



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

Machine Learning in a Day student guide	1
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	9
Section 3A: Upload CSV file to Blob Storage Container Manually.....	14
Section 3B (Optional): Upload CSV file to Blob Storage Container with Azure Data Factory V2.	17
Section 4: Build a clustering unsupervised model in Azure Machine Learning Studio.....	37
Section 5: Set up a Spark Cluster on Databricks and connect to Azure Blob Storage Account.....	51
Section 6: Implement Featuring Engineering Techniques to Enhance data for Machine Learning	59
Section 7: Apply Classification Supervised Model with Logistic Regression on Azure Databricks.....	68
Section 8: Export predicted DataFrame from Spark on Azure Databricks to Power BI for visualizations	71

Machine Learning in a Day

student guide

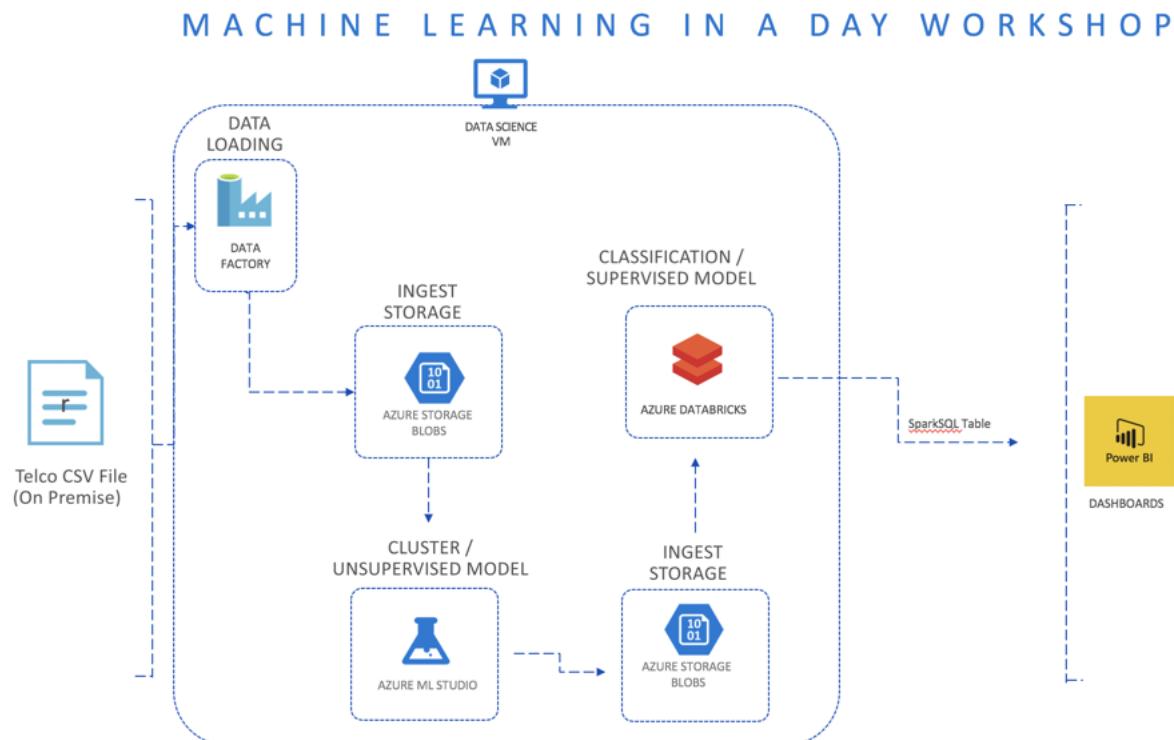
Abstract and learning objectives

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

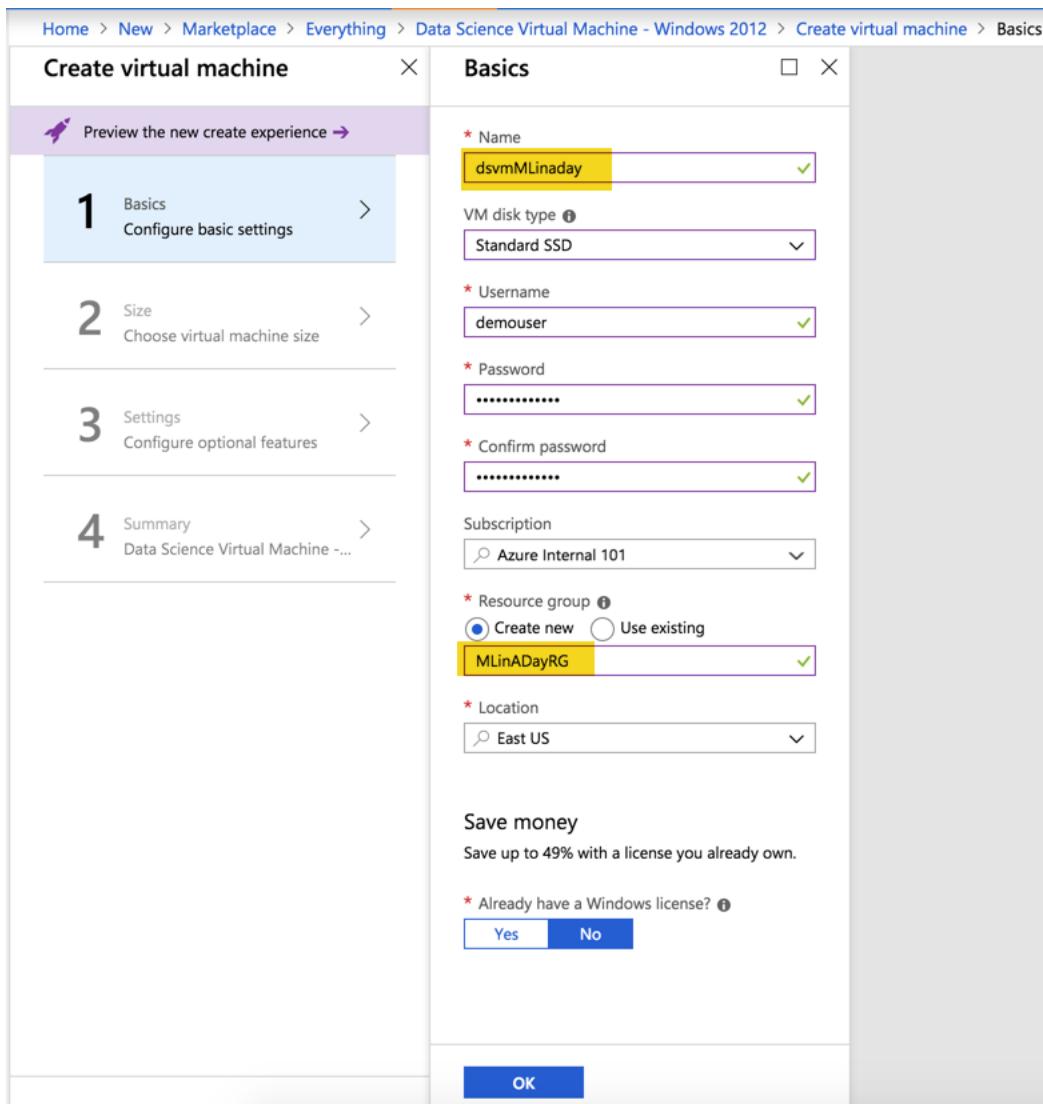


Section 1: Set up a Data Science Virtual Machine (DSVM) on Azure Portal

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 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 a search bar with 'data science virtual machine' and a results table. The first item in the table is 'Data Science Virtual Machine - Windows 2012', which is selected. The details pane on the right provides a description of the VM, mentioning tools like R, Python, and Julia. At the bottom of the details pane is a 'Create' button. Below it, a dropdown menu for 'Select a deployment model' shows 'Resource Manager' selected. A large red arrow highlights the path from the 'Create a resource' button through the search and selection process to the final '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:

SKU	Type	Compute Type	vCPUs	RAM	Data Disks	Max IOPS	Local SSD	Premium	Addition...	USD/Month
B1s	Standard	General purpose	1	1	2	800	4 GB	Yes		\$10.42
B1ms	Standard	General purpose	1	2	2	1600	4 GB	Yes		\$18.30
B2s	Standard	General purpose	2	4	4	3200	8 GB	Yes		\$43.15
B2ms	Standard	General purpose	2	8	4	4800	16 GB	Yes		\$73.66
B4ms	Standard	General purpose	4	16	8	7200	32 GB	Yes		\$146.57
B8ms	Standard	General purpose	8	32	16	10800	64 GB	Yes		\$293.14
D2s_v3	Standard	General purpose	2	8	4	3200	16 GB	Yes		\$139.87
D4s_v3	Standard	General purpose	4	16	8	6400	32 GB	Yes		\$279.74
D8s_v3	Standard	General purpose	8	32	16	12800	64 GB	Yes		\$559.49
D16s_v3	Standard	General purpose	16	64	32	25600	128 GB	Yes		\$1,118.98
D32s_v3	Standard	General purpose	32	128	32	51200	256 GB	Yes		\$2,237.95
D64s_v3	Standard	General purpose	64	256	32	80000	512 GB	Yes		\$4,475.90
E2s_v3	Standard	Memory optimized	2	16	4	3200	32 GB	Yes		\$167.40

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.

Select

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:

Network Security Group *?*
Basic Advanced

* Network security group (firewall) *?* (new) dsvmMLinaDay-nsg

Extensions
No extensions

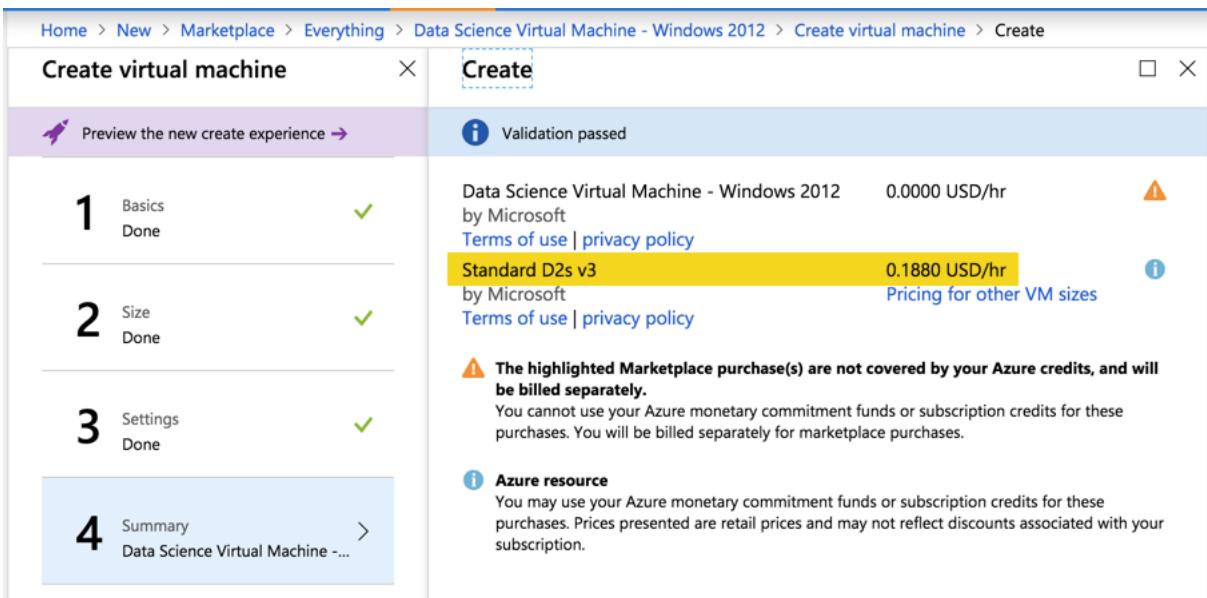
Auto-shutdown
Enable auto-shutdown *?*
 Off On

Shutdown time *?*
7:00:00 PM

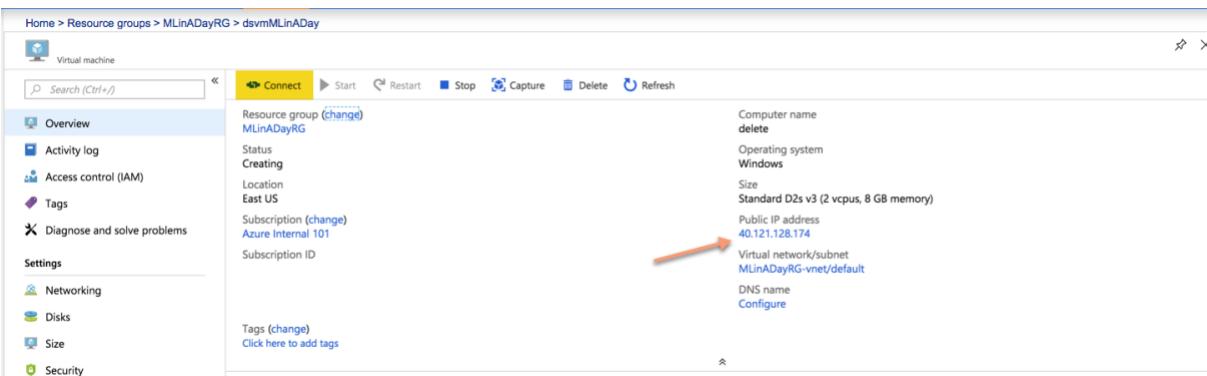
Time zone *?*
(UTC-05:00) Eastern Time (US & Canada)

Notification before shutdown *?*
 Off On

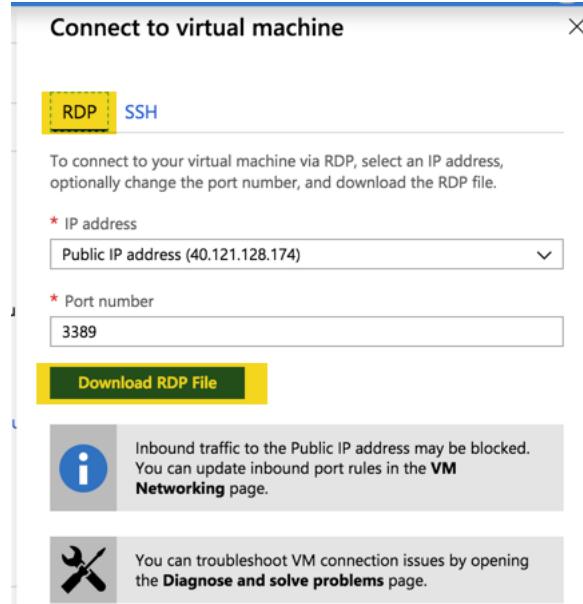
6. Click **OK** on the summary section and confirm everything that was selected is accurate and let the provisioning process of the DSVM begin by clicking on **Create** as seen in the following screenshot:



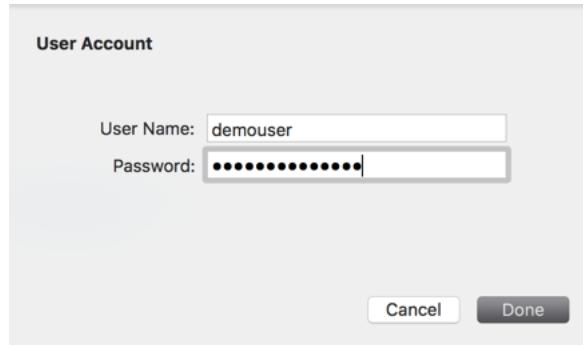
- Once the provisioning is complete, you can go to your resource and view your DSVM overview and obtain your **Public IP address** as seen in the following screenshot:



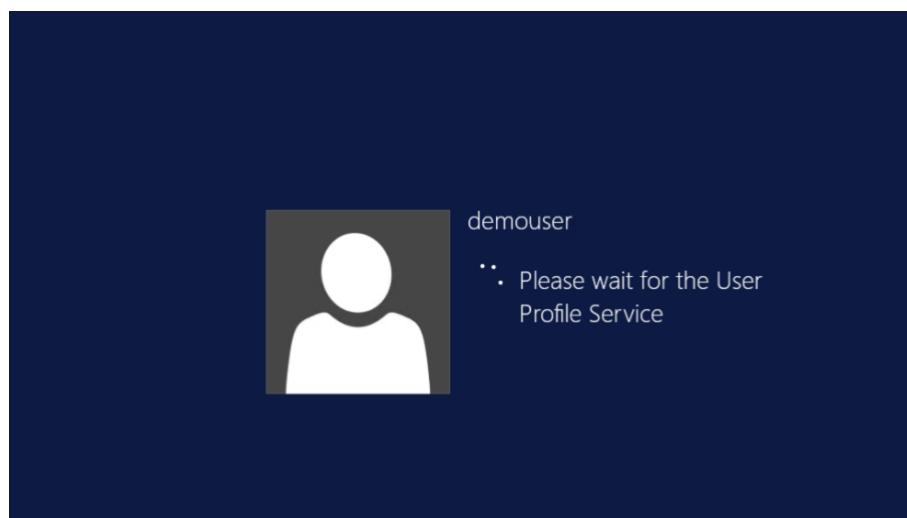
- Make sure that your DSVM is currently running and click on the **Connect** button to enter your virtual machine using **RDP** as seen in the following screenshot:



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:

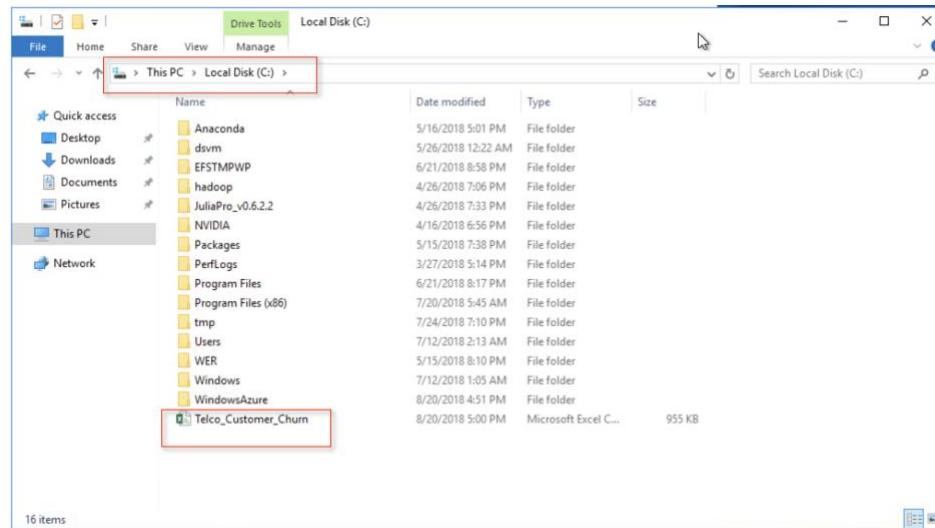


11. Once you log in using your credentials using the RDP approach, you should see the following remote server for the data science virtual machine as seen in the following screenshot:

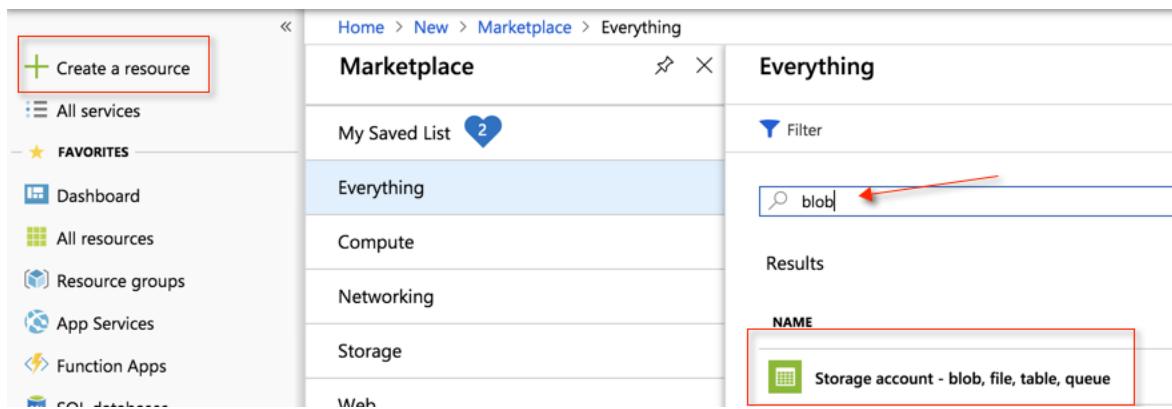


Section 2: Download Telco dataset and create Blob Storage Account and Container access

1. The Telco Churn Dataset ([TELCO_CUSTOMER_CHURN.CSV](https://blobmlinadahmed.blob.core.windows.net/datasources/Telco_Customer_Churn.csv)) can be downloaded from the following link:
https://blobmlinadahmed.blob.core.windows.net/datasources/Telco_Customer_Churn.csv
2. 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:



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):

Create storage account

Basics Advanced Tags Review + create

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below. [Learn more](#)

PROJECT DETAILS

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

* Subscription: Azure Internal 101
 Resource group: MLinADayRG
[Create new](#)

INSTANCE DETAILS

The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using the classic deployment model instead. [Choose classic deployment model](#)

* Storage account name: blobmlinaday
 Location: East US
 Performance: Standard (selected)
 Account kind: StorageV2 (general purpose v2)
 Replication: Locally-redundant storage (LRS)
 Access tier (default): Cool (selected)

Review + create Previous Next : Advanced >

5. Ensure that the Basics setting includes **Standard** Performance, **StorageV2** Account Kind, and **Cool** Access Tier.
6. Continue on to the Advanced Section of the settings and ensure that **secure transfer required** has been disabled as seen in the following screenshot:

Dashboard > New > Marketplace > Everything > Storage account - blob, file, table, queue > Create storage account

Create storage account

Basics Advanced Tags Review + create

SECURITY

Secure transfer required: Disabled Enabled

VIRTUAL NETWORKS

Allow access from: All networks Selected network
All networks will be able to access this storage account. [Learn more](#)

DATA LAKE STORAGE GEN2 (PREVIEW)

Hierarchical namespace: Disabled Enabled

7. Next we can just click on **Review and Create** and ensure that our validation has passed as seen in the following screenshot:

The screenshot shows the 'Create storage account' wizard in the Azure portal. At the top, a green bar indicates 'Validation passed'. Below it, the 'Review + create' tab is selected. The configuration details are divided into 'BASICS' and 'ADVANCED' sections:

Section	Setting	Value
BASICS	Subscription	Azure Internal 101
	Resource group	MLinADayRG
	Location	East US
	Storage account name	blobmlinaday
	Deployment model	Resource manager
	Account kind	StorageV2 (general purpose v2)
	Replication	Locally-redundant storage (LRS)
	Performance	Standard
ADVANCED	Access tier (default)	Cool
	Secure transfer required	Disabled
	Allow access from	All networks
	Hierarchical namespace	Disabled

At the bottom, there are 'Create', 'Previous', and 'Next' buttons, along with a link to 'Download a template for automation'.

8. We can then just click on **Create** and begin the deployment process of the blob storage.
9. While the deployment is underway we can check the progress as seen in the following screenshot:

The screenshot shows the Azure Notifications center. It displays a single notification: 'Deployment in progress...' with a status of 'Running'. The notification message states: 'Deployment to resource group 'MLinADayRG' is in progress.' and was 'a few seconds ago'.

10. 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:

The screenshot shows the 'Notifications' page with a single notification: 'Deployment succeeded' at 1:27 PM. The message states: 'Deployment 'Microsoft.StorageAccount-20180820132637' to resource group 'MLinADayRG' was successful.' Below the message are two buttons: 'Go to resource' and 'Pin to dashboard'. A red arrow points to the 'Go to resource' button.

