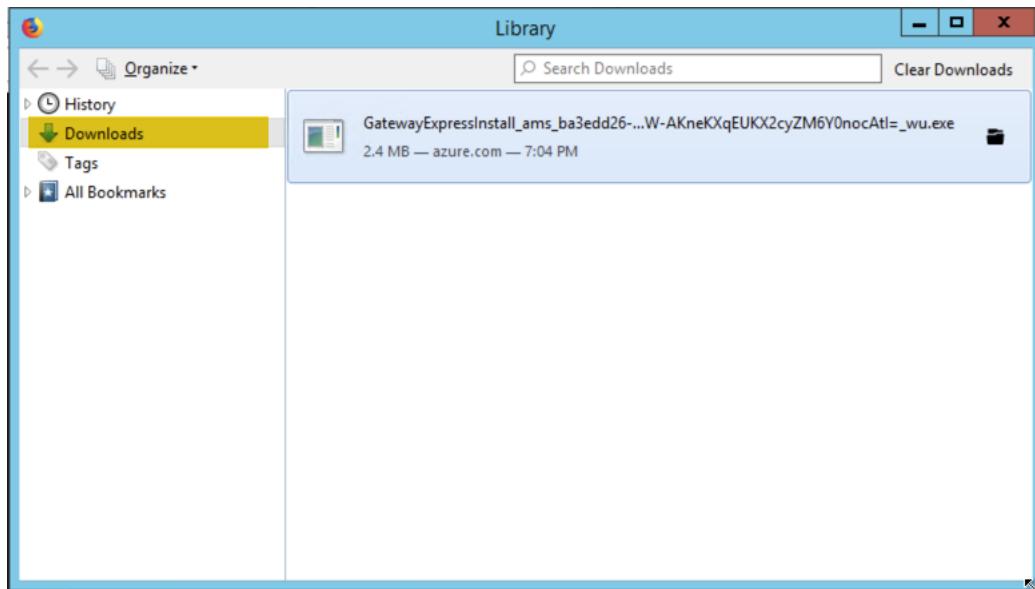
15. Select **Option 1: Express Setup** for the Integration Runtime as seen in the following screenshot:



16. Save the Express Setup File (usually it will default to the Downloads folder) as seen in the following screenshot:



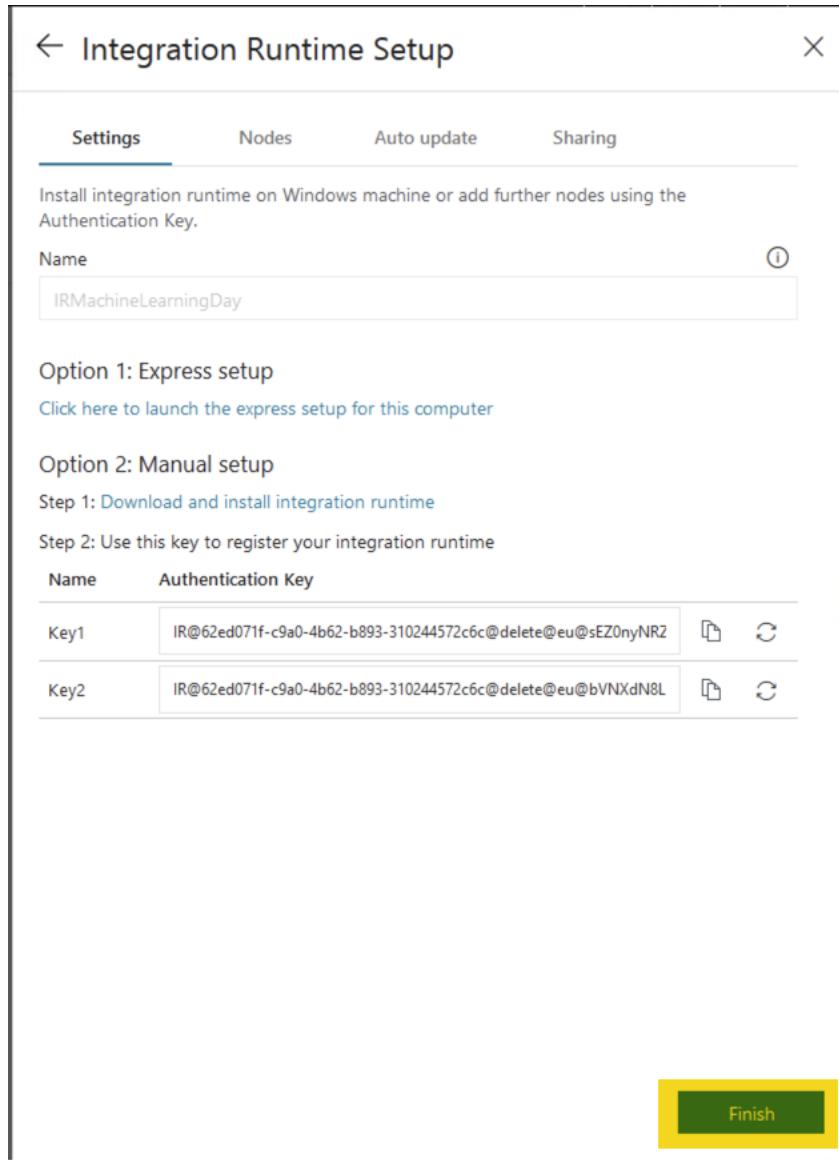
17. Execute the File from the **Downloads** folder as seen in the following screenshot:



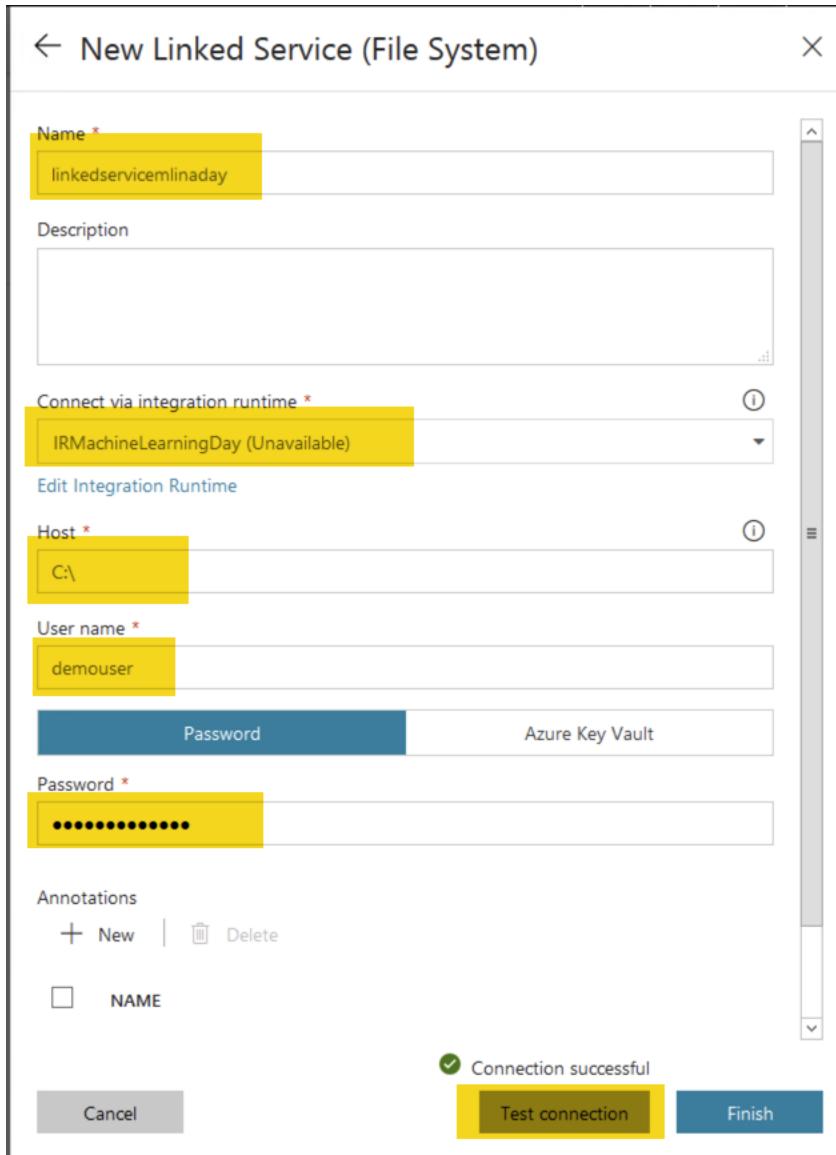
18. The Integration Runtime should begin installation. Once complete, select **Close** as seen in the following screenshot:



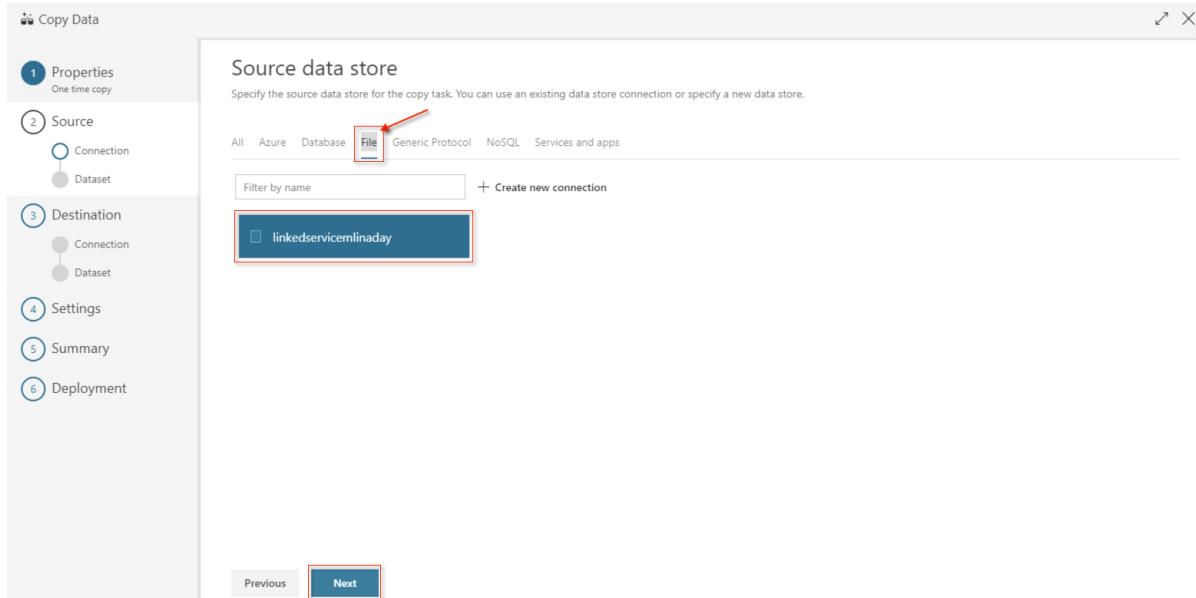
19. Select **Finish** on the **Integration Runtime Setup** as seen in the following screenshot:



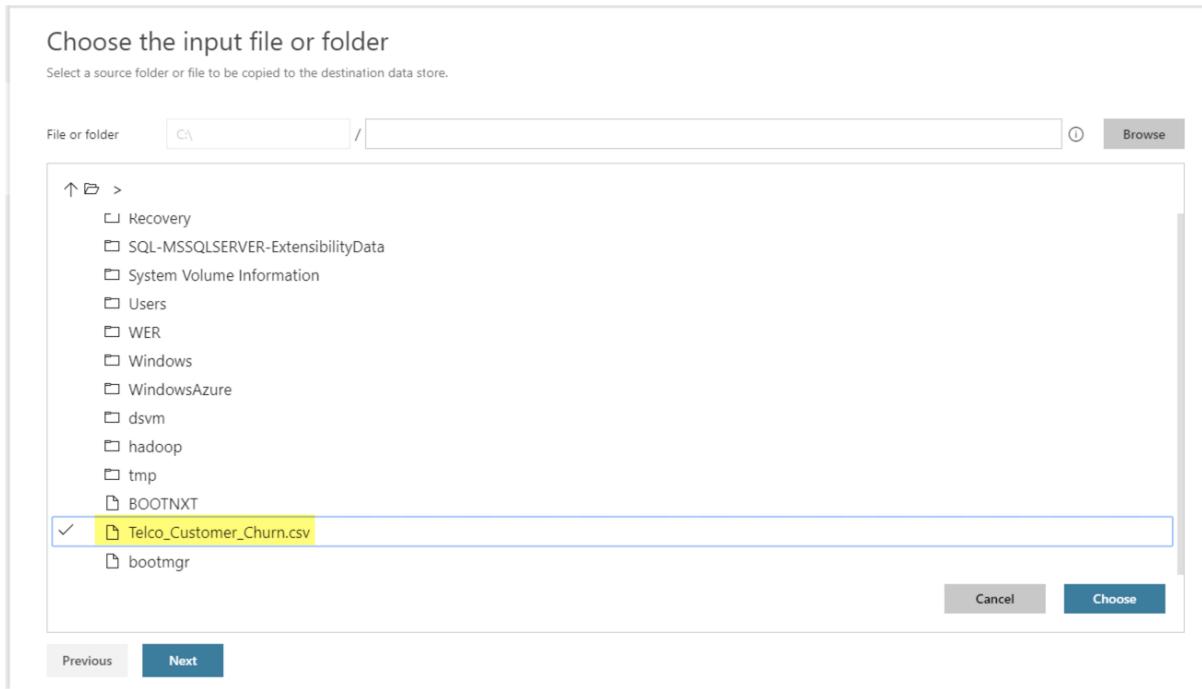
20. We can return now to the **Linked Service (File System)** and enter our user credentials that we initially created for our Data Science Virtual Machine (DSVM) (**User name = demouser & Password = #MLinaDay20!8**) and enter them into our Service as well as the location of the csv file (**C:\**) and select **Test Connection** as seen in the following screenshot:



21. Hopefully we will get a **Connection Successful** message next to a green checkmark. If so, go ahead and select **Finish**.
22. Now that our Connection is set up to our virtual machine with an integration runtime, we can specify the data store type as **File** and select **Next** as seen in the following screenshot:



23. Choose the input file or folder as seen in the following screenshot for the **Telco\_Customer\_Churn.csv**:



24. Specify the **File Format**, **Column Delimiter**, and select **Column Names in the first row** and select **Next** as seen in the following screenshot (we should see a preview of our dataset):

**File format settings**

File format: Text format  
Column delimiter: Comma (,)  
Row delimiter: Carriage Return + Line feed (\r\n)  
Skip line count: 0  
Column names in the first row:

customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV
7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	No	No	No
5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes	No	No

Preview Schema

25. We are now ready to specify our **Destination data store**. We can do so by **+ Create new connection** to our Blob store container as seen in the following screenshot:

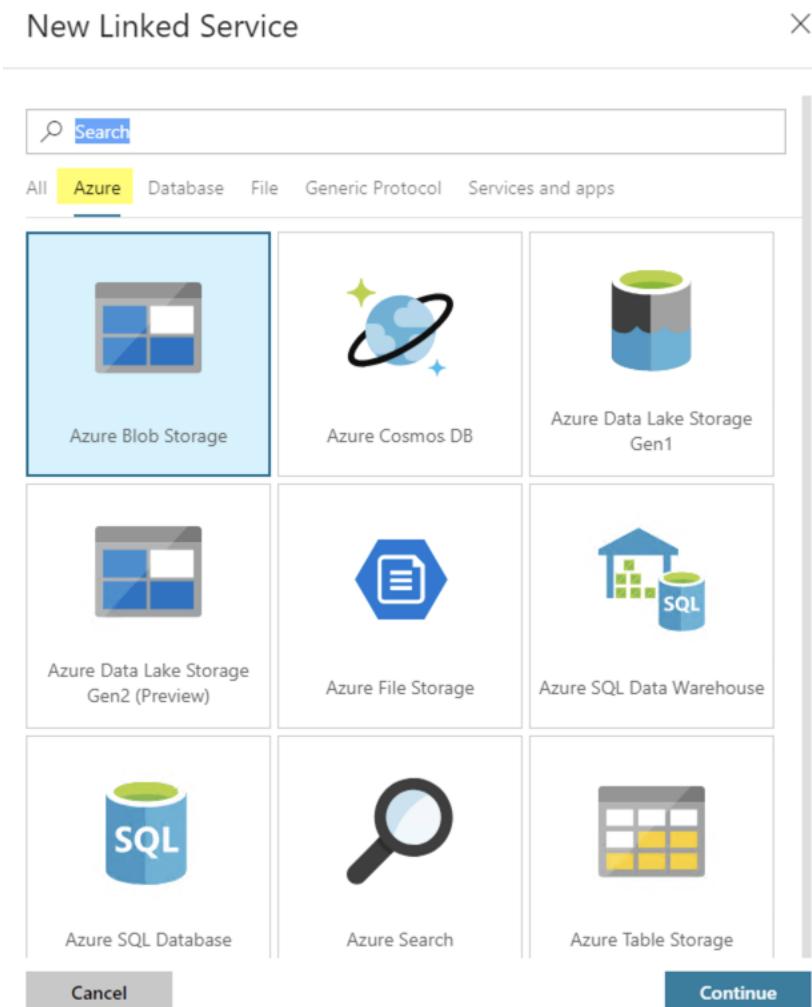
## Destination data store

Specify the destination data store for the copy task. You can use an existing data store connection or specify a new data store.