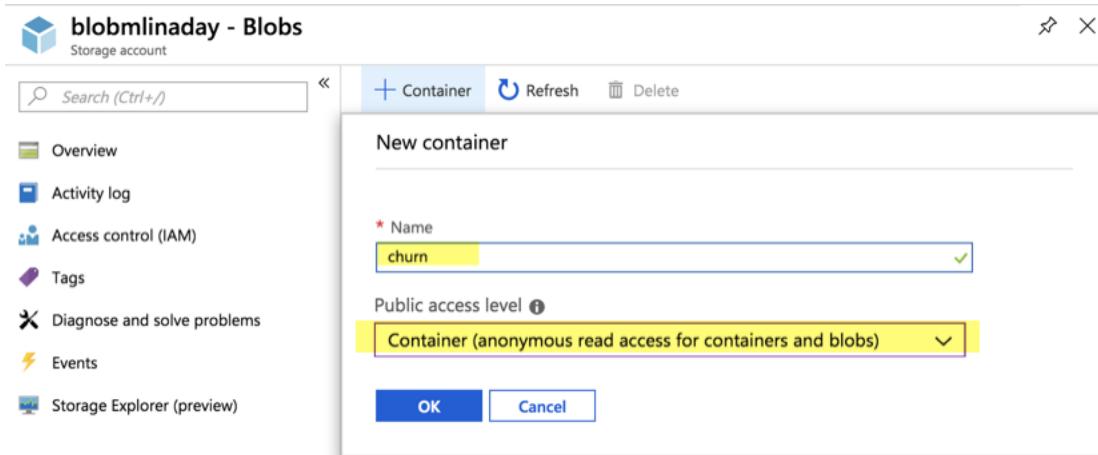
11. Once inside the resource, select **Blobs** under the **Services** section as seen in the following screenshot:

The screenshot shows the 'blobmlinaday' storage account overview page. In the 'Services' section, the 'Blobs' service is highlighted with a red arrow pointing to its icon. The 'Blobs' service is described as 'REST-based object storage for unstructured data'.

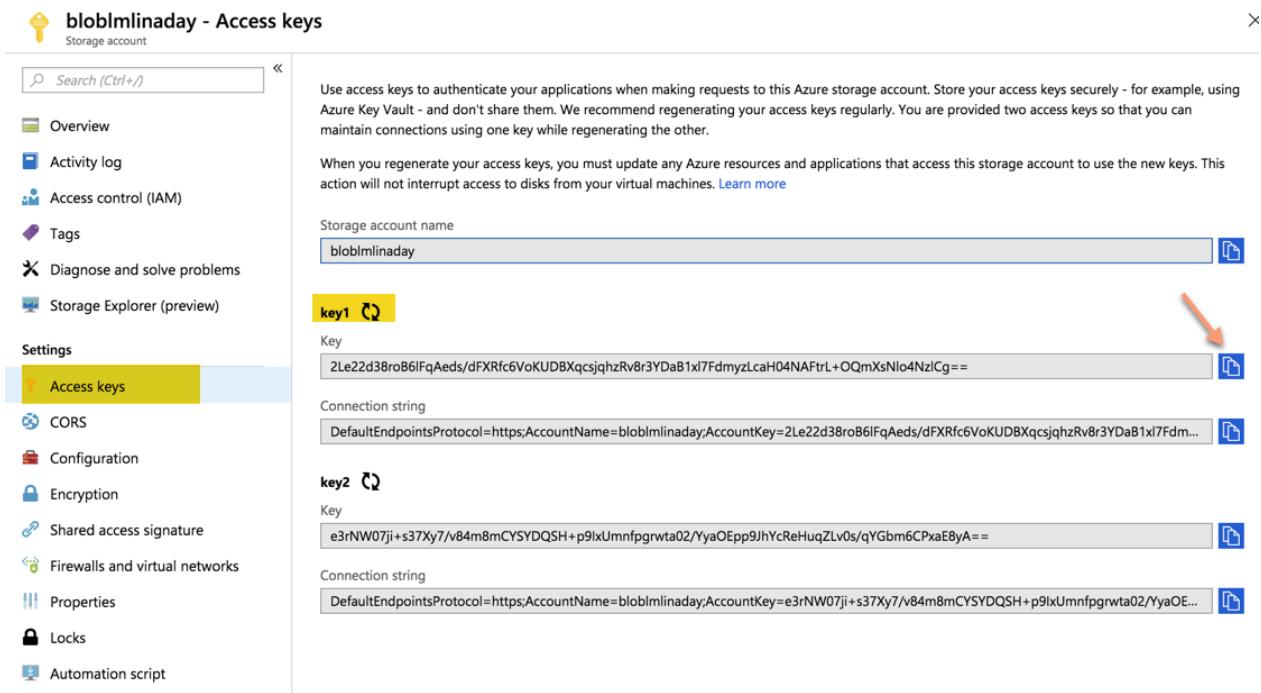
12. ****Important Step **** Ensure that your blob account has the following setting configured to ensure that **Secure Transfer Required** is **Disabled** by selecting Configuration as seen in the following screenshot:

The screenshot shows the 'blobmlinaday - Configuration' page. In the 'Configuration' section, the 'Secure transfer required' setting is shown as a toggle switch. The 'Disabled' option is selected, highlighted with a yellow box and a red arrow. Other settings visible include 'Performance' (Standard), 'Replication' (Locally-redundant storage (LRS)), and 'Azure Active Directory authentication for Azure Files (Preview)' (Enabled).

13. 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.**
14. 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:



15. 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:



16. Once the container has been successfully created, we can move onto the next section of adding data to the container using a couple of different options.

Section 3A: Upload CSV file to Blob Storage Container Manually

- ***This section will manually upload our CSV file to blob storage without using an ETL tool like Azure Data Factory V2. If you wish to use ADF V2 instead, skip this section and move onto Section 3B. Otherwise, continue on with this section.***
- Access your blob storage container once again by selecting **Storage Accounts** on the left-hand side of the Portal menu, selecting your Blob account, and finally selecting **Blobs** under **Blob Service** as seen in the following screenshot:

The screenshot shows the Microsoft Azure portal interface. On the left, the navigation menu is open, with 'Storage accounts' selected under the 'Load balancers' category. In the center, the 'Storage accounts' blade is displayed, showing a list of storage accounts. One account, 'blobmlinaday', is selected and highlighted with a yellow box. To the right of the account list, there's a detailed view of the selected account, including its name, status (Primary: Available), location (East US), and account kind (StorageV2 (general purpose v2)). Below this, there are sections for 'Services' (Blobs, Files, Tables, Queues) and 'Monitoring'. The 'Blobs' service section is also highlighted with a yellow box.

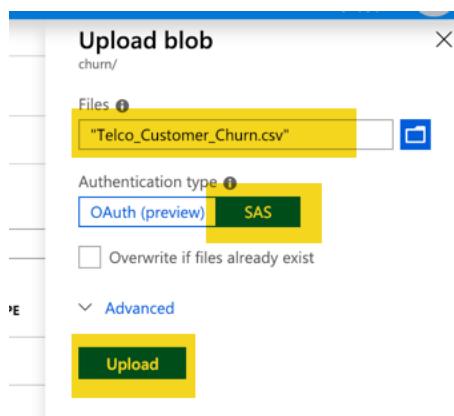
- Select your **container** which should be called churn as seen in the following screenshot:

The screenshot shows the 'blobmlinaday - Blobs' blade. On the left, the 'Blobs' item under the 'Blob service' section is highlighted with a yellow box. The main area displays a table of containers. One container, named 'churn', is selected and highlighted with a yellow box. The table columns include NAME, LAST MODIFIED, PUBLIC ACCESS ..., and LEASE STATE. The 'churn' container was last modified on 11/25/2018, 11:20:31 AM, has 'Container' public access, and is in an 'Available' lease state.

- Once inside your container, select the upload icon to begin the uploading of your csv spreadsheet as seen in the following screenshot:

The screenshot shows the Azure Storage Accounts interface for the 'churn' container. On the left, there's a sidebar with options like Overview, Access Control (IAM), Settings, Access policy, Properties, Metadata, and Editor (preview). The main area shows a search bar, a toolbar with Upload, Refresh, Delete, Acquire lease, Break lease, View snapshots, and Create snapshot, and a table header for NAME, MODIFIED, ACCESS TIER, and BLOB TYPE. A message says 'No blobs found.' Below the table is a search bar for blobs by prefix.

- Select your file from your local hard drive and use **SAS authentication type**. Next select Upload as seen in the following screenshot:



- You should now see your CSV file in your container as seen in the following screenshot:

The screenshot shows the 'churn' container page again. The table lists the uploaded file 'Telco_Customer_Churn.csv' with details: Modified on 11/25/2018, 11:28:06 AM, Access Tier Cool (Inferred), Blob type Block blob, Size 954.59 KB, and Lease state Available. There's also a 'Show deleted blobs' checkbox at the top right.

- Your file should be ~ 955 kB. If you click on your CSV file and select **Edit Blob**, you can preview your actual data that is in the CSV file within the **Blob** container as seen in the following screenshot:

The screenshot shows the Azure Storage Blob service interface. On the left, there's a sidebar with 'Upload', 'Refresh', 'More', 'Location:churn', and a search bar for blobs. Below that is a 'NAME' section with a list containing 'Telco_Customer_Churn.csv'. The main area is titled 'Telco_Customer_Churn.csv' and shows the file content as a table. The first few rows of data are:

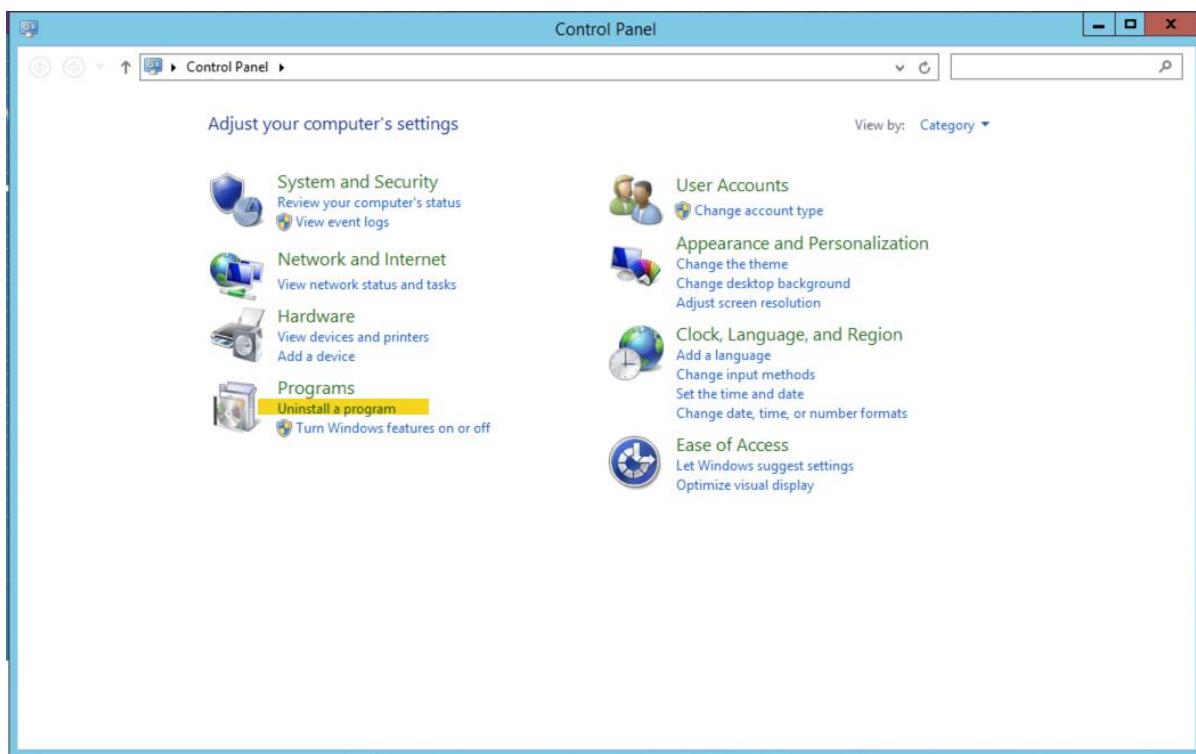
	CustomerID	gender	SeniorCitizen	Partner	Dependents	Tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	Churn	
1	7590-VHVEG	Female	0	Yes	No	1	No	No	phone service	DSL	No	Yes	No	No	No	Month-to-month	Yes	Electronic check	Yes
2	5575-GNVDDE	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes	No	No	No	One year	No	Mailed check	56.95
3	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	No	No	No	No	Month-to-month	Yes	Mailed check	53.85
4	7795-CFOCW	Male	0	No	No	45	No	No	phone service	DSL	Yes	No	Yes	Yes	No	One year	No	Bank transfer (automatic)	108.15
5	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	No	No	No	No	No	Month-to-month	Yes	Electronic check	70.7
6	9305-CDSKC	Female	0	No	No	8	Yes	Yes	Fiber optic	No	No	Yes	No	Yes	Yes	Month-to-month	Yes	Electronic check	1452-KIOVK
7	1452-KIOVK	Male	0	No	No	22	Yes	Yes	Fiber optic	No	Yes	No	Yes	No	No	Month-to-month	Yes	Credit card (automatic)	6713-OKOMC
8	6713-OKOMC	Female	0	No	No	10	No	No	phone service	DSL	Yes	No	No	No	No	Month-to-month	No	Mailed check	29
9	7892-POOKP	Female	0	Yes	No	28	Yes	Yes	Fiber optic	No	No	Yes	Yes	Yes	Yes	Month-to-month	Yes	Electronic check	6388-TABGU
10	6388-TABGU	Male	0	No	Yes	62	Yes	No	DSL	Yes	Yes	No	No	No	No	One year	No	Bank transfer (automatic)	56.15
11	9763-GRSKD	Male	0	Yes	Yes	13	Yes	No	DSL	Yes	No	No	No	No	No	Month-to-month	Yes	Mailed check	49.95
12	7469-LKBCI	Male	0	No	No	16	Yes	No	No	No	No	No	No	No	No	Month-to-month	Yes	No internet service	8891-TTVAZ
13	8891-TTVAZ	Male	0	Yes	No	58	Yes	Yes	Fiber optic	No	No	Yes	Yes	Yes	Yes	One year	No	Credit card (automatic)	0280-XJGEK
14	0280-XJGEK	Male	0	No	No	49	Yes	Yes	Fiber optic	No	Yes	Yes	Yes	Yes	Yes	Month-to-month	Yes	Bank transfer (automatic)	5129-JLPIS
15	5129-JLPIS	Male	0	No	No	25	Yes	No	Fiber optic	Yes	No	Yes	Yes	Yes	Yes	Month-to-month	Yes	Electronic check	3655-SNOYZ
16	3655-SNOYZ	Female	0	Yes	Yes	69	Yes	Yes	Fiber optic	Yes	Yes	Yes	Yes	Yes	Yes	Two year	No	Credit card (automatic)	8191-XWSZG
17	8191-XWSZG	Female	0	No	No	52	Yes	No	No	No	No	No	No	No	No	No	No	No internet service	9959-WOFGKT
18	9959-WOFGKT	Male	0	No	Yes	71	Yes	Yes	Fiber optic	Yes	Yes	Yes	Yes	Yes	Yes	Two year	No	Bank transfer (automatic)	4190-MFLUN
19	4190-MFLUN	Female	0	Yes	Yes	10	Yes	No	DSL	No	No	Yes	Yes	No	No	Month-to-month	No	Credit card (automatic)	4183-HYFRB
20	4183-HYFRB	Female	0	No	No	21	Yes	No	Fiber optic	No	Yes	Yes	No	No	No	Month-to-month	Yes	Electronic check	8779-QRMWV
21	8779-QRMWV	Male	1	No	No	1	No	No	phone service	DSL	No	No	Yes	No	No	Month-to-month	Yes	Electronic check	1680-VDCWN
22	1680-VDCWN	Male	0	Yes	No	12	Yes	No	No	No	No	No	No	No	No	No	No	No internet service	1866-JKSGK
23	1866-JKSGK	Male	0	No	No	1	Yes	No	No	No	No	No	No	No	No	No	No	No internet service	3638-WEABW
24	3638-WEABW	Female	0	Yes	No	58	Yes	Yes	DSL	No	Yes	Yes	No	No	No	Two year	Yes	Credit card (automatic)	6322-HRPFM
25	6322-HRPFM	Male	0	Yes	Yes	49	Yes	No	DSL	Yes	Yes	No	Yes	No	No	Month-to-month	No	Credit card (automatic)	6865-JZNKO
26	6865-JZNKO	Female	0	No	No	30	Yes	No	DSL	Yes	Yes	No	No	No	No	Month-to-month	Yes	Bank transfer (automatic)	
27																			

At the bottom right, there are buttons for 'Edit blob', 'Generate SAS', 'Csv', and 'Download'.

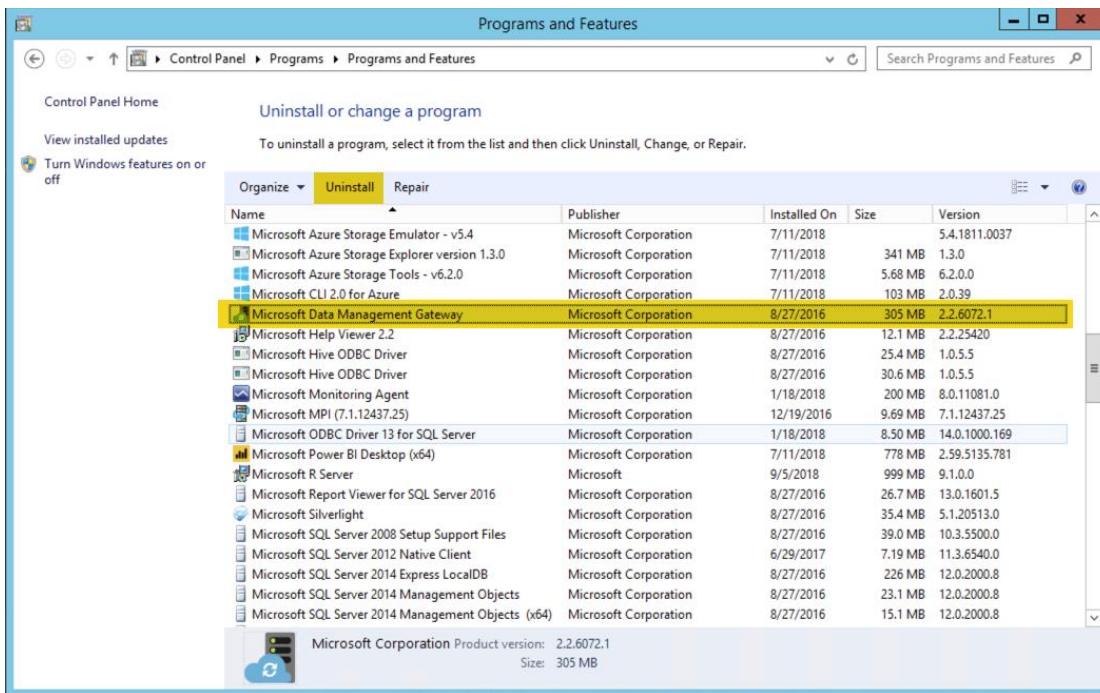
8. We have now confirmed our file is in our Blob account and we can continue onto Section 4.

Section 3B (Optional): Upload CSV file to Blob Storage Container with Azure Data Factory V2.

1. **Continue with this approach if you did not proceed with Section 3A. Otherwise, you can move onto Section 4.**
2. 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:



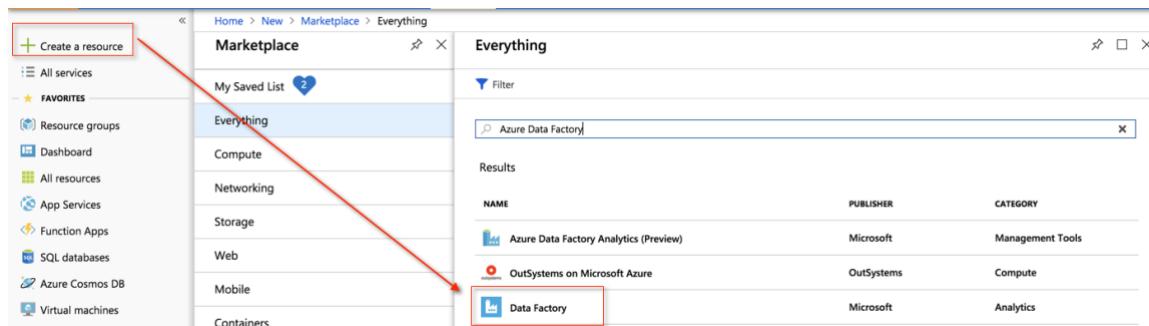
3. Next we will go ahead and **Uninstall Microsoft Data Management Gateway** as seen in the following screenshot:



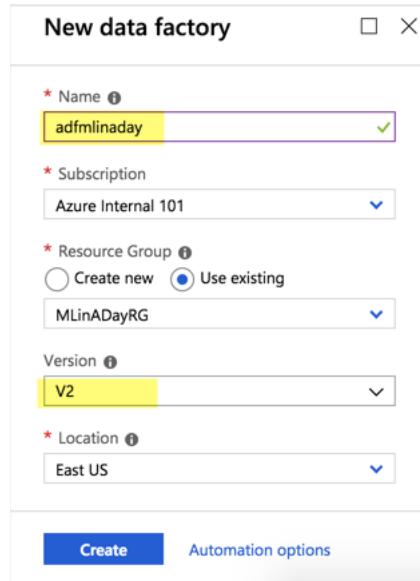
4. Select **Yes** to get rid of all data as seen in the following screenshot:



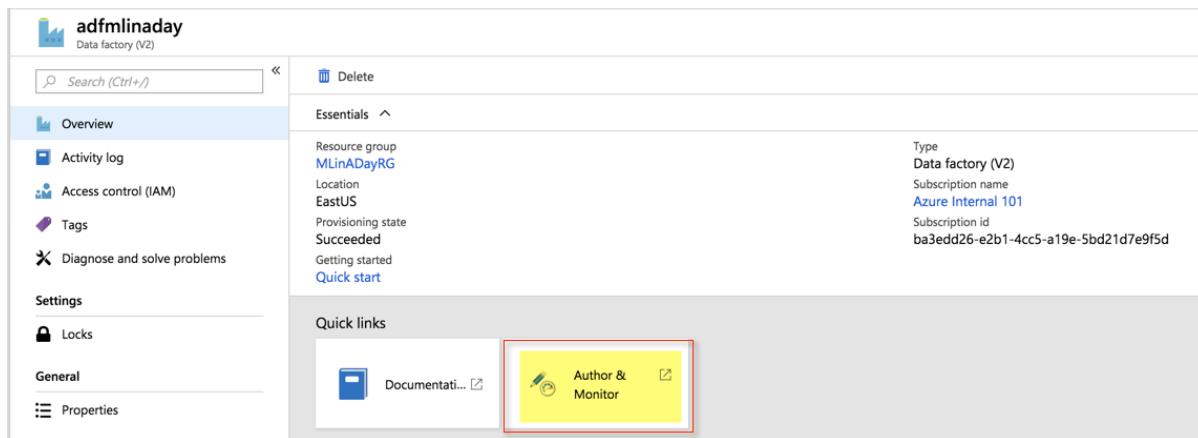
5. As we move forward in this section, we will require accessing the Azure Portal (<https://portal.azure.com>) inside of a browser within the data science virtual machine to set up **Azure Data Factory V2**.
6. Once again, as we did with the Blob Storage as well as with the Data Science Virtual Machine, 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:



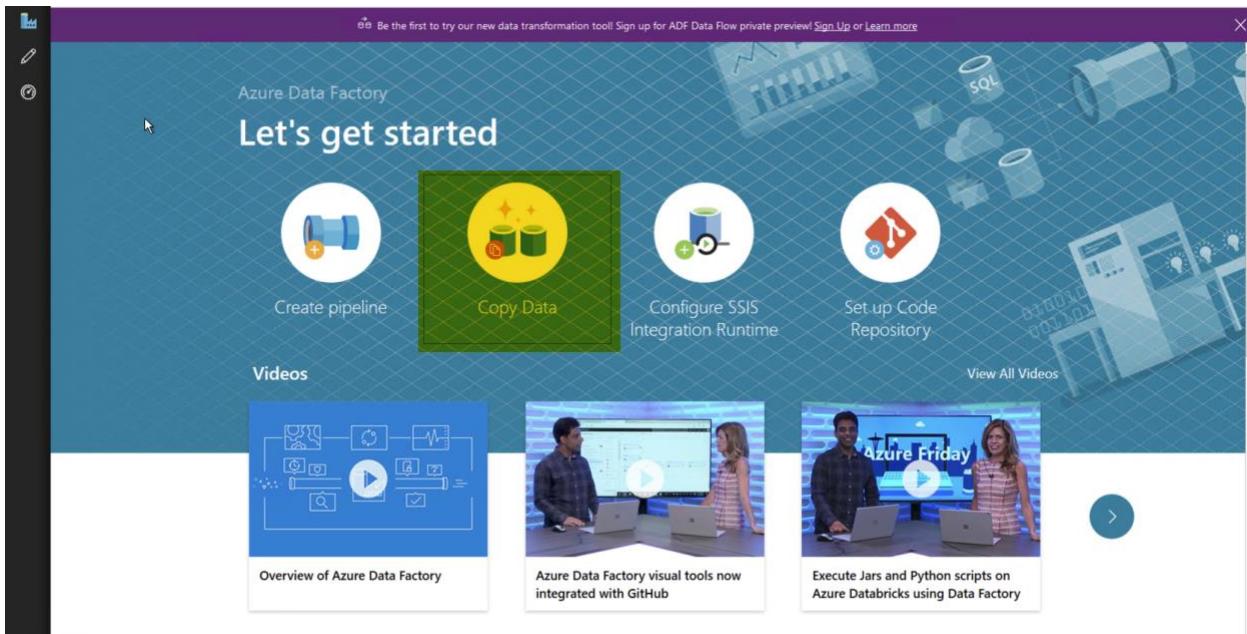
7. Once the resource is created, we will want to specify the following options and then click on **Create**:



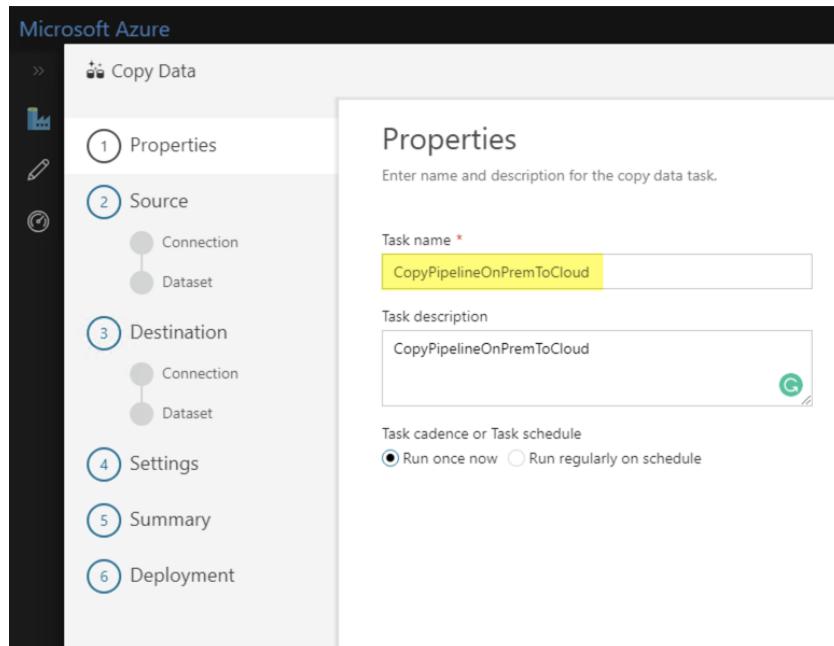
8. Once the resource is completed, we can go to the resource and click on the icon **Author & Monitor** as seen in the following screenshot:



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



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



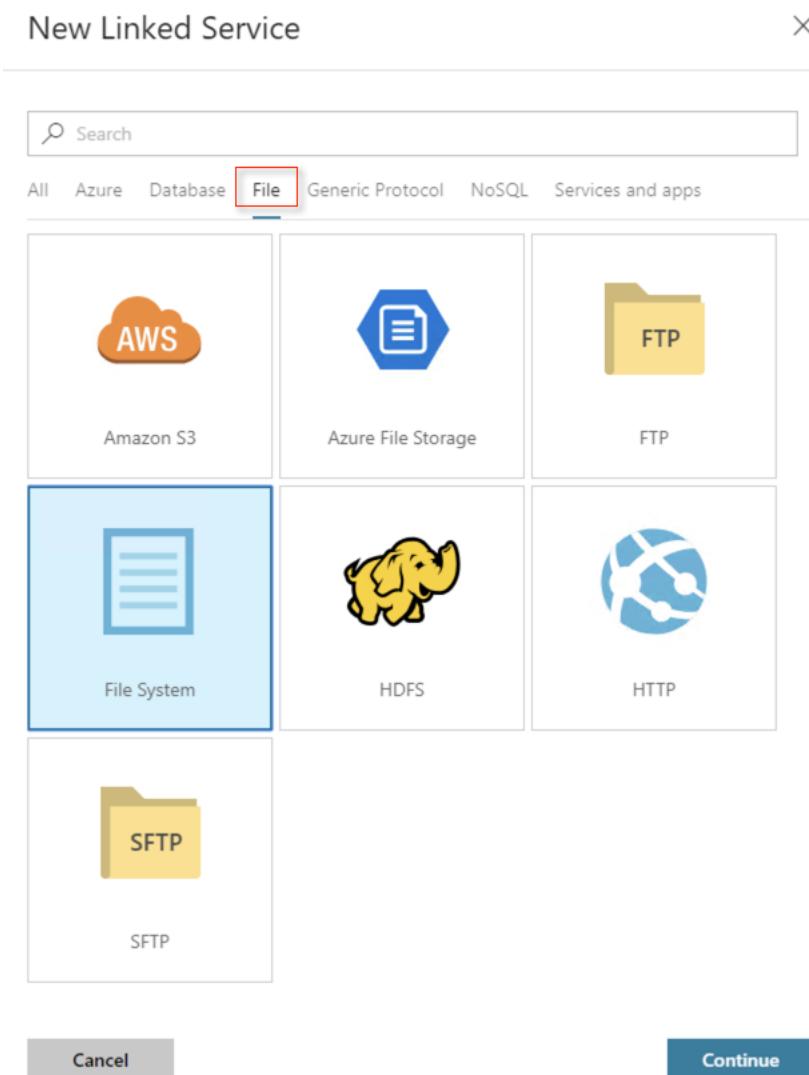
11. We named the task **CopyPipelineOnPremToCloud**. The next step is to specify a data source. Since this source will be on-premise on our virtual machine, we will need to set up + **Create a new connection** as seen in the following screenshot:

Source data store

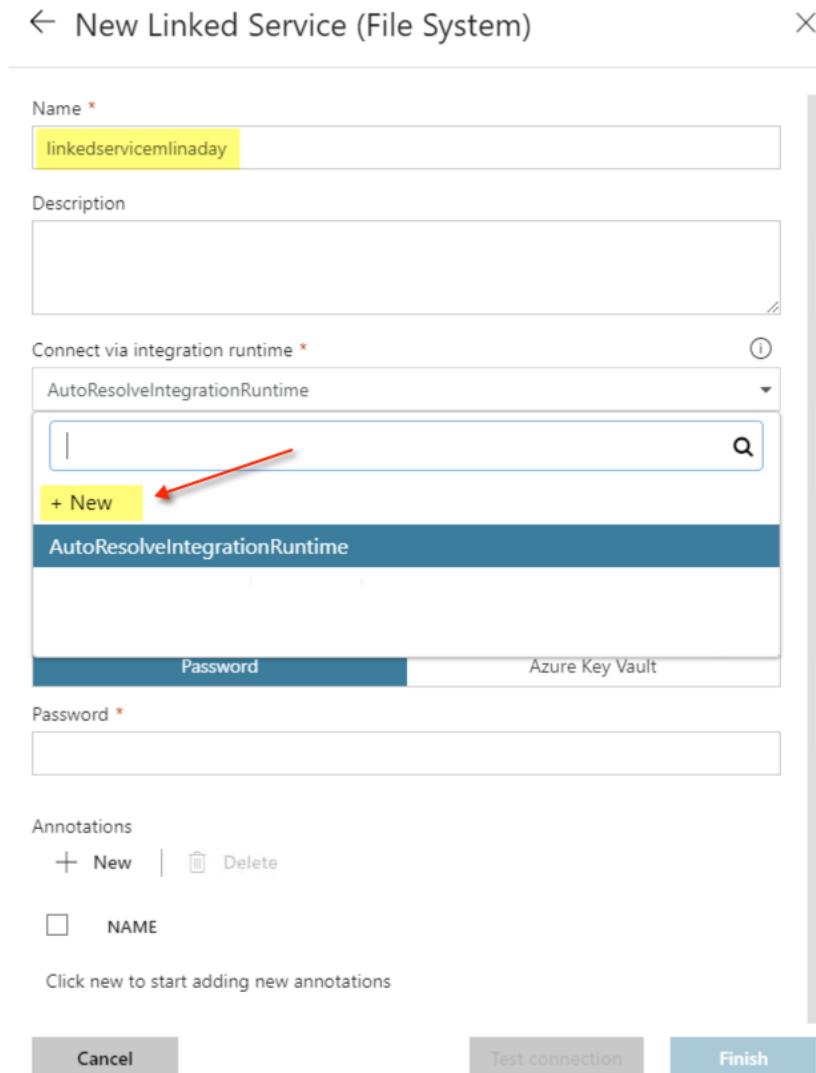
Specify the source data store for the copy task. You can use an existing data store connection or specify a new data store.

The screenshot shows a search interface for data stores. At the top, there are tabs: All, Azure, Database, File, Generic Protocol, NoSQL, and Services and apps. The 'File' tab is highlighted with a red border. Below the tabs is a search bar with the placeholder 'All'. To the right of the search bar is a button labeled 'Filter by name'. Further to the right is a yellow button with a plus sign and the text '+ Create new connection'.

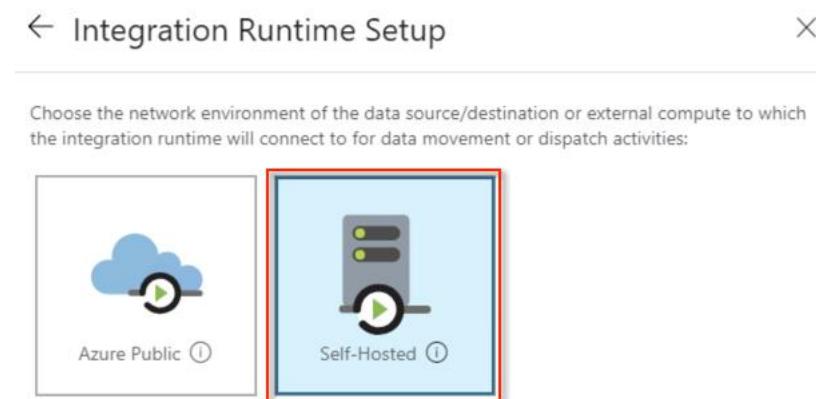
12. Select **File** and then **File System** for a **new Linked Service** between on-premise and in Azure. Then select **Continue** as seen in the following screenshot:



13. 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:

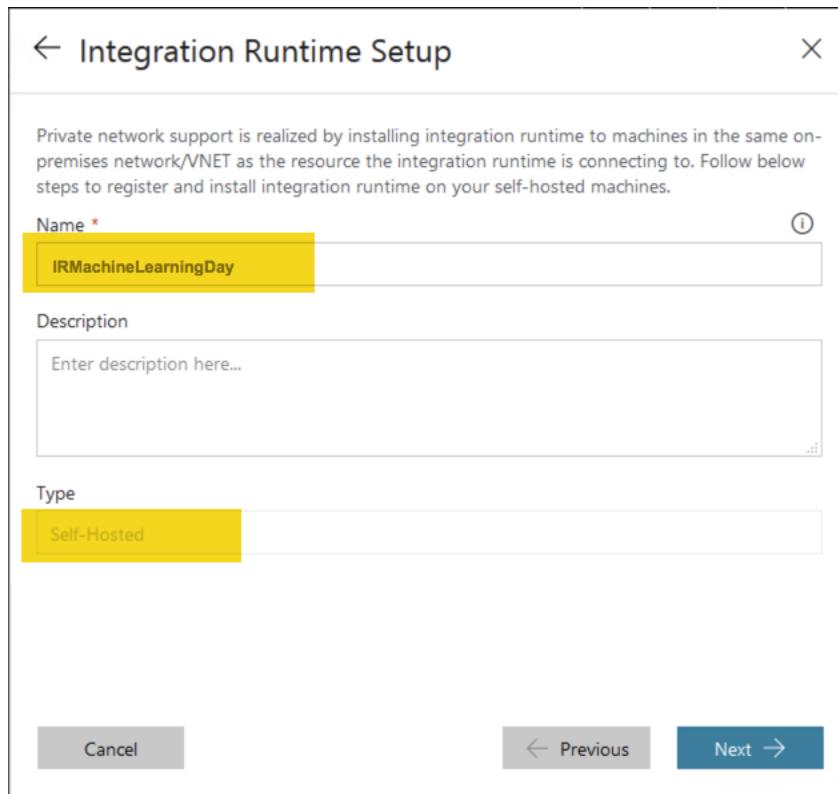


14. Select a **Self-Hosted** setup for the integration runtime as seen in the following screenshot:

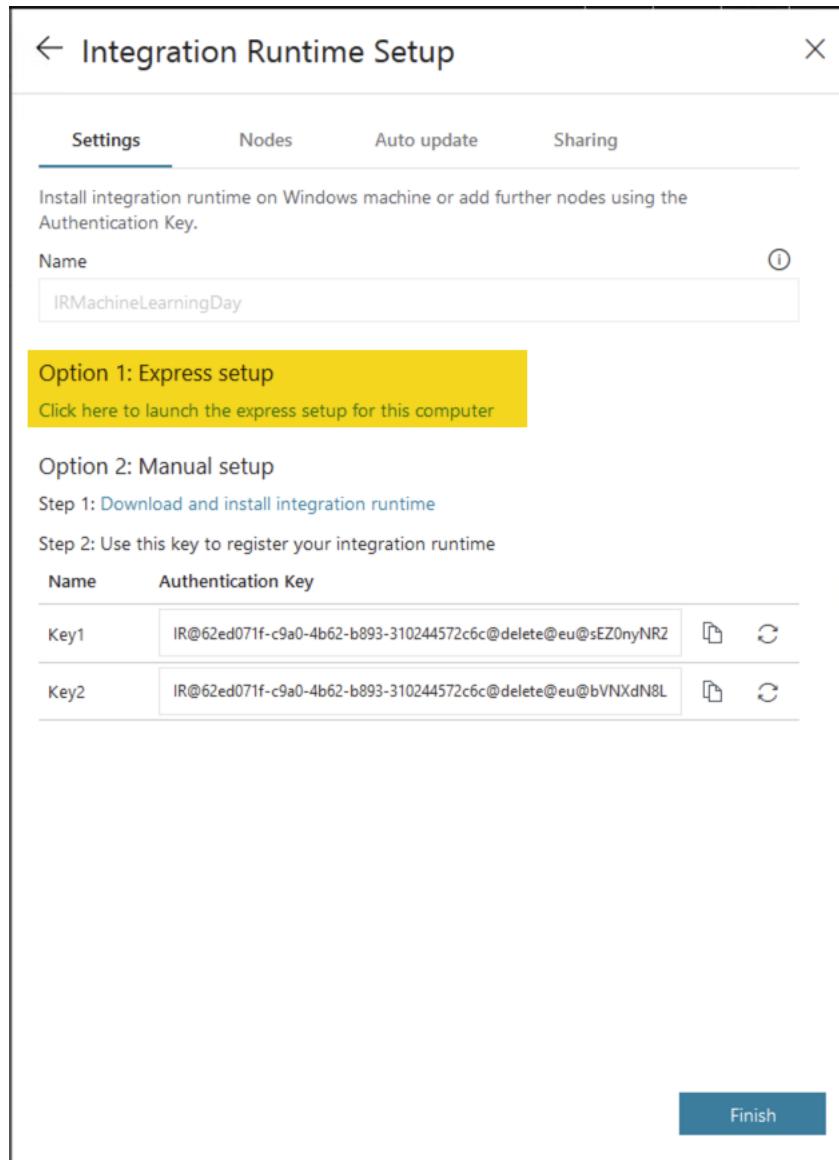


Choose an **Azure Public** environment if you are accessing services with public accessible endpoints. Choose **Self-Hosted** if you are accessing services in a private network like your on-premises environment, or in Virtual Network Environments like Azure Virtual Network. For our purposes, we will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

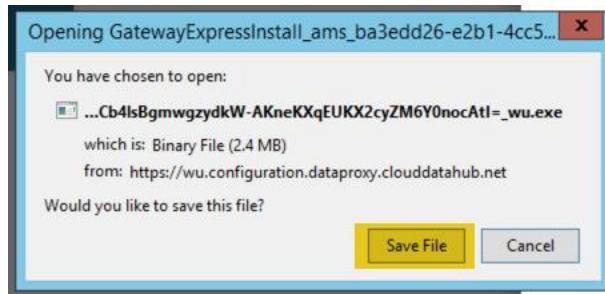
15. Private network support is realized by installing integration runtime to machines in the same on-premises network/VNET as the resource the integration runtime is connecting to. Follow below steps to register and install integration runtime on your self-hosted machines with a name called **IRMachineLearningDay**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



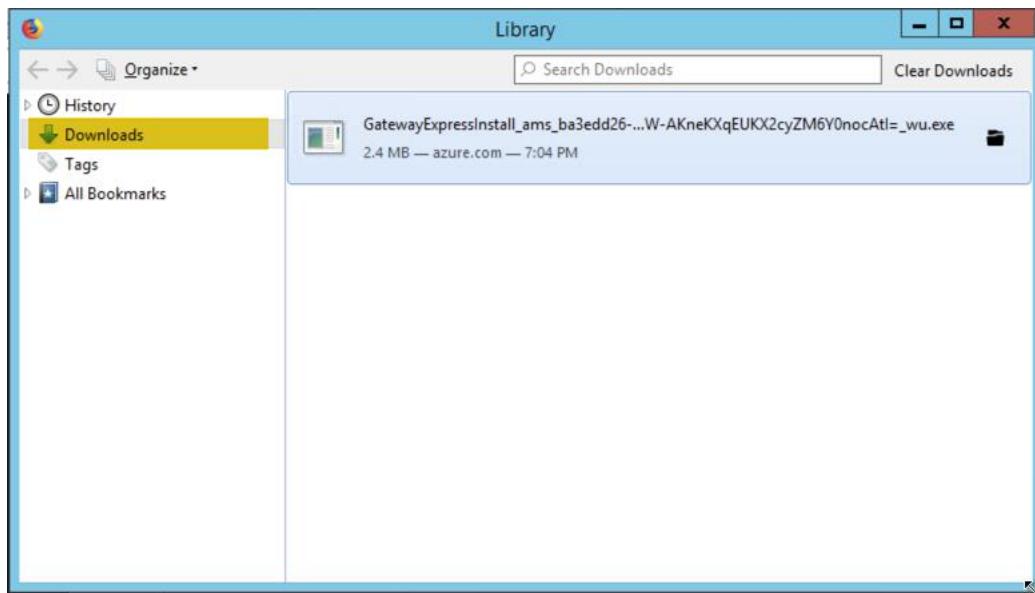
16. Select **Option 1: Express Setup** for the Integration Runtime as seen in the following screenshot:



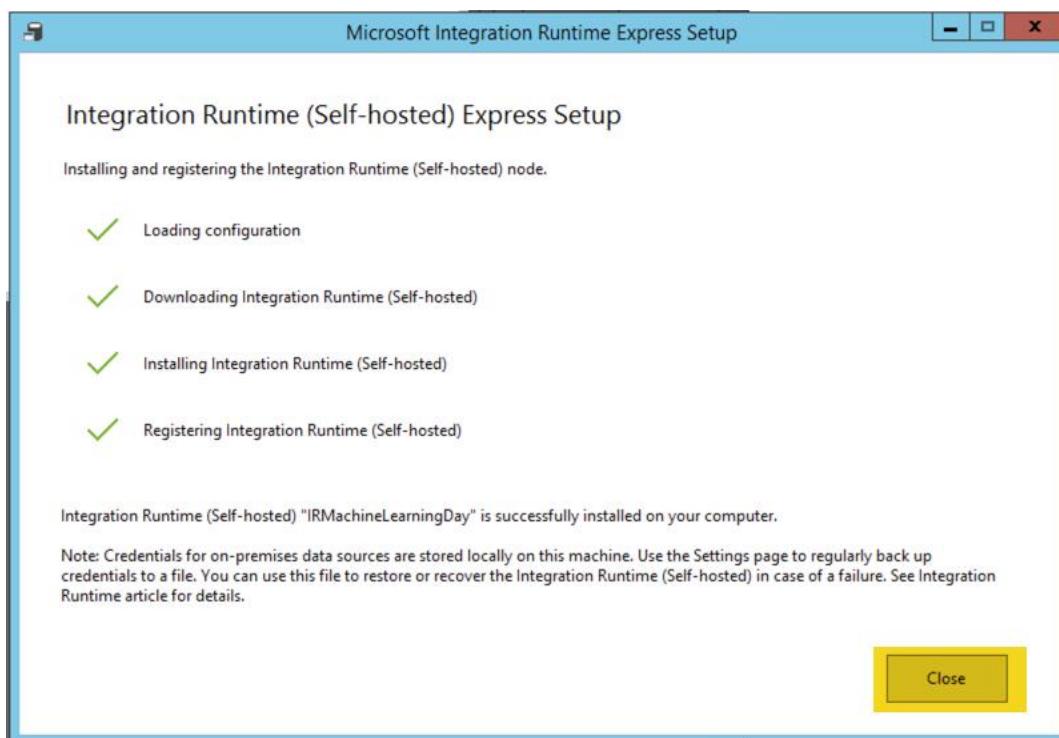
17. Save the Express Setup File (usually it will default to the Downloads folder) as seen in the following screenshot:



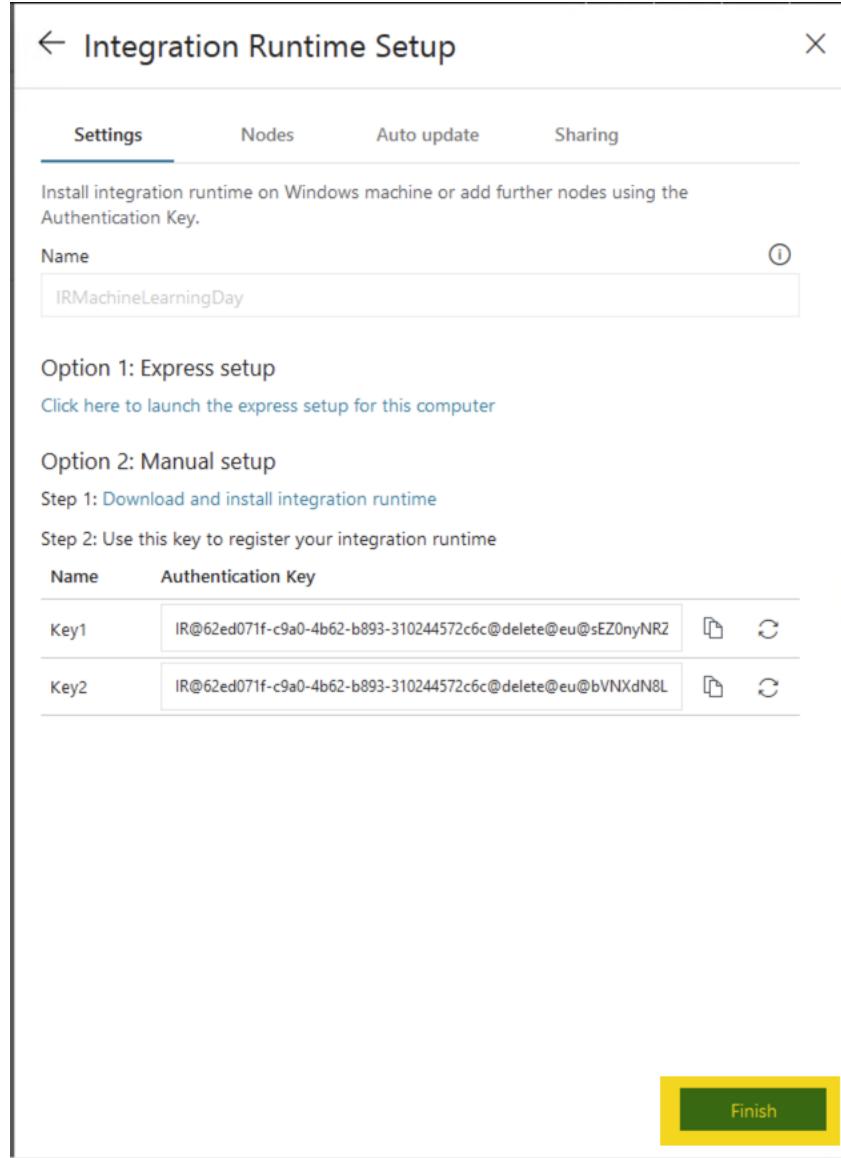
18. Execute the File from the **Downloads** folder as seen in the following screenshot:



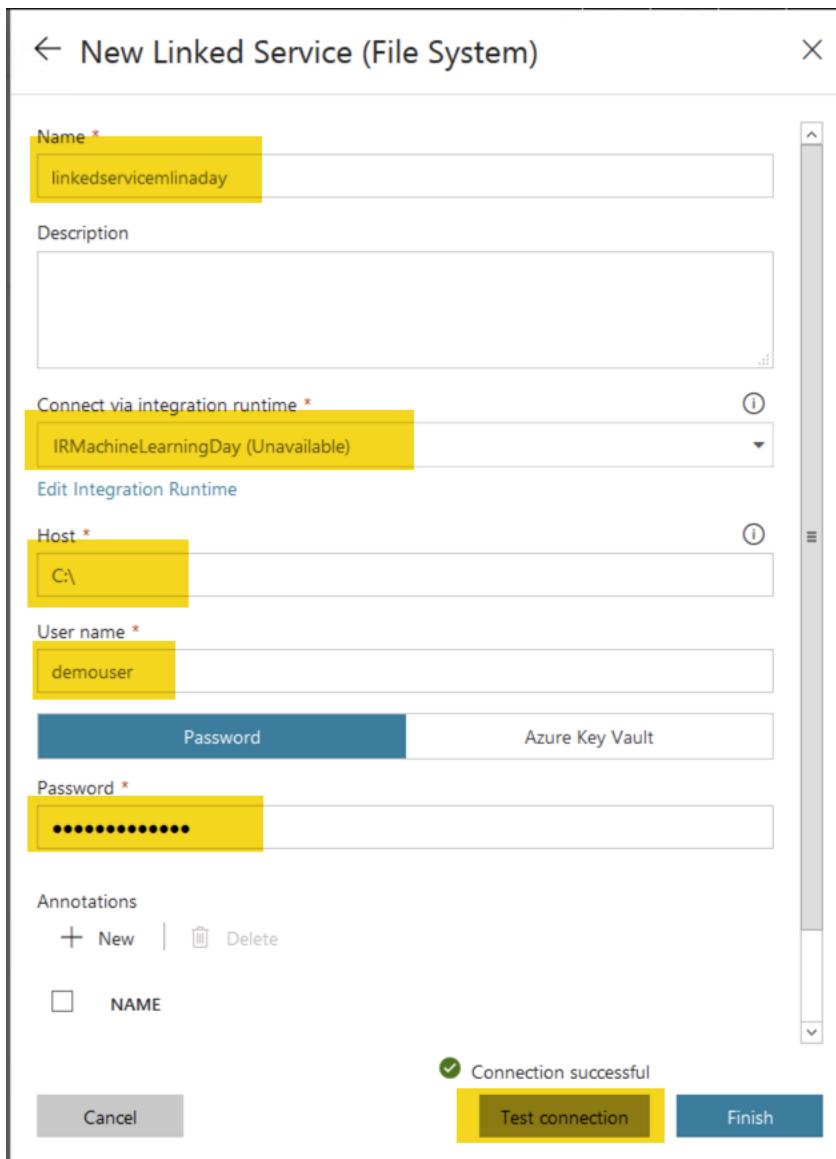
19. The Integration Runtime should begin installation. Once complete, select **Close** as seen in the following screenshot:



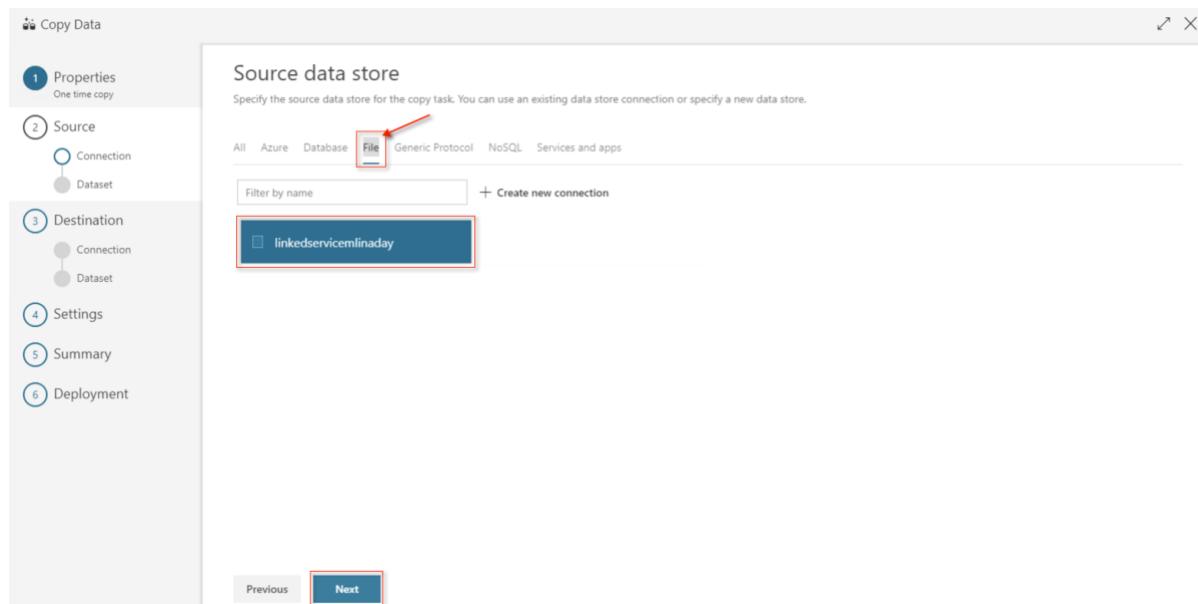
20. Select **Finish** on the **Integration Runtime Setup** as seen in the following screenshot:



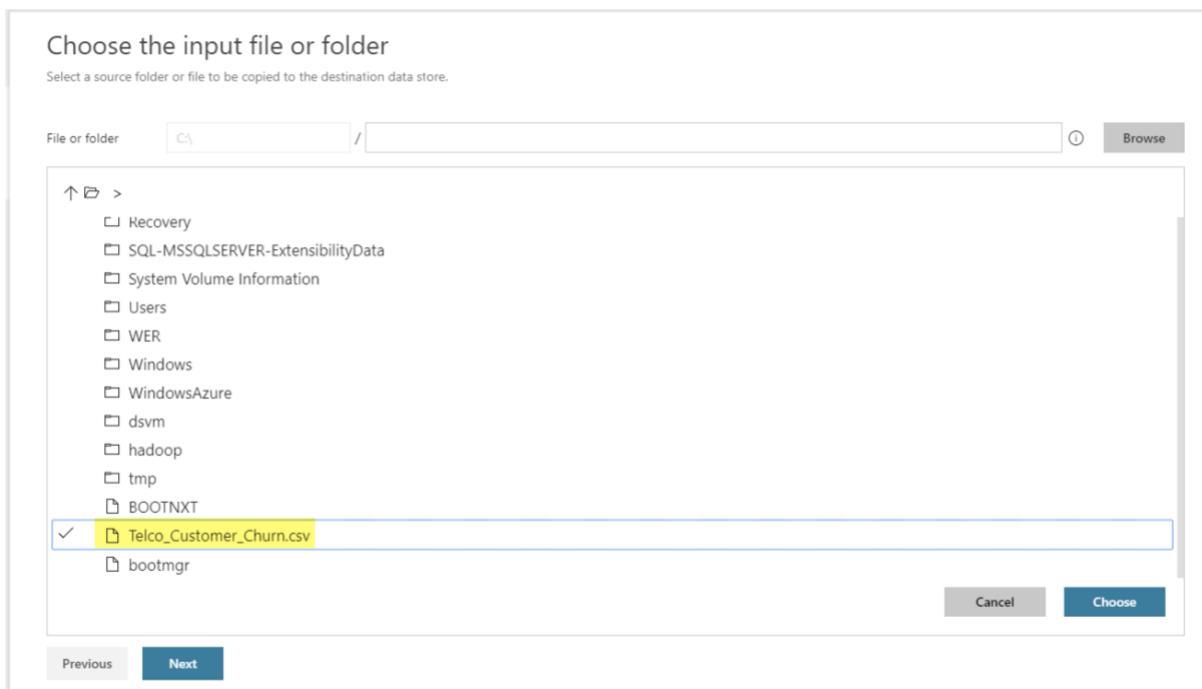
21. We can return now return to the **Linked Service (File System)** and enter our user credentials that we initially created for our Data Science Virtual Machine (DSVM) (**User name** = demouser & **Password** = #MLinaDay20!8) and enter them into our Service as well as the location of the csv file (C:\)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



22. Hopefully we will get a **Connection Successful** message next to a green checkmark. If so, go ahead and select **Finish**.
23. 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:



24. Choose the input file or folder as seen in the following screenshot for the **Telco_Customer_Churn.csv**:



25. 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 Detect Text Format

Column delimiter Comma (,) Use custom delimiter

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

Skip line count 0

Column names in the first row

Preview Schema

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

Previous Next

26. 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:

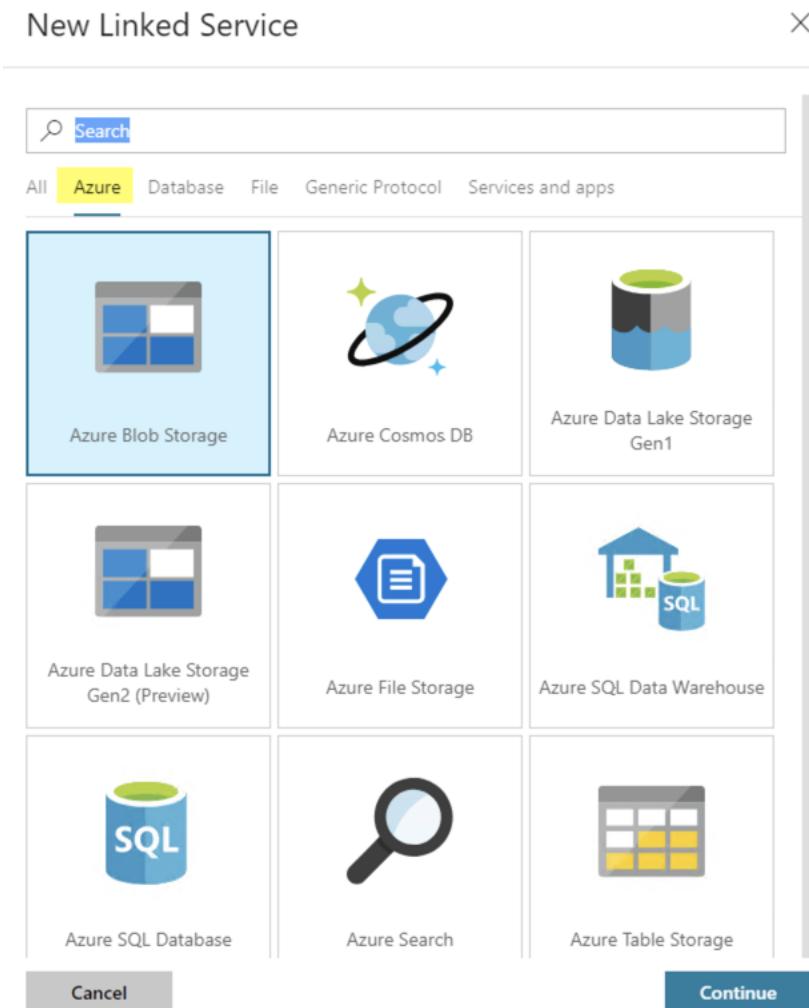
Destination data store

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

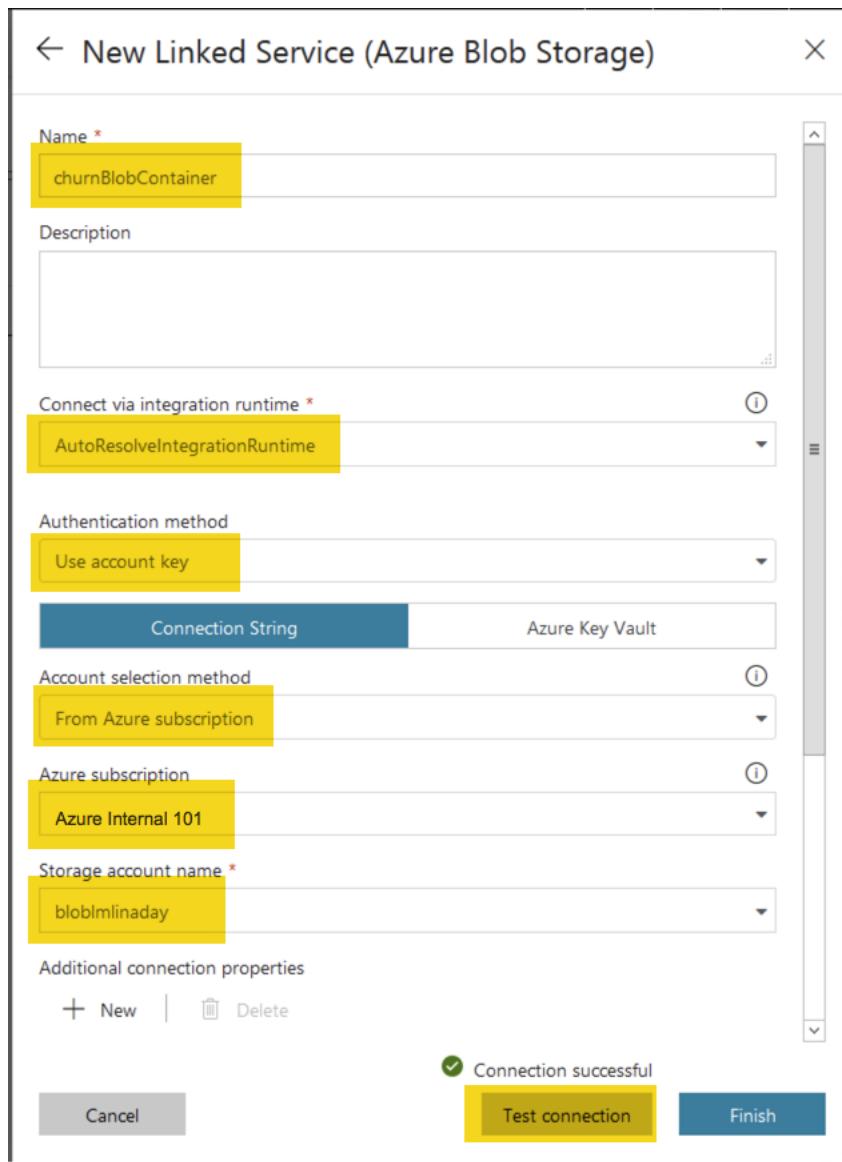
All Azure Database File Generic Protocol NoSQL Services and apps

All Filter by name + Create new connection

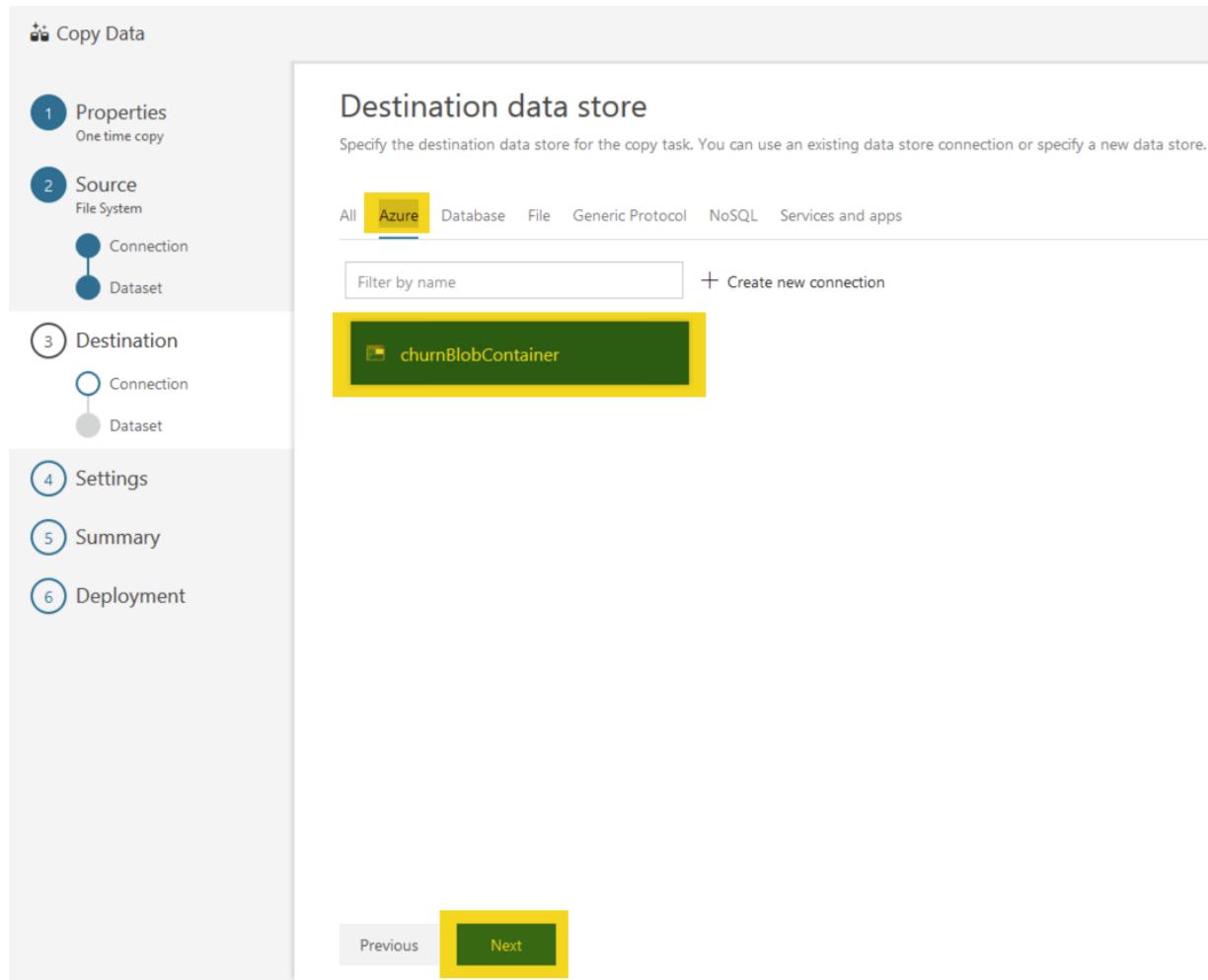
27. Search under **Azure** for **Azure Blob Store** as seen in the following screenshot:



28. 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:



29. Once connection is successful, click on **Finish**.
30. 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:



31. 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' step of the wizard. It has a title 'Choose the output file or folder' and a subtitle 'Specify a folder that will contain output files or a specific output file in the destination data store.' There are four input fields: '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 are 'Previous' and 'Next' buttons, with 'Next' being highlighted with a yellow box.

32. 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 ⓘ

Add header to file

► Advanced

Previous Next

33. 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

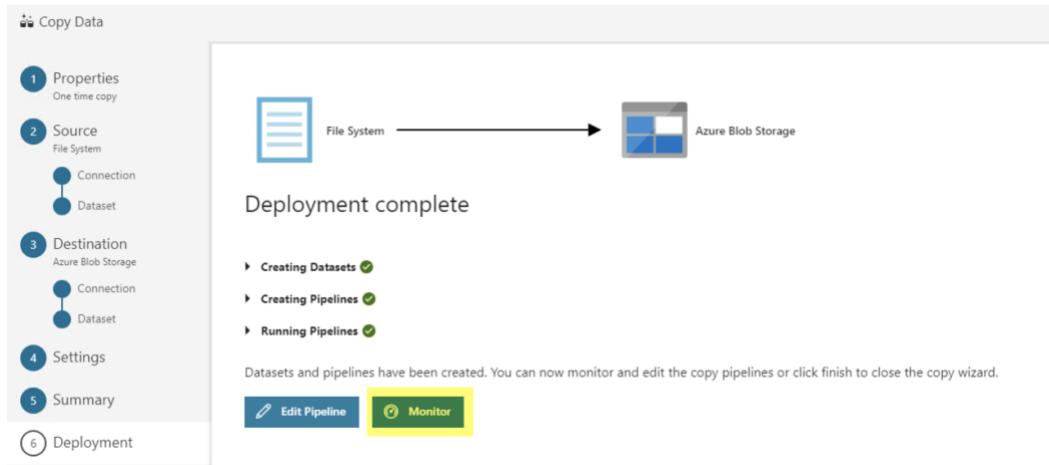
34. Review your summary pipeline to confirm all selections are accurate and select **Next** as seen in the following screenshot:

Summary

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



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



36. 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:

Microsoft Azure

adfmlinaday | Monitor Pipeline Runs

Last 24 Hours

Time Zone: (UTC+00:00) Monrovia, Reykjavik

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-17f4-4b16-a9c2-3c83829c756d

37. 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:

a. we first go to <https://portal.azure.com>

b. 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

c. 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

38. 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 > blobmlinaday - Blobs > churn

churn Container

Overview Access Control (IAM) Settings

NAME: churn.csv

MODIFIED: [date]

ACCESS TIER: Hot (Inferred)

BLOB TYPE: Block blob

SIZE: 954.59 KB

LEASE STATE: Available

39. 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

Save Discard Refresh Download Delete

Overview Snapshots Edit blob Generate SAS

customerID,gender,SeniorCitizen,Partner,Dependents,tenure,PhoneService,MultipleLines,InternetService,Online,VHS,Female,Yes,No,1,No,No,phone service,DSL,No,Yes,No,No,No,Month-to-month,Yes,Electronic check,56.95,1889.5,No

5575-GNVEDE,Male,0,No,No,34,Yes,No,DSL,Yes,No,Yes,No,No,One year,0,Mailed check,56.95,1889.5,No

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

7795-CFOCW,Male,0,No,No,45,No,No,phone service,DSL,Yes,No,Yes,No,No,0,One year,0,Bank transfer (automatic),97.71

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!

9305-CDSKC,Female,0,No,No,8,Yes,Yes,Fiber optic,No,Yes,No,Yes,Yes,Month-to-month,Yes,Electronic check,99.

1452-KIOVK,Male,0,No,Yes,22,Yes,Yes,Fiber optic,No,Yes,No,Yes,No,Month-to-month,Yes,Credit card (automatic),5129-JLPIS,Male,0,No,No,25,No,Fiber optic,Yes,No,Yes,Yes,Yes,Month-to-month,Yes,Electronic check,10!

6713-OKMC,Female,0,No,No,10,No,phone service,DSL,Yes,No,No,No,No,Month-to-month,0,Mailed check,29.75

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,0,Bank transfer (automatic),56.15,3487

9763-GRSKD,Male,0,Yes,Yes,13,Yes,No,DSL,Yes,No,No,No,No,Month-to-month,Yes,Mailed check,49.95,587.45,No

7469-LKBCI,Male,0,No,No,16,Yes,No,No,0,internet service,No,internet service,No,internet service,8091-TTVAX,Male,0,Yes,No,58,Yes,Yes,Fiber optic,No,No,Yes,No,Yes,One year,0,Credit card (automatic),10

0280-XJGEX,Male,0,No,No,49,Yes,Yes,Fiber optic,No,Yes,No,Yes,Month-to-month,Yes,Bank transfer (auto

5129-JLPIS,Male,0,No,No,25,No,Fiber optic,Yes,No,Yes,Yes,Yes,Month-to-month,Yes,Electronic check,10!

3655-SNOYZ,Female,0,Yes,Yes,69,Yes,Yes,Fiber optic,Yes,Yes,Yes,Yes,Two year,0,Credit card (automat

8191-XWSZG,Female,0,No,No,52,Yes,No,No,internet service,No,internet service,No,internet service,9959-WOKFT,Male,0,No,Yes,71,Yes,Yes,Fiber optic,Yes,No,Yes,Two year,0,Bank transfer (automatic),49.95,587.45

4190-MFLUW,Female,0,Yes,Yes,10,Yes,No,DSL,0,0,0,0,0,0,Month-to-month,0,Credit card (automatic),55.0

4183-MYFRB,Female,0,No,No,21,Yes,No,Fiber optic,No,0,0,0,0,0,0,Month-to-month,0,Electronic check,90.

8779-QRDMV,Male,1,No,No,1,No,No,phone service,DSL,0,0,0,0,0,0,0,Month-to-month,0,Electronic check,30

1680-VDCWW,Male,0,Yes,No,12,Yes,No,No,internet service,No,internet service,No,internet service,24,1066-JKSGK,Male,0,No,No,1,Yes,No,No,0,internet service,No,internet service,No,internet service,3638-WEABW,Female,0,Yes,No,58,Yes,Yes,DSL,0,0,0,0,Two year,0,Credit card (automatic),59.9,350!

6322-HRPFA,Male,0,Yes,Yes,49,Yes,No,DSL,0,0,0,0,0,0,0,Month-to-month,0,Credit card (automatic),59.6,

6865-JZK0,Female,0,No,No,30,Yes,No,DSL,0,0,0,0,0,0,0,0,Month-to-month,0,Bank transfer (automatic),55.

6467-CHFZW,Male,0,Yes,Yes,47,Yes,Yes,Fiber optic,0,0,0,0,0,0,0,0,Month-to-month,0,Electronic check,95

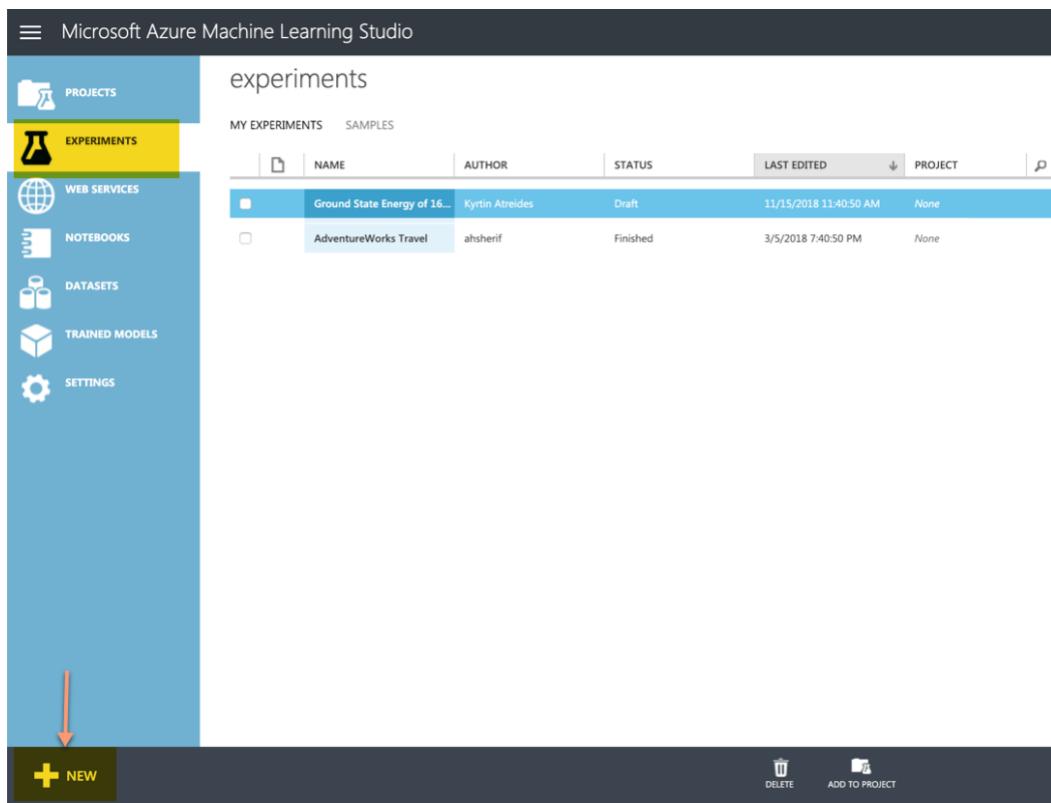
8665-UTDHZ,Male,0,Yes,Yes,1,0,No,phone service,DSL,0,0,0,0,0,0,0,0,Month-to-month,0,Electronic check,90

5248-YGIJN,Male,0,Yes,No,72,Yes,Yes,DSL,0,0,0,0,0,0,0,0,Two year,0,Credit card (automatic),90.25,0

Csv

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

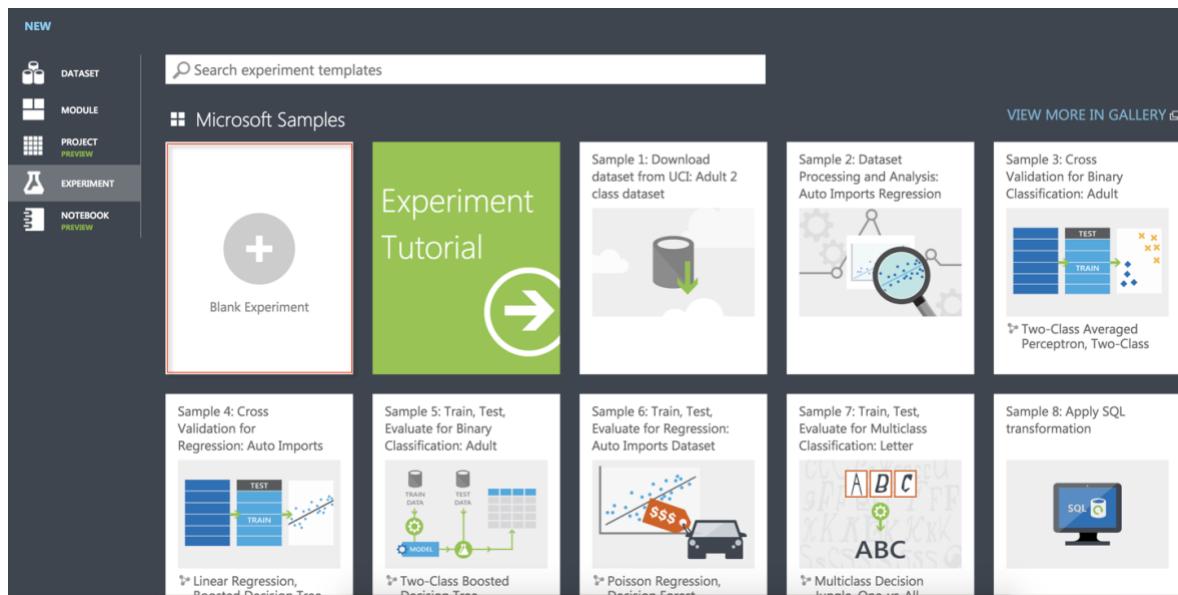
1. Clustering is the process of grouping similar entities together. The goal of this unsupervised machine learning technique is to find similarities in the fields that we have about our customers and group them into 10 distinct clusters.
2. Sign into Azure Machine Learning Studio using your Azure login 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 as seen in the following screenshot. All current and future experiments will reside in this section.



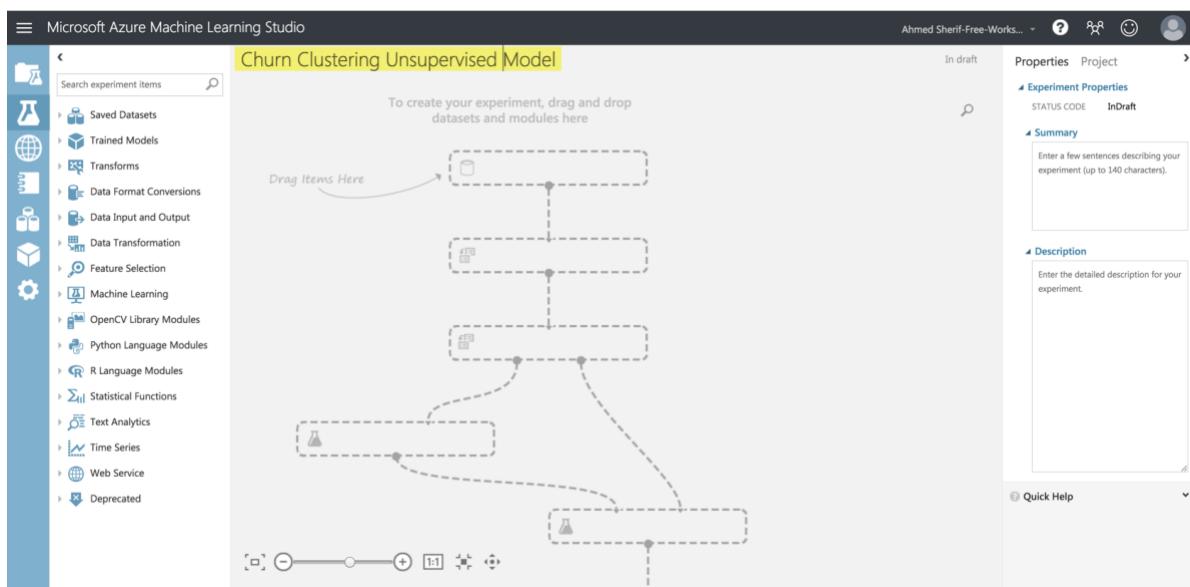
The screenshot shows the Microsoft Azure Machine Learning Studio interface. On the left, there is a vertical sidebar with icons for PROJECTS, EXPERIMENTS (which is highlighted in yellow), WEB SERVICES, NOTEBOOKS, DATASETS, TRAINED MODELS, and SETTINGS. Below the sidebar, a red arrow points down to a green button labeled '+ NEW'. The main area is titled 'experiments' and contains a table with two rows. The columns are labeled NAME, AUTHOR, STATUS, LAST EDITED, and PROJECT. The first row has a checkbox, the name 'Ground State Energy of 16...', author 'Kyrtin Atreides', status 'Draft', last edited '11/15/2018 11:40:50 AM', and project 'None'. The second row also has a checkbox, the name 'AdventureWorks Travel', author 'ahsherif', status 'Finished', last edited '3/5/2018 7:40:50 PM', and project 'None'.

	NAME	AUTHOR	STATUS	LAST EDITED	PROJECT
<input type="checkbox"/>	Ground State Energy of 16...	Kyrtin Atreides	Draft	11/15/2018 11:40:50 AM	None
<input type="checkbox"/>	AdventureWorks Travel	ahsherif	Finished	3/5/2018 7:40:50 PM	None

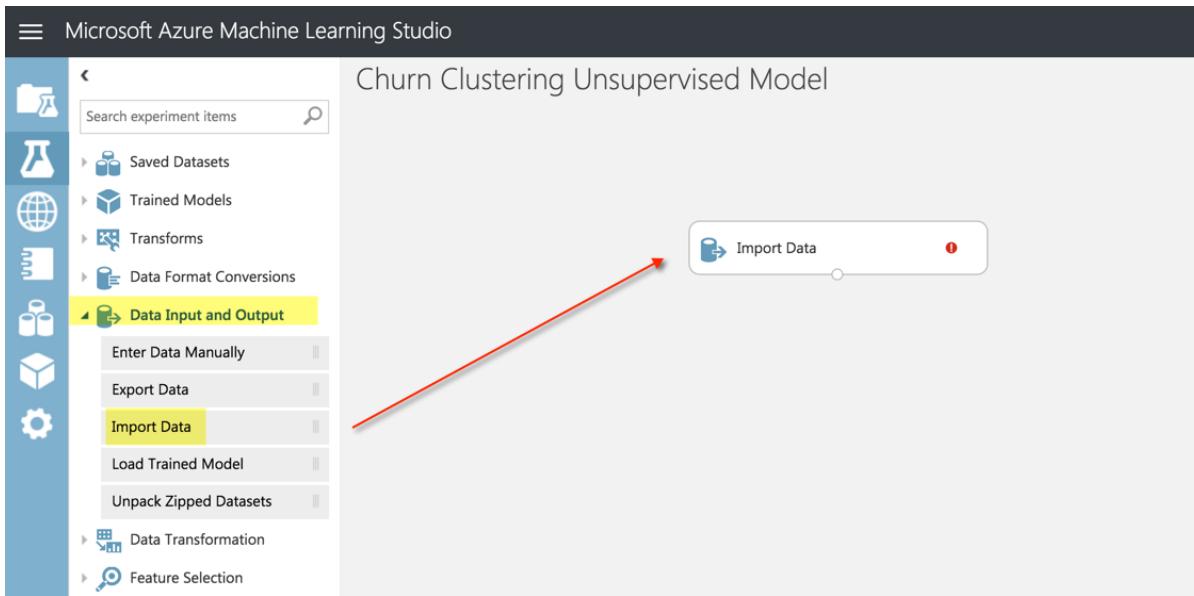
4. Next click on **+NEW** on the button of the screen and select **Blank Experiment** as seen in the following screenshot:



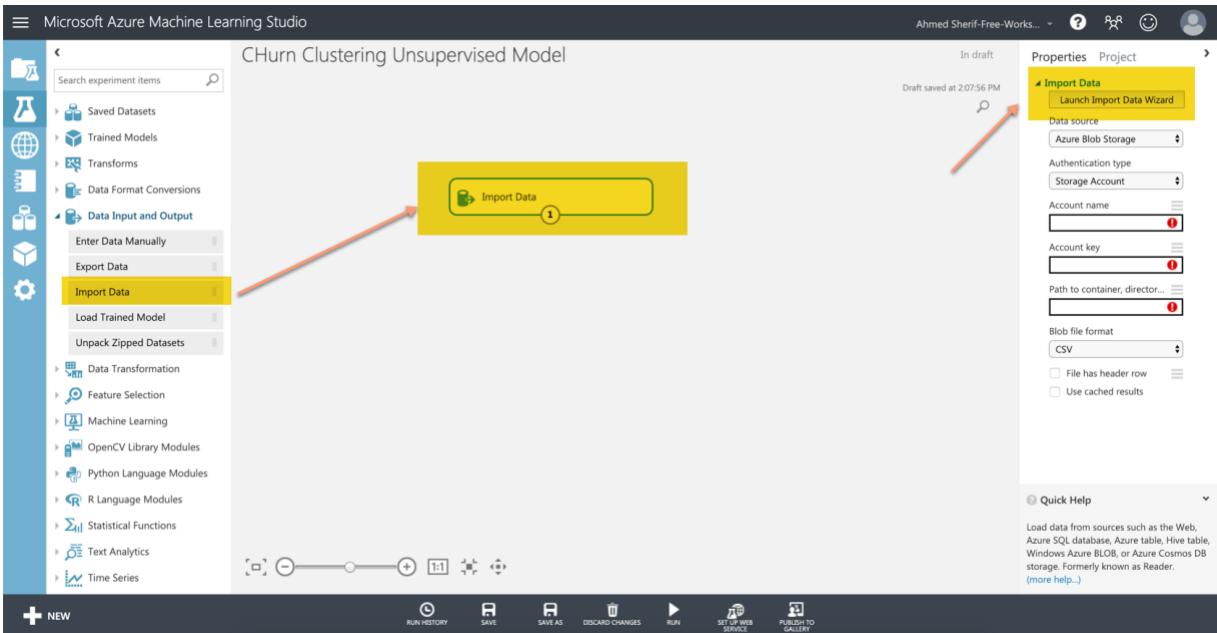
- Click on the title of the experiment and change the name to **Churn Clustering Unsupervised Model** as seen in the following screenshot:



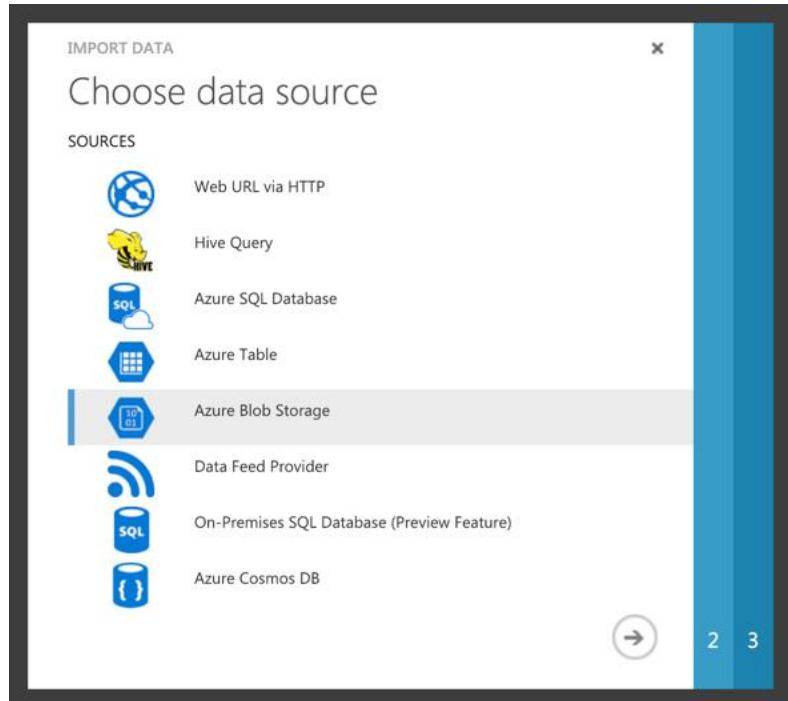
- Expand the **Data Input and Output** category and drag the **Import Data** into the Workflow as seen in the following screenshot:



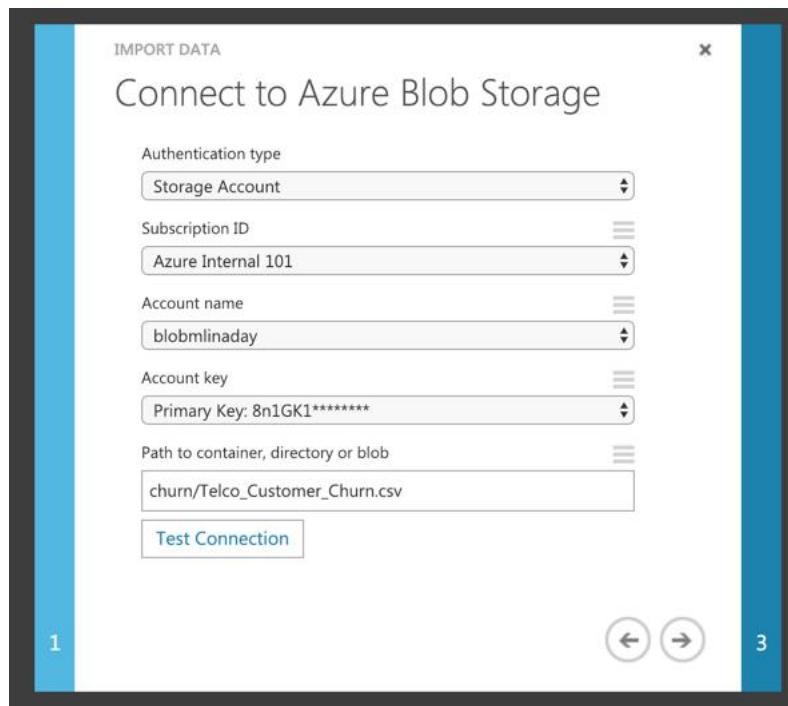
- As you click on the **Import Data** icon in the workflow, you will have the option to configure your data to pull from the Azure Blob Container you created a few sections ago using the Launch Import Wizard as seen in the following screenshot:



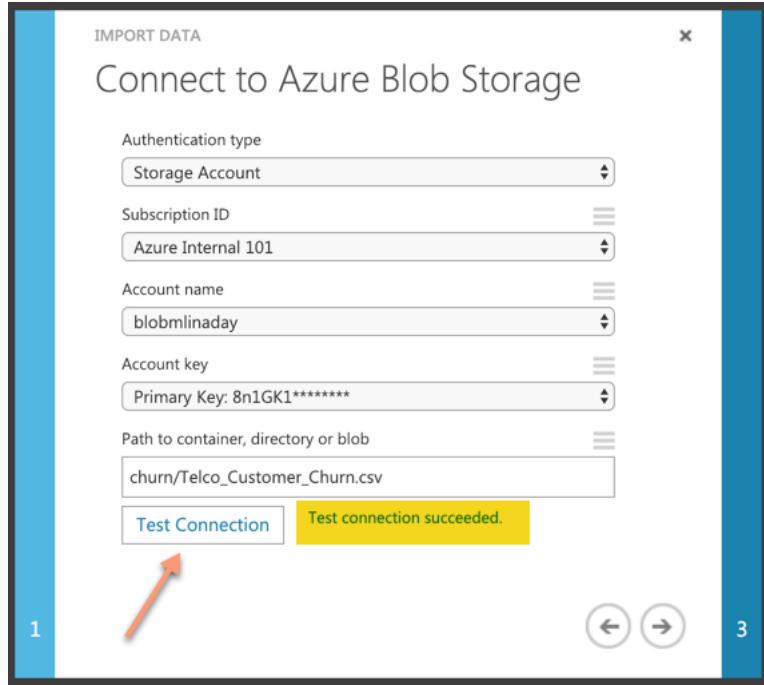
- Select Azure Blob Storage



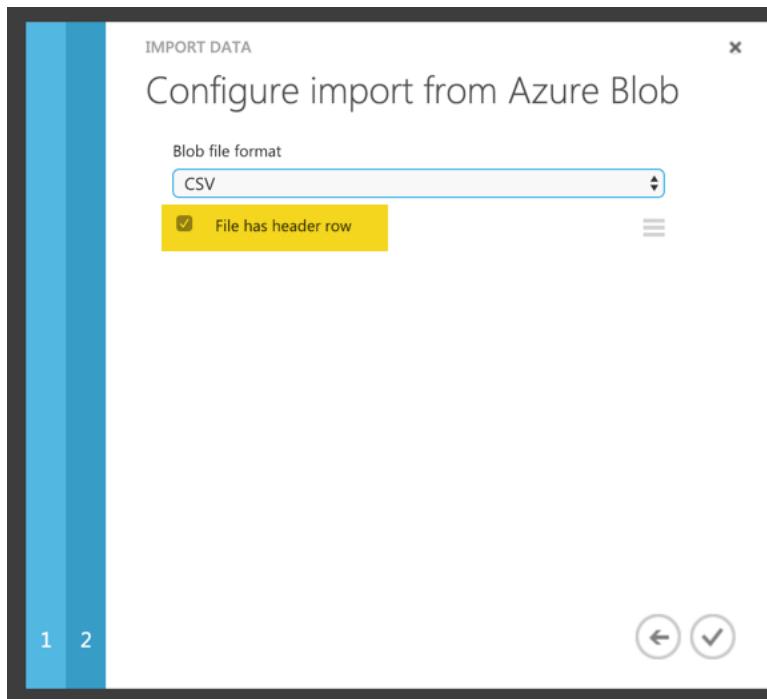
9. Specify the following credentials from the drop-down boxes for your accounts:



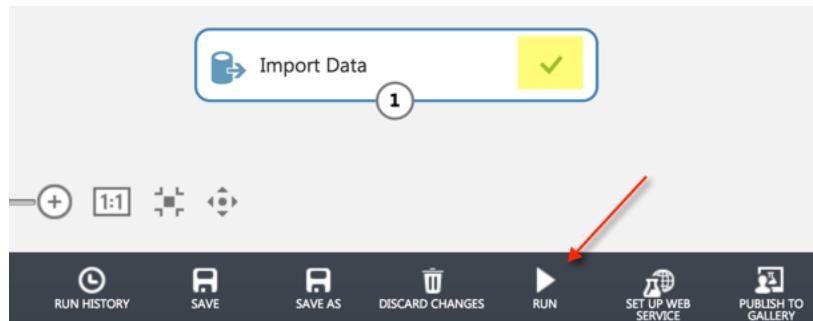
10. Test the connection to make sure the credentials work. Once it is validated you should see a green checkmark as seen in the following screenshot:



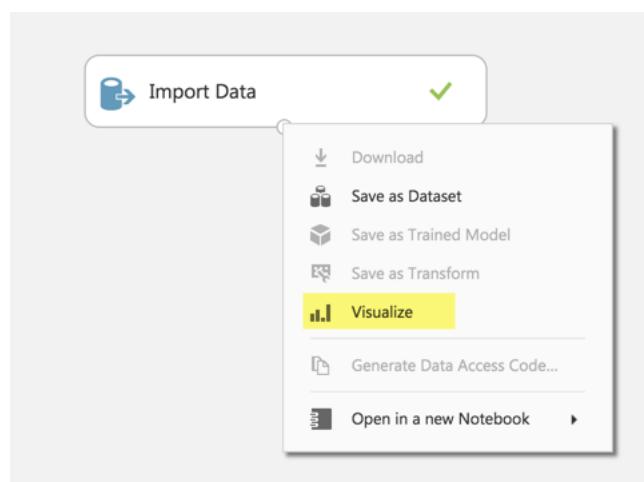
11. Check to include a **file header row** for the data and maintain a **CSV** file structure as seen in the following screenshot:



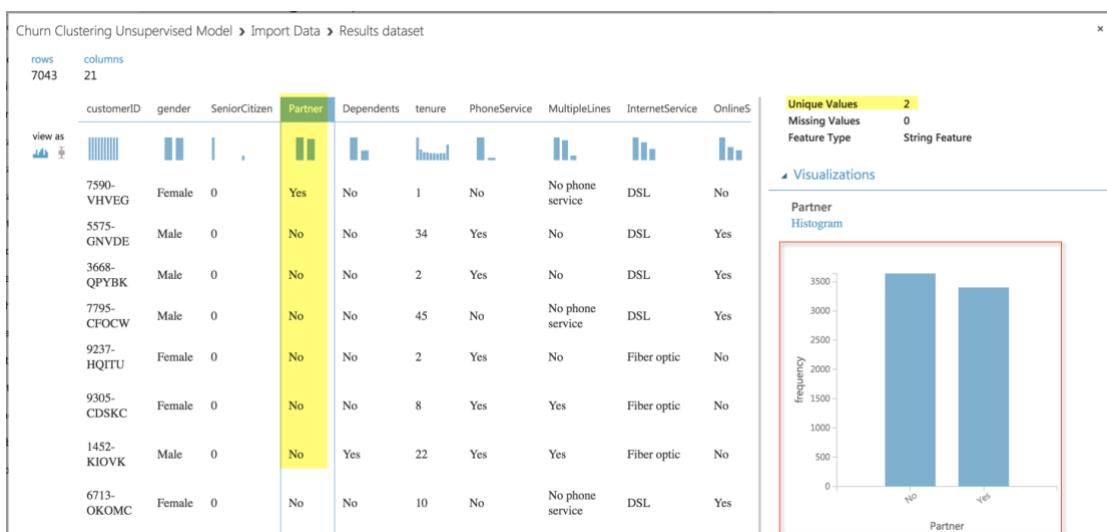
12. Back in the Azure ML Studio workflow, the credentials can be confirmed by clicking on the **RUN** icon at the bottom of the screen and seeing the green check mark next to **Import Data**.



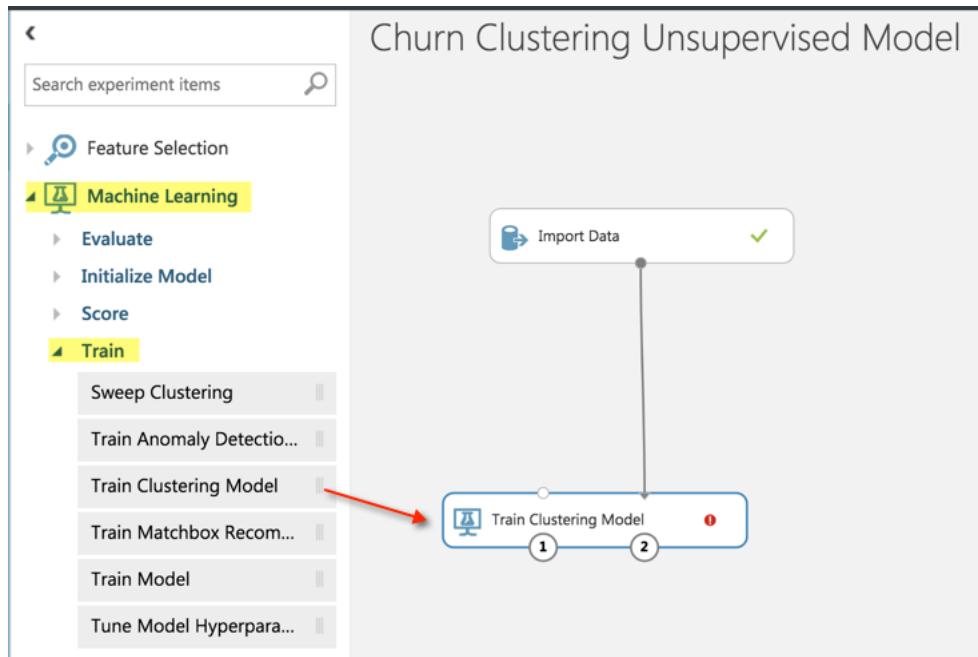
13. ***Every time a new node is added and attached to an existing node, you will need to select the RUN icon to see the new results***
14. Once we have confirmed the connection is working, we can right-click on the number **1** on the first workflow and **Visualize** the dataset as seen in the following screenshot:



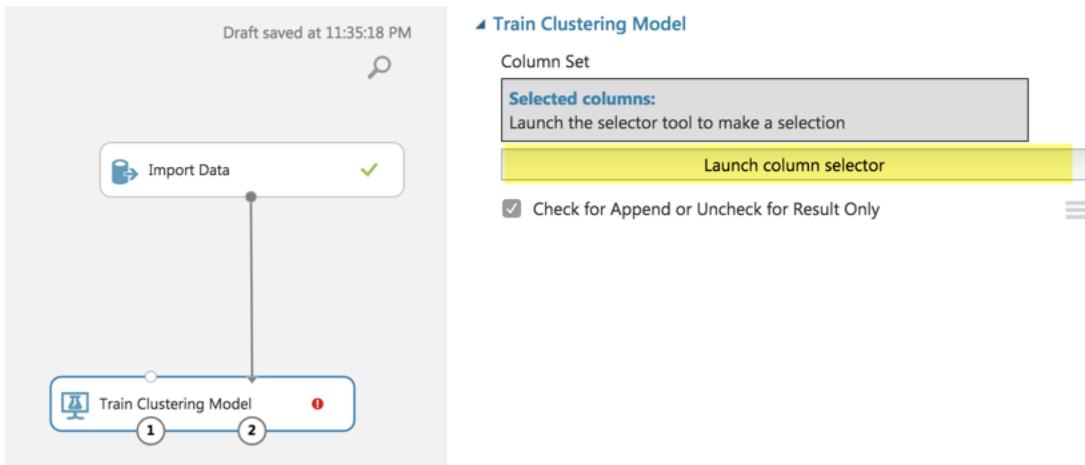
15. Take some time to click on the columns to identify the summary statistics for both the numeric fields and the categorical fields as seen in the following screenshot:



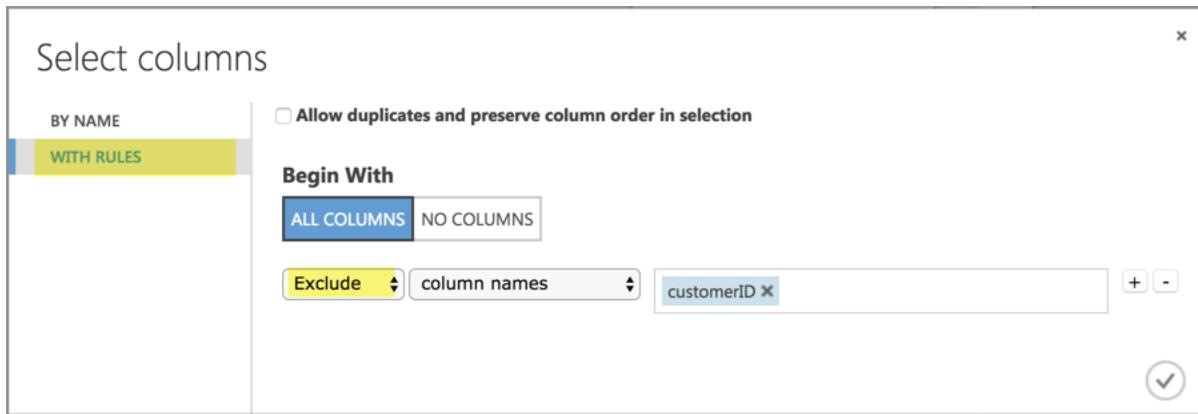
16. Import **Train Clustering Model** from the **Machine Learning** Node and connect it to the **Import Data** module as seen in the following screenshot:



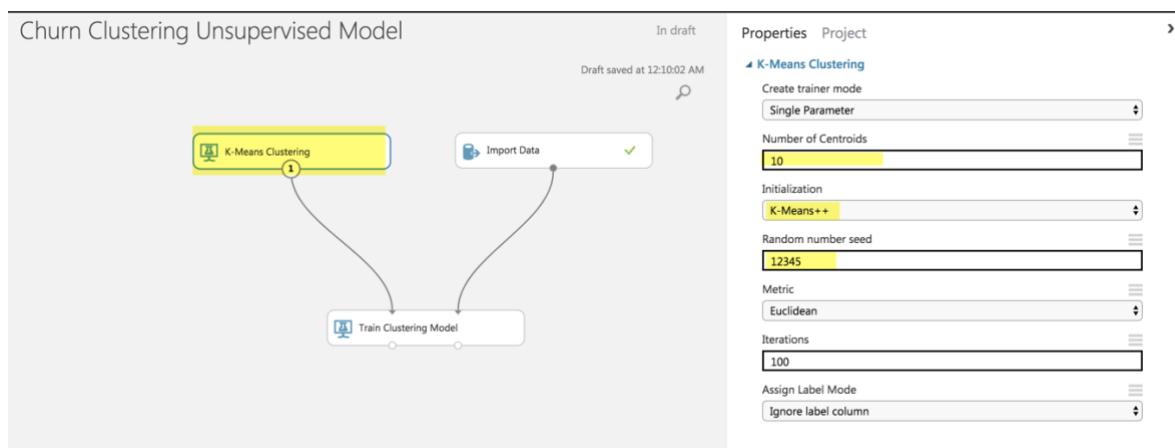
17. Next, click on the **Train Clustering Model** module, **Launch Column Selector**, connect the node back to the **Import Data** node, and configure the columns to be selected as seen in the following screenshot:



18. **CustomerID** is not a column that will help us out in clustering customers into 1 of 10 categories; therefore, we will remove it from the cluster algorithm.
19. So 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:

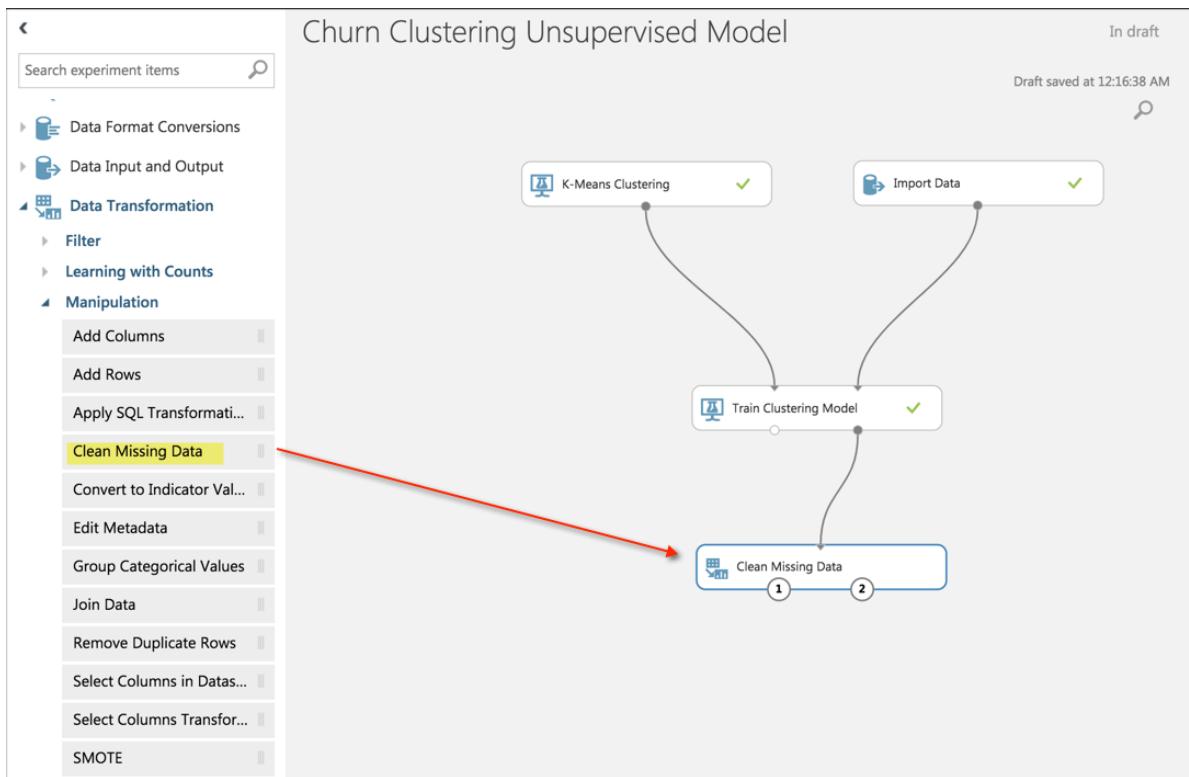


20. Next, we will attach the **K-Means Clustering** model to the **Train Clustering Model** and assign the following parameters to the clustering model, as seen in the following screenshot:

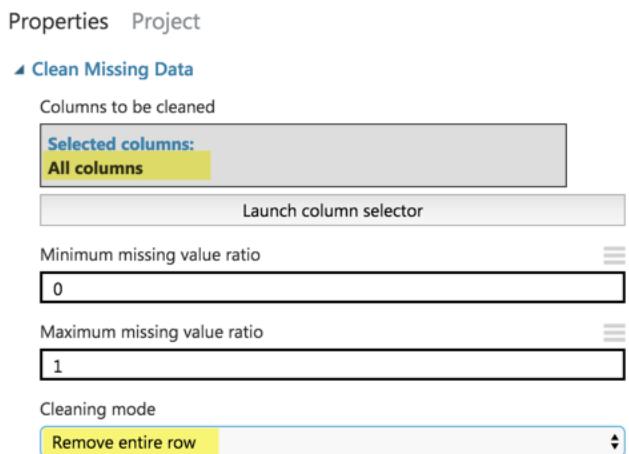


21. Select **RUN**

22. 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:



23. 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:

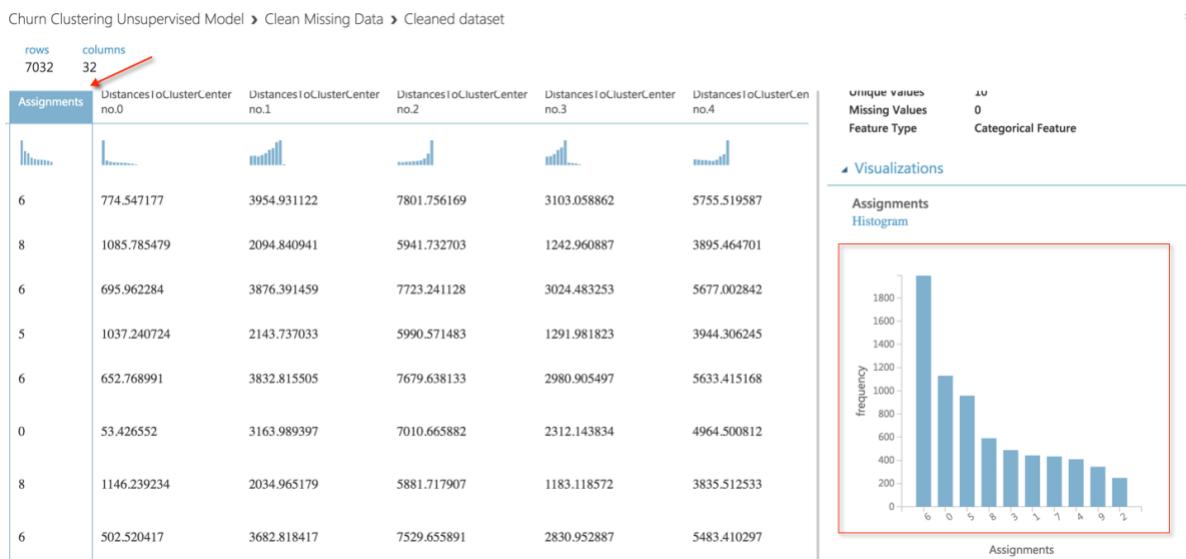


24. Select **RUN**

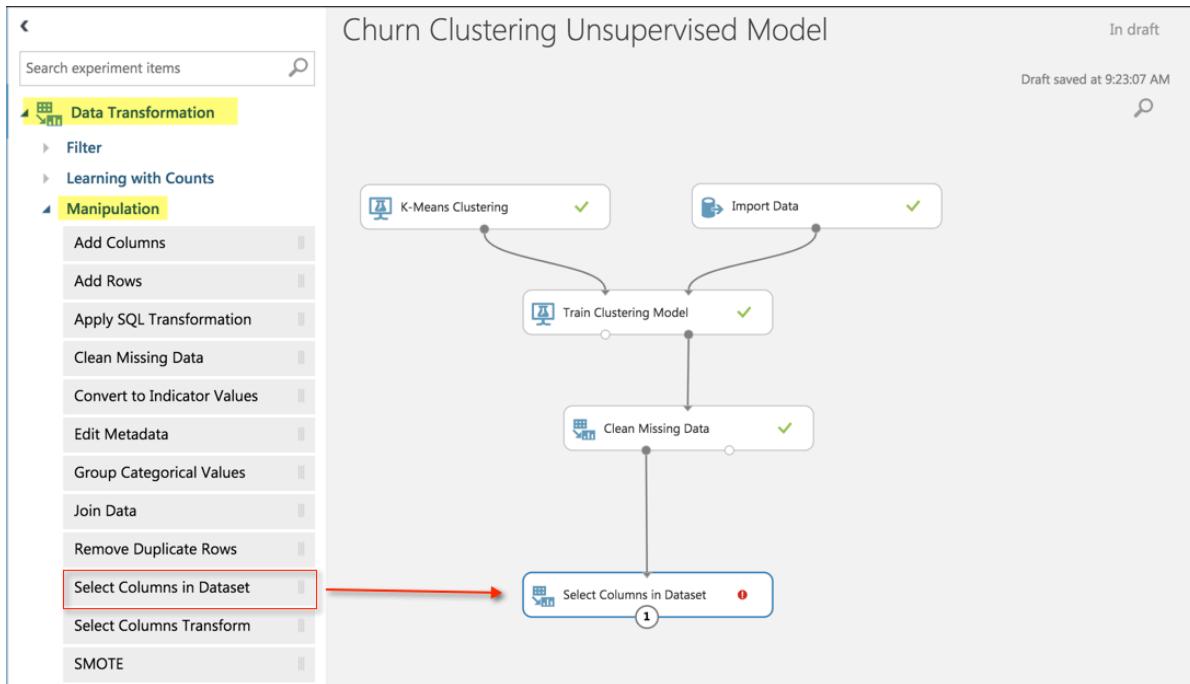
25. The clustering outcome produces several columns that are not necessary for our purposes for this workshop as can be seen by visualizing the **Clean Missing Data** module as seen in the following screenshot:

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	

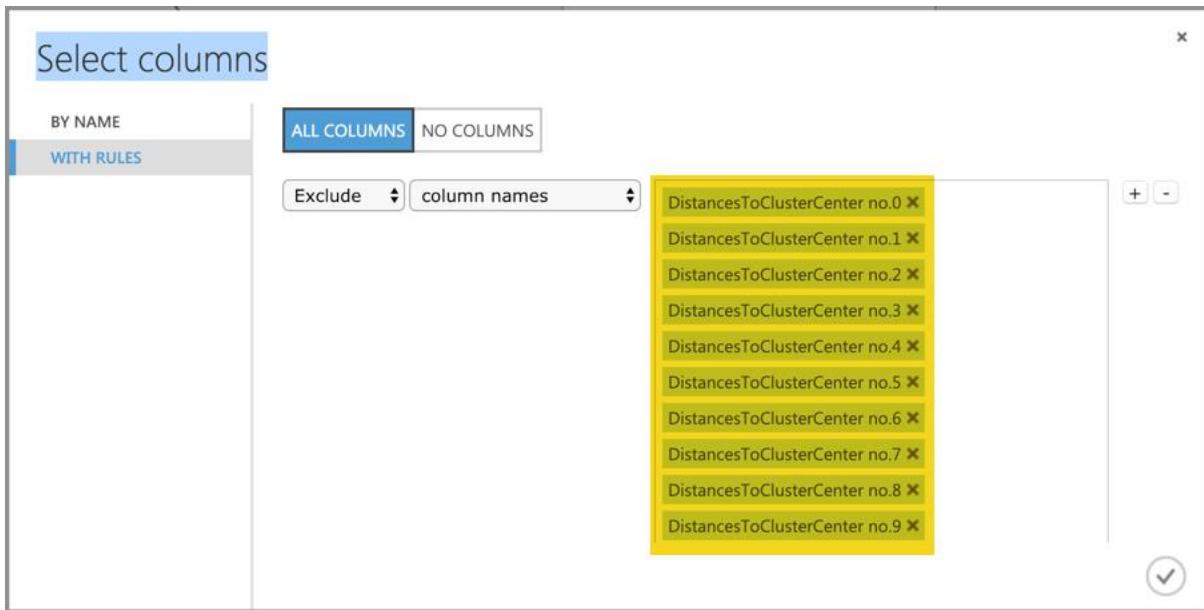
26. We are truly only interested in the assignments, which is grouping each customer into a category of **0** through **9** as seen when we visualize a histogram of the unique values as seen in the following screenshot:



27. 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:

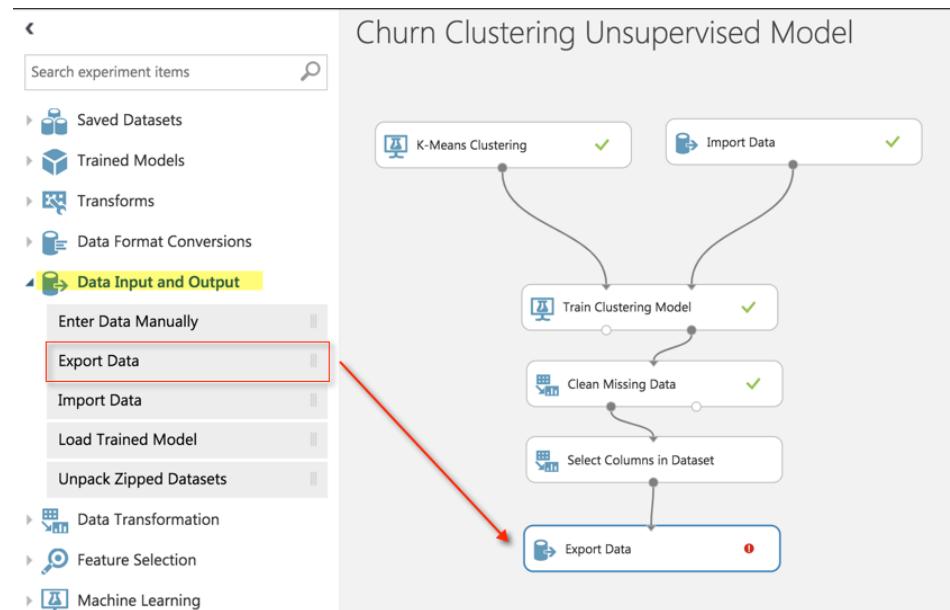


28. We can then configure the module for **Select Columns in Dataset** by **Launching Column Selector**, select **WITH RULES**, and exclude column names DistancetoClusterCenter no.0 through DistancetoClusterCenter no.9, as seen in the following screenshot:

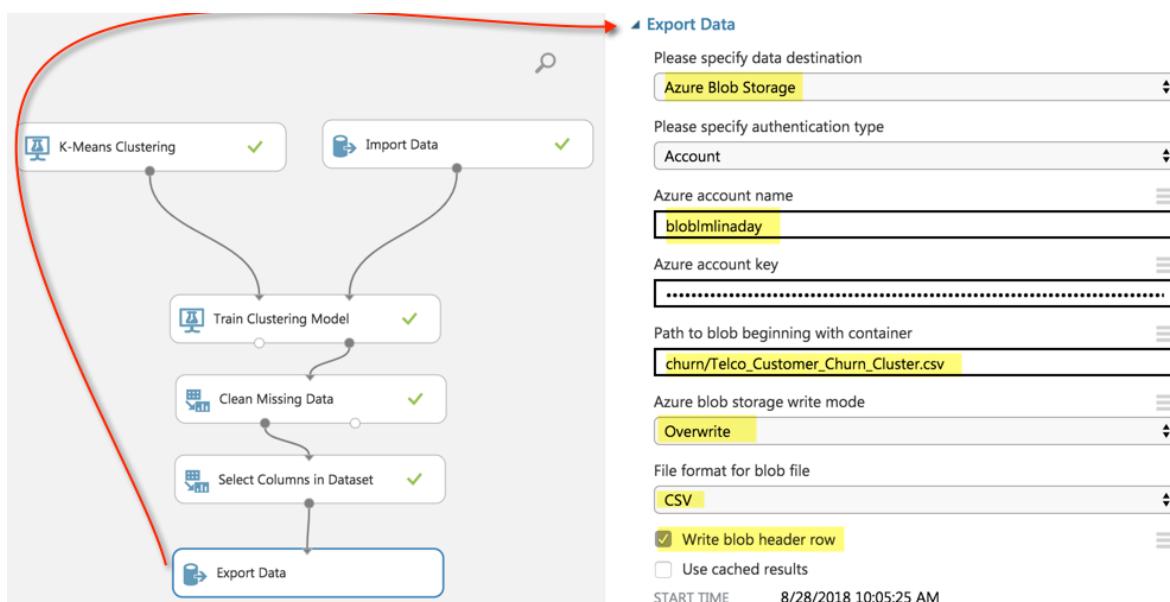


29. Select **RUN**

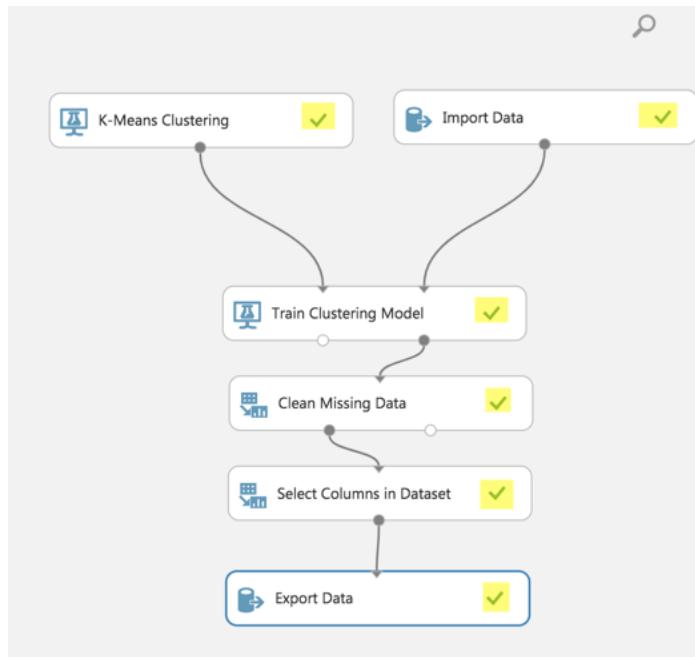
30. We are finally ready to send out revised dataset with only the columns that we need back to blob storage using the **Export Data** module as seen in the following screenshot:



31. Unlike the Import Data module, we do not have a wizard to help us out. So we will have to input our credentials manually.
32. 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 saved to a notepad from an earlier section*):



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

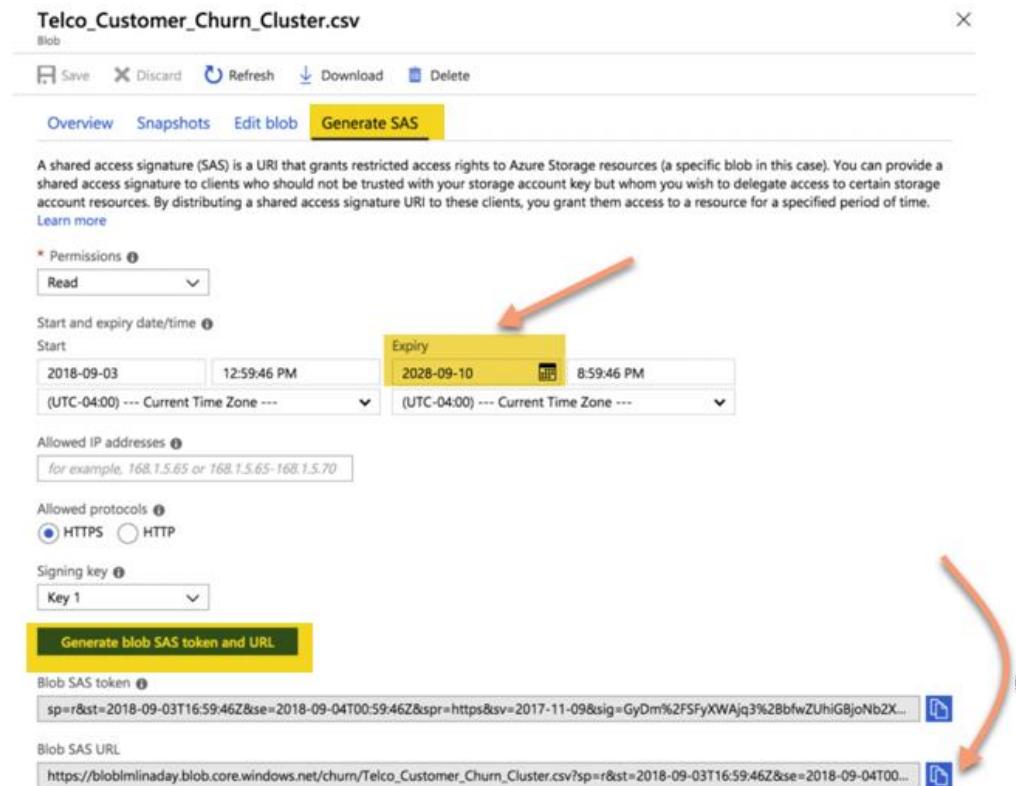


35. We can revisit our Blob container in the Azure Portal and view our newly added dataset alongside our existing dataset as seen in the following screenshot:

NAME	MODIFIED	BLOB TYPE
Telco_Customer_Churn_Cluster.csv	2018/10/02 10:05:29 AM	Block blob
Telco_Customer_Churn.csv	2018/10/02 11:32:04 AM	Block blob

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

37. 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:



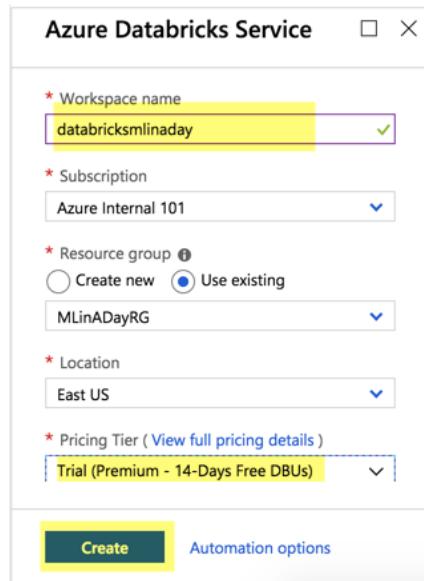
38. Keep a copy of the SAS URL as it will come in handy in the next section when we are reading the file from Blob to Azure Databricks. Additionally, make sure to set expiry date sometime in the future so that it does not expire while you are working on the workshop. Set a time expiration for at least 1 month from the day you are performing the workshop.
39. We are now ready to bring our new dataset with the additional features from clustering into Azure Databricks to do some supervised learning with **SparkML**.

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

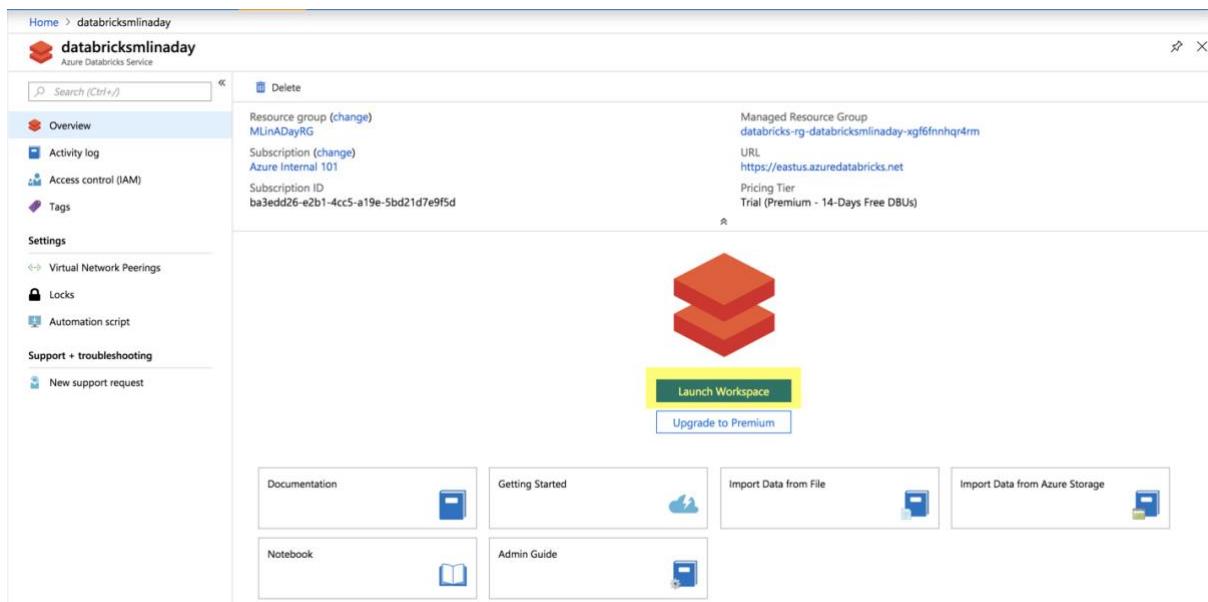
1. Return to the Azure portal (<https://portal.azure.com>) and provision an **Azure Databricks** workspace by selecting Azure Databricks from the **marketplace** as seen in the following screenshot:

The screenshot shows the Microsoft Azure portal interface. The top navigation bar includes 'Preview', 'Microsoft Azure', 'Report a bug', and a search bar. Below the navigation is a breadcrumb trail: Home > New > Azure Databricks. The main content area is titled 'Azure Databricks' and describes it as a 'Fast, easy, and collaborative Apache Spark-based analytics platform'. It highlights features like accelerating innovation, driving productivity, building on a secure cloud, and scaling without limits. A 'Create' button is prominently displayed at the bottom.

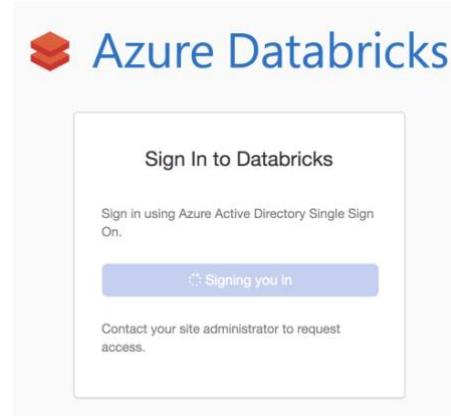
2. Enter the **Workspace** specifications as seen in the following screenshot:



3. Make sure you select the **Trial (Premium 14-Days Free DBUs)** for this specific workshop. Your IT team will thank you later on!
4. Once it has been provisioned, you can go to the resource and **launch** the **workspace** as seen in the following screenshot:



5. 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:



6. 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 top navigation bar shows "Microsoft Azure" and "PORTAL". A message "Free trial ends in 14 days. Upgrade to Premium in Azure Portal" is displayed. The main header is "Azure Databricks". The page is divided into several sections:

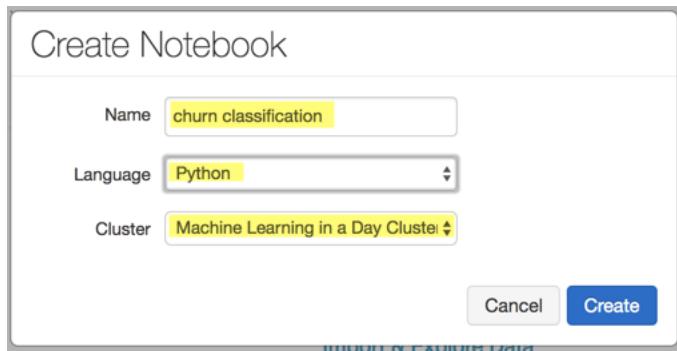
- Explore the Quickstart Tutorial:** "Spin up a cluster, run queries on preloaded data, and display results in 5 minutes."
- Import & Explore Data:** "Drop files or click to browse" and "Quickly import data, preview its schema, create a table, and query it in a notebook."
- Create a Blank Notebook:** "Create a notebook to start querying, visualizing, and modeling your data."
- Common Tasks:** A list including "New Notebook", "Upload Data", "Create Table", "New Cluster" (which is highlighted in yellow), "New Job", "Import Library", and "Read Documentation".
- Recents:** A placeholder for recently used files.
- Documentation:** Links to "Databricks Guide", "Python, R, Scala, SQL", and "Importing Data".

The left sidebar includes links for Home, Workspace, Recent, Data, Clusters, Jobs, and Search, with "Clusters" being the active tab.

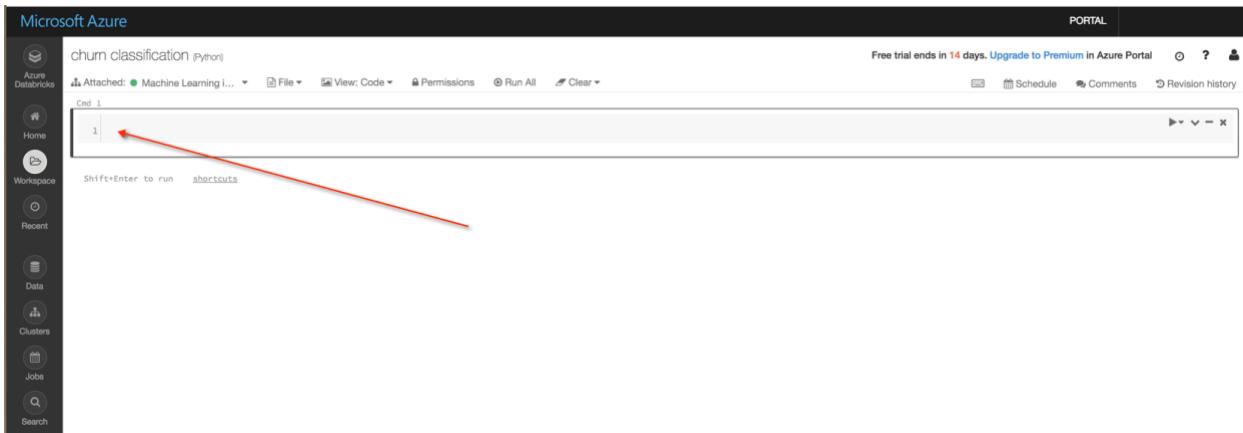
7. For our purposes for this workshop, we will provision a **Standard** mode cluster with **Python Version 3** support called **Machine learning in a Day Cluster** as seen in the following screenshot:

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

- 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:



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



11. 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%2BI2OHKAO0Nw%3D&sr=b'
```

Please note that your SAS_url will be different than the one in this documentation

12. Next we will read our dataset into a Python DataFrame known as pandas as seen in the following screenshot:

```
import pandas as pd
pandas_df = pd.read_csv(SAS_url)

df = spark.createDataFrame(pandas_df)
```

13. The concept of a Dataframe comes from the world of statistical software used in empirical research; it generally refers to "tabular" data: a data structure representing cases (rows), each of which consists of a number of observations or measurements (columns). Think of a spreadsheet in a CSV or Excel format or a table in a SQL database.

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

```

Azure Databricks
churn classification [Python]
Attached: Machine Learning i...
File View: Code Permissions Run All Clear
PORTAL
Free trial ends in 11 days. Upgrade to Premium in Azure Portal
Home Workspace Recent Data Clusters Jobs Search
churn classification [Python]
Attached: Machine Learning i...
File View: Code Permissions Run All Clear
PORTAL
Free trial ends in 11 days. Upgrade to Premium in Azure Portal
Home Workspace Recent Data Clusters Jobs Search
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=Gy0mz2FSFyXWAjq3N2BbfZU1gBjOnDkXzF81n2B120HKA0Nw3D6srb"
Command took 0.01 seconds -- by ahsherif@microsoft.com at 9/3/2018, 1:07:50 PM on Machine Learning in a Day Cluster
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, 1:08:03 PM on Machine Learning in a Day Cluster
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
1

```

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

```
display(df)
```

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

Cmd 4

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
800LJDNHKC	Female	0	No	No	8	Yes	Yes	Fiber optic	Mn	Mn	Yes	Mn	Yes	Yes

Showing the first 1000 rows.

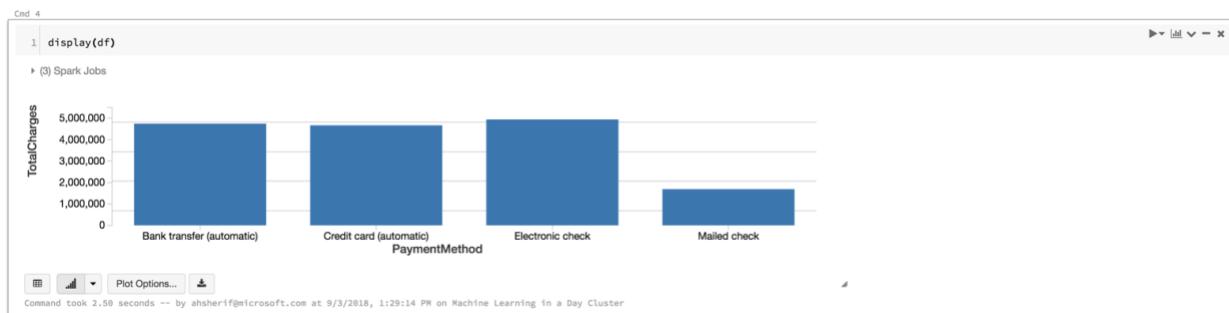
Bar □

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

3. 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 by clicking on the bar chart icon on the bottom of the DataFrame as seen in the following screenshot:

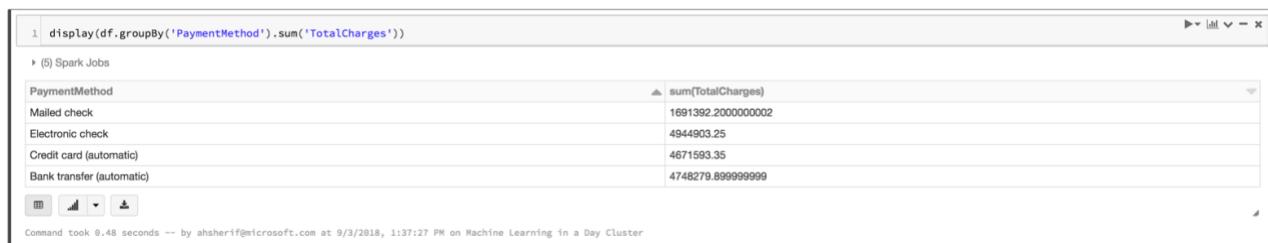


- Once we apply the **keys**, **groupings**, and **values** we can see the output directly within the cell of the notebook as seen in the following screenshot:



- Querying and manipulating DataFrames in 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:



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

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

The screenshot shows a Databricks notebook titled "churn classification (Python)". The notebook interface includes a header with user information and a "Free trial ends in 11 days. Upgrade to Premium in Azure Portal" message. Below the header are tabs for "Attached" (Machine Learning), "File", "View: Code", "Permissions", "Run All", and "Clear". The main area contains two code cells:

Cmd 6