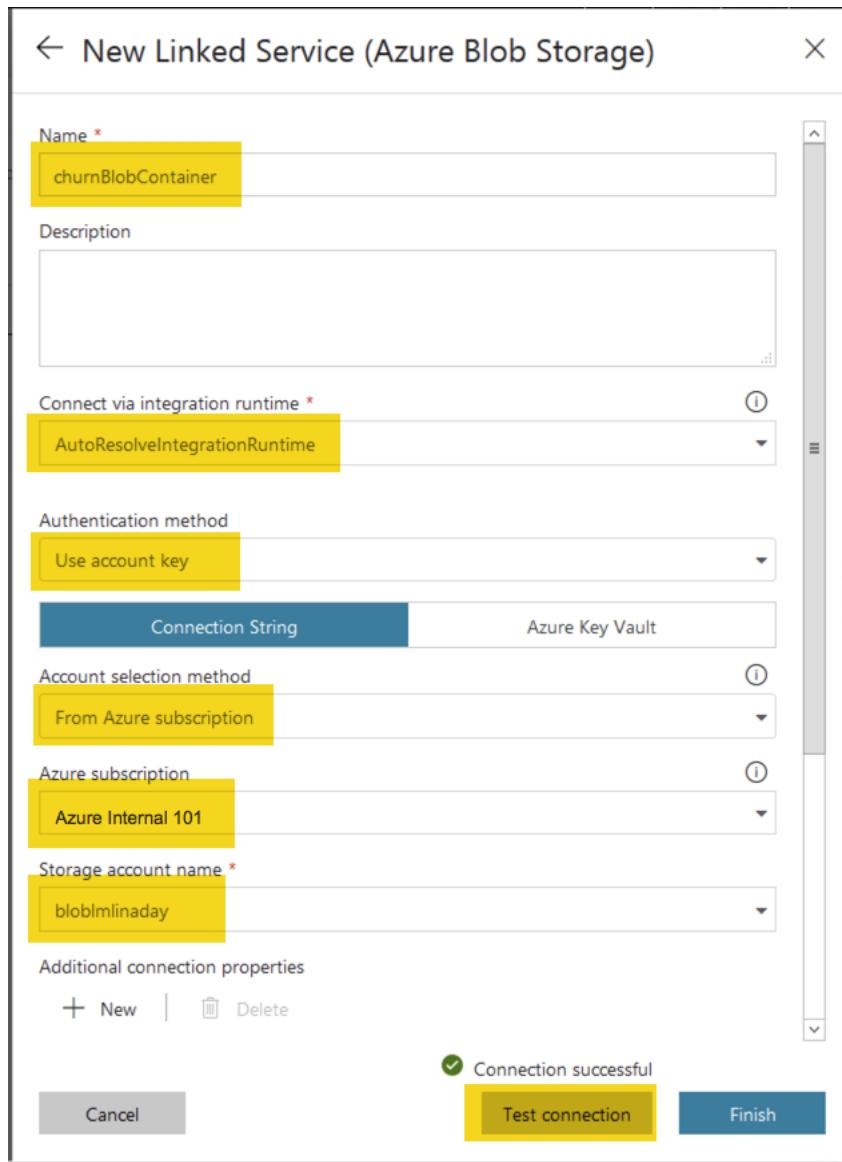
All Azure Database File Generic Protocol NoSQL Services and apps

All Filter by name + Create new connection

26. Search under **Azure** for **Azure Blob Store** as seen in the following screenshot:



27. Specify a name for the Linked Service to Azure Blob Storage such as **churnBlobContainer**. Additionally, select your **Azure Subscription** as well as your Blob Storage Account name and **Test the Connection** as seen in the following screenshot:



28. Once connection is successful, click on **Finish**.
29. Select the **Azure** tab as the **Destination Data Store** while also selecting the **churnBlobContainer** as the connection and then click on **Next** as seen in the following screenshot:

The screenshot shows the 'Copy Data' wizard interface. On the left, a vertical navigation bar lists steps 1 through 6: Properties, Source, Destination, Settings, Summary, and Deployment. Step 3 (Destination) is currently selected. The main area is titled 'Destination data store' and instructs the user to specify a destination data store. It includes tabs for All, Azure, Database, File, Generic Protocol, NoSQL, and Services and apps. Below the tabs is a search bar labeled 'Filter by name' and a 'Create new connection' button. A yellow box highlights the 'churnBlobContainer' item in the list. At the bottom, there are 'Previous' and 'Next' buttons, with 'Next' being highlighted.

30. Specify the **folder path** which is the container (**churn**) and give a name to your file such as **Telco\_Customer\_Churn.csv** as seen in the following screenshot (you can leave Compression Type and Copy Behavior as None):

The screenshot shows the 'Choose the output file or folder' settings page. It includes fields for 'Folder path' (containing 'churn'), 'File name' (containing 'Telco\_Customer\_Churn.csv'), 'Compression Type' (set to 'None'), and 'Copy behavior' (set to 'None'). At the bottom, there are 'Previous' and 'Next' buttons, with 'Next' being highlighted.

31. Specify the following **File format settings** and select **Next** as seen in the following screenshot:

## File format settings

File format  ⓘ  
Text format

Column delimiter  
Comma (,)

Use custom delimiter

Row delimiter  
Carriage Return + Line feed (\r\n)

Use custom delimiter

Skip line count  ⓘ  
0

Add header to file

► Advanced

Previous Next

32. Set **Fault tolerance setting** to **Skip incompatible Rows** as seen in the following screenshot:

Settings

More options for data movement

◀ Fault tolerance settings  
Fault tolerance  Skip incompatible rows  ⓘ

◀ Performance settings  
 Enable Staging ⓘ

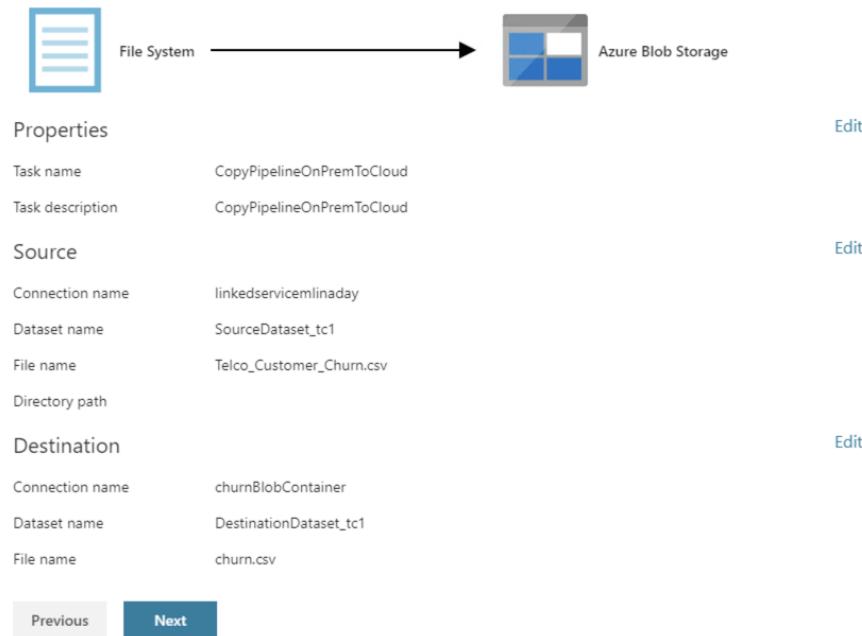
► Advanced settings

Previous Next

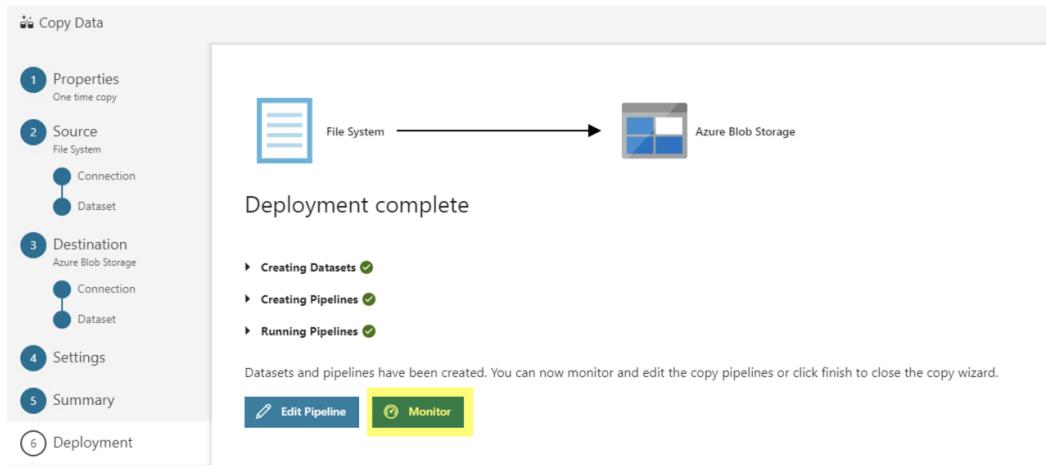
33. Review your summary pipeline to confirm all selections are accurate and select **Next** as seen in the following screenshot:

## Summary

You are running pipeline to copy data from File System to Azure Blob Storage.



34. Deployment is now complete and progress can be seen by clicking on **Monitor** as seen in the following screenshot:



35. The **Monitor** tab will give us some summary statistics as to whether or not our deployment succeeded and how long it took as seen in the following screenshot:

The screenshot shows the 'Pipeline Runs' monitor for the 'adfmlinaday' workspace. It displays a table of runs for the last 24 hours. One run is listed: 'CopyPipelineOnPremToCloud' with a duration of 00:00:22, triggered by 'Manual trigger', and a status of 'Succeeded'. A blue arrow points to the 'Time Zone' dropdown, which is set to '(UTC+00:00) Monrovia, Reykjavik'. A callout bubble says 'You can change your Time Zone here'.

Pipeline Name	Actions	Run Start	Duration	Triggered By	Status	Parameters	Error	RunID
CopyPipelineOnPremToCloud	▶	00:00:22		Manual trigger	Succeeded			8ee860ff-f7f4-4b16-a9c2-3c03829c756d

36. We can return to our blob container and confirm that our file made it there successfully. Anytime we wish to see anything we have deployed:

- we first go to <https://portal.azure.com>
- Then we select **Resource Groups** on the left hand side as seen in the following screenshot:

NAME	SUBSCRIPTION	LOCATION
MLinADayRG	Azure Internal 101	East US

- Finally, we can view any of the resources we have created as seen in the following screenshot:

NAME	TYPE	LOCATION
adfmlinaday	Data factory (V2)	East US
blobmlinaday	Storage account	East US

37. We can do a final test and see that our container now holds the **churn.csv** file in our blob container as seen in the following screenshot:

Home > Resource groups > MLinADayRG > blobminaday - Blobs > churn

**churn**  
Container

Search (Ctrl+)

Upload Refresh Delete Acquire lease Break lease View snapshots Create snapshot

Location: churn

Search blobs by prefix (case-sensitive)

Show deleted blobs

NAME	MODIFIED	ACCESS TIER	BLOB TYPE	SIZE	LEASE STATE
churn.csv	Hot (Inferred)	Block blob	954.59 KiB	Available	...

Overview Access Control (IAM) Settings  
Access policy Properties Metadata Editor (preview)

38. A simple click on the **churn.csv** file in the container and selecting **Edit blob** will reveal the contents of the file and confirm that our deployment to the cloud from on-premise was successful as seen in the following screenshot:

churn.csv

Blob

Upload Refresh More

Save Discard Refresh Download Delete

Location: churn

Search blobs by prefix (case-sensitive)

Show deleted blobs

NAME

churn.csv

Overview Snapshots Edit blob Generate SAS

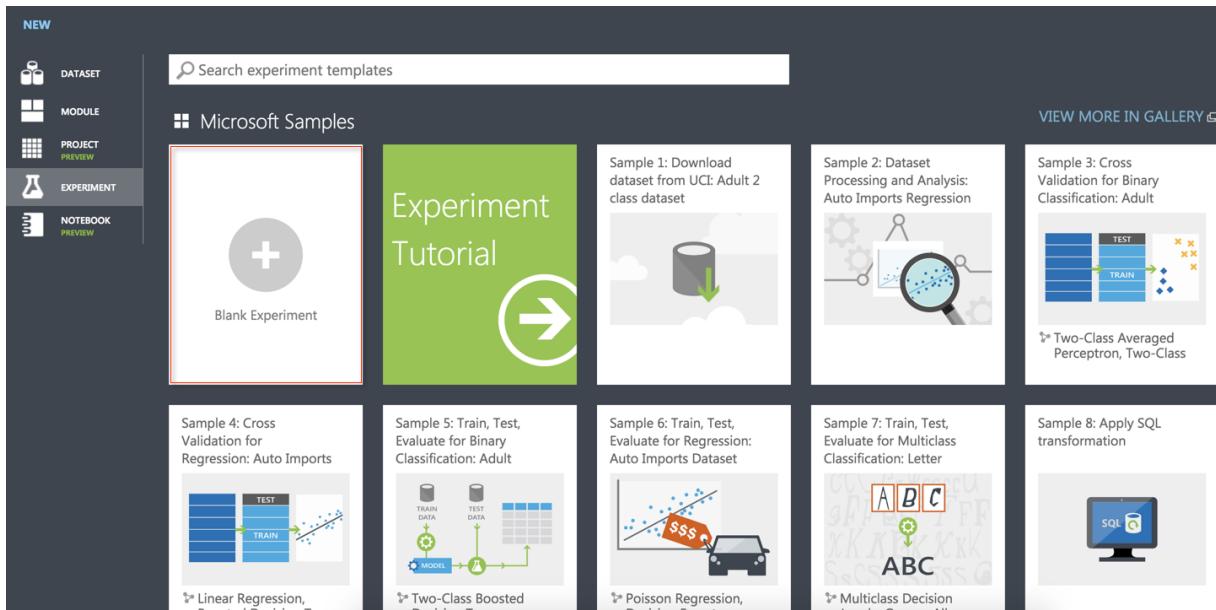
```

1 customerID,gender,SeniorCitizen,Partner,Dependents,tenure,PhoneService,MultipleLines,InternetService,Online
2 7590-VHVEG,Female,0,Yes,No,1,No,No phone service,DSL,No,Yes,No,No,No,Month-to-month,Yes,Electronic check,
3 5575-GNVDL,Male,0,No,No,34,Yes,No,DSL,Yes,No,Yes,No,No,One year,No,Mailed check,56.95,1889.5,No
4 3668-QPYBK,Male,0,No,No,2,Yes,No,DSL,Yes,Yes,No,No,No,Month-to-month,Yes,Mailed check,53.85,108.15,Yes
5 7795-CFOCW,Male,0,No,No,45,No,No phone service,DSL,Yes,No,Yes,Yes,No,No,One year,No,Bank transfer (automatic)
6 9237-HQITU,Female,0,No,No,2,Yes,No,Fiber optic,No,No,No,No,No,Month-to-month,Yes,Electronic check,70.7,1
7 9305-CDSKC,Female,0,No,No,8,Yes,Yes,Fiber optic,No,No,Yes,No,Yes,Yes,Month-to-month,Yes,Electronic check,99.
8 1452-KIOVK,Male,0,No,Yes,22,Yes,Yes,Fiber optic,No,Yes,No,No,Yes,No,Month-to-month,Yes,Credit card (automat
9 6713-OKOMC,Female,0,No,No,10,No,No phone service,DSL,Yes,No,No,No,No,Month-to-month,No,Mailed check,29.7
10 7892-POOKP,Female,0,Yes,No,28,Yes,Yes,Fiber optic,No,No,Yes,Yes,Yes,Yes,Month-to-month,Yes,Electronic check,
11 6388-TABGU,Male,0,No,Yes,62,Yes,No,DSL,Yes,Yes,No,No,No,One year,No,Bank transfer (automatic),56.15,3487.
12 9763-GRSKD,Male,0,Yes,13,Yes,No,DSL,Yes,Yes,No,No,No,Month-to-month,Yes,Mailed check,49.95,587.45,No
13 7469-LKBCI,Male,0,No,No,16,Yes,No,No,internet service,No,internet service,No,internet service,No,internet
14 8091-TTVAX,Male,0,Yes,No,58,Yes,Yes,Fiber optic,No,No,Yes,Yes,Yes,One year,No,Credit card (automatic),100
15 0280-XJGEK,Male,0,No,No,49,Yes,Yes,Fiber optic,No,Yes,Yes,Yes,Yes,Month-to-month,Yes,Bank transfer (auto
16 5129-JLPIS,Male,0,No,No,25,Yes,No,Fiber optic,Yes,Yes,Yes,Yes,Yes,Month-to-month,Yes,Electronic check,10
17 3655-SNQYZ,Female,0,Yes,69,Yes,Yes,Fiber optic,Yes,Yes,Yes,Yes,Yes,Two year,No,Credit card (automatic)
18 8191-XWSZG,Female,0,No,No,52,Yes,No,No,internet service,No,internet service,No,internet service,No,internet
19 9959-WOFKT,Male,0,No,Yes,71,Yes,Yes,Fiber optic,Yes,No,Yes,Yes,Yes,Two year,No,Bank transfer (automatic)
20 4190-MFLUW,Female,0,Yes,10,Yes,No,DSL,Yes,Yes,Yes,Yes,Yes,Month-to-month,No,Credit card (automatic),55.
21 4183-MYFRB,Female,0,No,No,21,Yes,No,Fiber optic,No,Yes,Yes,Yes,Yes,Month-to-month,Yes,Electronic check,90.
22 8779-QRDMV,Male,1,No,No,1,No,No phone service,DSL,No,No,Yes,Yes,No,No,Month-to-month,Yes,Electronic check,35
23 1680-VDCMW,Male,0,Yes,No,12,Yes,No,No,internet service,No,internet service,No,internet service,No,internet
24 1066-JKSGK,Male,0,No,No,1,Yes,No,No,No,internet service,No,internet service,No,internet service,No,internet
25 3638-WEABW,Female,0,Yes,No,58,Yes,Yes,DSL,Yes,Yes,No,No,Two year,Yes,Credit card (automatic),59.9,350
26 6322-HRPFA,Male,0,Yes,49,Yes,No,DSL,Yes,Yes,Yes,Yes,Month-to-month,No,Credit card (automatic),59.6
27 6865-JZNKO,Female,0,No,No,30,Yes,No,DSL,Yes,Yes,No,No,No,Month-to-month,Yes,Bank transfer (automatic),55.
28 6467-CHFWZ,Male,0,Yes,47,Yes,Yes,Fiber optic,No,Yes,No,No,Yes,Yes,Month-to-month,Yes,Electronic check,95
29 8665-UTDHZ,Male,0,Yes,1,Yes,No phone service,DSL,Yes,Yes,No,No,No,Month-to-month,No,Electronic check,30
30 5248-YGIJN,Male,0,Yes,No,72,Yes,Yes,DSL,Yes,Yes,Yes,Yes,Yes,Two year,Yes,Credit card (automatic),90.25,
```

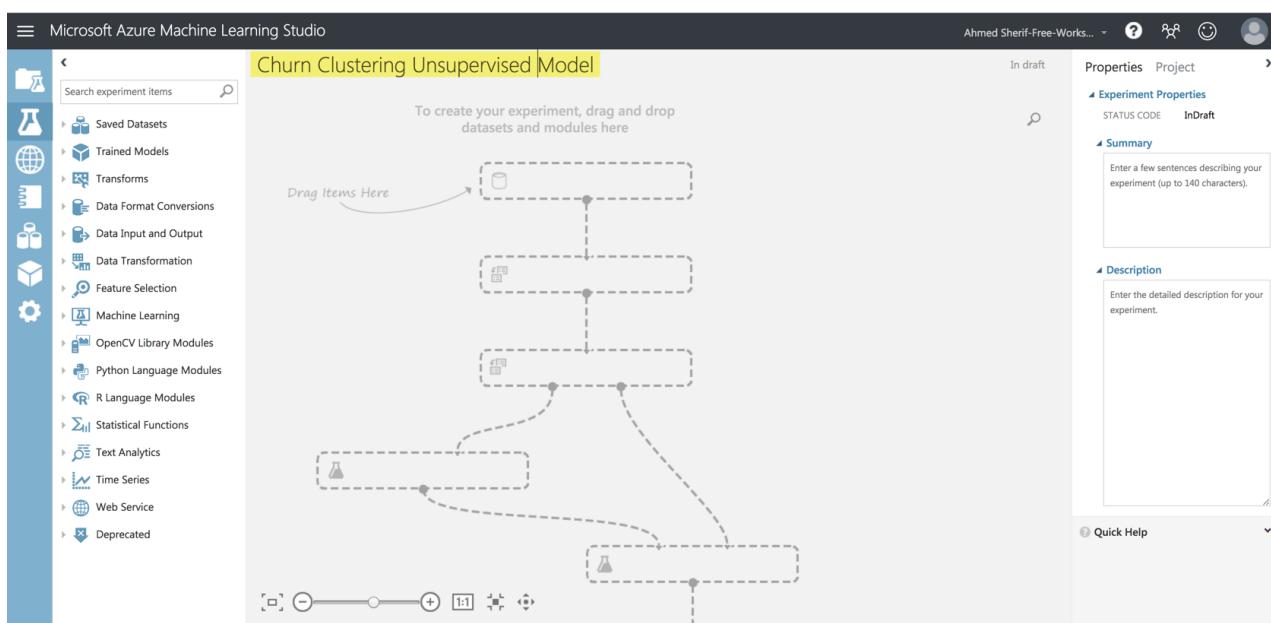
Preview Csv

## Section 4: Build a clustering unsupervised model in Azure Machine Learning Studio

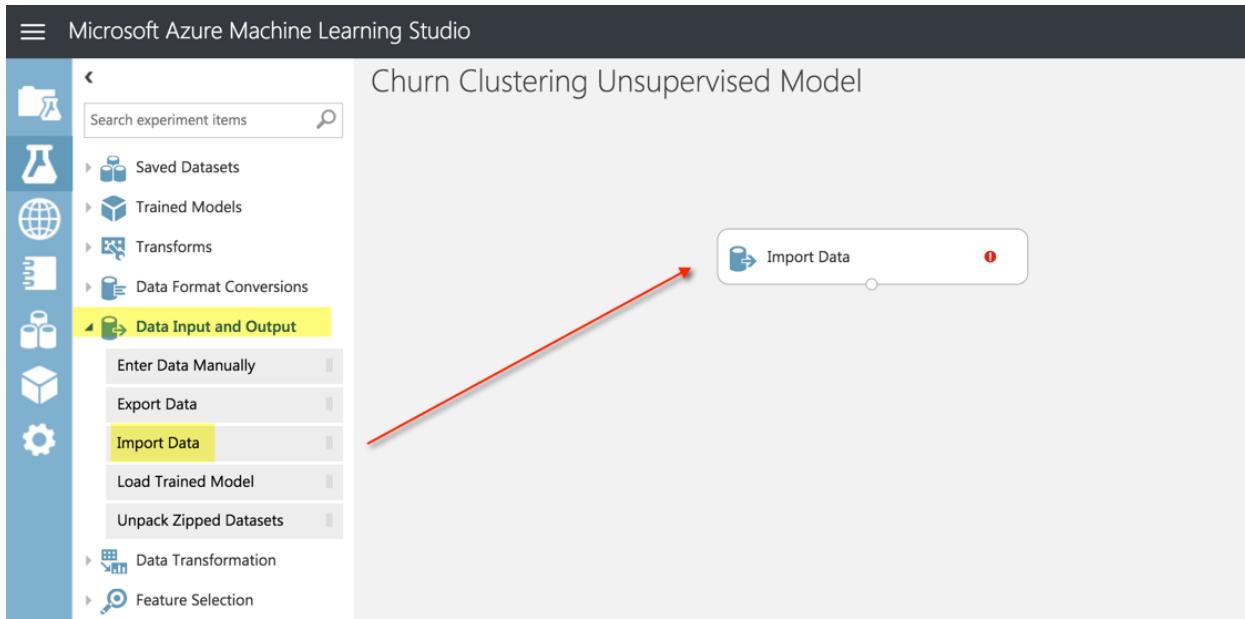
1. Clustering is the process of grouping similar entities together. The goal of this unsupervised machine learning technique is to find similarities in the fields that we have about our customers and group them into 10 distinct clusters.
2. Sign into Azure Machine Learning Studio using your Azure login Credentials in the following website:  
<https://studio.azureml.net/>
3. If you are not already directed to create a new experiment, go ahead and select the Experiments tab on the menu on the left-hand side. All current and future experiments will reside in this section.
4. Next click on **+NEW** on the button of the screen and select **Blank Experiment** as seen in the following screenshot:



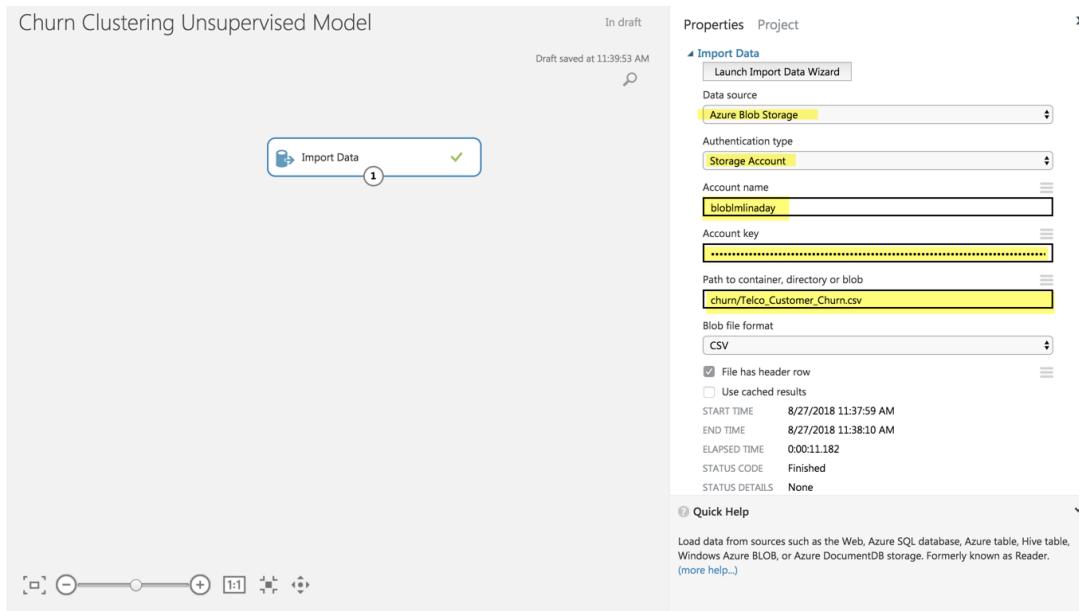
5. Click on the title of the experiment and change the name to **Churn Clustering Unsupervised Model** as seen in the following screenshot:



6. Expand the **Data Input and Output** category and drag the **Import Data** into the Workflow as seen in the following screenshot:



7. As you click on the **Import Data** icon in the workflow, you will have the option to configure your data to pull from the Azure Blob Container you created a few sections ago. Enter the following credentials to import your dataset into Azure ML Studio through a Blob Container connection:



8. The Account Key is the same one that we saved a few sections ago when we created our Blob storage account and pasted it onto a Notepad editor. Once again, we will walk through the steps to obtain the **Account Key** field by going back to the Azure Portal ([portal.azure.com](https://portal.azure.com)) and copying **Key1** from the **Access Keys** section for the **blobmlinaday** account as seen in the following screenshot:

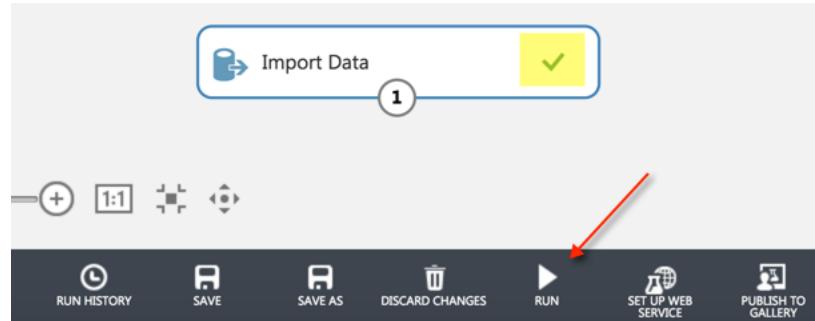
blobmlinaday - Access keys

Storage account name: blobmlinaday

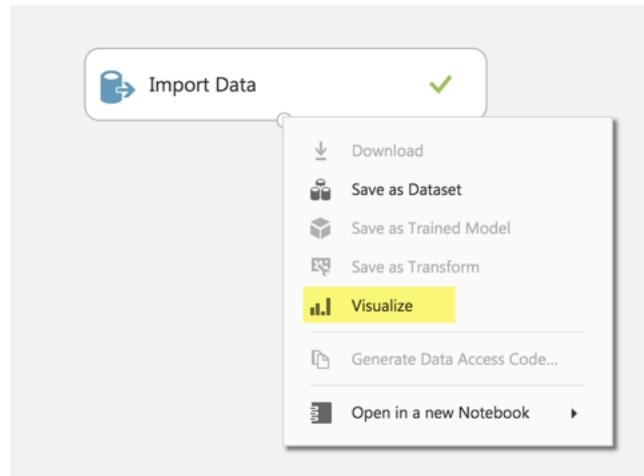
key1: 2Le22d38roB6lFqAeds/dXRfc6VoKUDBXqcsjhzRv8r3YDaB1x7FdmyzLcaH04NAFrL+OQmXsNlo4NzICg==

key2: e3rNW07ji+s37Xy7/v84m8mCYSDQSH+p9ixUmnfpgrwta02/yaoEpp9jhYcReHuqZLx0s/qYgbm6CPxaE8yA==

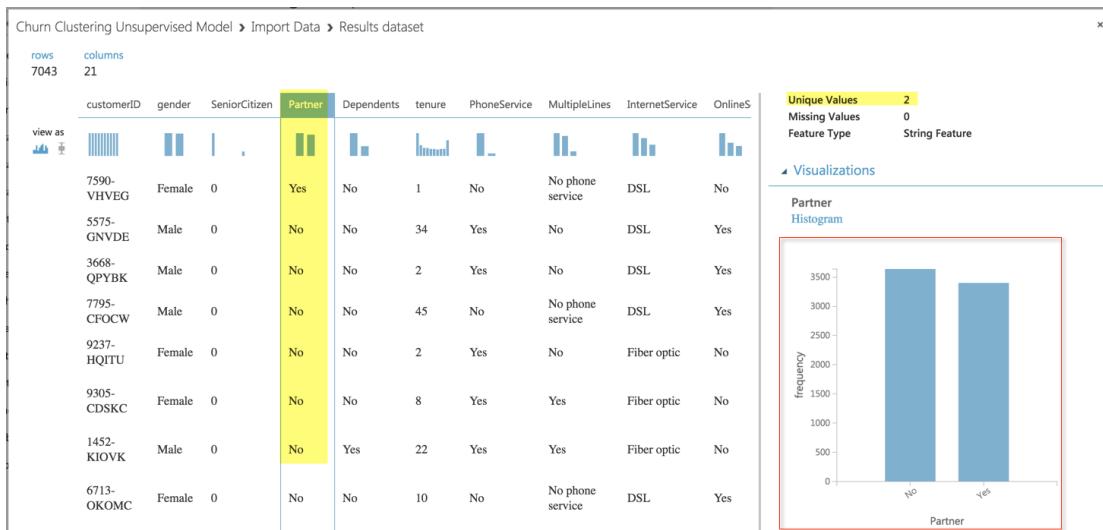
- Back in the Azure ML Studio workflow, the credentials can be confirmed by clicking on the **RUN** icon at the bottom of the screen and seeing the green check mark next to **Import Data**.



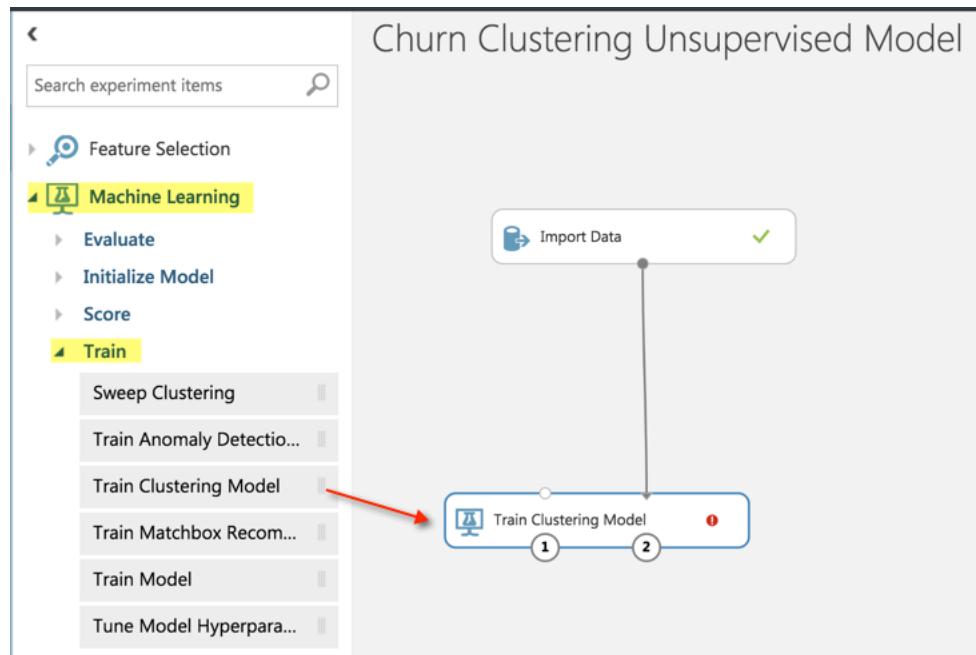
- Once we have confirmed the connection is working, we can right-click on the number **1** on the first workflow and **Visualize** the dataset as seen in the following screenshot:



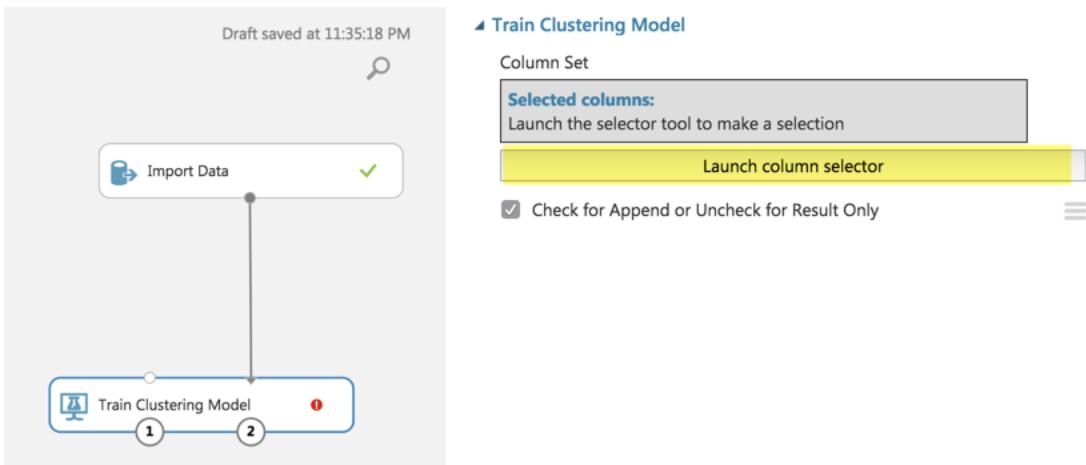
- Take some time to click on the columns to identify the summary statistics for both the numeric fields and the categorical fields as seen in the following screenshot:



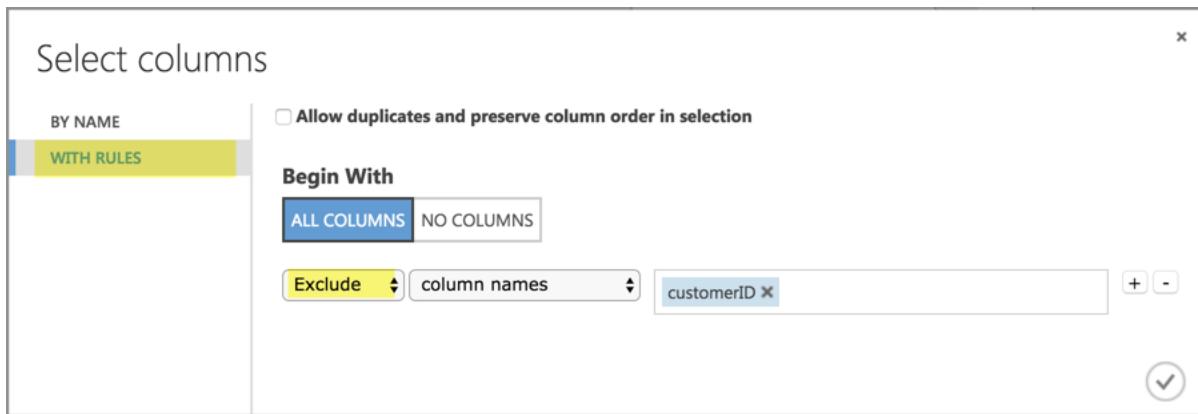
12. Import **Train Clustering Model** from the **Machine Learning** Node and connect it to the **Import Data** module as seen in the following screenshot:



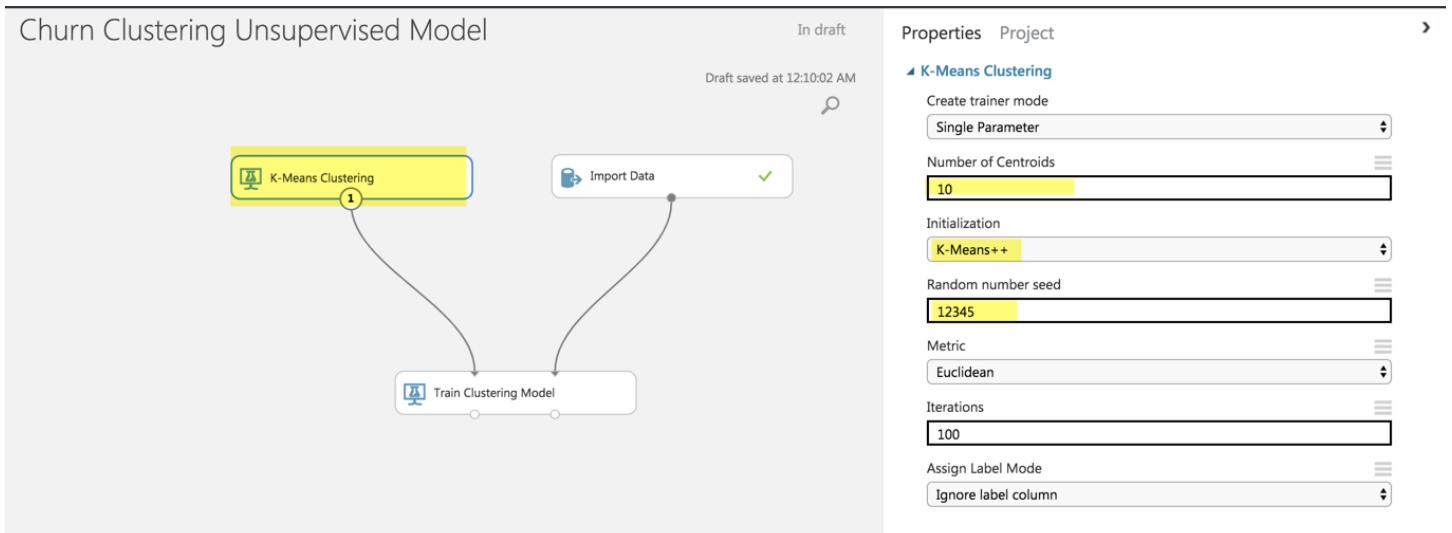
13. Next, click on the **Train Clustering Model** module, **Launch Column Selector**, connect the node back to the **Import Data** node, and configure the columns to be selected as seen in the following screenshot:



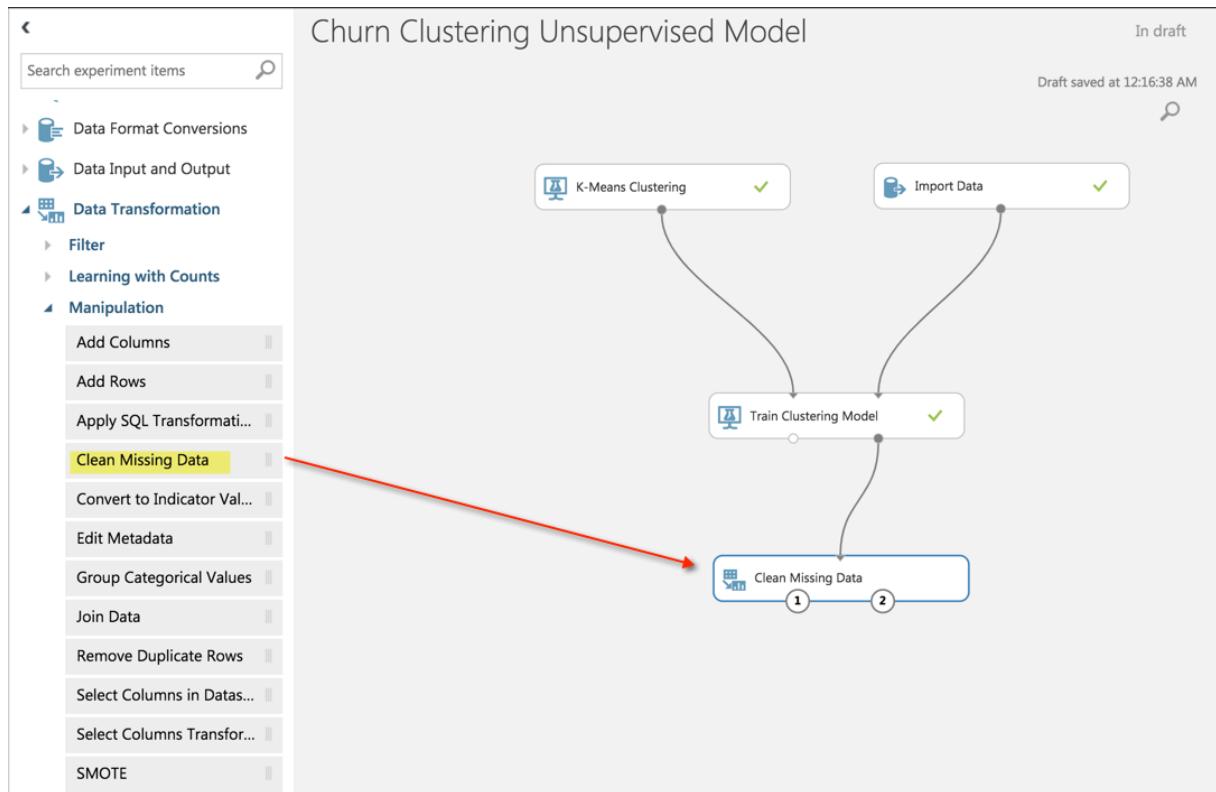
14. Next, you will need to configure the **clustering model** and **Select All columns** from the dataset except for the column called **customerID** as seen in the following screenshot:



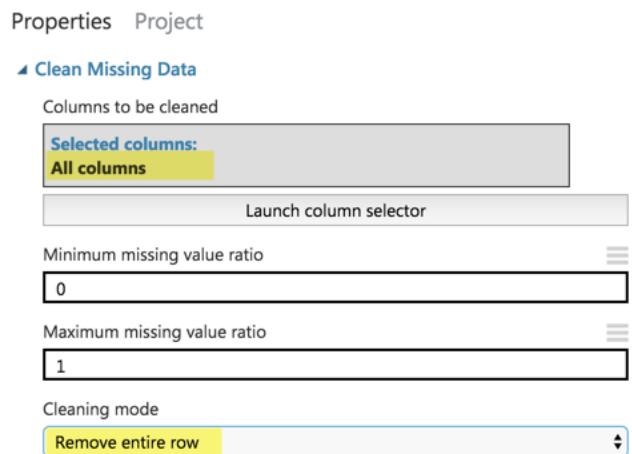
15. Next, we will attach the **K-Means Clustering** model to the **Train Clustering Model** and assign the following parameters to the clustering model, as seen in the following screenshot:



16. Next, we will do some data cleanup and remove any rows that have missing values by pulling in the **Clean Missing Data** module and connecting it to the **Train Clustering Model** as seen in the following screenshot:



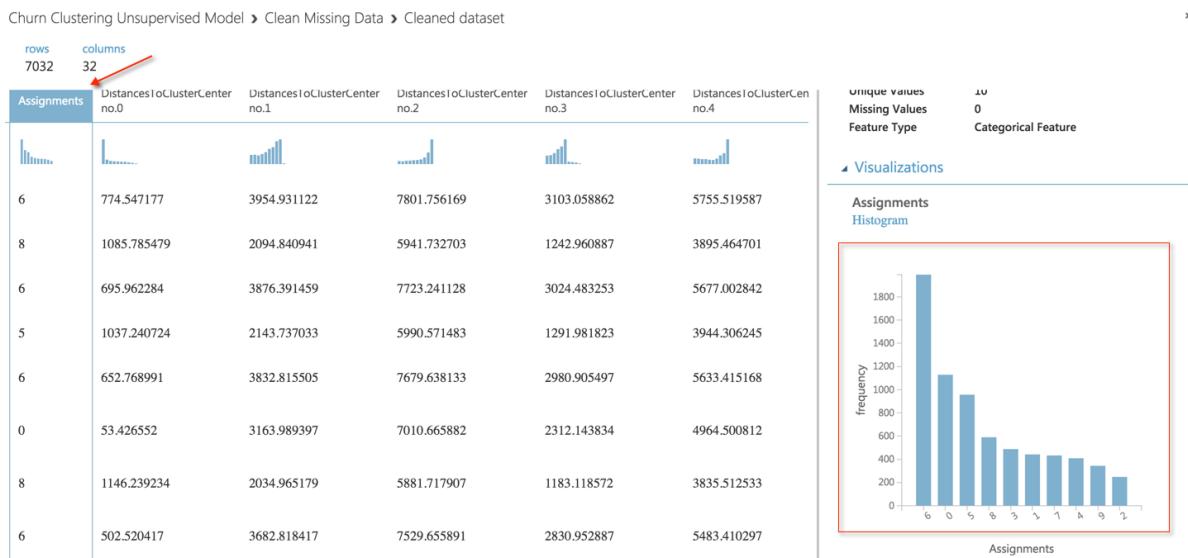
17. We can configure the **Clean Missing Data** module by clicking on it and selecting the following configuration to remove any rows if they meet the following criteria:



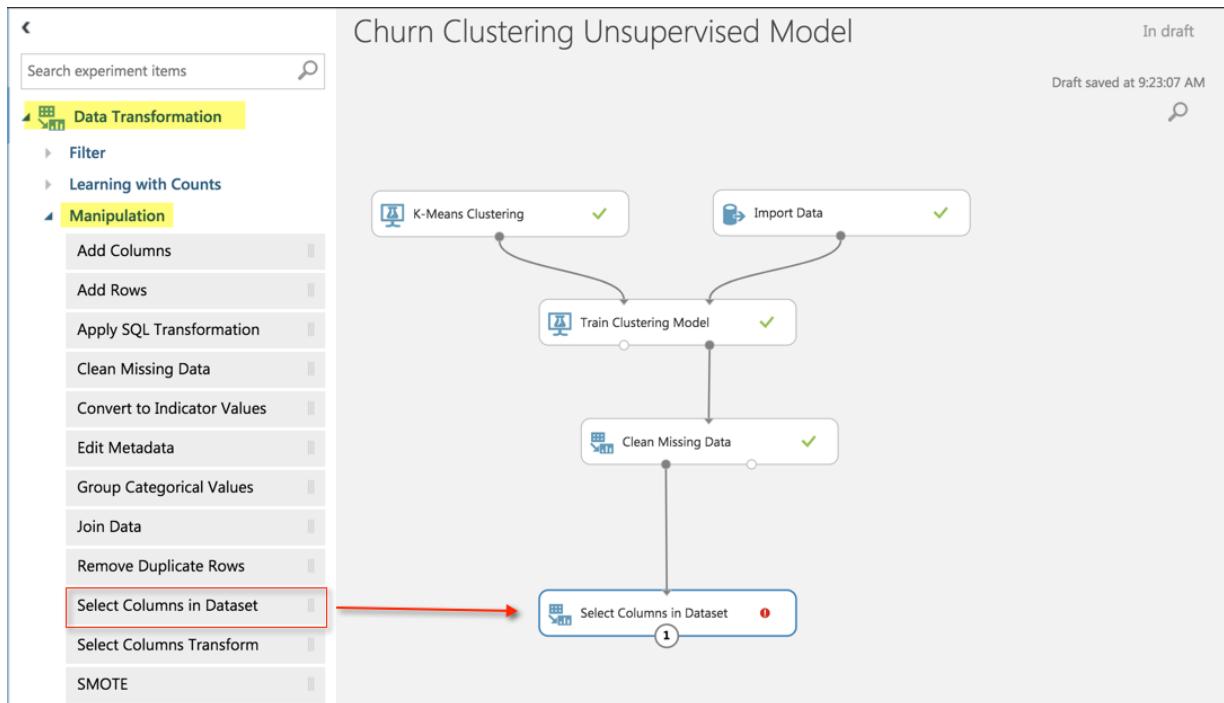
18. The clustering outcome produces several columns that are not necessary for our purposes for this workshop as can be seen by visualizing the **Clean Missing Data** module as seen in the following screenshot:

Churn Clustering Unsupervised Model > Clean Missing Data > Cleaned dataset							
rows	columns						
Churn	Assignments	DistancesToClusterCenter no.0	DistancesToClusterCenter no.1	DistancesToClusterCenter no.2	DistancesToClusterCenter no.3	DistancesToClusterCenter no.4	DistancesToC no.4
No	6	774.547177	3954.931122	7801.756169	3103.058862	5755.519587	
No	8	1085.785479	2094.840941	5941.732703	1242.960887	3895.464701	
Yes	6	695.962284	3876.391459	7723.241128	3024.483253	5677.002842	
No	5	1037.240724	2143.737033	5990.571483	1291.981823	3944.306245	
Yes	6	652.768991	3832.815505	7679.638133	2980.905497	5633.415168	
Yes	0	53.426552	3163.989397	7010.665882	2312.143834	4964.500812	
No	8	1146.239234	2034.965179	5881.717907	1183.118572	3835.512533	
No	6	502.520417	3682.818417	7529.655891	2830.952887	5483.410297	

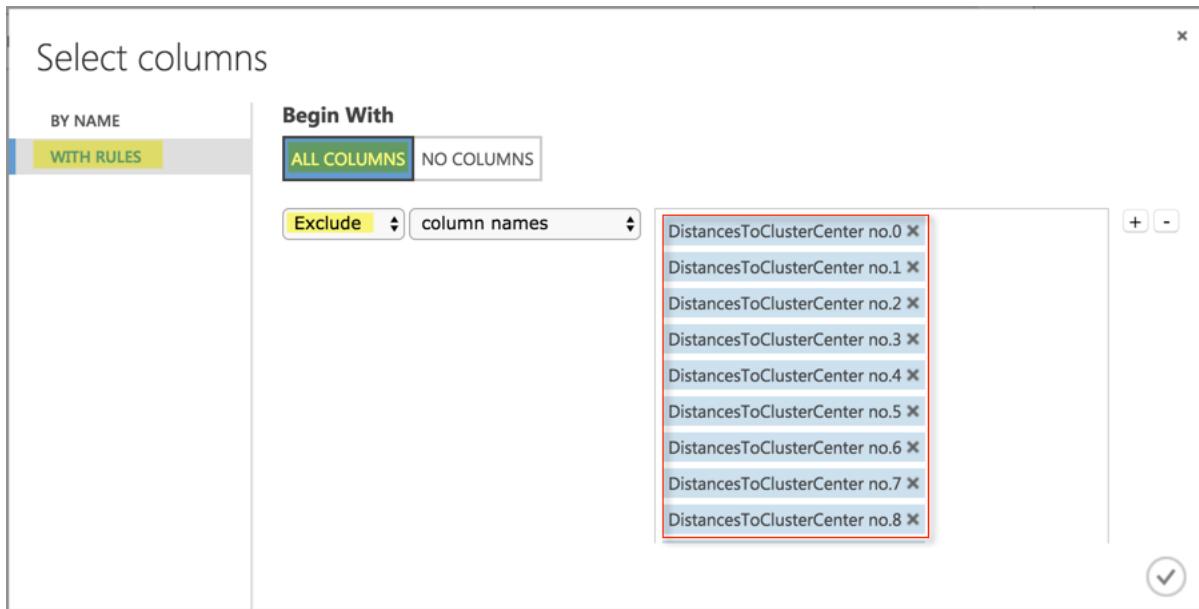
19. We are truly only interested in the assignments, which is grouping each customer into a category of **0** through **9** as seen when we visualize a histogram of the unique values as seen in the following screenshot:



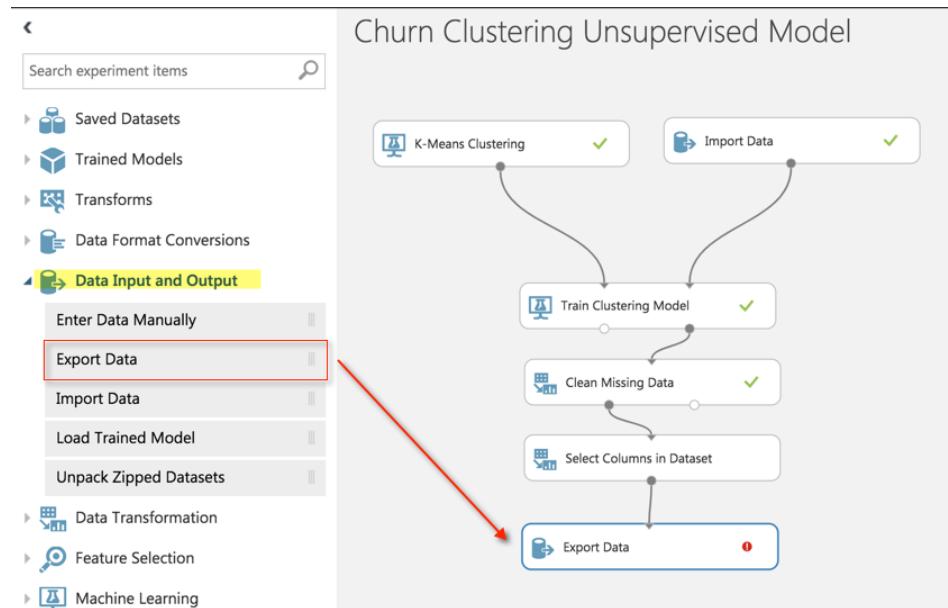
20. All other columns that start with **DistancesToClusterCenter** are not relevant for our purposes and can be removed. The easiest way to do this is to drag into our workflow **Select Columns in Dataset** as seen in the following screenshot:



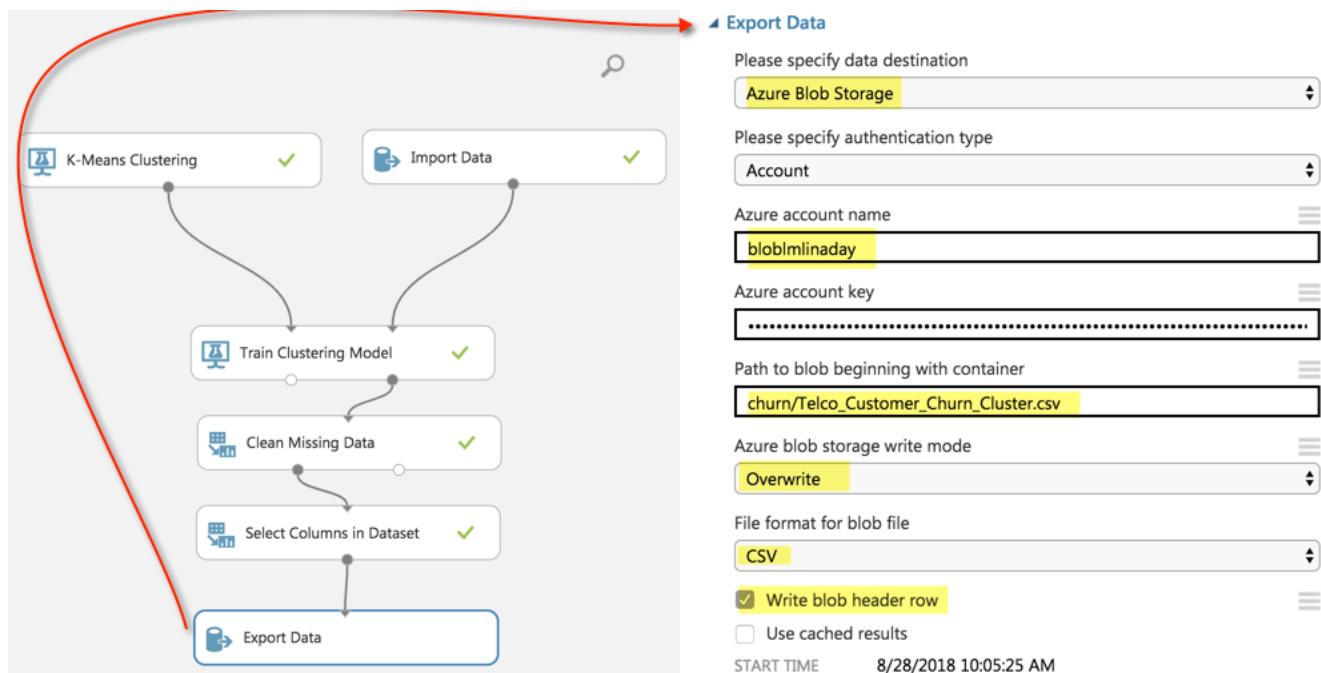
21. We can then configure the module for **Select Columns in Dataset** by **Launching Column Selector**, select **WITH RULES**, and exclude column names as seen in the following screenshot:



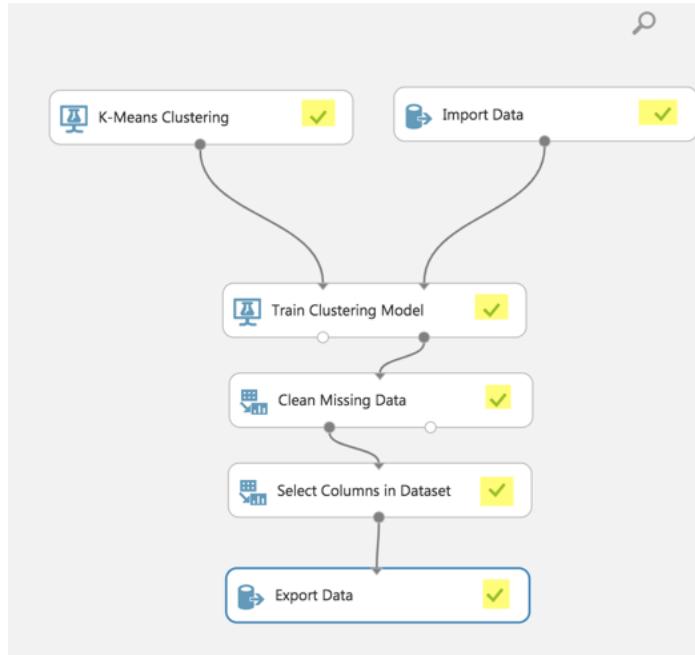
22. We are finally ready to send out revised dataset with only the columns that we need back to blob storage using the Export Data module as seen in the following screenshot:



23. Just as we did with importing the data into AML Studio from Blob, we will need to configure the connection to export the new dataset, **Telco\_Customer\_Churn\_Cluster.csv**, back to the same blob container, **churn**, using the following configuration as seen in the following screenshot (*please note that you should continue to use the same Key1 from Blob storage that was used earlier on in the section when importing the data set from Blob*):



24. Once we are done, we can execute our workflow one last time by clicking on the **RUN** button. Once completed, we should hopefully see all green checkmarks next to each step of our workflow as seen in the following screenshot:



25. We can revisit our Blob container in the Azure Portal and view our newly added dataset alongside our existing dataset as seen in the following screenshot:

Home > Resource groups > MLinADayRG > blobmlinaday - Blobs > churn

**churn**  
Container

Search (Ctrl+ /) <<

Upload Refresh Delete Acquire lease Break lease View snapshots Create snapshot

Overview

Access Control (IAM)

Settings

- Access policy
- Properties
- Metadata
- Editor (preview)

Location: churn

Search blobs by prefix (case-sensitive)

NAME	MODIFIED	BLOB TYPE
Telco_Customer_Churn_Cluster.csv		Block blob
Telco_Customer_Churn.csv		Block blob

26. Finally, we will right-click on the csv file and obtain our **Generate SAS** as seen in the following screenshot:

The screenshot shows the Azure Storage Blob service interface. A blob named 'Telco\_Customer\_Churn\_Cluster.csv' is selected. A context menu is open over it, with the 'Generate SAS' option highlighted by a yellow box. Other options in the menu include 'View/edit blob', 'Download', 'Blob properties', 'View snapshots', 'Create snapshot', 'Acquire lease', 'Break lease', and 'Delete'.

27. A **shared access signature (SAS)** is a URL that grants restricted access rights to Azure Storage resources (a specific blob in this case). You can provide a shared access signature to clients who should not be trusted with your storage account key but whom you wish to delegate access to certain storage account resources. By distributing a shared access signature URI to these clients, you grant them access to a resource for a specified period of time. Select Generate blob SAS token and URL as seen in the following screenshot:

The screenshot shows the blob properties page for 'Telco\_Customer\_Churn\_Cluster.csv'. The 'Generate SAS' button is highlighted with a yellow box. An orange arrow points from the 'Expiry' field (set to 2028-09-10) to a download icon. Another orange arrow points from the generated SAS URL to a download icon.

**Permissions:** Read  
**Start and expiry date/time:**  
 Start: 2018-09-03 12:59:46 PM (UTC-04:00) --- Current Time Zone ---  
 Expiry: 2028-09-10 8:59:46 PM (UTC-04:00) --- Current Time Zone ---  
**Allowed IP addresses:** for example, 168.1.5.65 or 168.1.5.65-168.1.5.70  
**Allowed protocols:** HTTPS (selected)  
**Signing key:** Key 1  
**Generate blob SAS token and URL**

**Blob SAS token:** sp=r&st=2018-09-03T16:59:46Z&se=2018-09-04T00:59:46Z&spr=https&sv=2017-11-09&sig=GyDm%2FSFyXWAjq3%2BbfwZUhjGBjoNb2X...  
**Blob SAS URL:** https://blobmlinaday.blob.core.windows.net/churn/Telco\_Customer\_Churn\_Cluster.csv?sp=r&st=2018-09-03T16:59:46Z&se=2018-09-04T00:59:46Z&spr=https&sv=2017-11-09&sig=GyDm%2FSFyXWAjq3%2BbfwZUhjGBjoNb2X...

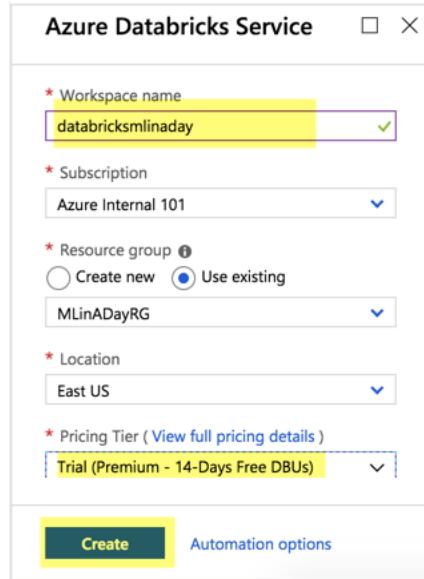
28. Keep a copy of the SAS URL as it will come in handy in the next section when we are reading the file from Blob to Azure Databricks. Additionally, make sure to set expiry date sometime in the future so that it does not expire while you are working on the workshop:

## Section 5: Set up a Spark Cluster on Databricks and connect to Azure Blob Storage Account

1. Return to the Azure portal (<https://portal.azure.com>) and provision an **Azure Databricks** workspace by selecting Azure Databricks from the **marketplace** as seen in the following screenshot:

The screenshot shows the Microsoft Azure portal interface. At the top, there's a navigation bar with 'Preview' (orange), 'Microsoft Azure' (blue), 'Report a bug' (orange), and a search bar ('Search resources, services, and documents'). Below the navigation bar, the breadcrumb trail shows 'Home > New > Azure Databricks'. The main content area is titled 'Azure Databricks' under the 'Microsoft' heading. It features a brief description: 'Fast, easy, and collaborative Apache Spark-based analytics platform' and 'Accelerate innovation by enabling data science with a high-performance analytics platform that's optimized for Azure.' A large yellow button labeled '+ Create a resource' is visible on the left sidebar. The sidebar also lists various Azure services: All services, Favorites (Resource groups, Dashboard, All resources, App Services, Function Apps, SQL databases, Azure Cosmos DB, Virtual machines, Load balancers, Storage accounts, Virtual networks, Azure Active Directory, Monitor, Advisor, Security Center, Cost Management + Billing, Help + support). On the right side of the main content area, there's a 'Save for later' button and a preview window showing the Azure Databricks portal with a 'Create' button.

2. Enter the **Workspace** specifications as seen in the following screenshot:



3. Once it has been provisioned, you can go to the resource and **launch the workspace** as seen in the following screenshot:

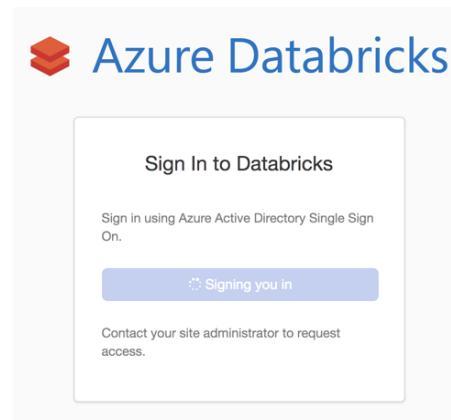
The screenshot shows the 'databricksmlinaday' workspace overview. Key details include:

- Resource group: MLinADayRG
- Subscription: Azure Internal 101
- Subscription ID: ba3edd26-e2b1-4cc5-a19e-5bd21d7e9f5d
- Managed Resource Group: databricks-rg-databricksmlinaday-xgf6fnhqr4rm
- URL: <https://eastus.azuredatabricks.net>
- Pricing Tier: Trial (Premium - 14-Days Free DBUs)

Below the details is a large red Databricks logo. To its right are 'Launch Workspace' and 'Upgrade to Premium' buttons. At the bottom is a grid of links:

- Documentation
- Getting Started
- Import Data from File
- Import Data from Azure Storage
- Notebook
- Admin Guide

4. You will then be taken to the Databricks workspace and signed in automatically through your **Azure Active Directory** credentials as seen in the following screenshot:



5. You will then land in the Azure Databricks home page. The first thing we will need to do is provision a **new cluster**, as seen in the following screenshot:

A screenshot of the Azure Databricks home page. The header shows 'Microsoft Azure' and 'PORTAL'. It features a central 'Azure Databricks' logo. Below the logo are three main sections: 'Explore the Quickstart Tutorial', 'Import &amp; Explore Data', and 'Create a Blank Notebook'. On the left, a sidebar lists 'Common Tasks' including 'New Notebook', 'Upload Data', 'Create Table', 'New Cluster' (which is highlighted with a yellow background), 'New Job', 'Import Library', and 'Read Documentation'. The 'Recent' section shows a single file named 'Untitled Notebook'. The 'Documentation' section links to 'Databricks Guide', 'Python, R, Scala, SQL', and 'Importing Data'.

6. For our purposes for this workshop, we will provision a **Standard mode cluster with Python Version 3 support called Machine learning in a Day Cluster** as seen in the following screenshot:

Create Cluster

New Cluster | Cancel | Create Cluster | 2-8 Workers: 28.0-112.0 GB Memory, 8-32 Cores, 1.5-6 DBU  
1 Driver: 14.0 GB Memory, 4 Cores, 0.75 DBU Cost \$0.55 per DBU

**Cluster Name**  
Machine Learning in a Day Cluster

**Cluster Mode**  
 High Concurrency  Standard  
Optimized to run concurrent SQL, Python, and R workloads.  
Does not support Scala. Previously known as Serverless.

**Databricks Runtime Version** 4.2 (includes Apache Spark 2.3.1, Scala 2.11)

**Python Version** 3

**Driver Type** Same as worker

**Worker Type** Standard\_DS3\_v2 | Min Workers: 2 | Max Workers: 8 |  Enable autoscaling

**Auto Termination**  Terminate after 120 minutes of inactivity

- Once we select **Create Cluster**, it will begin provisioning and we can move onto creating our first notebook as seen in the following screenshot:

Free trial ends in 14 days. Upgrade to Premium in Azure Portal ?

**Azure Databricks**

**Explore the Quickstart Tutorial**  
Spin up a cluster, run queries on preloaded data, and display results in 5 minutes.

**Import & Explore Data**  
Quickly import data, preview its schema, create a table, and query it in a notebook.

**Create a Blank Notebook**  
Create a notebook to start querying, visualizing, and modeling your data.

**Common Tasks**

- New Notebook**
- Upload Data
- Create Table
- New Cluster
- New Job
- Import Library
- Read Documentation

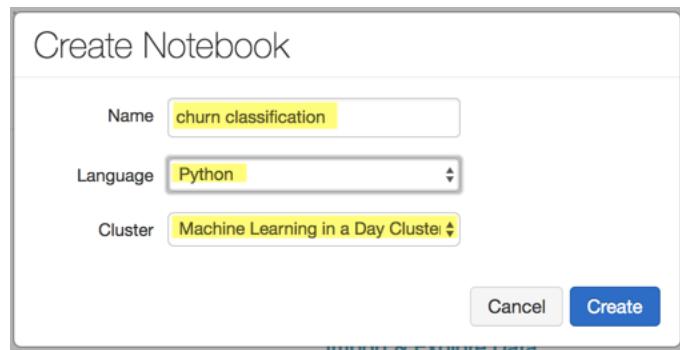
**Recents**

Recent files appear here as you work.