```
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

Cmd 7

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

Spark Jobs (1)

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

Command took 0.71 seconds -- by ahsherif@microsoft.com at 9/3/2018, 1:53:24 PM on Machine Learning in a Day Cluster

9. 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: Implementing 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 scientist's 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())
```

Cmd 10

Churn	count
No	5163
Yes	1869

Command took 0.85 seconds -- by ahsherif@microsoft.com at 9/3/2018, 8:05:14 PM on Machine Learning in a Day Cluster

Cmd 11

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;
```

Cmd 11

Assignments	Total_Count
6	1992
0	1129
5	957
8	590
3	488
1	442
7	433
4	409
9	344
2	248

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:

```
Cmd 9
1 df.printSchema()

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)

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 (Python Script is whitespace sensitive, so keep the formatting on the tabs and indents, otherwise you will receive an error):

```
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.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
Cnd 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'))
```

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',

```

```

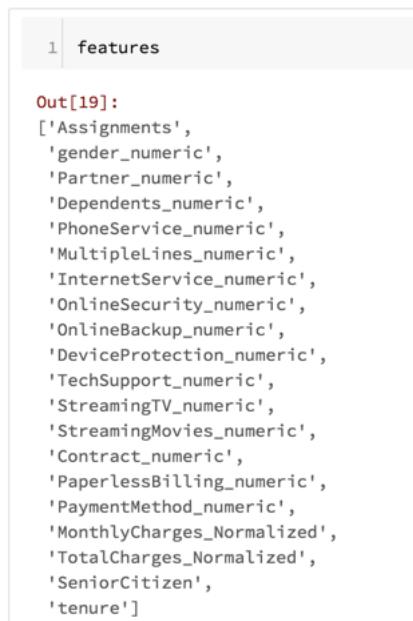
'Partner_numeric',
'Dependents_numeric',
'PhoneService_numeric',
'MultipleLines_numeric',
'InternetService_numeric',
'OnlineSecurity_numeric',
'OnlineBackup_numeric',
'DeviceProtection_numeric',
'TechSupport_numeric',
'StreamingTV_numeric',
'StreamingMovies_numeric',
'Contract_numeric',
'PaperlessBilling_numeric',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

'MonthlyCharges_Normalized',
'TotalCharges_Normalized',

'SeniorCitizen',
'tenure'
]

```

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



The screenshot shows a Jupyter Notebook cell with the title "1 | features". The cell contains the following code output:

```

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

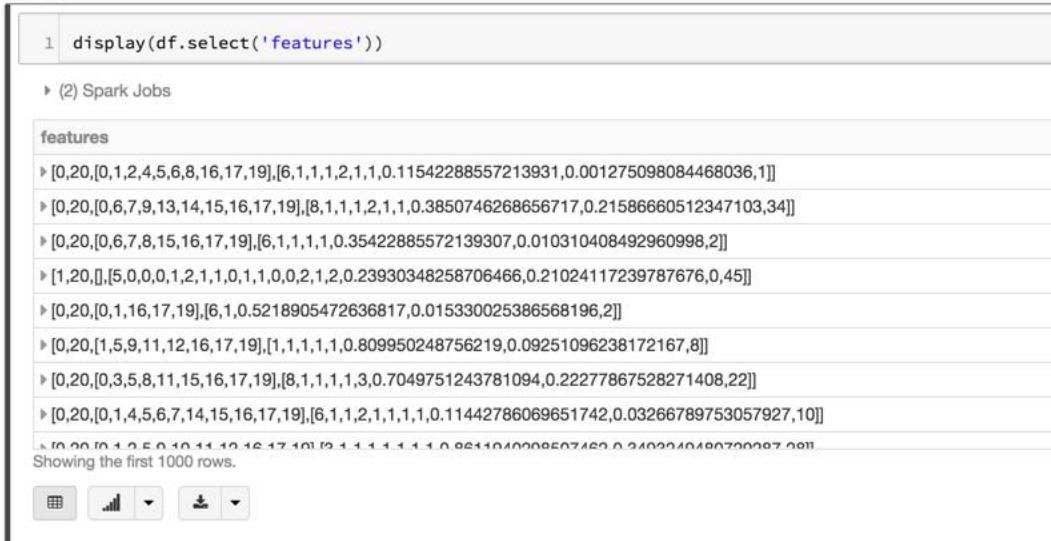
feature_vectors = VectorAssembler(
    inputCols = features,
    outputCol = "features")

```

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



```
1 display(df.select('features'))
▶ (2) Spark Jobs
```

features

```
▶ [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,],[5,0,0,0,1,2,1,1,0,1,1,0,2,1,2,0.23930348258706466,0.21024117239787676,0,45]
▶ [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]]
```

Showing the first 1000 rows.

Grid View | Data View | Download | Copy | Cell

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:



```
1 display(df.select('Churn', 'features'))
▶ (2) Spark Jobs
```

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,],[5,0,0,0,1,2,1,1,0,1,1,0,2,1,2,0.23930348258706466,0.21024117239787676,0,45]
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]]

Showing the first 1000 rows.

Grid View | Data View | Download | Copy | Cell

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. Logistic regression is a method for classifying data into discrete outcomes. For example, we might use logistic regression to classify a customer as **Y** or **N** for Churn.
2. 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.
3. 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)

▶ [1] trainDF: pyspark.sql.dataframe.DataFrame = [customerID: string, gender: string ... 38 more fields]
▶ [2] 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
```

4. The randomness is set to **1234** for reproducibility.
5. 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
```

6. 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)
```

7. 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)
```

8. 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'))
```

1	display(predictedDF.select('label', 'probability', 'prediction'))
▶ (3) Spark Jobs	
label	probability
0	▶ [1.2,][0.2453414700108091,0.7546585299891908]
0	▶ [1.2,][0.8531850801244731,0.14681491987552686]
0	▶ [1.2,][0.9156653394194328,0.0843346605805672]
0	▶ [1.2,][0.5615028194673514,0.4384971805326486]
1	▶ [1.2,][0.5183021325512303,0.4816978674487698]
1	▶ [1.2,][0.6351426553451417,0.36485734465485836]
1	▶ [1.2,][0.5640926911816028,0.4359073088183972]
0	▶ [1.2,][0.873286076625449,0.12671392337455098]
Showing the first 1000 rows.	

9. 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.
10. 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'))
```

1	display(predictedDF.crosstab('label', 'prediction'))
▶ (5) Spark Jobs	
label_prediction	0.0 1.0
1	171 206
0	910 105

11. We made **910 + 206 = 1,116** correct predictions and **171+105 = 276** false predictions for an overall accuracy of ~ **80%**
12. 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
```

13. 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 $(1 - \text{specificity})$.
14. 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
```

15. 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

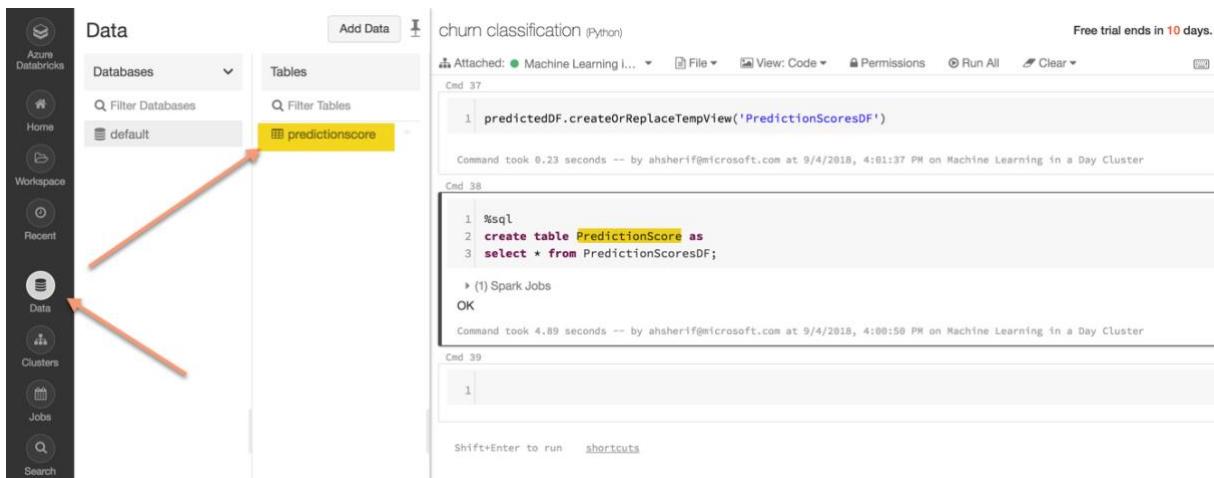
1. We are able to stream our DataFrames from Spark on Azure Databricks to other Business Intelligence tools such as Power BI, Tableau, and Alteryx.
2. First, we will create another view that called **PredictionScoresDF** using the following script:

```
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:



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 under the 'Data' tab. On the left sidebar, the 'Clusters' icon is highlighted. In the main area, a table named 'predictionscore' is selected. The 'Schema' section displays the columns and their data types:

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 'Sample Data' section shows two rows of sample data:

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 interi
0164-XAIRP	Female	0	No	No	24	Yes	No	No	No internet service	No internet service	No interi

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 interface under the 'Clusters' tab. On the left sidebar, the 'Clusters' icon is highlighted. The main area lists 'Interactive Clusters'. A cluster named 'Machine Learning in a Day Cluster' is selected, indicated by a green dot and highlighted with a yellow box. An orange arrow points from the left sidebar to this highlighted cluster.

Name	State	Nodes	Driver	Worker	Runtime	Creator	Actions
Machine Learning in a Day Cluster	Running	3	Standard_DS3_v2	Standard_DS3_v2	4.2 (includes Apache...)	ahsheel@microsoft...	1 0 ...

The 'Job Clusters' section below shows 'No clusters found'.

7. 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**.

Clusters / Machine Learning in a Day Cluster

Machine Learning in a Day Cluster

Configuration [Notebooks \(1\)](#) [Libraries \(0\)](#) [Event Log](#) [Spark UI](#) [Driver Logs](#) [Spark Cluster UI - Master](#)

Worker Type: Standard_DS3_v2 14.0 GB Memory, 4 Cores, 0.75 DBU

Min Workers: 2 Max Workers: 8 Enable autoscaling

Auto Termination: Terminate after 120 minutes of inactivity

JDBC/ODBC [Permissions](#)

Server Hostname: eastus.azuredatabricks.net

Port: 443

Protocol: HTTPS

HTTP Path: sql/protocolv1/o/2535530955171549/0831-164140-ayes780 (unique)
sql/protocolv1/o/2535530955171549/machine-learning-in-a-day-cluster (alias, not guaranteed unique)

JDBC URL: jdbc:hive2://eastus.azuredatabricks.net:443/default;transportMode=http;ssl=true;h
ttpPath\$sql/protocolv1/o/2535530955171549/0831-164140-ayes780
jdbc:hive2://eastus.azuredatabricks.net:443/default;transportMode=http;ssl=true;h
ttpPath\$sql/protocolv1/o/2535530955171549/machine-learning-in-a-day-cluster

[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:

Documentation > User Guide > Business Intelligence Tools

Business Intelligence Tools

Business Intelligence (BI) tools can connect to Azure Databricks clusters to query data in tables. Every Azure Databricks cluster runs a JDBC/ODBC server on the driver node. This section provides general instructions for connecting BI tools to Azure Databricks clusters, along with specific instructions for popular BI tools.

General instructions

- [Connecting BI Tools](#)

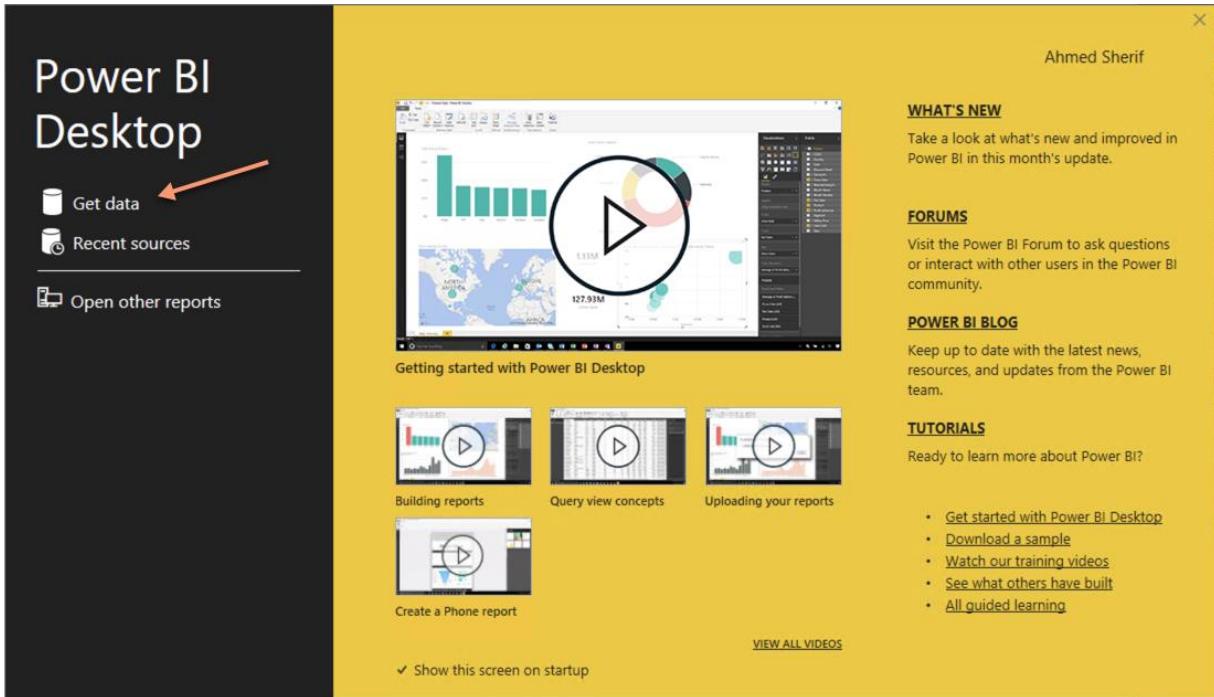
Tools

- Tableau
- Power BI**
- Alteryx
- Looker
- SQL Workbench/J

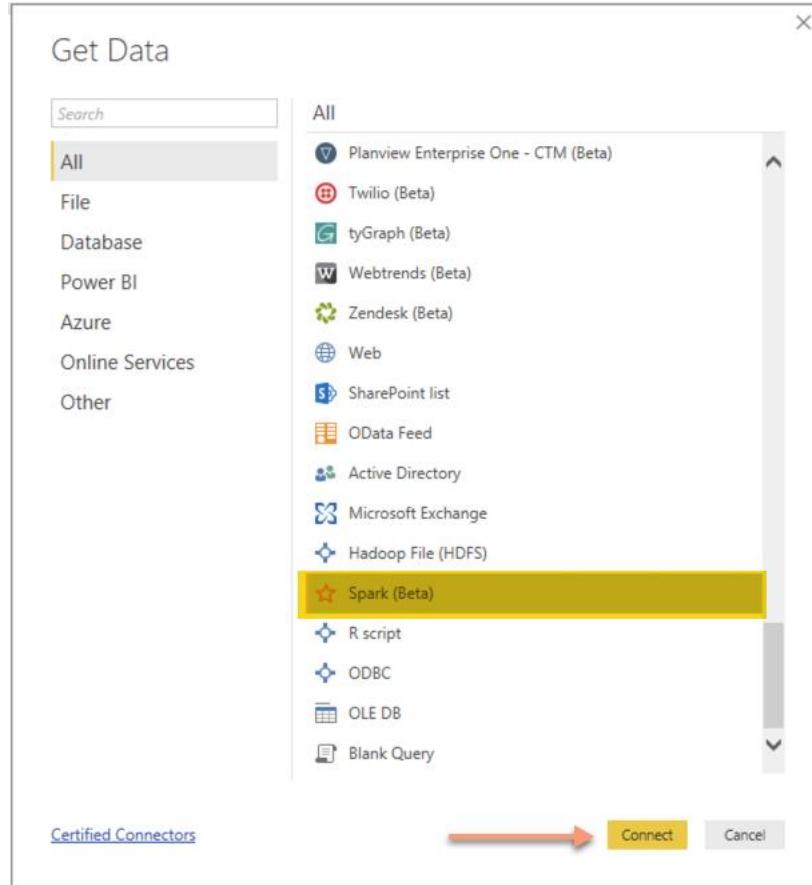
[PREVIOUS](#) [NEXT](#)

© Databricks 2018. All rights reserved. Apache, Apache Spark, Spark, and the Spark logo are trademarks of the Apache Software Foundation.
[Privacy Policy](#) | [Terms of Use](#)

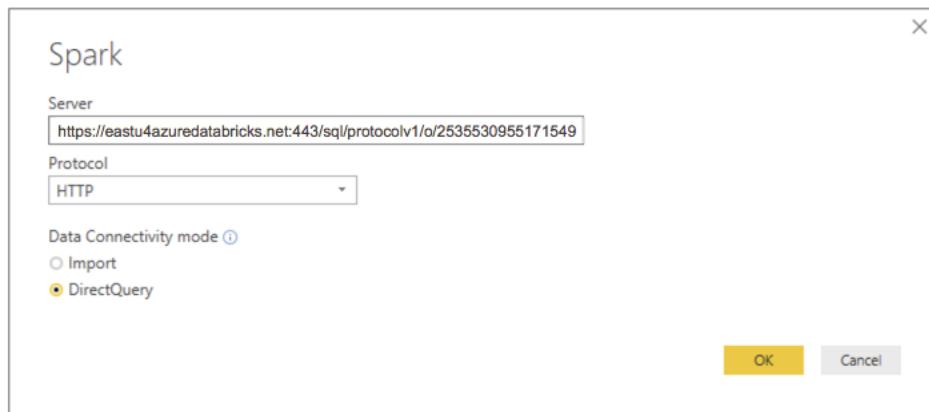
9. Databricks should have step-by-step directions for connecting Tables from Spark to Power BI as found in the following link: <https://docs.azuredatabricks.net/user-guide/bi/power-bi.html>
10. There should be a Power BI desktop installation already available in our Data Science Virtual machine that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



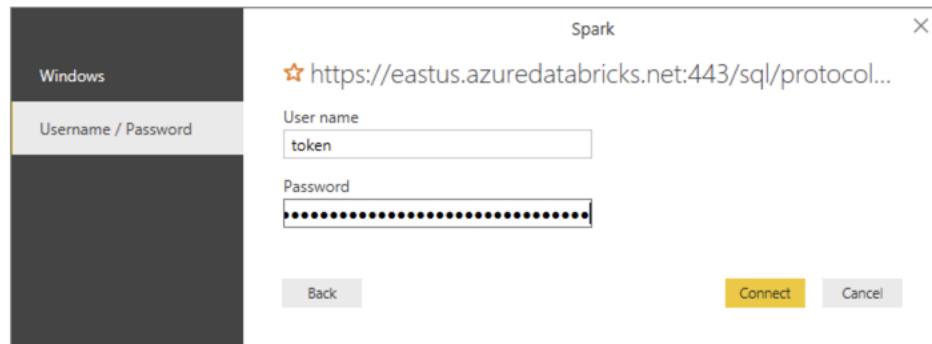
11. Select **Spark (Beta)** and **Connect** as seen in the following screenshot:



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:

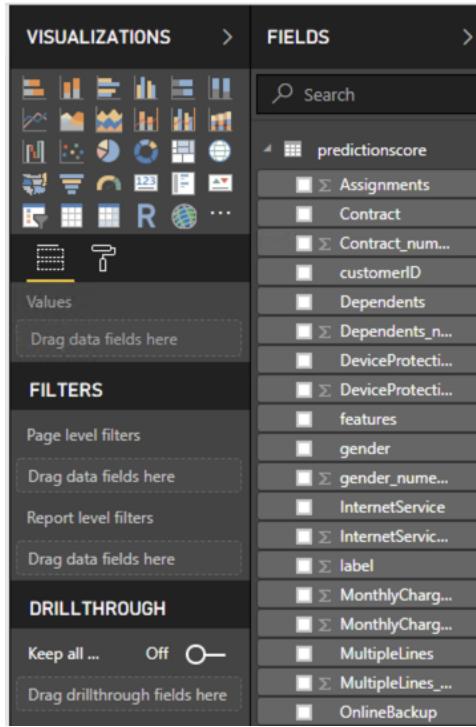


14. We can now view our data from the **predictionscore** table in Spark now viewed directly within Power BI as seen in the following screenshot:

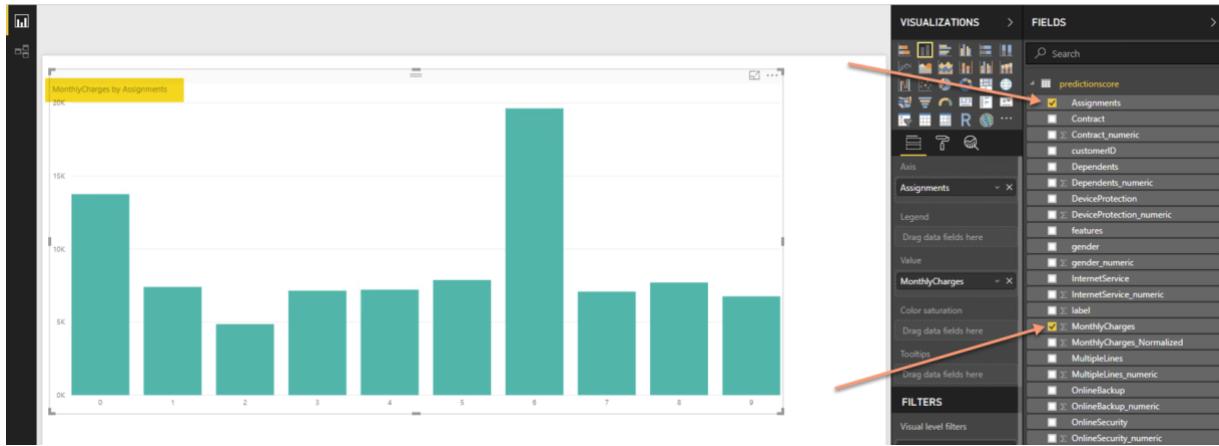
customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup
0064-SUDOG	Female		0 Yes	Yes	22	Yes	No	No	No internet service	No internet service
0164-XAIRP	Female		0 No	No	24	Yes	No	No	No internet service	No internet service
0230-UBYRQ	Male		1 Yes	No	63	No	No phone service	DSL	Yes	No
0298-XACET	Male		0 Yes	Yes	52	No	No phone service	DSL	No	Yes
0422-0XKAP	Female		0 Yes	No	51	Yes	Yes	Fiber optic	No	No
0644-OQMDK	Male		1 No	No	4	Yes	No	Fiber optic	No	No
0657-DOGUM	Female		0 Yes	No	48	Yes	Yes	DSL	Yes	No
0674-EYVZ	Female		0 No	No	1	Yes	No	DSL	No	No
0872-CASJZ	Male		0 Yes	No	59	Yes	Yes	DSL	Yes	No
0886-QGENL	Female		1 Yes	No	27	Yes	No	Fiber optic	Yes	No
0902-RPHOF	Male		0 No	No	38	Yes	No	No	No internet service	No internet service

The data in the preview has been truncated due to size limits.

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:



17. We have successfully completed this workshop!!