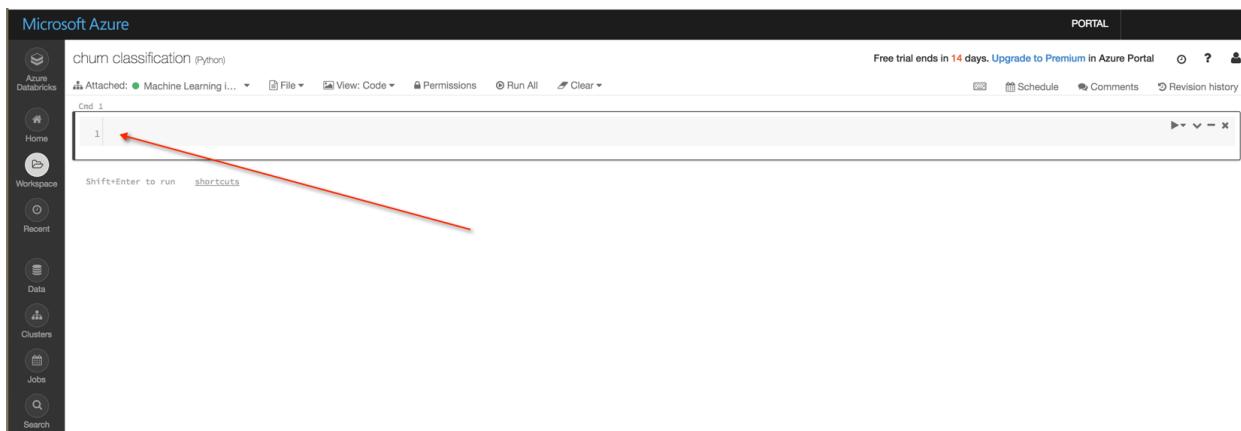
**Documentation**

- Databricks Guide
- Python, R, Scala, SQL
- Importing Data

- We name our notebook, **churn classification**, and assign it a default language of **Python** with the cluster we just created as seen in the following screenshot:



9. We are now in our notebook where we will be building our machine learning classification model in cells as seen in the following screenshot:



10. We are going to construct our URL to our blob file. This will require using the SAS URL link that we copied from step 26 from the previous section. We will set the URL to a variable and read it into a Spark DataFrame using the following script in the cell:

```
SAS_url =  
'https://blobmlinaday.blob.core.windows.net/churn/Telco_Customer_Churn_Cluster.csv  
?sp=r&st=2018-09-03T16:59:46Z&se=2018-09-04T00:59:46Z&spr=https&sv=2017-11-  
09&sig=GyDm%2FSFyXWAjq3%2BbfwZUhiGBjoNb2X%2F81%2BI2OHKA0Nw%3D&sr=b'  
  
import pandas as pd  
pandas_df = pd.read_csv(SAS_url)  
  
df = spark.createDataFrame(pandas_df)
```

Please note that your SAS\_url will be different than the one in this documentation

11. The script appears as the following in the notebook:

Microsoft Azure PORTAL

churn classification (Python)

Attached: Machine Learning i... File View: Code Permissions Run All Clear

Free trial ends in 11 days. Upgrade to Premium in Azure Portal

Home Workspace Recent Data Clusters Jobs Search

```

Cmd 1
1 # Please note your SAS_url link will be different
2 SAS_url = 'https://blobmlinaday.blob.core.windows.net/churn/Telco_Customer_Churn_Cluster.csv?sp=r&st=2018-09-03T16:59:46Z&se=2018-09-04T00:59:46Z&spr=https&sv=2017-11-09&sig=GyDm%2FSFyXWqj3%2BbfwZUhGBjoNb2X%2F8l%2BI20HKA0Nw%3D&sr=b'

Command took 0.01 seconds -- by ahsherif@microsoft.com at 9/3/2018, 1:07:58 PM on Machine Learning in a Day Cluster

Cmd 2
1 import pandas as pd
2 pandas_df = pd.read_csv(SAS_url)

Command took 0.26 seconds -- by ahsherif@microsoft.com at 9/3/2018, 1:09:03 PM on Machine Learning in a Day Cluster

Cmd 3
1 df = spark.createDataFrame(pandas_df)

> df: pyspark.sql.dataframe.DataFrame = [customerID: string, gender: string ... 20 more fields]

Command took 1.33 seconds -- by ahsherif@microsoft.com at 9/3/2018, 1:10:23 PM on Machine Learning in a Day Cluster

Cmd 4
1

```

12. Our Spark DataFrame, **df**, is created and can be viewed by executing `display(df)` as seen in the following screenshot:

Cmd 4

1 `display(df)`

(2) Spark Jobs

customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	Stream
7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	No	No	No	No
5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes	No	No	No
3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	No	No	No	No
7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	No	Yes	Yes	No	No
9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	No	No	No	No	No
9305-CDSKC	Female	0	No	No	8	Yes	Yes	Fiber optic	No	No	Yes	No	Yes	Yes

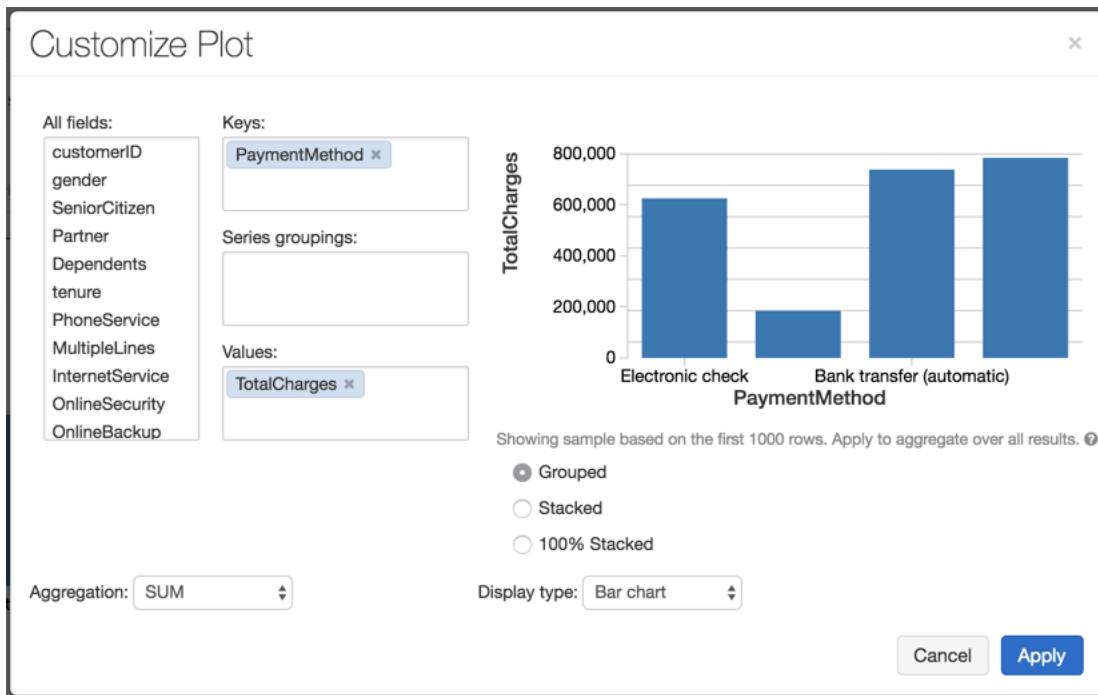
Showing the first 1000 rows.

Command took 2.12 seconds -- by ahsherif@microsoft.com at 9/3/2018, 1:21:44 PM on Machine Learning in a Day Cluster

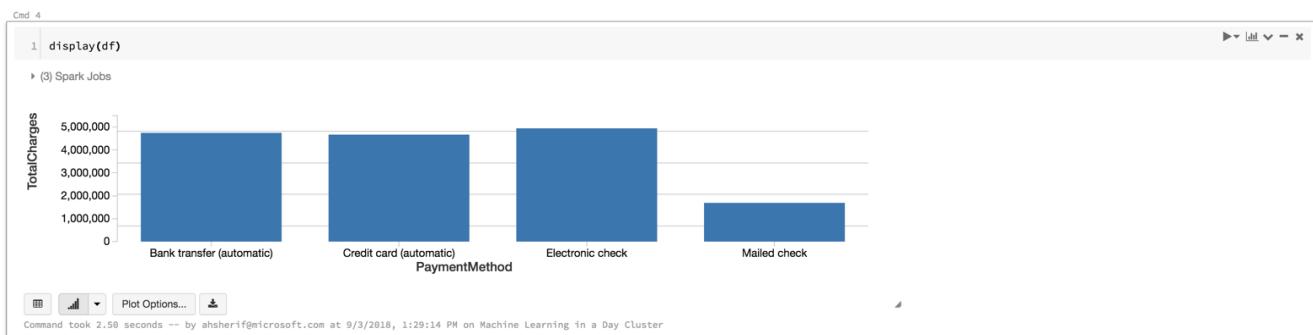
13. The easiest way to create a visualization in Databricks is to call the following script:

```
display(<dataframe-name>)
```

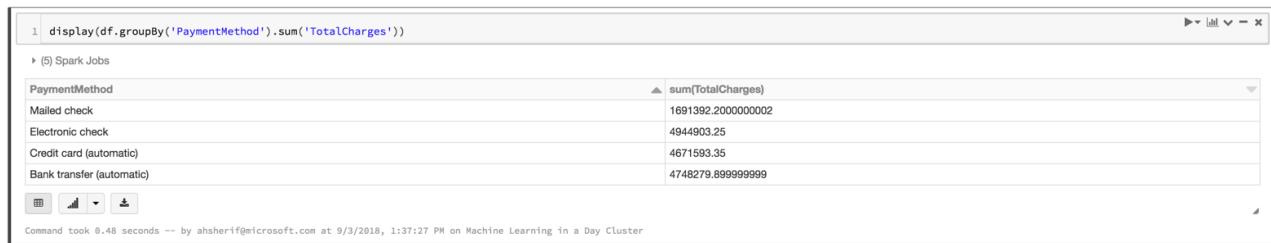
14. Databricks gives us the option to convert DataFrames into visualizations on the fly without having to code them using the bar chart icon underneath the DataFrame. For example, we can create a bar chart with **PaymentMethod** on the x-axis and **TotalCharges** on the y-axis by sum as seen in the following screenshot:



15. Once we apply the **keys**, **groupings**, and **values** we can see the output directly within the cell of the notebook as seen in the following screenshot:



16. Spark is not that hard to learn, especially if you have some existing skills in Python, SQL, or R. We can use Python syntax to do some data analysis and view **TotalCharges** by **PaymentMethod** using the following Python syntax script `display(df.groupBy('PaymentMethod').sum('TotalCharges'))` as seen in the following screenshot:



17. However, we can also do the same data analysis using SQL syntax. First, we create a view using the following script:  
`df.createTempView('sqlView')`.

18. Then we create a cell with SQL syntax and execute the following script:

```
%SQL
```

```
select PaymentMethod, sum(TotalCharges) from sqlView group by 1 order by 1 asc;
```

19. The output of the script can be seen in the following screenshot:

The screenshot shows a Databricks notebook interface. At the top, it says "churn classification (Python)". Below that, there's a toolbar with "Attached: Machine Learning i...", "File", "View: Code", "Permissions", "Run All", "Clear", and links for "Free trial ends in 11 days. Upgrade to Premium in Azure Portal", "Schedule", "Comments", and "Revision history".

In the code editor, there are two cells:

```
1 df.createTempView('sqlView')

Command took 0.06 seconds -- by ahsherif@microsoft.com at 9/3/2018, 1:52:46 PM on Machine Learning in a Day Cluster
```

```
1 %sql
2 select PaymentMethod, sum(TotalCharges) from sqlView group by 1 order by 1 asc;
```

Below the code editor, the results of the second cell are displayed in a table:

PaymentMethod	sum(TotalCharges)
Bank transfer (automatic)	4748279.899999999
Credit card (automatic)	4671593.35
Electronic check	4944903.25
Mailed check	1691392.200000004

At the bottom of the interface, there are icons for file operations and a command bar.

20. The results should match exactly what we had in the Python Script output. At this point we are ready to move on from just doing some data analysis with Spark on Databricks.

## Section 6: Implement Featuring Engineering Techniques to Enhance data for Machine Learning

1. Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. Feature engineering is fundamental to the application of machine learning, and is both difficult and expensive. It is often stated that feature engineering takes roughly 80% of the time from a data scientists day job.
2. The classification goal is to predict whether the customer will leave or churn (**Yes/No**).
3. We can take a look at our distribution of **Churn** by frequency to see what our distribution of churned customers we have in our sample set by executing the following script as seen in the following screenshot:

```
display(df.groupBy('Churn').count())
```

The screenshot shows a Databricks notebook interface. At the top, it says "Cmd 10". Below that, there's a toolbar with "Attached: Machine Learning i...", "File", "View: Code", "Permissions", "Run All", "Clear", and links for "Free trial ends in 11 days. Upgrade to Premium in Azure Portal", "Schedule", "Comments", and "Revision history".

In the code editor, there is one cell:

```
1 display(df.groupBy('Churn').count())
```

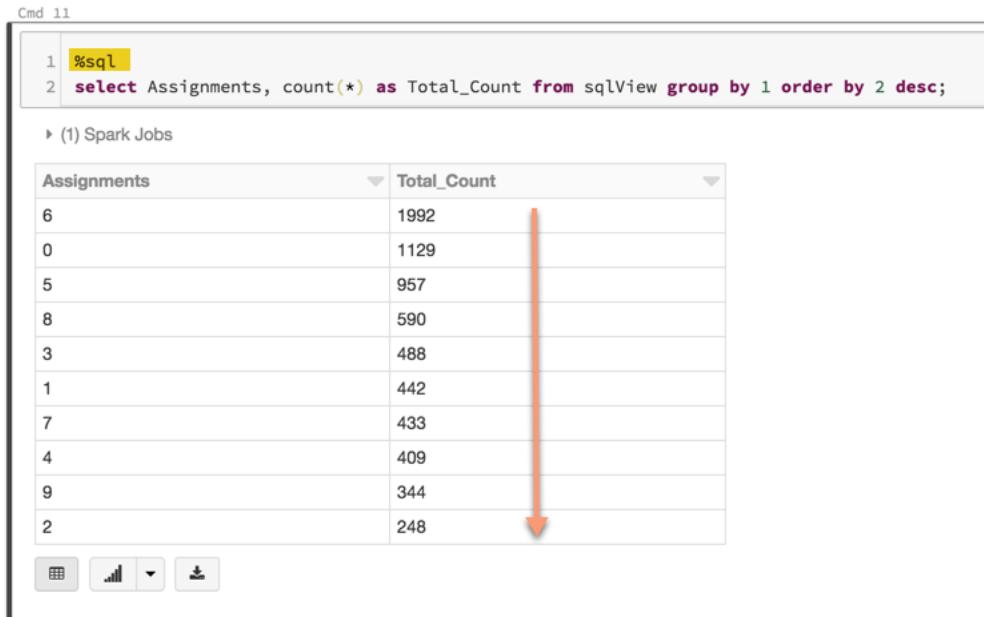
Below the code editor, the results of the cell are displayed in a table:

Churn	count
No	5163
Yes	1869

At the bottom of the interface, there are icons for file operations and a command bar.

4. So, about 27% or **1,869** of the customers that we have were churned versus 73% or **5,163** that remained with the company within the last year. Finally, we can take a look at the clusters that we created over in AML studio and see once again how they were distributed amongst the 10 clusters (0-9) that we created for them using the following script, , as seen in the following screenshot:

```
%sql
select Assignments, count(*) as Total_Count from sqlView group by 1 order by 2 desc;
```



5. We have the ability to see the schema of the data within the DataFrame by executing `df.printSchema()` as seen in the following screenshot:

The screenshot shows a Jupyter Notebook cell titled "Cmd 9". The code block contains one line: `df.printSchema()`. The output shows the schema of the DataFrame `df`:

```

root
|-- customerID: string (nullable = true)
|-- gender: string (nullable = true)
|-- SeniorCitizen: long (nullable = true)
|-- Partner: string (nullable = true)
|-- Dependents: string (nullable = true)
|-- tenure: long (nullable = true)
|-- PhoneService: string (nullable = true)
|-- MultipleLines: string (nullable = true)
|-- InternetService: string (nullable = true)
|-- OnlineSecurity: string (nullable = true)
|-- OnlineBackup: string (nullable = true)
|-- DeviceProtection: string (nullable = true)
|-- TechSupport: string (nullable = true)
|-- StreamingTV: string (nullable = true)
|-- StreamingMovies: string (nullable = true)
|-- Contract: string (nullable = true)
|-- PaperlessBilling: string (nullable = true)
|-- PaymentMethod: string (nullable = true)
|-- MonthlyCharges: double (nullable = true)
|-- TotalCharges: double (nullable = true)
|-- Churn: string (nullable = true)
|-- Assignments: long (nullable = true)

```

At the bottom of the cell, it says "Command took 0.02 seconds -- by ahsherif@microsoft.com at 9/3/2018, 7:37:56 PM on Machine Learning in a Day Clus".

6. While most fields are string, we do have some that are **long** integer and some that are **double**. Other than **customerID**, all of the variables will be our independent variables (**what we will use to predict**) and **Churn** will be our dependent variable (**what we are predicting**).
7. For the purposes of machine learning, these string or categorical variables will need to be encoded to numeric variables. First we will start with the **Churn** column and assign a value of 0 for **No** and 1 for **Yes** using the following script as seen in the following screenshot:

```
from pyspark.sql import functions as F
df = df.withColumn('Churn', F.when(df['Churn']=='Yes', 1).otherwise(0))
```

```
1 from pyspark.sql import functions as F
2 df = df.withColumn('Churn', F.when(df['Churn']=='Yes', 1).otherwise(0))

  df: pyspark.sql.dataframe.DataFrame
    customerID: string
    gender: string
    SeniorCitizen: long
    Partner: string
    Dependents: string
    tenure: long
    PhoneService: string
    MultipleLines: string
    InternetService: string
    OnlineSecurity: string
    OnlineBackup: string
    DeviceProtection: string
    TechSupport: string
    StreamingTV: string
    StreamingMovies: string
    Contract: string
    PaperlessBilling: string
    PaymentMethod: string
    MonthlyCharges: double
    TotalCharges: double
    Churn: integer
    Assignments: long

Command took 0.13 seconds -- by ahsherif@microsoft.com at 9/3/2018, 8:34:14 PM on Machine Learning in a Day Cluster
```

8. `pyspark.sql.functions` help apply SQL-type functionality to DataFrames for manipulation.
9. The **Churn** column is now converted to an **integer** data type from a **string** data type. We can identify all of our string or categorical columns by executing the following script and seeing the output in the following screenshot:

```
categorical_variables = [i[0] for i in df.dtypes if i[1] == 'string']
```

```
Cmd 13
1 categorical_variables = [i[0] for i in df.dtypes if i[1] == 'string']

Command took 0.01 seconds -- by ahsherif@microsoft.com at 9/3/2018, 8:51:36 PM on Machine Learning in a Day Cluster

Cmd 14
1 categorical_variables

Out[35]:
['customerID',
 'gender',
 'Partner',
 'Dependents',
 'PhoneService',
 'MultipleLines',
 'InternetService',
 'OnlineSecurity',
 'OnlineBackup',
 'DeviceProtection',
 'TechSupport',
 'StreamingTV',
 'StreamingMovies',
 'Contract',
 'PaperlessBilling',
 'PaymentMethod']

Command took 0.01 seconds -- by ahsherif@microsoft.com at 9/3/2018, 8:51:44 PM on Machine Learning in a Day Cluster
```

10. We have 16 categorical variables, but only 15 of them are useful from a machine learning perspective as **customerID** does not provide any intrinsic value. Therefore, we can remove it from our list of columns by executing the following script as seen in the following screenshot:

```
categorical_variables = categorical_variables[1:]
categorical_variables
```

```
1 categorical_variables = categorical_variables[1:]
2 categorical_variables

Out[37]:
['gender',
 'Partner',
 'Dependents',
 'PhoneService',
 'MultipleLines',
 'InternetService',
 'OnlineSecurity',
 'OnlineBackup',
 'DeviceProtection',
 'TechSupport',
 'StreamingTV',
 'StreamingMovies',
 'Contract',
 'PaperlessBilling',
 'PaymentMethod']

Command took 0.01 seconds -- by ahsherif@microsoft.com at 9/3/2018, 8:55:35 PM on Machine Learning in a Day Cluster
```

11. **StringIndexer** encodes a string column of labels to a column of label indices. The indices are in [0, numLabels), ordered by label frequencies, so the most frequent label gets index 0. The unseen labels will be put at index numLabels if user chooses to keep them. If the input column is numeric, we cast it to string and index the string values.
12. We import **StringIndexer** to transform each categorical column into a numeric column with a **\_numeric** suffix using the following script:

```
from pyspark.ml.feature import StringIndexer
indexers = []
for categoricalCol in categorical_variables:
    stringIndexer = StringIndexer(inputCol=categoricalCol,
outputCol=categoricalCol+"_numeric")
    indexers += [stringIndexer]
```

```
1 from pyspark.ml.feature import StringIndexer
2 indexers = []
3 for categoricalCol in categorical_variables:
4     stringIndexer = StringIndexer(inputCol=categoricalCol, outputCol=categoricalCol+"_numeric")
5     indexers += [stringIndexer]

Command took 0.11 seconds -- by ahsherif@microsoft.com at 9/3/2018, 9:12:53 PM on Machine Learning in a Day Cluster
```

13. Next, we will apply the transformation on the DataFrame using the following script:

```
models = []
for model in indexers:
    indexer_model = model.fit(df)
    models+=[indexer_model]

for i in models:
    df = i.transform(df)
```

14. The output of the DataFrame should now reveal the newly added columns when executing the `df.printSchema` script as seen in the following screenshot:

```

1 models = []
2 for model in indexers:
3     indexer_model = model.fit(df)
4     models+=[indexer_model]
5
6 for i in models:
7     df = i.transform(df)

(15) Spark Jobs
▶ df: pyspark.sql.DataFrame[customerID: string, gender: string ... 35 more fields]

Command took 2.72 seconds -- by ahsherif@microsoft.com at 9/3/2018, 9:12:54 PM on Machine Learning in a Day Cluster
Cmd 18

1 df.printSchema

Out[22]: <bound method DataFrame.printSchema of DataFrame[customerID: string, gender: string, SeniorCitizen: bigint, Partner: string, Dependents: string, tenure: bigint, PhoneService: string, MultipleLines: string, InternetService: string, OnlineSecurity: string, OnlineBackup: string, DeviceProtection: string, TechSupport: string, StreamingTV: string, StreamingMovies: string, Contract: string, PaperlessBilling: string, PaymentMethod: string, MonthlyCharges: double, TotalCharges: double, Churn: int, Assignments: bigint, gender_numeric: double, Partner_numeric: double, Dependents_numeric: double, PhoneService_numeric: double, MultipleLines_numeric: double, InternetService_numeric: double, OnlineSecurity_numeric: double, OnlineBackup_numeric: double, DeviceProtection_numeric: double, TechSupport_numeric: double, StreamingTV_numeric: double, StreamingMovies_numeric: double, Contract_numeric: double, PaperlessBilling_numeric: double, PaymentMethod_numeric: double]>

Command took 0.01 seconds -- by ahsherif@microsoft.com at 9/3/2018, 10:04:56 PM on Machine Learning in a Day Cluster

```

15. Every column ending with **\_numeric** has a **double** precision data type. We can do a side-by-side comparison of one of the columns, `PaymentMethod`, to see how the **\_numeric** column compares to the string column by executing the following script and seeing the output in the following screenshot:

```
display(df.select('PaymentMethod', 'PaymentMethod_numeric'))
```

1 display(df.select('PaymentMethod', 'PaymentMethod_numeric'))	
(2) Spark Jobs	
PaymentMethod	PaymentMethod_numeric
Electronic check	0
Mailed check	1
Mailed check	1
Bank transfer (automatic)	2
Electronic check	0
Electronic check	0
Credit card (automatic)	3
Mailed check	1
Electronic check	0

Showing the first 1000 rows.

Command took 0.30 seconds -- by ahsherif@microsoft.com at 9/3/2018, 10:46:17 PM on Machine Learning in a Day Cluster

16. We have two columns, **MonthlyCharges** and **TotalCharges**, that need to be normalized from their original values. They are considered continuous variables and not categorical variables.
17. Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information. Normalization is also required for some algorithms to model the data correctly. For example, assume your input dataset contains one column with values ranging from 0 to 1, and another column with values ranging from 10,000 to 100,000. The great difference in the scale of the numbers could cause problems when you attempt to combine the values as features during modeling.
18. Normalization avoids these problems by creating new values that maintain the general distribution and ratios in the source data, while keeping values within a scale applied across all numeric columns used in the model.
19. We can make these transformations using the following script as seen in the output in the following screenshot:

```

from pyspark.sql.functions import min, max

minMonthly = df.agg(min(df.MonthlyCharges)).head()[0]
maxMonthly = df.agg(max(df.MonthlyCharges)).head()[0]

minTotal = df.agg(min(df.TotalCharges)).head()[0]
maxTotal = df.agg(max(df.TotalCharges)).head()[0]

df = df.withColumn('MonthlyCharges_Normalized',
                    (df.MonthlyCharges - minMonthly) / (maxMonthly-minMonthly))
df = df.withColumn('TotalCharges_Normalized',
                    (df.TotalCharges - minTotal) / (maxTotal-minTotal))

```

```

1 from pyspark.sql.functions import min, max
2
3 minMonthly = df.agg(min(df.MonthlyCharges)).head()[0]
4 maxMonthly = df.agg(max(df.MonthlyCharges)).head()[0]
5
6 minTotal = df.agg(min(df.TotalCharges)).head()[0]
7 maxTotal = df.agg(max(df.TotalCharges)).head()[0]
8
9
10 df = df.withColumn('MonthlyCharges_Normalized', (df.MonthlyCharges - minMonthly)/(maxMonthly-minMonthly))
11 df = df.withColumn('TotalCharges_Normalized', (df.TotalCharges - minTotal)/(maxTotal-minTotal))
12

```

▶ (4) Spark Jobs  
▶ df: pyspark.sql.dataframe.DataFrame = [customerID: string, gender: string ... 22 more fields]

Command took 0.83 seconds -- by ahsherif@microsoft.com at 9/4/2018, 10:07:01 AM on Machine Learning in a Day Cluster

20. We can do a side-by-side comparison of the original **MonthlyCharges** column with the newly added **MonthlyCharges\_Normalized** column by executing the following script and seeing the output in the following screenshot:

MonthlyCharges	MonthlyCharges_Normalized
29.85	0.11542288557213931
56.95	0.3850746268656717
53.85	0.35422885572139307
42.3	0.23930348258706466
70.7	0.5218905472636817
99.65	0.809950248756219
89.1	0.7049751243781094
29.75	0.11442786069651742

Showing the first 1000 rows.

Command took 0.21 seconds -- by ahsherif@microsoft.com at 9/4/2018, 10:07:04 AM on Machine Learning in a Day Cluster

21. We have now completed our feature engineering process and we are ready to assign the columns that we will use as the final features for making our machine learning model work as efficiently as possible. The final **features** will be selected using the following script:

```

features = [
    'Assignments',

```

```

'gender_numeric',
'Partner_numeric',
'Dependents_numeric',
'PhoneService_numeric',
'MultipleLines_numeric',
'InternetService_numeric',
'OnlineSecurity_numeric',
'OnlineBackup_numeric',
'DeviceProtection_numeric',
'TechSupport_numeric',
'StreamingTV_numeric',
'StreamingMovies_numeric',
'Contract_numeric',
'PaperlessBilling_numeric',
'PaymentMethod_numeric',

'MonthlyCharges_Normalized',
'TotalCharges_Normalized',

'SeniorCitizen',
'tenure'
]

```

22. The output of the new list of columns can be seen in the following screenshot:

```

1 features

Out[19]:
['Assignments',
 'gender_numeric',
 'Partner_numeric',
 'Dependents_numeric',
 'PhoneService_numeric',
 'MultipleLines_numeric',
 'InternetService_numeric',
 'OnlineSecurity_numeric',
 'OnlineBackup_numeric',
 'DeviceProtection_numeric',
 'TechSupport_numeric',
 'StreamingTV_numeric',
 'StreamingMovies_numeric',
 'Contract_numeric',
 'PaperlessBilling_numeric',
 'PaymentMethod_numeric',
 'MonthlyCharges_Normalized',
 'TotalCharges_Normalized',
 'SeniorCitizen',
 'tenure']

```

23. We will apply **VectorAssembler** to all of our features. VectorAssembler is a transformer that combines a given list of columns into a single vector column. It is useful for combining raw features and features generated by different feature transformers into a single feature vector, in order to train ML models like logistic regression. We can specify which columns will be vectorized by applying the following script:

```

from pyspark.ml.feature import VectorAssembler

feature_vectors = VectorAssembler(
    inputCols = features,
    outputCol = "features")

```

24. We can transform the DataFrame to include the newly created column, **features**, by executing the following script:
- ```

df = feature_vectors.transform(df)

```

25. We can see the output of the field, **features**, by executing `display(df.select('features'))`, as seen in the following screenshot:

A screenshot of a Jupyter Notebook cell. The code entered is `display(df.select('features'))`. Below the code, a section titled '(2) Spark Jobs' shows the output of the `display` command. The output is a list of 1000 rows of vector data, each represented as a list of lists. The first few rows are as follows:

```
[0,20,[0,1,2,4,5,6,8,16,17,19],[6,1,1,1,2,1,1,0.11542288557213931,0.001275098084468036,1]]  
[0,20,[0,6,7,9,13,14,15,16,17,19],[8,1,1,1,2,1,1,0.3850746268656717,0.21586660512347103,34]]  
[0,20,[0,6,7,8,15,16,17,19],[6,1,1,1,1,0.35422885572139307,0.010310408492960998,2]]  
[1,20,[[5,0,0,0,1,2,1,1,0,1,1,0,0,2,1,2,0.23930348258706466,0.21024117239787676,0,45]]]  
[0,20,[0,1,16,17,19],[6,1,0.5218905472636817,0.015330025386568196,2]]  
[0,20,[1,5,9,11,12,16,17,19],[1,1,1,1,1,0.809950248756219,0.09251096238172167,8]]  
[0,20,[0,3,5,8,11,15,16,17,19],[8,1,1,1,1,3,0.7049751243781094,0.22277867528271408,22]]  
[0,20,[0,1,4,5,6,7,14,15,16,17,19],[6,1,1,2,1,1,1,1,0.11442786069651742,0.03266789753057927,10]]  
...
```

The output shows the first 1000 rows of the `features` column.

26. There are only two columns now that are of interest to us: **Churn** and **features**. **Churn** is our label column that we are trying to predict and **features** is our predictor column that is combined with all of our predictor variables in one vector as seen in the following screenshot:

A screenshot of a Jupyter Notebook cell. The code entered is `display(df.select('Churn', 'features'))`. Below the code, a section titled '(2) Spark Jobs' shows the output of the `display` command. The output is a table with two columns: **Churn** and **features**. The **Churn** column contains binary values (0 or 1), and the **features** column contains the same vector data as the previous screenshot. The first few rows are as follows:

| Churn | features                                                                                         |
|-------|--------------------------------------------------------------------------------------------------|
| 0     | [0,20,[0,1,2,4,5,6,8,16,17,19],[6,1,1,1,2,1,1,0.11542288557213931,0.001275098084468036,1]]       |
| 0     | [0,20,[0,6,7,9,13,14,15,16,17,19],[8,1,1,1,2,1,1,0.3850746268656717,0.21586660512347103,34]]     |
| 1     | [0,20,[0,6,7,8,15,16,17,19],[6,1,1,1,1,0.35422885572139307,0.010310408492960998,2]]              |
| 0     | [1,20,[[5,0,0,0,1,2,1,1,0,1,1,0,0,2,1,2,0.23930348258706466,0.21024117239787676,0,45]]]          |
| 1     | [0,20,[0,1,16,17,19],[6,1,0.5218905472636817,0.015330025386568196,2]]                            |
| 1     | [0,20,[1,5,9,11,12,16,17,19],[1,1,1,1,1,0.809950248756219,0.09251096238172167,8]]                |
| 0     | [0,20,[0,3,5,8,11,15,16,17,19],[8,1,1,1,1,3,0.7049751243781094,0.22277867528271408,22]]          |
| 0     | [0,20,[0,1,4,5,6,7,14,15,16,17,19],[6,1,1,2,1,1,1,1,0.11442786069651742,0.03266789753057927,10]] |
| 1     | [0,20,[0,1,2,4,5,6,8,16,17,19],[6,1,1,1,2,1,1,0.11542288557213931,0.001275098084468036,1]]       |

The output shows the first 1000 rows of the `Churn` and `features` columns.

27. One final modification we will make is convert the name **Churn** into **label** as it will better generalize the purpose of that field. We execute the following script, `df = df.withColumnRenamed('Churn', 'label')`, to rename the existing field to **label**. The output of the script can be seen by scrolling down to the location where **Churn** was and now seeing it replaced with **label** as seen in the following screenshot:

```
1 df = df.withColumnRenamed('Churn', 'label')

▶ df: pyspark.sql.dataframe.DataFrame
  MonthlyCharges: double
  TotalCharges: double
  label: integer
  Assignments: long
  MonthlyCharges_Normalized: double
  TotalCharges_Normalized: double
  gender_numeric: double
  Partner_numeric: double
  Dependents_numeric: double
  PhoneService_numeric: double
  MultipleLines_numeric: double
  InternetService_numeric: double
  OnlineSecurity_numeric: double
  OnlineBackup_numeric: double
  DeviceProtection_numeric: double
  TechSupport_numeric: double
  StreamingTV_numeric: double
  StreamingMovies_numeric: double
  Contract_numeric: double
  PaperlessBilling_numeric: double
  PaymentMethod_numeric: double
  ▶ features: udt
```

## Section 7: Apply Classification Supervised Model with Logistic Regression on Azure Databricks

1. In machine learning we usually split our data into two subsets: training data and testing data (sometimes we even split them into three: train, validate and test), and fit our model on the train data, in order to make predictions on the test data. This is used to help solve the problem of overfitting. Overfitting is the problem of fitting your model so well to the data that you have but not taking into consideration future data points. It lacks generality.
2. We split our data into a testing and training dataset using the following script with the output in the following screenshot: `trainDF, testDF = df.randomSplit([0.8, 0.2], seed = 1234)`

```
1 trainDF, testDF = df.randomSplit([0.8, 0.2], seed = 1234)

▶ trainDF: pyspark.sql.dataframe.DataFrame = [customerID: string, gender: string ... 38 more fields]
▶ testDF: pyspark.sql.dataframe.DataFrame = [customerID: string, gender: string ... 38 more fields]

Command took 0.05 seconds -- by ahsherif@microsoft.com at 9/4/2018, 10:53:01 AM on Machine Learning in a Day Cluster
```

3. The randomness is set to **1234** for reproducibility.
4. We can see the number of rows from our original DataFrame assigned to test versus train by executing the following script with the output seen in the following screenshot:

```
print('training data: '+str(trainDF.count()), ', testing data: '+str(testDF.count()))
```

```
1 print('training data: '+str(trainDF.count()), ', testing data: '+str(testDF.count()))

▶ (2) Spark Jobs
  training data: 5640 , testing data: 1392
```

5. Next we want to build out our **logistic regression classification** model pipeline and **fit** it on our training DataFrame using the following script:

```
from pyspark.ml.classification import LogisticRegression
logreg = LogisticRegression(labelCol="label", featuresCol="features", maxIter=10)
LogisticRegressionModel = logreg.fit(trainDF)
```

6. Next we test the model against our test DataFrame, **testDF**, using the following script to create a new DataFrame called **predictedDF**:

```
predictedDF = LogisticRegressionModel.transform(testDF)
```

7. The output of our predicted values along with the original label can be seen by executing the following script with the output in the following screenshot:

```
display(predictedDF.select('label', 'probability', 'prediction'))
```

| label | probability                                    | prediction |
|-------|------------------------------------------------|------------|
| 0     | [1,0],[0.2453414700108091,0.7546585299891908]  | 1          |
| 0     | [1,0],[0.8531850801244731,0.14681491987552686] | 0          |
| 0     | [1,0],[0.9156653394194328,0.0843346605805672]  | 0          |
| 0     | [1,0],[0.5615028194673514,0.4384971805326486]  | 0          |
| 1     | [1,0],[0.5183021325512303,0.4816978674487698]  | 0          |
| 1     | [1,0],[0.6351426553451417,0.36485734465485836] | 0          |
| 1     | [1,0],[0.5640926911816028,0.4359073088183972]  | 0          |
| 0     | [1,0],[0.873286076625449,0.12671392337455098]  | 0          |

Showing the first 1000 rows.

8. In the field of machine learning, a confusion matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class.
9. We can create a confusion matrix to identify False Positives, False Negatives, True Positives, and True Negatives using the following script with the output seen in the following screenshot:

```
display(predictedDF.crosstab('label', 'prediction'))
```

| label_prediction | 0.0 | 1.0 |
|------------------|-----|-----|
| 1                | 171 | 206 |
| 0                | 910 | 105 |

10. We made **910 + 206 = 1,116** correct predictions and **171+105 = 276** false predictions for an overall accuracy of ~ **80%**
11. We can calculate the accuracy of the model by executing the following script and view the output as seen in the following screenshot:

```
from sklearn import metrics

actual = predictedDF.select('label').toPandas()
predicted = predictedDF.select('prediction').toPandas()
print('accuracy score = {}'.format(metrics.accuracy_score(actual, predicted)*100))
```

```
1 from sklearn import metrics
2
3 actual = predictedDF.select('label').toPandas()
4 predicted = predictedDF.select('prediction').toPandas()
5 print('accuracy score = {}'.format(metrics.accuracy_score(actual, predicted)*100))

▶ (2) Spark Jobs
accuracy score = 80.17241379310344
```

12. In addition, the ROC (Receiver Operating Characteristic) Curve or Area Under the Curve is another way to measure the performance of the model. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true-positive rate is also known as sensitivity, recall or probability of detection in machine learning. The false-positive rate is also known as the fall-out or probability of false alarm[1] and can be calculated as  $(1 - \text{specificity})$ .
13. We can calculate the ROC or Area Under the Curve by executing the following script and see the output as seen in the following screenshot:

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator

evaluator = BinaryClassificationEvaluator()
print('ROC score = {}'.format(round(evaluator.evaluate(predictedDF)*100,2)))
```

```
1 from pyspark.ml.evaluation import BinaryClassificationEvaluator
2
3 evaluator = BinaryClassificationEvaluator()
4 print('ROC score = {}'.format(round(evaluator.evaluate(predictedDF)*100,2)))

▶ (4) Spark Jobs
ROC score = 85.32
```

14. We have a high ROC score of **85.32%**. The higher the score or closer it is to 100% the better.

## Section 8: Export predicted DataFrame from Spark on Azure Databricks to Power BI for visualizations

1. We are able to stream our DataFrames from Spark on Azure Databricks to other Business Intelligence tools such as Power BI, Tableau, and Alteryx.
2. First, we will create another view that called **PredictionScoresDF** using the following script:

```
predictedDF.createOrReplaceTempView('PredictionScoresDF')
```

3. Next we will create a table, **PredictionScore**, accessible outside of Spark using the following script:

```
%sql
create table PredictionScore as
select * from PredictionScoresDF;
```

4. We can then see this table, **PredictionScore**, outside of the notebook and within the Data section within Databricks as seen in the following screenshot:

The screenshot shows the Azure Databricks interface. On the left, there's a sidebar with icons for Home, Workspace, Recent, Clusters, Jobs, and Search. The 'Data' icon is highlighted with a red arrow. The main area is titled 'churn classification (Python)'. It shows a 'Tables' section with a table named 'predictionscore' selected. Below it, a code editor window displays two commands:

```

Cmd 37
1 predictedDF.createOrReplaceTempView('PredictionScoresDF')

Command took 0.23 seconds -- by ahsherif@microsoft.com at 9/4/2018, 4:01:37 PM on Machine Learning in a Day Cluster

Cmd 38
1 %sql
2 create table PredictionScore as
3 select * from PredictionScoresDF;

▶ (1) Spark Jobs
OK

Command took 4.89 seconds -- by ahsherif@microsoft.com at 9/4/2018, 4:00:50 PM on Machine Learning in a Day Cluster

```

Cmd 39

Shift+Enter to run [shortcuts](#)

5. When clicking on the table, **predictionscore**, within Databricks, we can view both the **sample data** as well as the **schema** inside of the table as seen in the following screenshot:

The screenshot shows the Azure Databricks interface with the 'Data' sidebar selected. In the main area, a table named 'predictionscore' is selected. The top part shows the 'Schema' for the table:

| col_name         | data_type | comment |
|------------------|-----------|---------|
| customerID       | string    | null    |
| gender           | string    | null    |
| SeniorCitizen    | bigint    | null    |
| Partner          | string    | null    |
| Dependents       | string    | null    |
| tenure           | bigint    | null    |
| PhoneService     | string    | null    |
| MultipleLines    | string    | null    |
| InternetService  | string    | null    |
| OnlineSecurity   | string    | null    |
| OnlineBackup     | string    | null    |
| DeviceProtection | string    | null    |
| TechSupport      | string    | null    |
| StreamingTV      | string    | null    |
| StreamingMovies  | string    | null    |
| Contract         | string    | null    |

The bottom part shows the 'Sample Data' for the table:

| customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | InternetService | OnlineSecurity      | OnlineBackup        | DeviceF  |
|------------|--------|---------------|---------|------------|--------|--------------|---------------|-----------------|---------------------|---------------------|----------|
| 0064-SUDOG | Female | 0             | Yes     | Yes        | 12     | Yes          | No            | No              | No internet service | No internet service | No inter |
| 0164-XAIRP | Female | 0             | No      | No         | 24     | Yes          | No            | No              | No internet service | No internet service | No inter |

6. Next we will click on the **Clusters** icon on the left-hand side of the menu and select our current cluster, **Machine Learning in a Day Cluster**, as seen in the following screenshot:

The screenshot shows the Azure Databricks Clusters page. On the left is a sidebar with icons for Home, Workspace, Recent, Data, Clusters, Jobs, and Search. The main area is titled 'Clusters' and shows a table for 'Interactive Clusters'. A single row is listed: 'Machine Learning in a Day Cluster' (Status: Running, Nodes: 3, Driver: Standard\_DS3\_v2, Worker: Standard\_DS3\_v2, Runtime: 4.2 (includes Apache...), Creator: ahsherif@microsoft..., Actions). A red arrow points from the 'Clusters' icon in the sidebar to the 'Machine Learning in a Day Cluster' row.

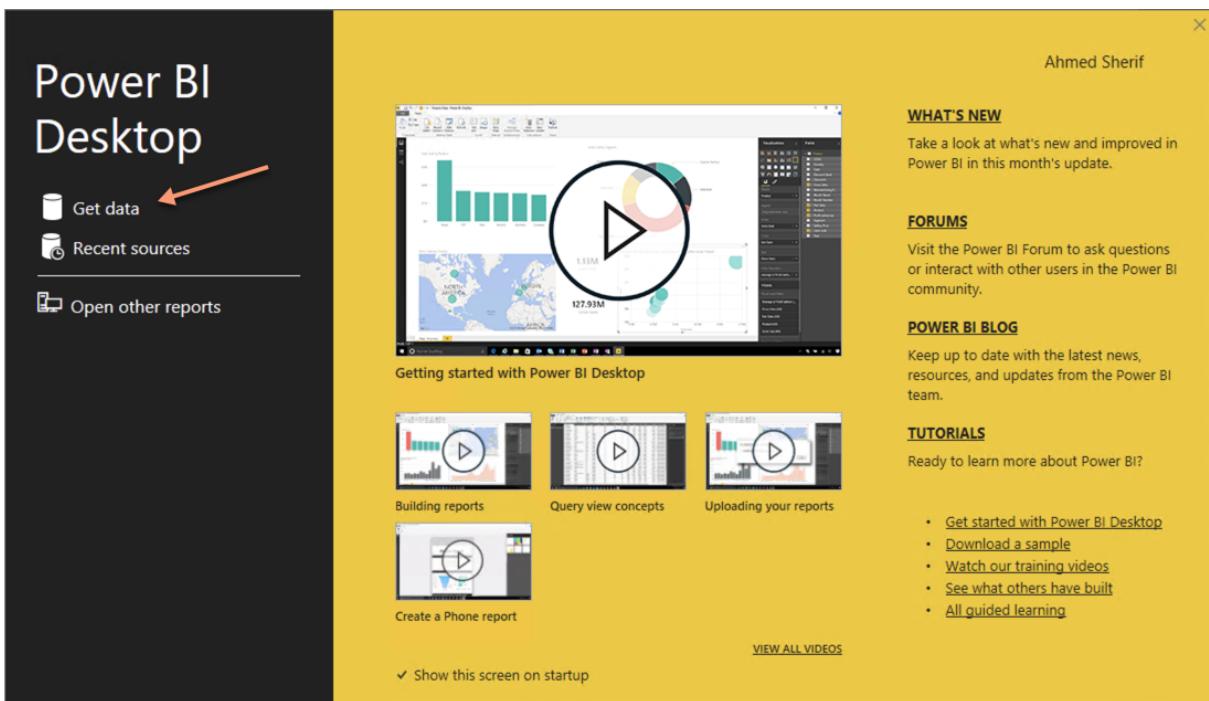
- Once inside the cluster, click on the **JDBC/ODBC** tab and click on the link in the bottom **To Learn More About Connecting your Favorite BI Tools to Databricks**.

The screenshot shows the 'Machine Learning in a Day Cluster' configuration page. The sidebar on the left includes icons for Home, Workspace, Recent, Data, Clusters, Jobs, and Search. The main content area has tabs for Configuration, Notebooks (1), Libraries (0), Event Log, Spark UI, Driver Logs, and Spark Cluster UI - Master. The 'JDBC/ODBC' tab is selected, highlighted with a yellow background. Below it, there are fields for Server Hostname (eastus.azuredatabricks.net), Port (443), Protocol (HTTPS), and HTTP Path (sql/protocolv1/o/2535530955171549/0831-164140-ayes780 (unique) /sql/protocolv1/o/2535530955171549/machine-learning-in-a-day-cluster (alias, not guaranteed unique)). At the bottom of the JDBC URL section, a red arrow points to a yellow button labeled 'Learn more about connecting your favorite BI tool to Databricks.'

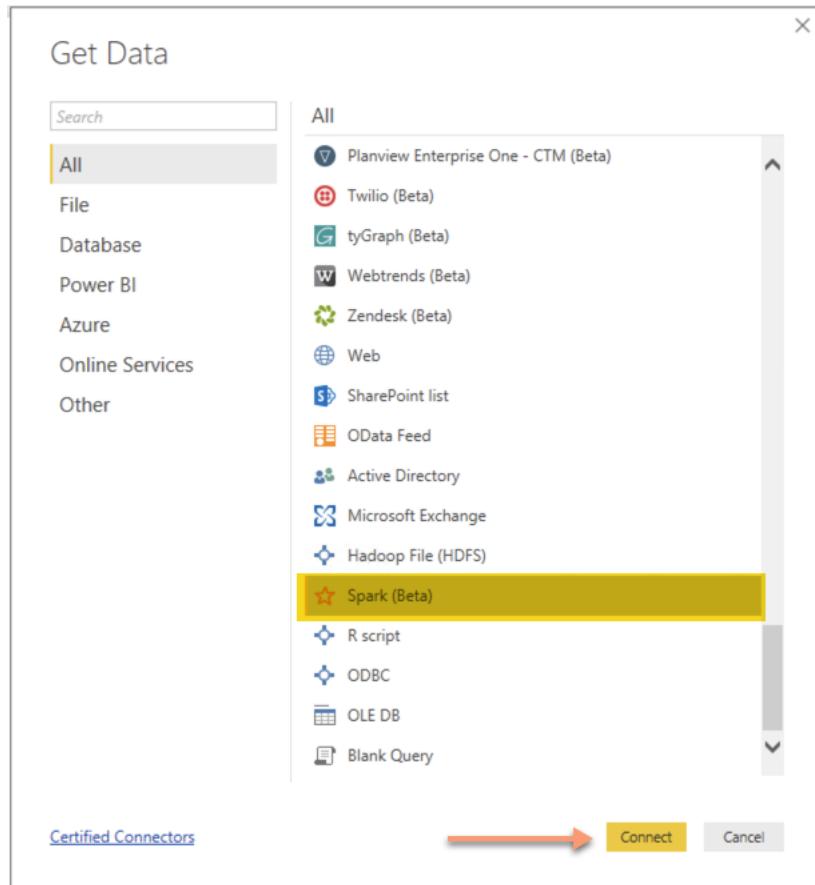
- Select **Power BI** as the **Business Intelligence tool** of choice for connecting to Databricks as seen in the following screenshot:

The screenshot shows the Azure Databricks documentation for Business Intelligence Tools. The left sidebar contains links for User Guide, Getting Started Guide, and various Databricks features like Clusters, Workspaces, Notebooks, and Visualizations. Under Business Intelligence Tools, there's a section for Connecting BI Tools with links for Tableau, Power BI, Alteryx, Looker, and SQL Workbench/J. The 'Power BI' link is highlighted with a yellow box. The main content area has a yellow header 'Business Intelligence Tools' and a paragraph about connecting BI tools to Azure Databricks clusters. Below that is a 'General instructions' section with a bullet point for 'Connecting BI Tools'. A 'Tools' section lists the same five options, with 'Power BI' again highlighted. Navigation buttons '< PREVIOUS' and 'NEXT >' are at the bottom.

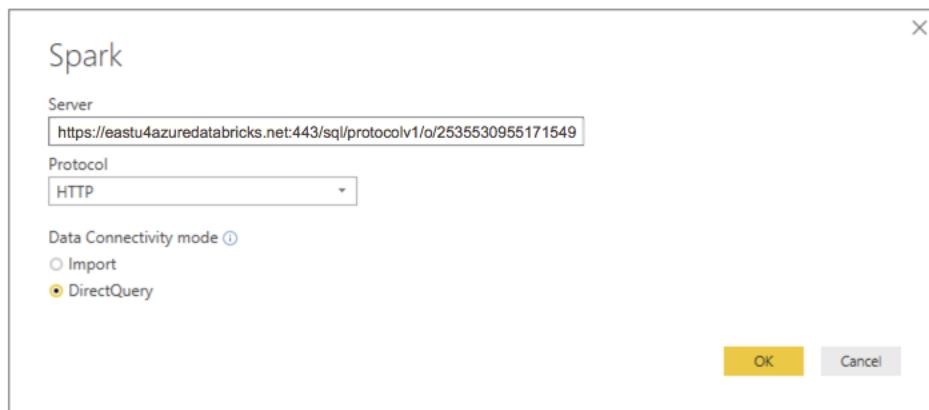
9. Databricks should have step-by-step directions for connecting Tables from Spark to Power BI as found in the following link: <https://docs.azuredatabricks.net/user-guide/bi/power-bi.html>
10. There should be a Power BI desktop installation already available in our Data Science Virtual machine that we can use. When we sign into Power BI, we will first select **Get Data** as seen in the following screenshot:



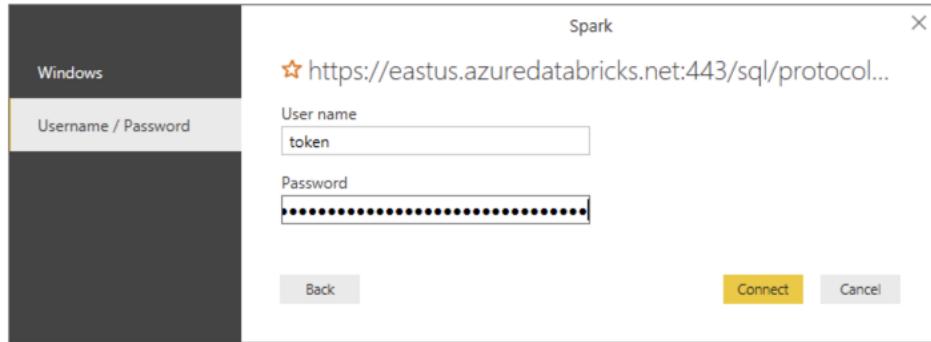
11. Select **Spark (Beta)** and **Connect** as seen in the following screenshot:



12. Enter the **Server** and **Protocol** Type as well as **DirectQuery** as the **data connectivity mode** as seen in the following screenshot:



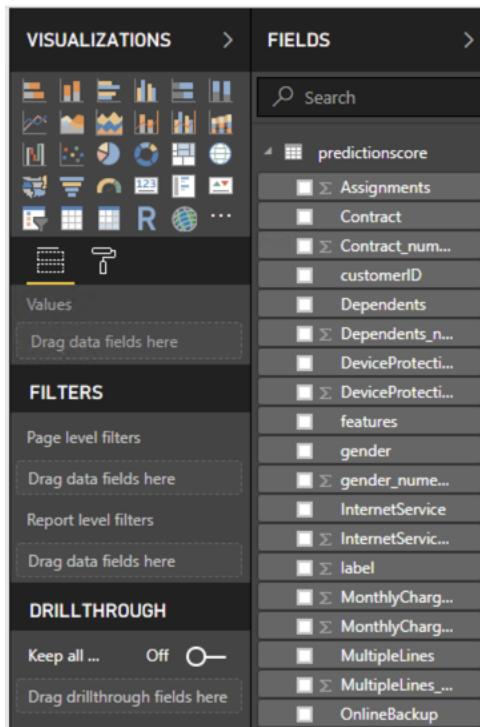
13. Next, Enter the User Name and Password credentials as seen in the following screenshot:



14. We can now view our data from the **predictionscore** table in Spark now viewed directly within Power BI as seen in the following screenshot:

A screenshot of the Power BI Navigator. On the left, there's a list of datasets: "predictionscore" (highlighted in yellow) and "https://eastus.azuredatabricks.net:443/sql/prot...". The main area shows a preview of the "predictionscore" table with columns: customerID, gender, SeniorCitizen, Partner, Dependents, tenure, PhoneService, MultipleLines, InternetService, OnlineSecurity, and OnlineBackup. The preview displays 10 rows of sample data. A note at the bottom says "The data in the preview has been truncated due to size limits." At the bottom right are "Load", "Edit", and "Cancel" buttons.

15. Once we select **Load**, the dataset with all of the columns will be available for us to visualize inside of Power BI as seen in the following screenshot:



16. We can then create a simple visualization of a bar chart using **MonthlyCharges by Assignment** as seen in the following screenshot:

