



# Microsoft Machine Learning in a Day Workshop

Student Guide

March 2019

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

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

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

© 2018 Microsoft Corporation. All rights reserved.

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

# Contents

<b>Machine Learning in a Day student guide.....</b>	<b>1</b>
Abstract and learning objectives.....	1
Architecture .....	2
Section 1: Set up a Data Science Virtual Machine (DSVM) on Azure Portal.....	3
Section 2: Download Telco dataset and create Blob Storage Account and Container access .....	9
Section 3: Upload CSV file to Blob Storage Container Manually.....	15
Section 4: Build a clustering unsupervised model in Azure Machine Learning Studio .....	18
Section 5: Set up a Spark Cluster on Databricks and connect to Azure Blob Storage Account.....	32
Section 6: Implement Feature Engineering Techniques to Enhance data for Machine Learning.....	40
Section 7: Apply Classification Supervised Model with Logistic Regression on Azure Databricks .....	49
Section 8: Export predicted DataFrame from Spark on Azure Databricks to Power BI for visualizations.....	52



# Machine Learning in a Day student guide

## Abstract and learning objectives

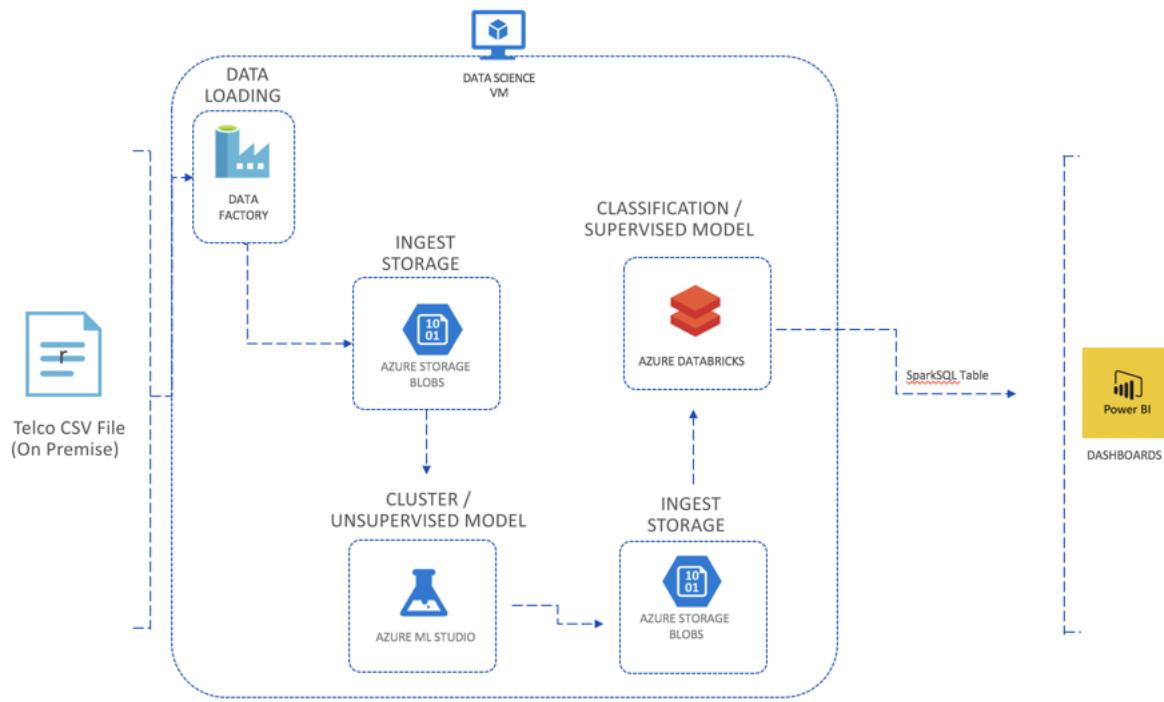
In this workshop, you will build a Machine Learning model that will be used to predict whether customers remain with business or move onto other services based on their behavior, demographics, and spending. You will leverage several Microsoft products and services on Microsoft Azure to deliver the Machine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

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

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

## Architecture

### MACHINE LEARNING IN A DAY WORKSHOP



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

1. The use of this documentation requires the subscription of a Microsoft Azure account either from a personal email account or a work email account.
2. Visit [portal.azure.com](https://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. On the left, there's a sidebar with various service icons and a 'Create a resource' button highlighted with a yellow box. The main area has a search bar with 'data science virtual machine' typed in. Below it is a table with columns 'NAME', 'PUBLISHER', and 'CATEGORY'. The first row, 'Data Science Virtual Machine - Windows 2012', is also highlighted with a yellow box. To the right of the table is a detailed description of the DSVM, mentioning tools like R, Python, Julia, and various databases. At the bottom right of the main panel, there's a '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.

### Create a virtual machine

[Basics](#) [Disks](#) [Networking](#) [Management](#) [Guest config](#) [Tags](#) [Review + create](#)

Create a virtual machine that runs Linux or Windows. Select an image from Azure marketplace or use your own customized image. Complete the Basics tab then Review + create to provision a virtual machine with default parameters or review each tab for full customization.

Looking for classic VMs? [Create VM from Azure Marketplace](#)

**PROJECT DETAILS**

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

\* Subscription [CCG Training Sandbox](#)  
 \* Resource group [DS-Practice](#) [Create new](#)

**INSTANCE DETAILS**

\* Virtual machine name [dsvmMLinaday](#)  
 \* Region [East US](#)  
 Availability options [No infrastructure redundancy required](#)  
 \* Image [Data Science Virtual Machine - Windows 2012](#) [Browse all images and disks](#)  
 \* Size [Standard DS3 v2](#)  
 4 vcpus, 14 GB memory [Change size](#)

**ADMINISTRATOR ACCOUNT**

\* Username [demouser](#)  
 \* Password [#MLinaDay2019](#)  
 \* Confirm password [#MLinaDay2019](#)

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 **#MLinaDay2019** (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).

- After clicking “Change size,” choose the following size VM (**D2S\_v3**) for the purposes of this workshop as seen in the following screenshot:

**Select a VM size**  
 Browse available virtual machine sizes and their features

Search by VM size... [X](#) [Clear all filters](#)

Size : Small (0-4) [Generation : Current](#) [Family : General purpose](#) [Premium disk : Supported](#) [+ Add filter](#)

Showing 13 of 198 VM sizes. | Subscription: CCG Training Sandbox | Region: East US | Current size: Standard\_DS3\_v2

VM SIZE	OFFERING	FAMILY	VCPUS	RAM (GB)	DATA DISKS	MAX IOPS	TEMPORARY STORA...	Premium disk sup...	COST/MONTH (ESTI...)
B1ms	Standard	General purpose	1	2	2	800	4 GB	Yes	\$18.30
B1s	Standard	General purpose	1	1	2	400	4 GB	Yes	\$10.42
B2ms	Standard	General purpose	2	8	4	2400	16 GB	Yes	\$67.85
B2s	Standard	General purpose	2	4	4	1600	8 GB	Yes	\$36.90
B4ms	Standard	General purpose	4	16	8	3600	32 GB	Yes	\$135.41
<b>D2S_v3</b>	Standard	General purpose	<b>2</b>	<b>8</b>	<b>4</b>	<b>3200</b>	<b>16 GB</b>	<b>Yes</b>	<b>\$139.87</b>
D4s_v3	Standard	General purpose	4	16	8	6400	32 GB	Yes	\$279.74
DS1_v2	Standard	General purpose	1	3.5	4	3200	7 GB	Yes	\$93.74

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

Create a virtual machine

Basics • Disks Networking Management Guest config Tags Review + create

Configure monitoring and management options for your VM.

**MONITORING**

Boot diagnostics  On  Off

OS guest diagnostics  On  Off

\* Diagnostics storage account

**IDENTITY**

System assigned managed identity  On  Off

**AUTO-SHUTDOWN**

Enable auto-shutdown  On  Off

Shutdown time

Time zone

Notification before shutdown  On  Off

**Review + create** **Previous** **Next : Guest config >**

6. In 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:

### Create a virtual machine

 Validation passed

Basics Disks Networking Management Guest config Tags Review + create

**PRODUCT DETAILS**

Data Science Virtual Machine - Windows 2012 by Microsoft	Not covered by credits ⓘ <b>0.0000 USD/hr</b>
Standard D2s v3 by Microsoft	Subscription credits apply ⓘ <b>0.1880 USD/hr</b> <a href="#">Pricing for other VM sizes</a>

**TERMS**

By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Marketplace offering(s) listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the offering(s), with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. See the [Azure Marketplace Terms](#) for additional details.

**BASICS**

Subscription	CCG Training Sandbox
Resource group	DS-Practice
Virtual machine name	mliadDSVMbees
Region	East US
Availability options	No infrastructure redundancy required
Username	demouser
Already have a Windows license?	No

**DISKS**

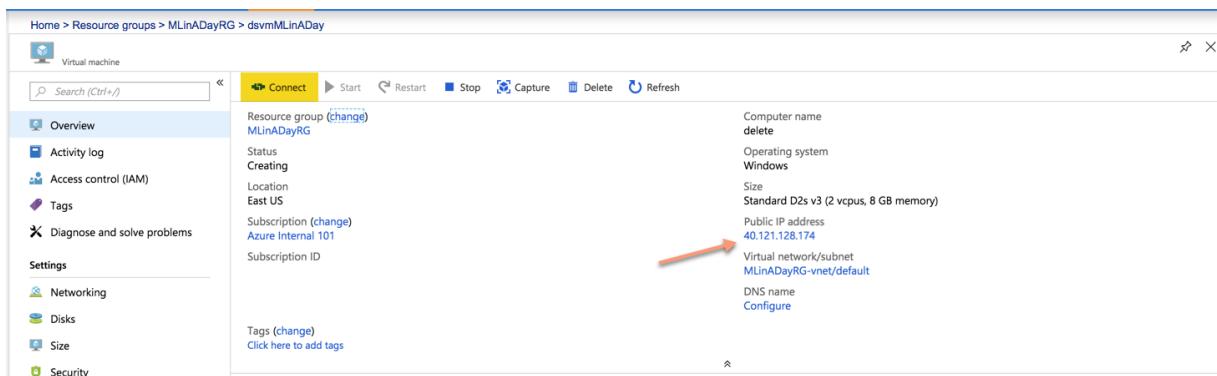
OS disk type	Premium SSD
Use managed disks	Yes

**NETWORKING**

Virtual network	CCG-INTERNAL
-----------------	--------------

**Buttons:** Create, Previous, Next, Download a template for automation

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



Home > Resource groups > MLinADayRG > dsvmMLinADay

Virtual machine

Search (Ctrl+F)

Connect Start Restart Stop Capture Delete Refresh

Overview

Activity log Access control (IAM) Tags Diagnose and solve problems

Settings

Networking Disks Size Security

Resource group (changed) MLinADayRG

Status Creating

Location East US

Subscription (change) Azure Internal 101

Subscription ID

Tags (change) Click here to add tags

Computer name delete

Operating system Windows

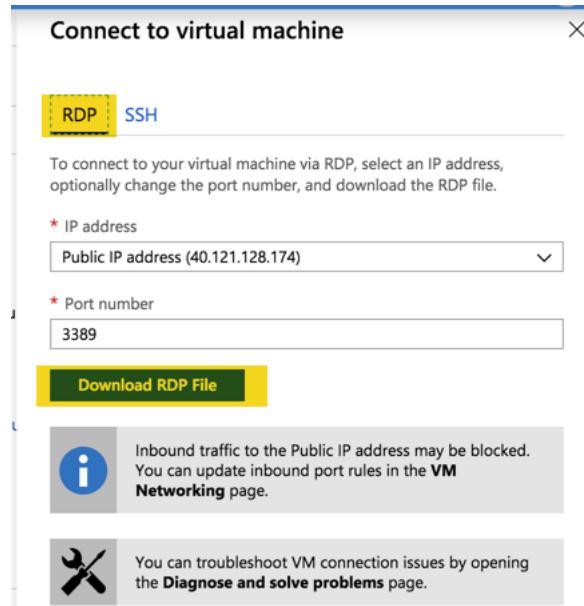
Size Standard D2s v3 (2 vcpus, 8 GB memory)

Public IP address 40.121.128.174

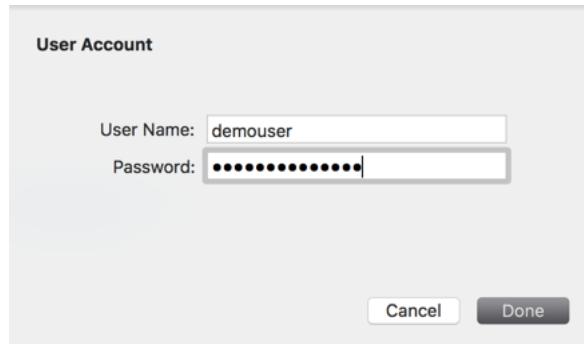
Virtual network/subnet MLinADayRG-vnet/default

DNS name Configure

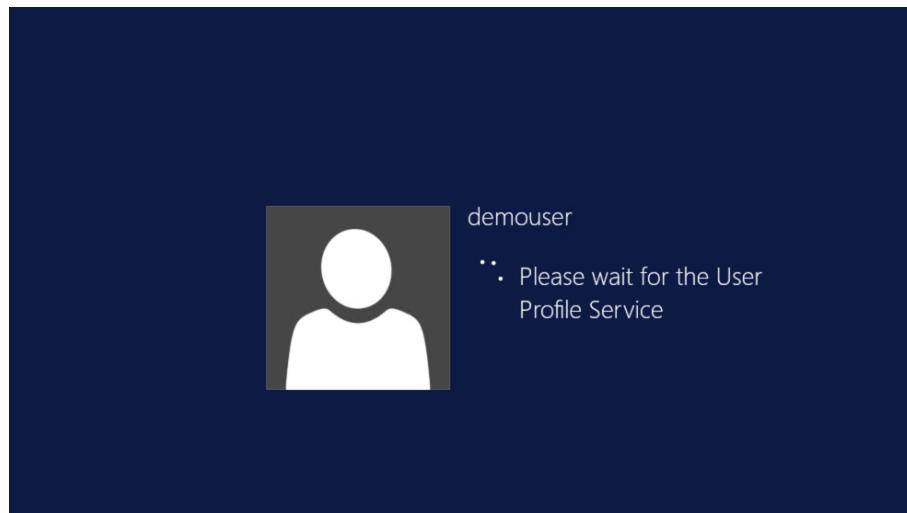
- Make sure that your DSVM is currently running and click on the **Connect** button to enter your virtual machine using **RDP** as seen in the following screenshot:



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



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

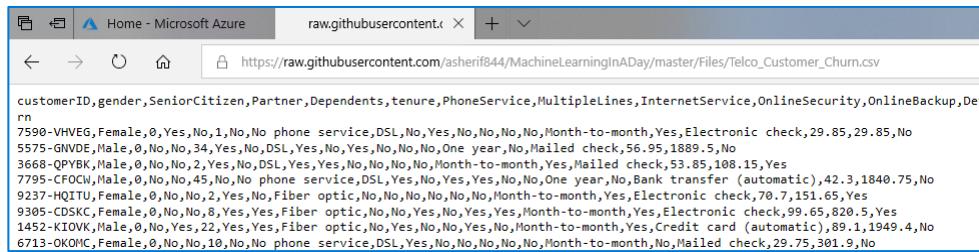


11. Once you log in using your credentials using the RDP approach, you should see the following remote server for the data science virtual machine as seen in the following screenshot:



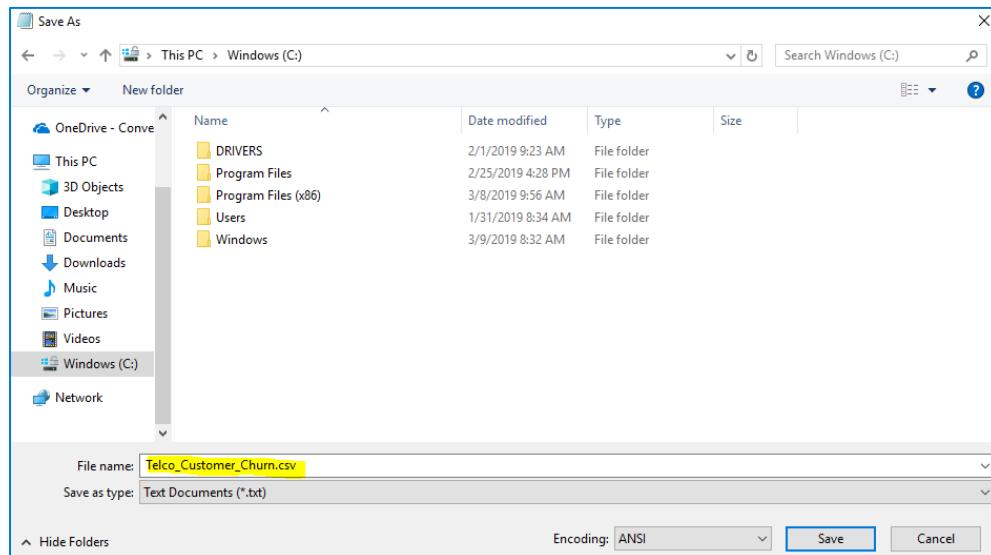
## Section 2: Download Telco dataset and create Blob Storage Account and Container access

- The Telco Churn Dataset ([TELCO\\_CUSTOMER\\_CHURN.csv](https://raw.githubusercontent.com/asherif844/MachineLearningInADay/master/Files/Telco_Customer_Churn.csv)) can be accessed through the following link:  
[https://raw.githubusercontent.com/asherif844/MachineLearningInADay/master/Files/Telco\\_Customer\\_Churn.csv](https://raw.githubusercontent.com/asherif844/MachineLearningInADay/master/Files/Telco_Customer_Churn.csv)



The screenshot shows a Microsoft Edge browser window with the URL [https://raw.githubusercontent.com/asherif844/MachineLearningInADay/master/Files/Telco\\_Customer\\_Churn.csv](https://raw.githubusercontent.com/asherif844/MachineLearningInADay/master/Files/Telco_Customer_Churn.csv). The page displays a large block of comma-separated values (CSV) data, which is the Telco Customer Churn dataset. The data includes columns such as customerID, gender, SeniorCitizen, Partner, Dependents, tenure, PhoneService, MultipleLines, InternetService, OnlineSecurity, OnlineBackup, Dev, and various service usage and payment details.

- Place your cursor in the screen and press **Ctrl + A** to select all the text. Next press **Ctrl + C** to copy all the text.
- Open a new notepad editor by clicking your windows menu, typing Notepad, and selecting the Notepad application when it appears in the search results.
- In your new Notepad window, press **Ctrl + V** to paste the data.
- Select **File** and **Save As...**
- For now, the dataset can be saved in the C:\ drive as seen in the following screenshot. Be sure to end the file name with **.csv** to specify what type of data the file contains.



- 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** as seen in the following screenshot:

The screenshot shows the Azure Marketplace interface. On the left, there's a sidebar with a 'Create a resource' button highlighted with a red box. Below it are sections for 'All services', 'FAVORITES' (containing 'Dashboard', 'All resources', 'Resource groups', 'App Services', 'Function Apps', and 'SQL databases'), and a 'My Saved List' section with 2 items. The main area is titled 'Marketplace' and shows a search bar with 'blob' typed in, also highlighted with a red box. A red arrow points from the search bar to the search icon. To the right, the results are listed under 'Everything', including 'Compute', 'Networking', 'Storage', and 'Web'. The 'Storage' section is expanded, showing a list of storage account types. One item, 'Storage account - blob, file, table, queue', is highlighted with a red box.

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

The screenshot shows the 'Create storage account' wizard in the 'Basics' step. At the top, there are tabs for 'Basics' (highlighted with a yellow box), 'Advanced', 'Tags', and 'Review + create'. Below the tabs, a descriptive text explains Azure Storage as a Microsoft-managed service providing cloud storage. It includes options to select a subscription ('Azure Internal 101') and a resource group ('MLinADayRG'). Under 'PROJECT DETAILS', there's a note about deployment models, with a link to choose classic deployment model. The 'INSTANCE DETAILS' section contains fields for 'Storage account name' (set to 'blobminaday'), 'Location' (set to 'East US'), 'Performance' (set to 'Standard'), 'Account kind' (set to 'StorageV2 (general purpose v2)'), 'Replication' (set to 'Locally-redundant storage (LRS)'), and 'Access tier (default)' (set to 'Cool'). At the bottom, there are navigation buttons: 'Review + create' (highlighted with a blue box), 'Previous', and 'Next : Advanced >'.

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

The screenshot shows the 'Create storage account' wizard with the 'Advanced' tab selected. Under 'SECURITY', 'Secure transfer required' is set to 'Disabled'. Under 'VIRTUAL NETWORKS', 'Allow access from' is set to 'All networks'. Under 'DATA LAKE STORAGE GEN2 (PREVIEW)', 'Hierarchical namespace' is set to 'Disabled'.

11. 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 with the 'Review + create' tab selected. A green bar at the top indicates 'Validation passed'. The 'BASICS' section shows the following configuration:

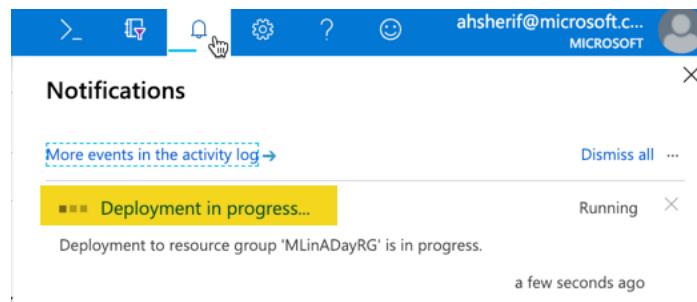
Subscription	Azure Internal 101
Resource group	MLinADayRG
Location	East US
Storage account name	blobmlinaday
Deployment model	Resource manager
Account kind	StorageV2 (general purpose v2)
Replication	Locally-redundant storage (LRS)
Performance	Standard
Access tier (default)	Cool

The 'ADVANCED' section shows:

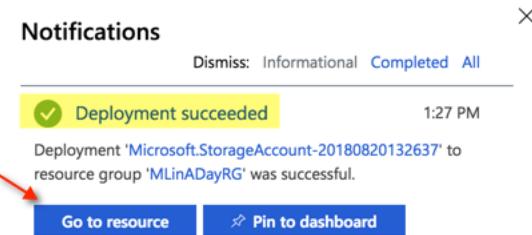
Secure transfer required	Disabled
Allow access from	All networks
Hierarchical namespace	Disabled

At the bottom, the 'Create' button is highlighted in yellow.

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



14. 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:



15. \*\*\*\*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:

This screenshot shows the 'blobmlinaday - Configuration' page in the Azure portal. The left sidebar lists navigation options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Storage Explorer (preview), Settings, Access keys, CORS, Configuration (selected), Encryption, Shared access signature, Firewalls and virtual networks, Advanced Threat Protection (pr...), Properties, and Locks. The main content area shows the cost of the storage account depends on usage and options chosen. It specifies 'Account kind: Storage (general purpose v1)'. A warning message states: 'This account can be upgraded to a General Purpose v2 account with additional features. Upgrading is permanent and will result in billing changes.' with an 'Upgrade' button. Under 'Performance', 'Standard' is selected. In the 'Configuration' section, 'Secure transfer required' is set to 'Disabled' (highlighted in yellow). Other settings include 'Locally-redundant storage (LRS)', 'Azure Active Directory authentication for Azure Files (Preview)' (disabled), and 'Data Lake Storage Gen2 (preview)' (disabled).

16. Back in the Overview section, select **Blobs** under the **Services** section as seen in the following screenshot:

blobmlinaday  
Storage account

Search (Ctrl+ /) | Open in Explorer | Move | Delete | Refresh

Resource group (change)  
MLinADayRG

Status  
Primary: Available

Location  
East US

Subscription (change)  
Azure Internal 101

Subscription ID  
ba3edd26-e2b1-4cc5-a19e-5bd21d7e9f5d

Tags (change)  
Click here to add tags

**Services**

- Blobs REST-based object storage for unstructured data [Learn more](#)

17. 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.**
18. 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:

blobmlinaday - Blobs  
Storage account

Search (Ctrl+ /) | + Container | Refresh | Delete

New container

\* Name: churn

Public access level: Container (anonymous read access for containers and blobs)

OK | Cancel

19. 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:

The screenshot shows the 'blobmlinaday - Access keys' page in the Azure portal. The left sidebar lists various account settings like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Storage Explorer (preview), and Settings. Under Settings, the 'Access keys' tab is selected and highlighted with a yellow box. A large orange arrow points from the top right towards the 'key1' section. The 'key1' section contains a key value: 2Le22d38roB6lFqAeds/dFXRfc6VoKUDBXqcsjhzRv8r3YDaB1xI7FdmyzLcaH04NAFrL+OQmXsNlo4NzICg==. Below it is a connection string: DefaultEndpointsProtocol=https;AccountName=blobmlinaday;AccountKey=2Le22d38roB6lFqAeds/dFXRfc6VoKUDBXqcsjhzRv8r3YDaB1xI7Fd... . The 'key2' section is partially visible below it.

20. Once the container has been successfully created, we can move onto the next section of adding data to the container. We'll do it manually to keep this workshop simple.

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

- 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. In the main content area, the 'blobmlinaday' storage account is displayed. Under the 'Blob service' section, the 'Blobs' option is highlighted with a yellow box. The right pane provides detailed information about the storage account, including its name, location (East US), and account kind (StorageV2). It also lists services like File, Table, and Queue.

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

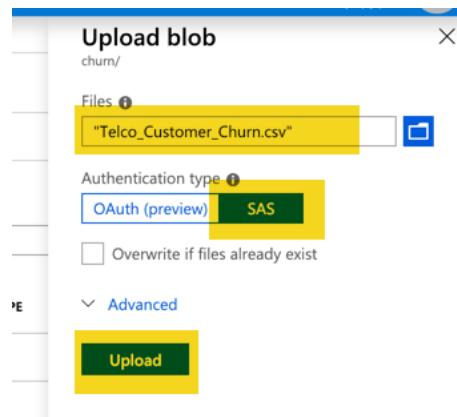
The screenshot shows the 'blobmlinaday - Blobs' blade. On the left, the 'Blobs' option under the 'Blob service' section is highlighted with a yellow box. The main area displays a table of containers. One row, containing the word 'churn', is highlighted with a yellow box. The table includes columns for NAME, LAST MODIFIED, PUBLIC ACCESS ..., and LEASE STATE.

NAME	LAST MODIFIED	PUBLIC ACCESS ...	LEASE STATE
churn	11/25/2018, 11:20:31 AM	Container	Available

- 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 has tabs for Upload, Refresh, Delete, Acquire lease, Break lease, View snapshots, and Create snapshot. A search bar at the top says 'Search blobs by prefix (case-sensitive)'. Below it, a table lists blobs with columns for NAME, MODIFIED, ACCESS TIER, and BLOB TYPE. A message says 'No blobs found.'

4. Select your file from your local hard drive and select Upload as seen in the following screenshot. The Authentication type option no longer appears in the latest version:



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

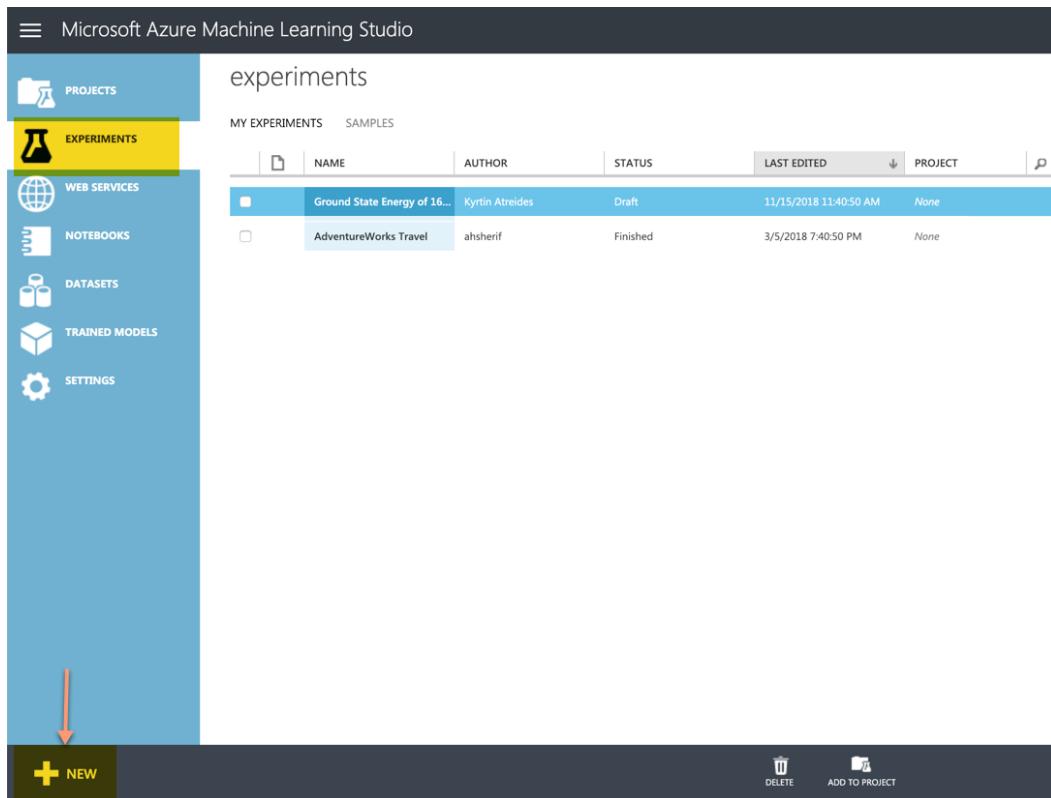
The screenshot shows the 'churn' container page again. The table now lists the 'Telco\_Customer\_Churn.csv' file. The columns include NAME, MODIFIED, ACCESS TIER, BLOB TYPE, SIZE, and LEASE STATE. The file details are: NAME 'Telco\_Customer\_Churn.csv', MODIFIED '11/25/2018, 11:28:06 AM', ACCESS TIER 'Cool (Inferred)', BLOB TYPE 'Block blob', SIZE '954.59 kB', and LEASE STATE 'Available'.

6. 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:

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

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

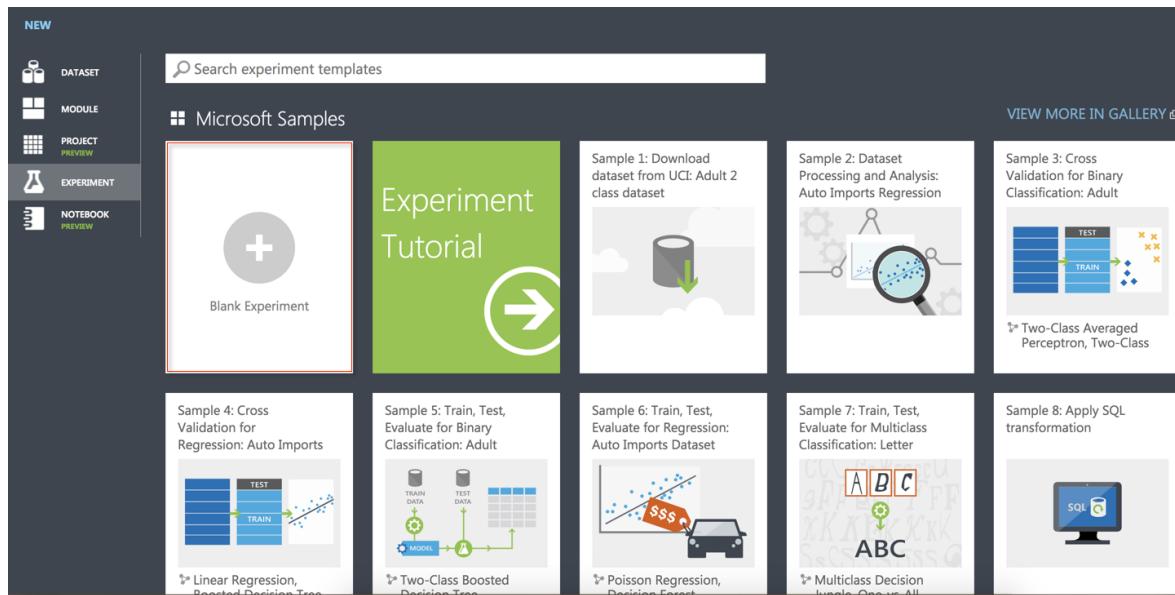
1. Clustering is the process of grouping similar entities together. The goal of this unsupervised machine learning technique is to find similarities in the fields that we have about our customers and group them into 10 distinct clusters.
2. Sign into Azure Machine Learning Studio using your Azure login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



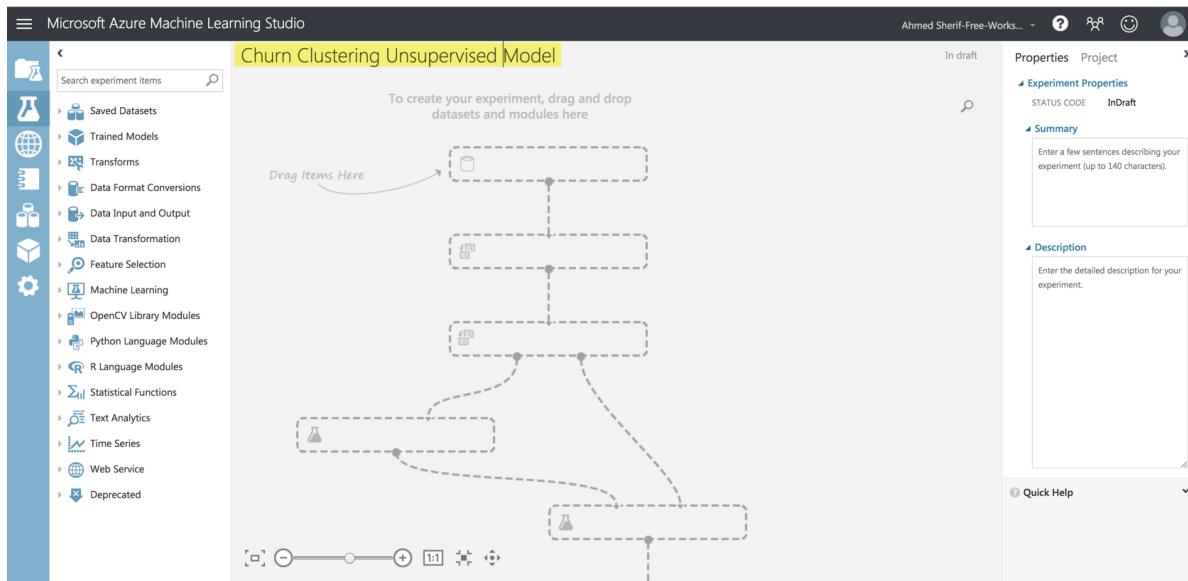
The screenshot shows the Microsoft Azure Machine Learning Studio interface. On the left, there is a vertical navigation bar with icons for Projects, Experiments (which is highlighted in yellow), Web Services, Notebooks, Datasets, Trained Models, and Settings. The main area is titled "experiments" and contains a table titled "MY EXPERIMENTS". The table has columns for NAME, AUTHOR, STATUS, LAST EDITED, and PROJECT. There are two rows: one for "Ground State Energy of 16..." (Author: Kyrrin Atreides, Status: Draft, Last Edited: 11/15/2018 11:40:50 AM, Project: None) and one for "AdventureWorks Travel" (Author: ahsherif, Status: Finished, Last Edited: 3/5/2018 7:40:50 PM, Project: None). At the bottom of the screen, there is a dark footer bar with a yellow "+ NEW" button, a "DELETE" icon, and an "ADD TO PROJECT" icon. An orange arrow points from the text in step 4 to the "+ NEW" button.

	NAME	AUTHOR	STATUS	LAST EDITED	PROJECT
<input type="checkbox"/>	Ground State Energy of 16...	Kyrrin 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

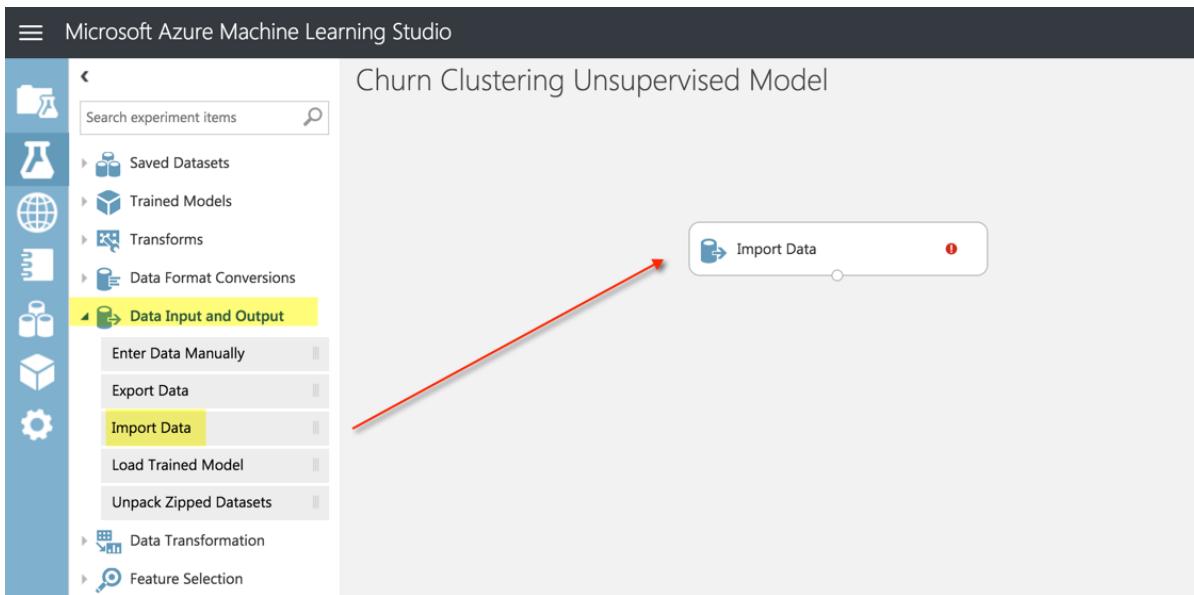
4. Next click on **+NEW** on the button of the screen and select **Blank Experiment** as seen in the following screenshot:



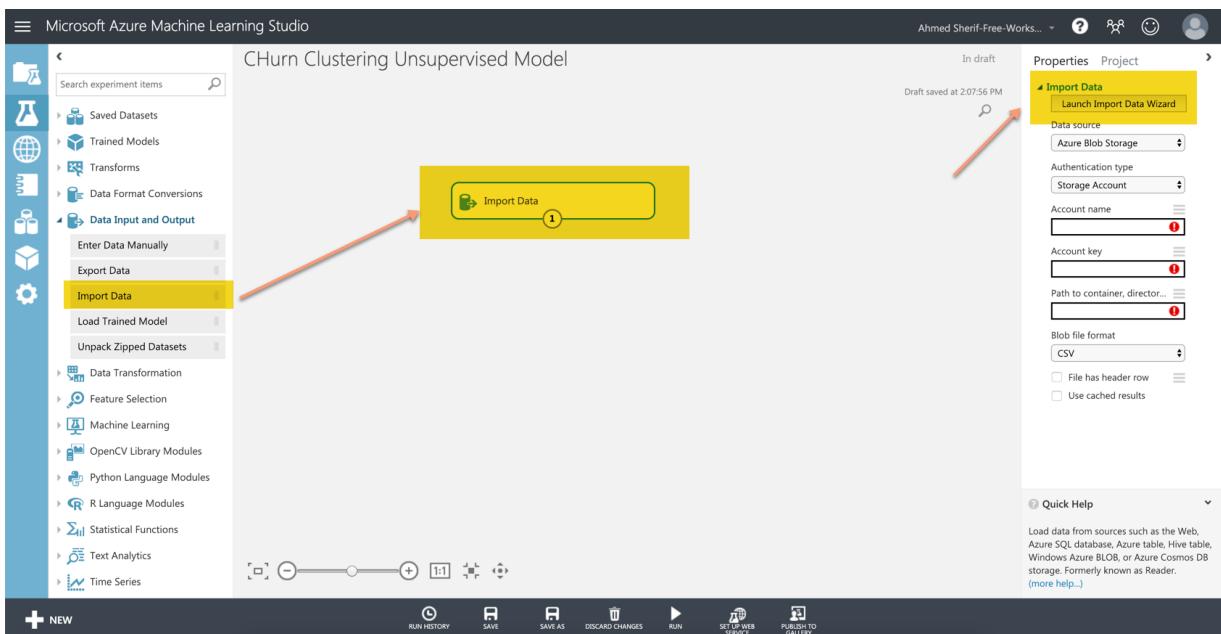
5. Click on the title of the experiment and change the name to **Churn Clustering Unsupervised Model** as seen in the following screenshot:



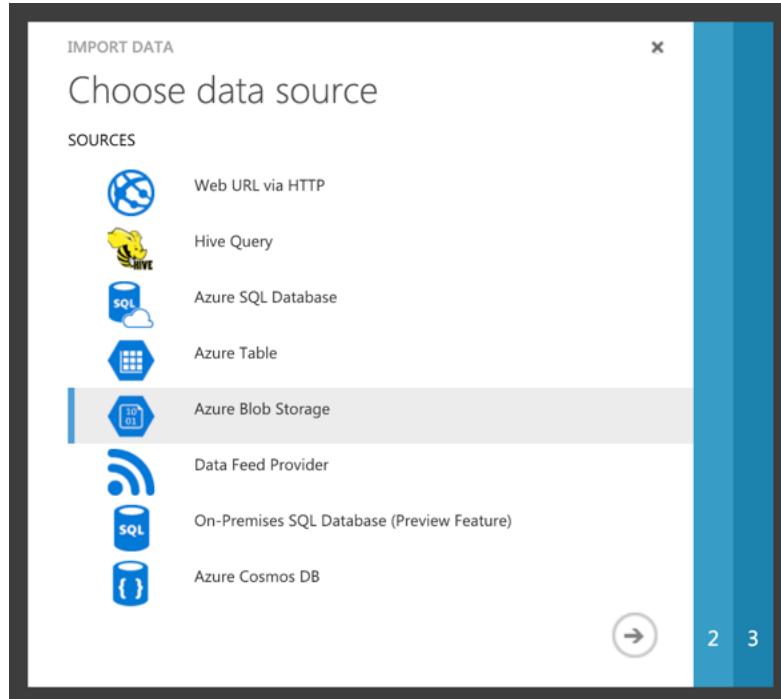
6. Expand the **Data Input and Output** category and drag the **Import Data** into the Workflow as seen in the following screenshot:



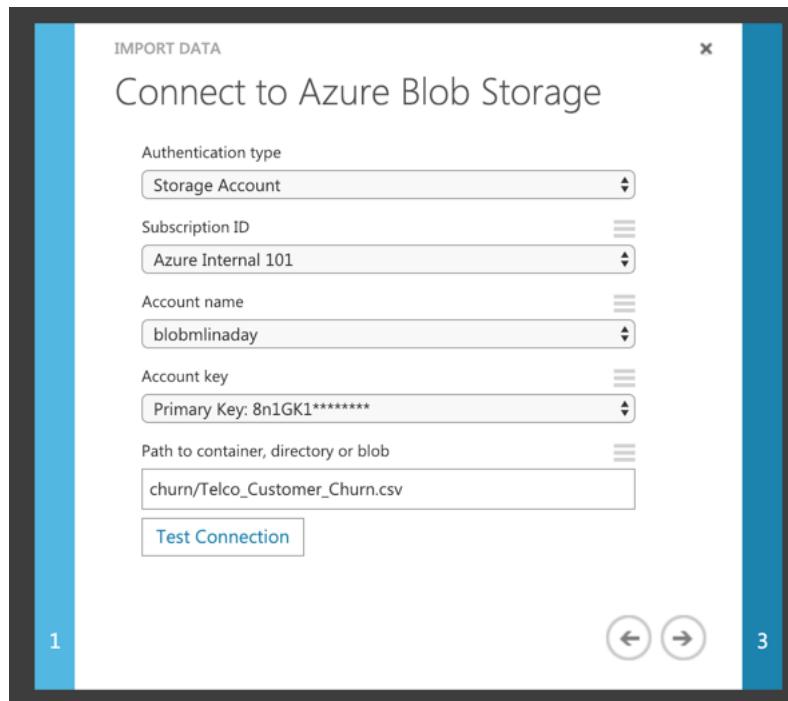
- As you click on the **Import Data** icon in the workflow, you will have the option to configure your data to pull from the Azure Blob Container you created a few sections ago using the Launch Import Wizard as seen in the following screenshot:



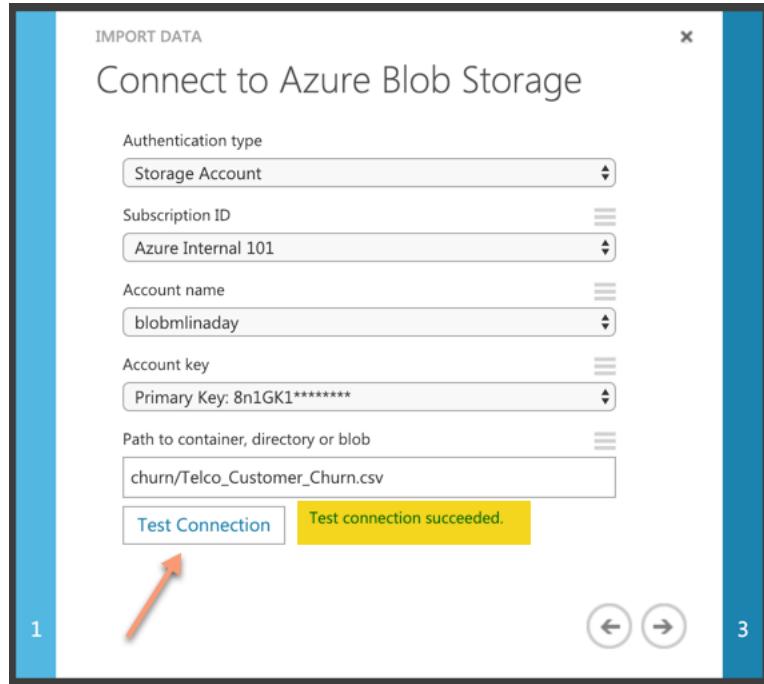
- Select Azure Blob Storage



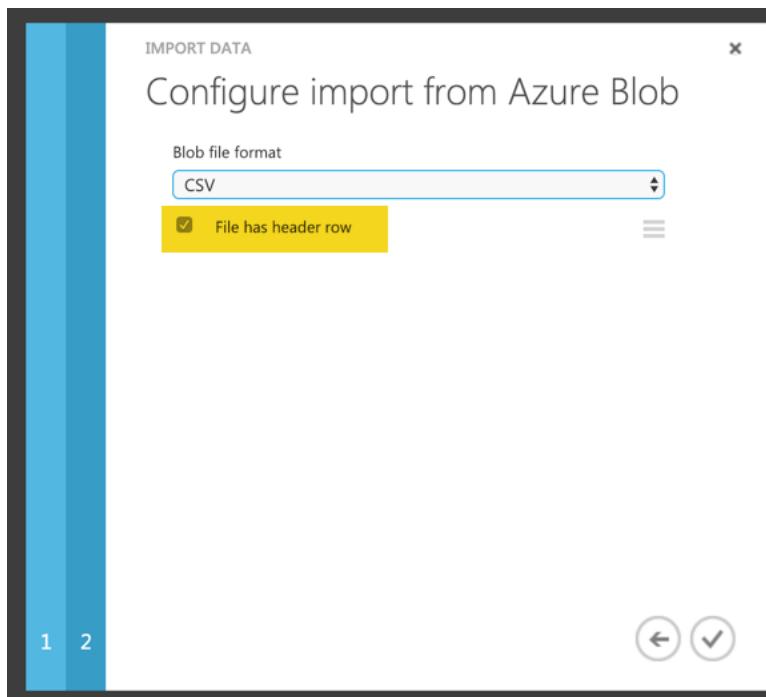
9. Specify the following credentials from the drop-down boxes for your accounts:



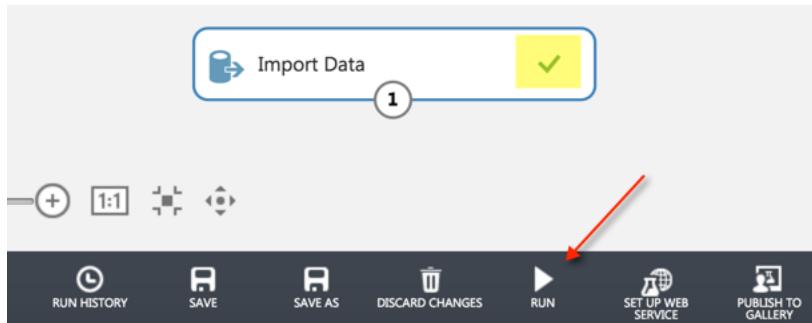
10. Test the connection to make sure the credentials work. Once it is validated you should see a green checkmark as seen in the following screenshot:



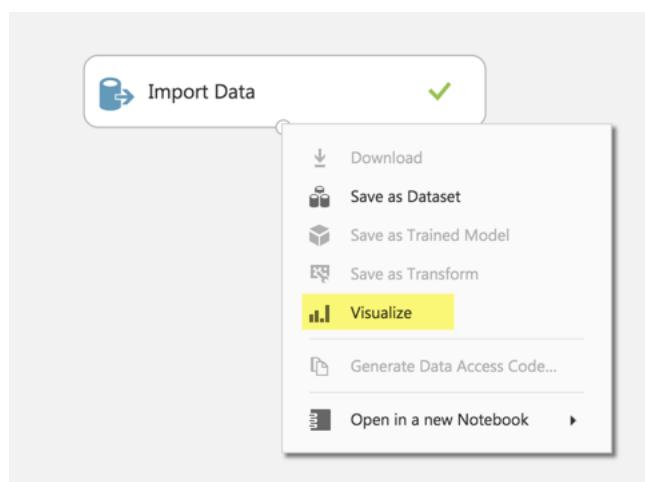
11. Check to include a **file header row** for the data and maintain a **CSV** file structure as seen in the following screenshot:



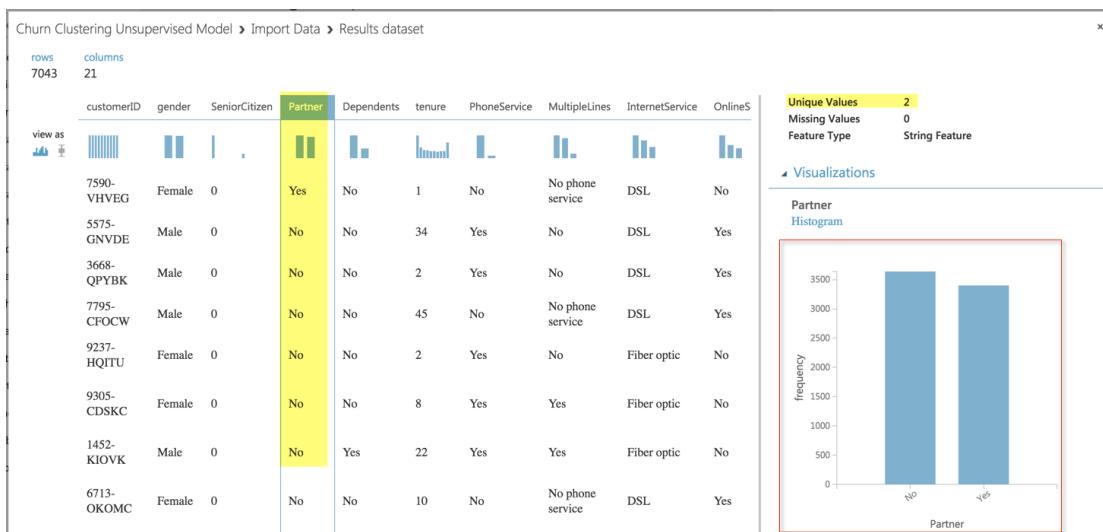
12. Back in the Azure ML Studio workflow, the credentials can be confirmed by clicking on the **RUN** icon at the bottom of the screen and seeing the green check mark next to **Import Data**.



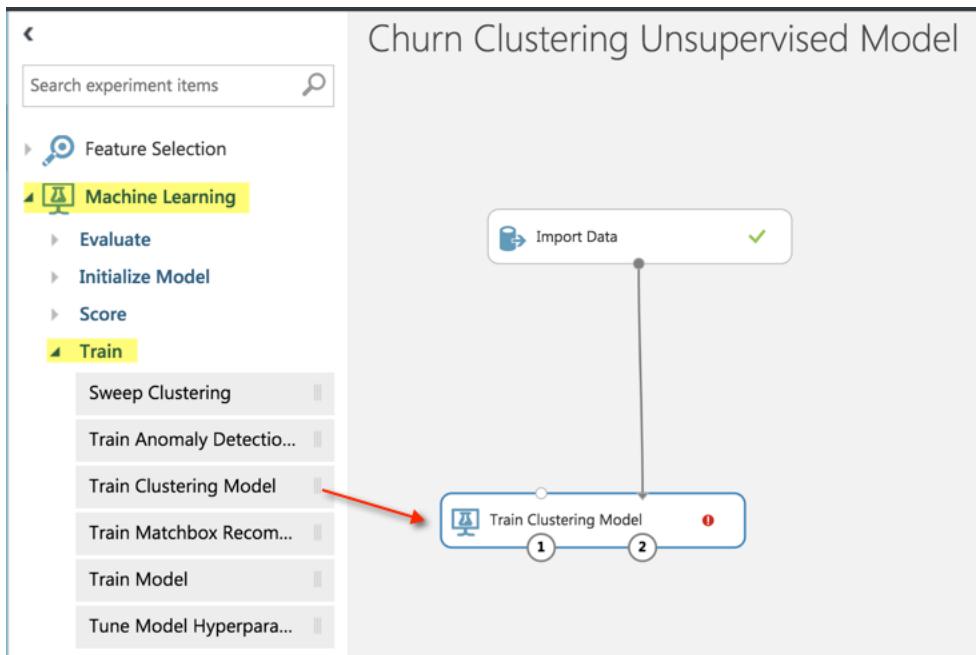
13. \*\*\*Every time a new node is added and attached to an existing node, you will need to select the RUN icon to see the new results\*\*\*
14. Once we have confirmed the connection is working, we can right-click on the number **1** on the first workflow and **Visualize** the dataset as seen in the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



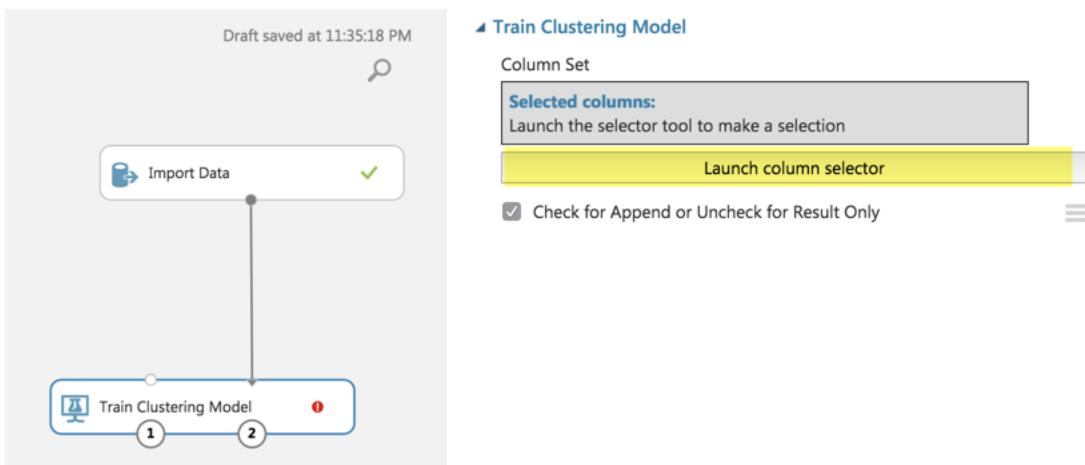
15. Take some time to click on the columns to identify the summary statistics for both the numeric fields and the categorical fields as seen in the following screenshot:



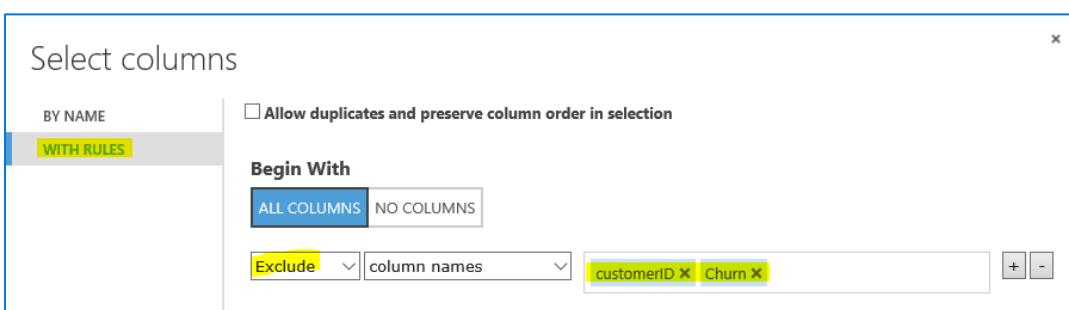
16. Import **Train Clustering Model** from the **Machine Learning** Node and connect it to the **Import Data** module as seen in the following screenshot:



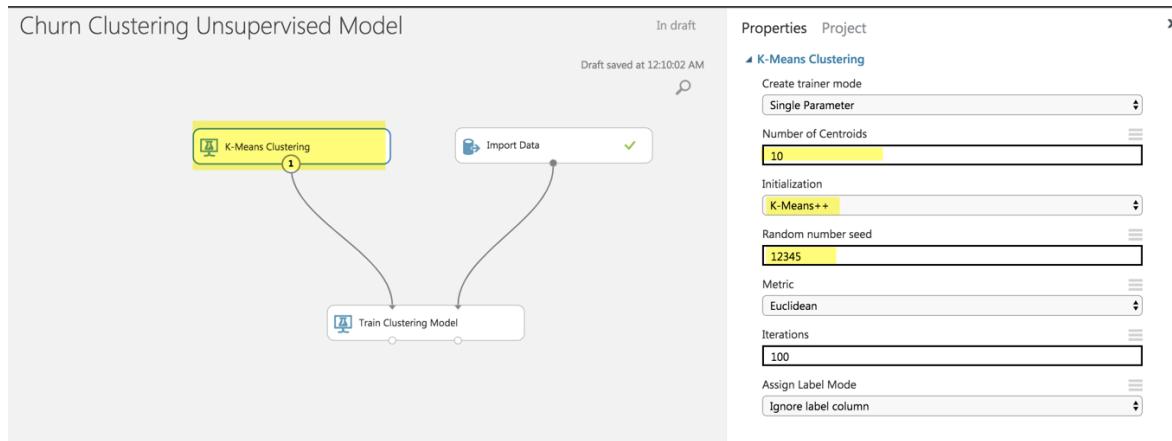
17. Next, click on the **Train Clustering Model** module, **Launch Column Selector**, connect the node back to the **Import Data** node, and configure the columns to be selected as seen in the following screenshot:



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. **Churn** is also not helpful, because customers who have churned are no longer customers.
19. So, you will need to configure the **clustering model** and **Select All columns** from the dataset except for the column called **customerID** and **Churn** as seen in the following screenshot:

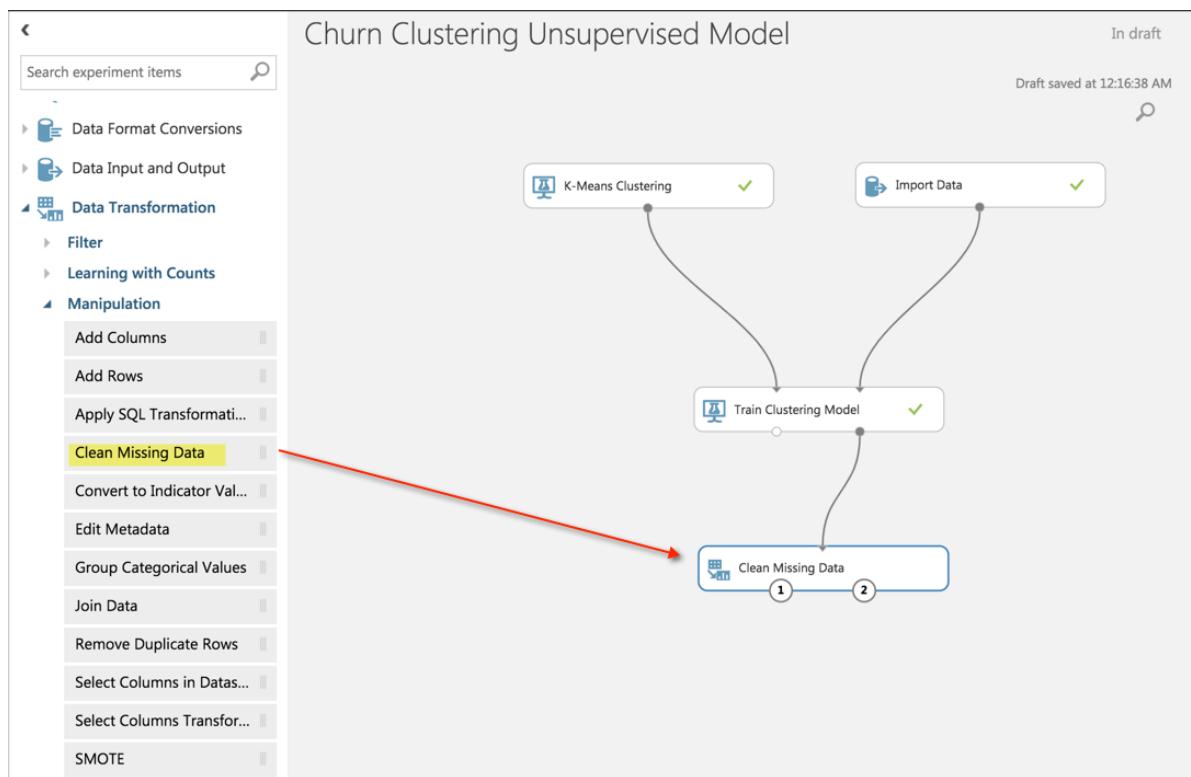


20. Next, we will add a **K-Means Clustering** model from the Machine Learning → Initialize Model → Clustering section. Attach it to the **Train Clustering Model** and assign the following parameters to the clustering model, as seen in the following screenshot:

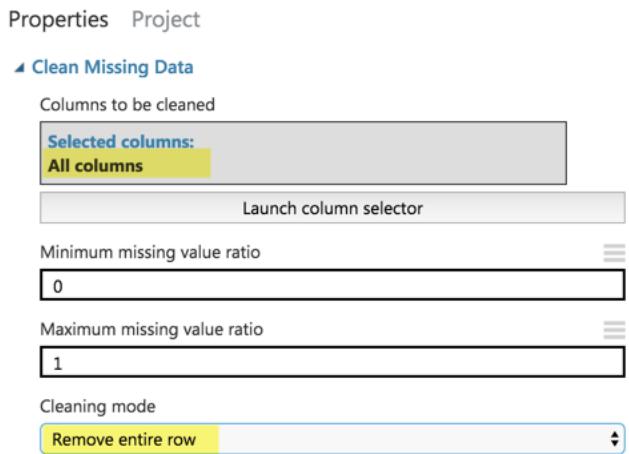


21. Select **RUN**

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:

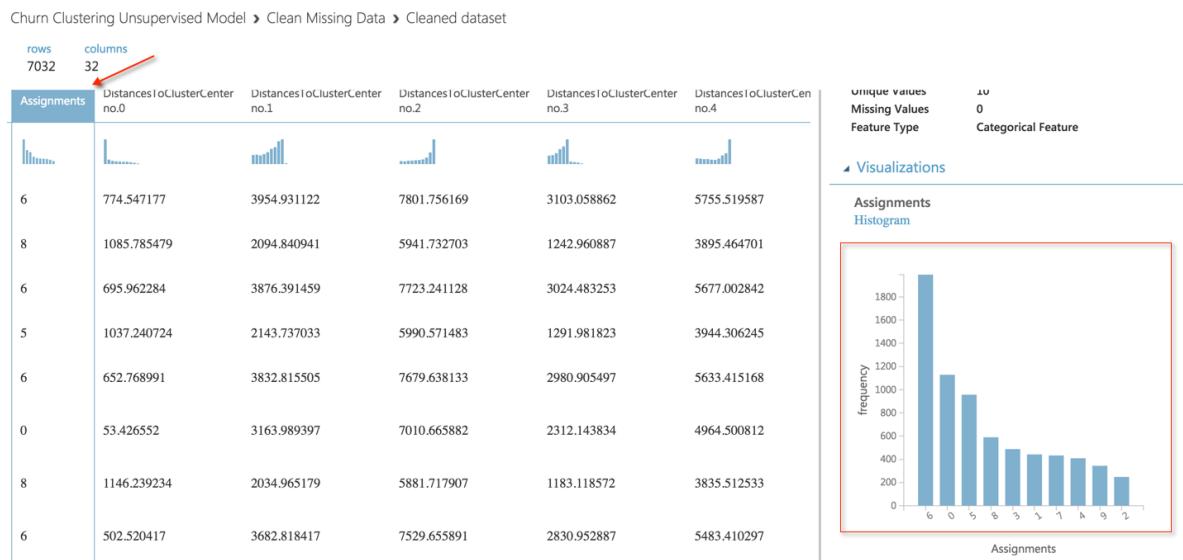


#### 24. Select RUN

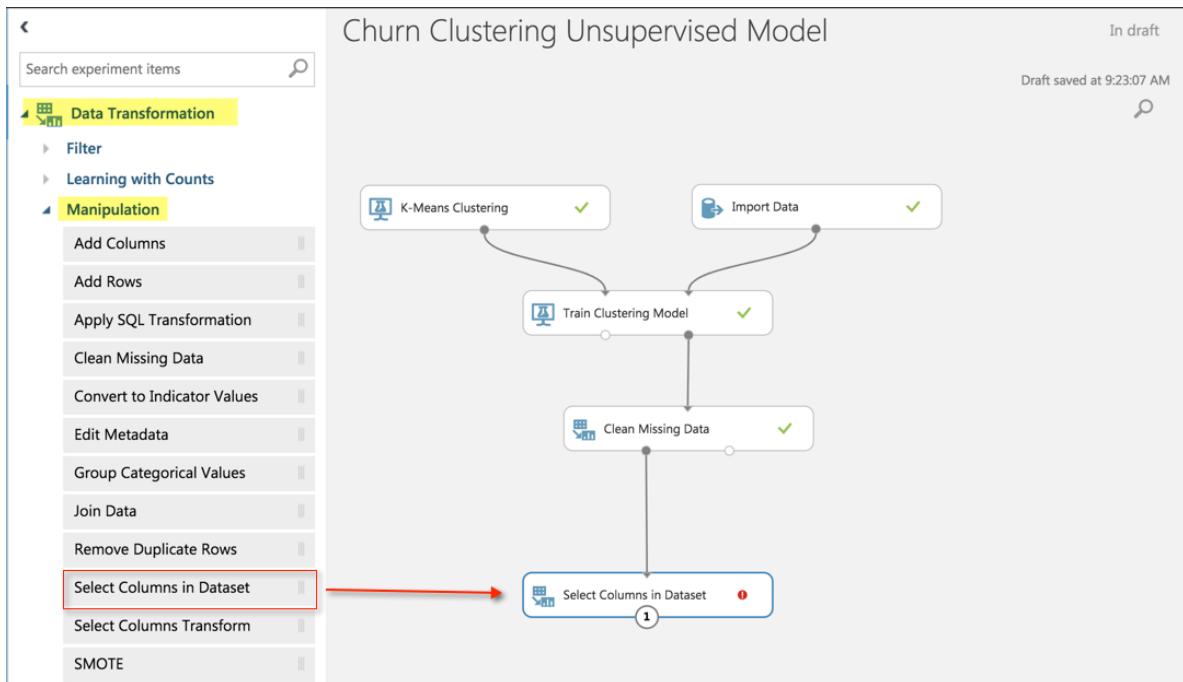
25. The clustering outcome produces several columns that are not necessary for our purposes for this workshop as can be seen by visualizing the **Clean Missing Data** module as seen in the following screenshot:

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
7032	32						
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	3835.512532	
No	6	502.520417	3682.818417	7529.655891	2830.952887	5483.410297	

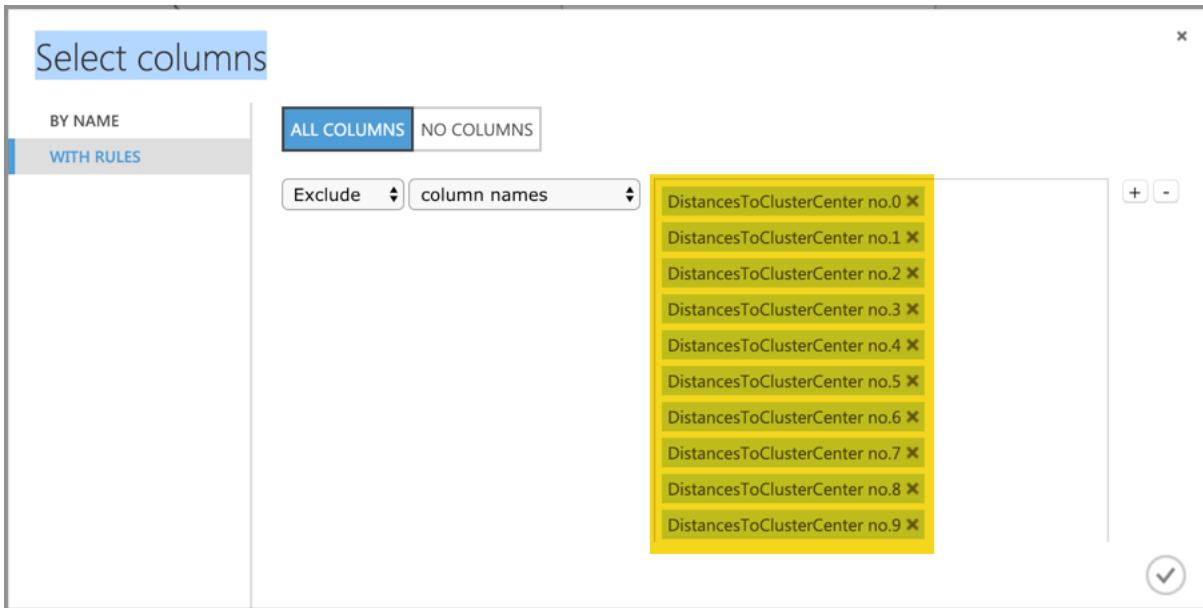
26. We are truly only interested in the assignments, which is grouping each customer into a category of **0** through **9** as seen when we visualize a histogram of the unique values as seen in the following screenshot:



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:

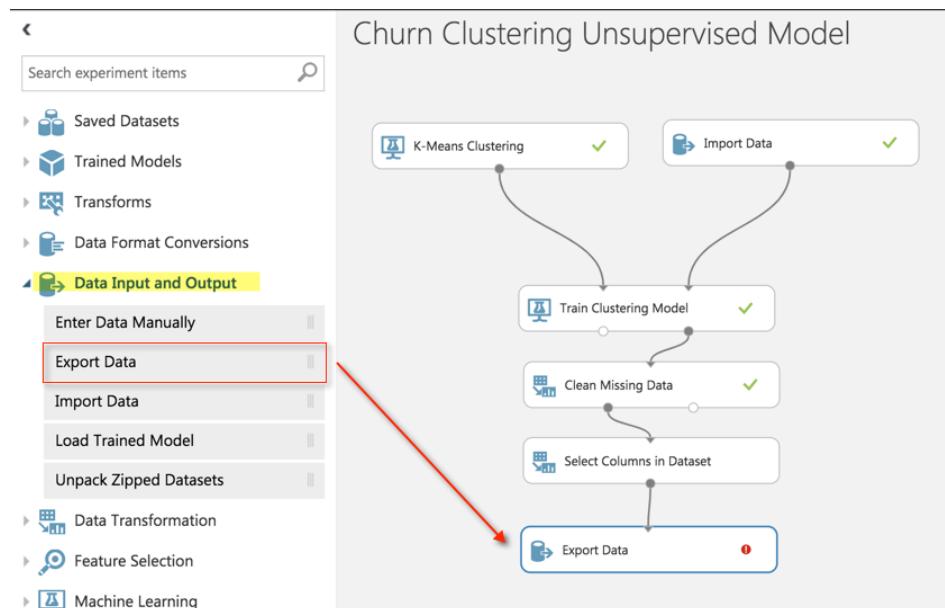


28. We can then configure the module for **Select Columns in Dataset** by **Launching Column Selector**, select **WITH RULES**, and exclude column names DistancetoClusterCenter no.0 through DistancetoClusterCenter no.9, as seen in the following screenshot:



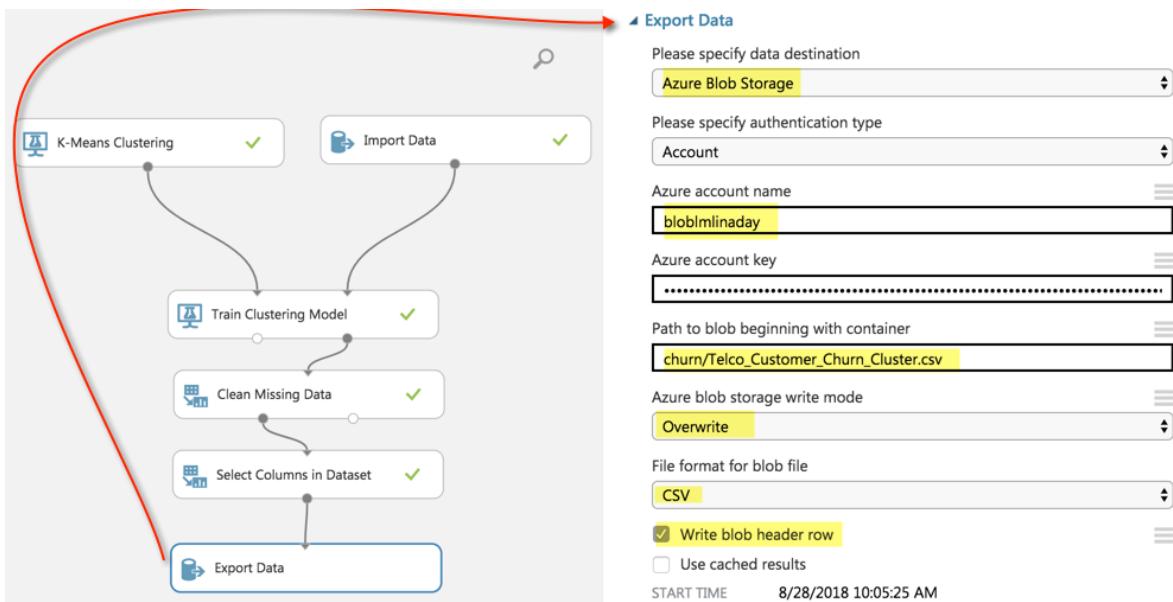
29. Select **RUN**

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:

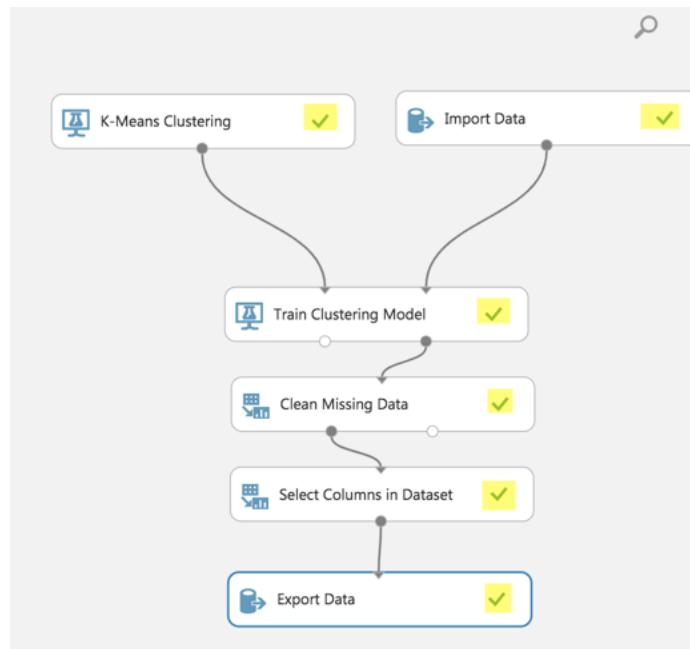


31. Unlike the Import Data module, we do not have a wizard to help us out. So, we will have to input our credentials manually.

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	2018-02-27 10:05:29 AM	Block blob
Telco_Customer_Churn.csv	2018-02-27 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:

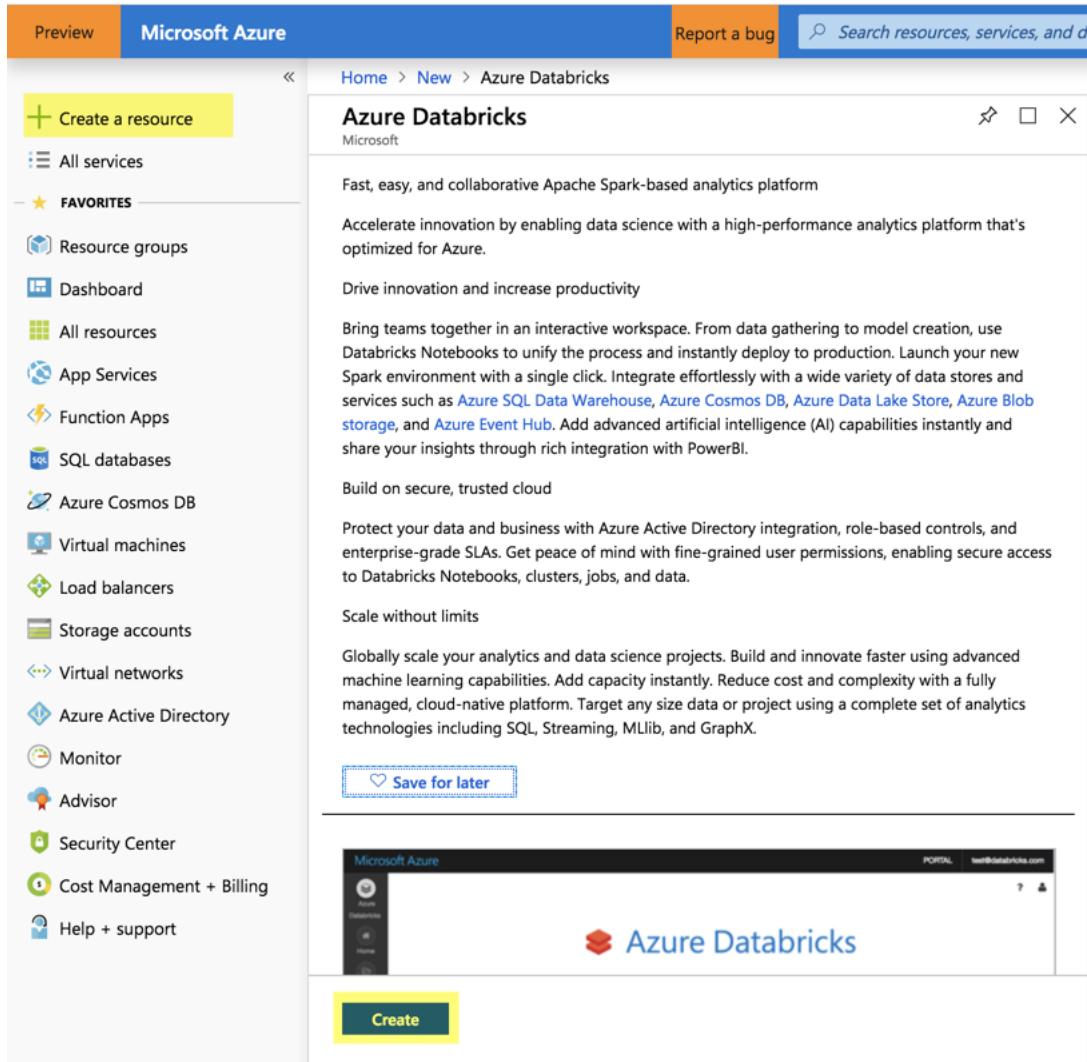
37. A **shared access signature (SAS)** is a URL that grants restricted access rights to Azure Storage resources (a specific blob in this case). You can provide a shared access signature to clients who should not be trusted with your storage account key but whom you wish to delegate access to certain 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:

The screenshot shows the Azure Storage Blob SAS generation interface for the file 'Telco\_Customer\_Churn\_Cluster.csv'. The 'Generate SAS' button is highlighted in yellow. A red arrow points to the 'Expiry' field, which is set to 2028-09-10 at 8:59:46 PM UTC. Another red arrow points to the 'Generate blob SAS token and URL' button. A third red arrow points to the generated SAS URL, which is a long string of characters starting with 'sp=r&st=2018-09-03T16:59:46Z&se='.

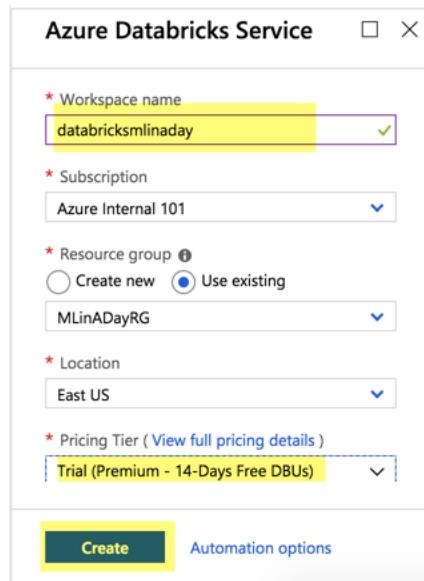
38. Keep a copy of the SAS URL as it will come in handy in the next section when we are reading the file from Blob to Azure Databricks. Additionally, make sure to set expiry date sometime in the future so that it does not expire while you are working on the workshop. Set a time expiration for at least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

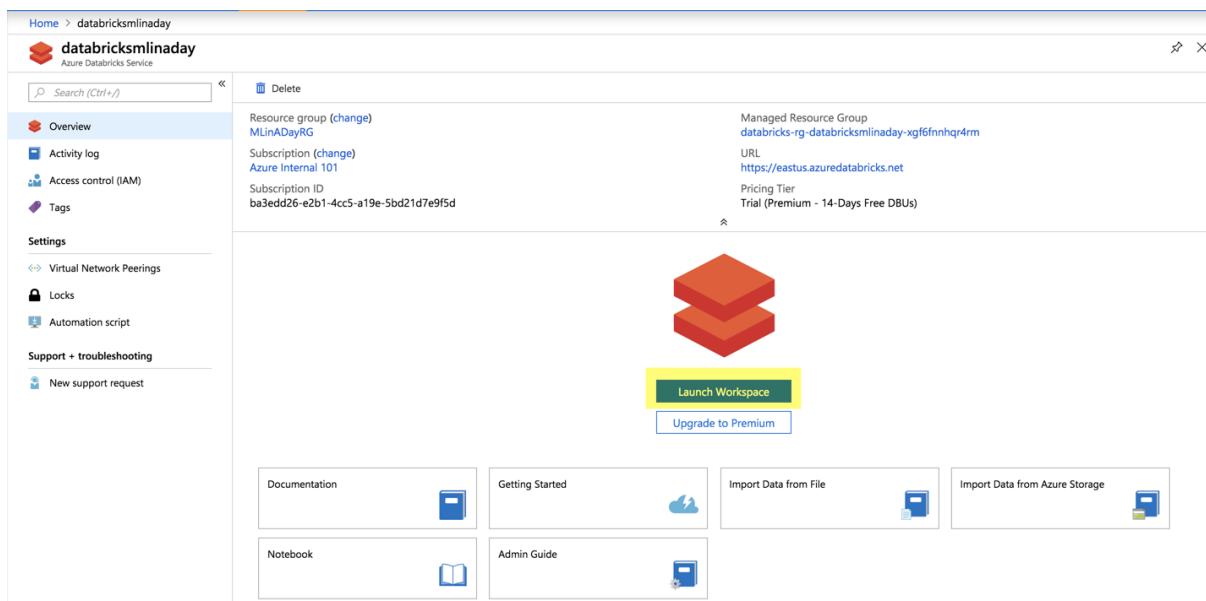
1. Return to the Azure portal (<https://portal.azure.com>) and provision an **Azure Databricks** workspace by selecting Azure Databricks from the **marketplace** as seen in the following screenshot:



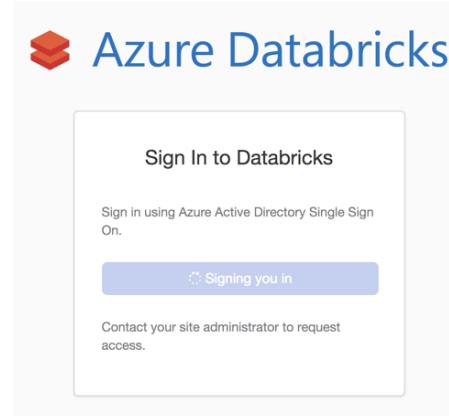
2. Enter the **Workspace** specifications as seen in the following screenshot:



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 left sidebar includes icons for Home, Workspace, Recent, Data, Clusters, Jobs, and Search. The main area features three cards: 'Explore the Quickstart Tutorial', 'Import &amp; Explore Data', and 'Create a Blank Notebook'. Below these are sections for 'Common Tasks' (New Notebook, Upload Data, Create Table, New Cluster, New Job, Import Library, Read Documentation), 'Recents' (Recent files), and 'Documentation' (Databricks Guide, Python, R, Scala, SQL, Importing Data). A 'PORTAL' tab is visible at the top right.

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:

Create Cluster

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

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

**Cluster Mode:** Standard (selected)

Optimized to run concurrent SQL, Python, and R workloads.  
Does not support Scala. Previously known as Serverless.

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

**Python Version:** 3

**Driver Type:** Same as worker (14.0 GB Memory, 4 Cores, 0.75 DBU)

Worker Type	Min Workers	Max Workers
Standard_DS3_v2 (14.0 GB Memory, 4 Cores, 0.75 DBU)	2	8

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

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

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

**Azure Databricks**

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

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

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

**Common Tasks**

- New Notebook (highlighted)
- Upload Data
- Create Table
- New Cluster
- New Job
- Import Library
- Read Documentation

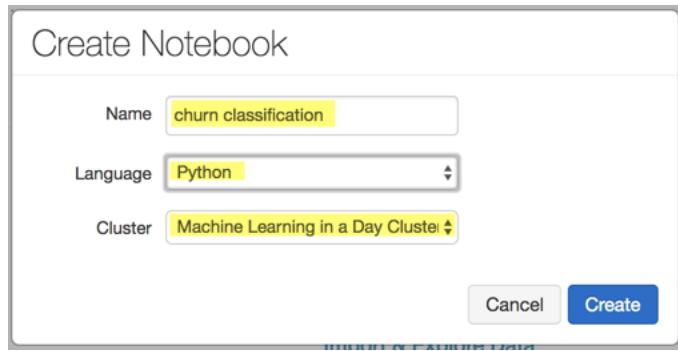
**Recents**

Recent files appear here as you work.

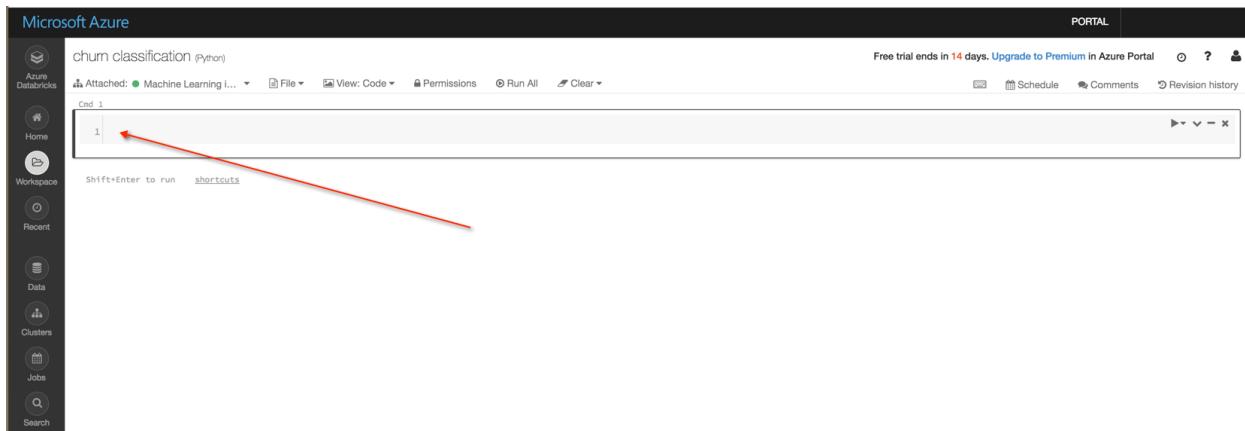
**Documentation**

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

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



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%2BI2OHKA0Nw%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:

The screenshot shows the Microsoft Azure Databricks workspace interface. On the left is a sidebar with icons for Home, Data, Clusters, Jobs, and Search. The main area has a header "churn classification (Python)" and a "PORTAL" button. It displays four command cells (Cmd 1 to Cmd 4) with Python code and their execution logs. The code reads a CSV file from a blob storage URL and creates a DataFrame named df.

```

1 # Please note your SAS_url link will be different
2 SAS_url = 'https://bloblinaday.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%2FSFyXWaqj3%2BbfwZUy1Bj0nb2x2F81%2B120HKA0Nw%3D&sr=b'

```

```

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

```

```

1 df = spark.createDataFrame(pandas_df)

```

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

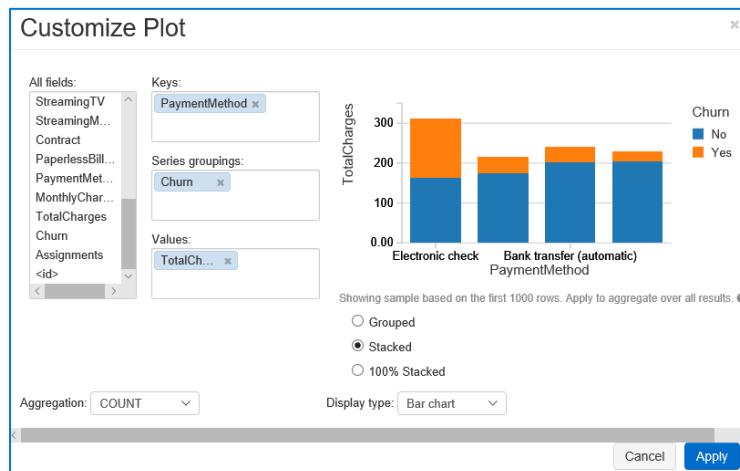
```
display(df)
```

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

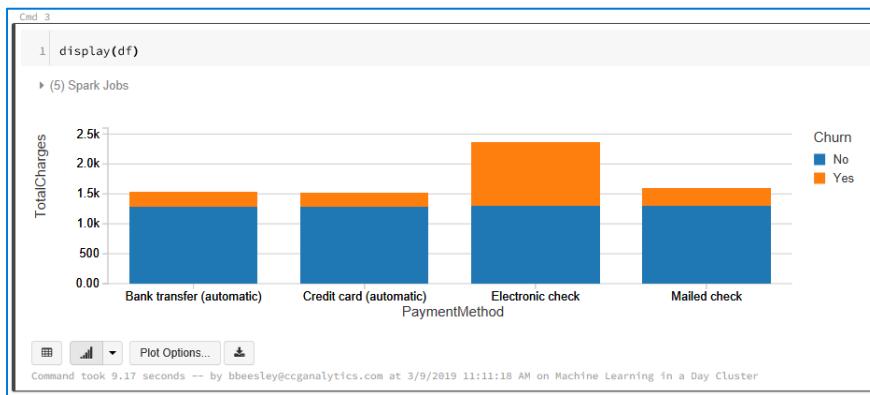
The screenshot shows the result of the `display(df)` command. It displays the first 1000 rows of a DataFrame with 21 columns. A red arrow points to the "More" button at the bottom left of the table preview, which allows for further exploration of the data.

customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	Stream
7580-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
0955-JNSKC	Female	0	No	No	8	Yes	Fiber optic	No	No	No	No	No	Yes	Yes

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



4. 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:



5. 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:

PaymentMethod	sum[TotalCharges]
Mailed check	1691392.200000002
Electronic check	4944903.25
Credit card (automatic)	4671593.35
Bank transfer (automatic)	4748279.899999999

6. However, we can also do the same data analysis using SQL syntax. First, we create a view using the following script: `df.createTempView('sqlView')`
7. 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:

PaymentMethod	sum[TotalCharges]
Mailed check	1691392.200000002
Electronic check	4944903.25
Credit card (automatic)	4671593.35
Bank transfer (automatic)	4748279.899999999

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: Implement Feature Engineering Techniques to Enhance data for Machine Learning

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

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

```
Cmd 10
1 display(df.groupBy('Churn').count())
▶ (5) Spark Jobs
Churn
No
Yes
count
5163
1869
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
1 %sql
2 select Assignments, count(*) as Total_Count from sqlView group by 1 order by 2 desc;
▶ (1) Spark Jobs
```

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 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 Cluster
```

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:]
```

```
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)

> (16) 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
```

Cmd 18

```
1 df.printSchema

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

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

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

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

1	display(df.select('PaymentMethod', 'PaymentMethod_numeric'))																				
▶ (2) Spark Jobs																					
<table border="1"> <thead> <tr> <th>PaymentMethod</th> <th>PaymentMethod_numeric</th> </tr> </thead> <tbody> <tr><td>Electronic check</td><td>0</td></tr> <tr><td>Mailed check</td><td>1</td></tr> <tr><td>Mailed check</td><td>1</td></tr> <tr><td>Bank transfer (automatic)</td><td>2</td></tr> <tr><td>Electronic check</td><td>0</td></tr> <tr><td>Electronic check</td><td>0</td></tr> <tr><td>Credit card (automatic)</td><td>3</td></tr> <tr><td>Mailed check</td><td>1</td></tr> <tr><td>Electronic check</td><td>0</td></tr> </tbody> </table>		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
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.																					

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:

```

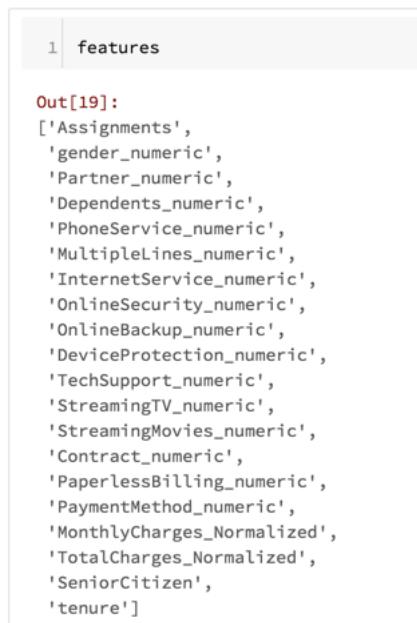
features = [
    'Assignments',

    '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 code cell contains the following 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 view 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,[],[5,0,0,0,1,2,1,1,0,1,0,0,2,1,2,0.23930348258706466,0.21024117239787676,0,45]]  
▶ [0,20,[0,1,16,17,19],[6,1,0.5218905472636817,0.015330025386568196,2]]  
▶ [0,20,[1,5,9,11,12,16,17,19],[1,1,1,1,1,0.809950248756219,0.09251096238172167,8]]  
▶ [0,20,[0,3,5,8,11,15,16,17,19],[8,1,1,1,1,3,0.7049751243781094,0.22277867528271408,22]]  
▶ [0,20,[0,1,4,5,6,7,14,15,16,17,19],[6,1,1,2,1,1,1,1,0.11442786069651742,0.03266789753057927,10]]  
▶ [0,20,[0,1,2,5,9,10,11,12,16,17,19],[2,1,1,1,1,1,1,1,0.8611040208507462,0.2402210480720087,28]]  
Showing the first 1000 rows.  
  

```

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:

| Churn | features                                                                                           |
|-------|----------------------------------------------------------------------------------------------------|
| 0     | ▶ [0,20,[0,1,2,4,5,6,8,16,17,19],[6,1,1,1,2,1,1,0.11542288557213931,0.001275098084468036,1]]       |
| 0     | ▶ [0,20,[0,6,7,9,13,14,15,16,17,19],[8,1,1,1,2,1,1,0.3850746268656717,0.21586660512347103,34]]     |
| 1     | ▶ [0,20,[0,6,7,8,15,16,17,19],[6,1,1,1,1,0.35422885572139307,0.010310408492960998,2]]              |
| 0     | ▶ [1,20,[],[5,0,0,0,1,2,1,1,0,1,1,0,0,2,1,2,0.23930348258706466,0.21024117239787676,0,45]]         |
| 1     | ▶ [0,20,[0,1,16,17,19],[6,1,0.5218905472636817,0.015330025386568196,2]]                            |
| 1     | ▶ [0,20,[1,5,9,11,12,16,17,19],[1,1,1,1,1,0.809950248756219,0.09251096238172167,8]]                |
| 0     | ▶ [0,20,[0,3,5,8,11,15,16,17,19],[8,1,1,1,1,3,0.7049751243781094,0.2227786752871408,22]]           |
| 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,[0,1,2,5,9,10,11,12,16,17,19],[2,1,1,1,1,1,1,0.9611040208507462,0.2402240480720087,28]]    |

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. Let's add a column indicating whether each prediction for our test set is correct or not.

```
from pyspark.sql.functions import abs
predictedDF = predictedDF.withColumn('CorrectPrediction', 1 -
abs(predictedDF["label"] - predictedDF["prediction"]))
```

9. 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',
'CorrectPrediction'))
```

|                                                                                          |                                                 |
|------------------------------------------------------------------------------------------|-------------------------------------------------|
| 1 display(predictedDF.select('label', 'probability', 'prediction', 'CorrectPrediction')) |                                                 |
| ▶ (2) Spark Jobs                                                                         |                                                 |
| label                                                                                    | probability                                     |
| 0                                                                                        | [1.2,][0.1668175095928016, 0.6331824904071965]  |
| 0                                                                                        | [1.2,][0.8164170437101321, 0.1815829562896787]  |
| 0                                                                                        | [1.2,][0.9054630351396837, 0.0945369648603162]  |
| 0                                                                                        | [1.2,][0.4418370486158937, 0.5581629513841063]  |
| 1                                                                                        | [1.2,][0.2513723804734345, 0.7486276195266565]  |
| 1                                                                                        | [1.2,][0.5030456390367141, 0.49695436096328593] |
| 1                                                                                        | [1.2,][0.8453026255234901, 0.15469737447650997] |
| 0                                                                                        | [1.2,][0.6827134985764811, 0.31728650142351883] |
| 1                                                                                        | [1.2,][0.79392141607364, 0.4136068324772251]    |
| Showing the first 1000 rows.                                                             |                                                 |
|                                                                                          |                                                 |
|                                                                                          |                                                 |

10. 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.
11. 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                                                      | 176 188 |
| 0                                                      | 876 114 |
|                                                        |         |
|                                                        |         |

12. We made **876 + 188 = 1,064** correct predictions and **176+114 = 290** false predictions for an overall accuracy of ~ **79%**
13. 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 = 78.58197932053176                                                   |  |

14. 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  $(1 - \text{specificity})$ .
15. 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 = 84.23
```

16. We have a high ROC score of **84.23%**. The higher the score or closer it is to 100% the better.

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

1. We are able to stream our DataFrames from Spark on Azure Databricks to other Business Intelligence tools such as Power BI, Tableau, and Alteryx.

2. First, we will create another view that called **PredictionScoresDF** using the following script:

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

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

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

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

The screenshot shows the Azure Databricks interface. On the left, a sidebar menu has 'Data' selected. In the main area, under 'Tables', a table named 'predictionscore' is highlighted with a yellow box and an orange arrow pointing from the sidebar's 'Data' icon. To the right, a code editor window displays two commands. The first command creates a temporary view:

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

The second command creates a permanent table:

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

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

The screenshot shows the Azure Databricks interface with the 'Data' sidebar selected. In the main area, under 'Tables', the 'predictionscore' table is selected and highlighted with a yellow box and an orange arrow. Below the table name, the word 'Schema:' is followed by a table showing the column names, data types, and comments. At the bottom, there is a section titled 'Sample Data:' with a table showing two rows of sample data.

| customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | InternetService | OnlineSecurity | OnlineBackup        | DeviceProtection    | TechSupport         | StreamingTV         | StreamingMovies     | Contract            |
|------------|--------|---------------|---------|------------|--------|--------------|---------------|-----------------|----------------|---------------------|---------------------|---------------------|---------------------|---------------------|---------------------|
| 0064-SUDOG | Female | 0             |         | Yes        | Yes    | 12           | Yes           | No              | No             | No internet service |
| 0164-XAIPP | Female | 0             |         | No         | No     | 24           | Yes           | No              | No             | No internet service |

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

The screenshot shows the Azure Databricks Clusters page. On the left, there's a sidebar with icons for Home, Workspace, Recent, Data, and Clusters. The 'Clusters' icon is highlighted with a red arrow. The main area shows a table of clusters. One row is selected, showing the cluster name 'Machine Learning in a Day Cluster', its state 'Running', and other details like nodes, driver, worker, runtime, and creator. A message at the bottom says 'No clusters found'.

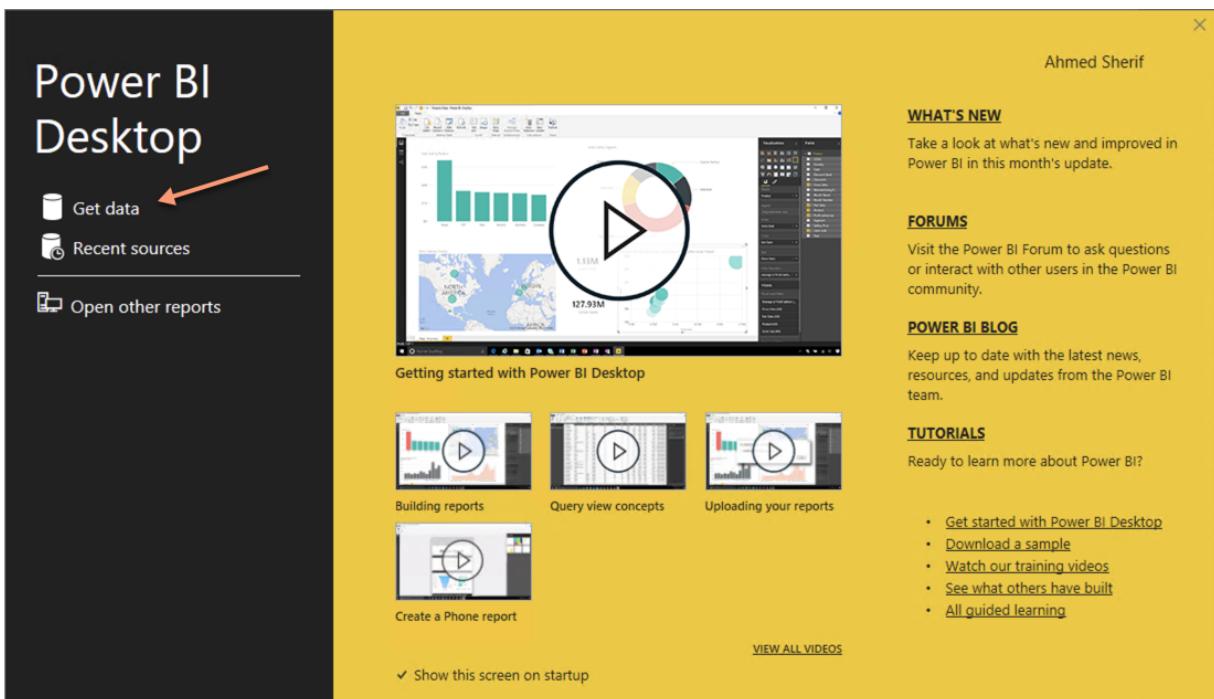
7. Once inside the cluster, click on the **JDBC/ODBC** tab and click on the link in the bottom **To Learn More About Connecting your Favorite BI Tools to Databricks**.

The screenshot shows the configuration page for the 'Machine Learning in a Day Cluster'. The sidebar has icons for Home, Workspace, Recent, Data, Clusters (which is selected), Jobs, and Search. The main page shows cluster settings like worker type (Standard\_DS3\_v2), auto-termination (terminate after 120 minutes of inactivity), and connection details. The 'JDBC/ODBC' tab is active. At the bottom, there's a link 'Learn more about connecting your favorite BI tool to Databricks.' which is highlighted with a red arrow.

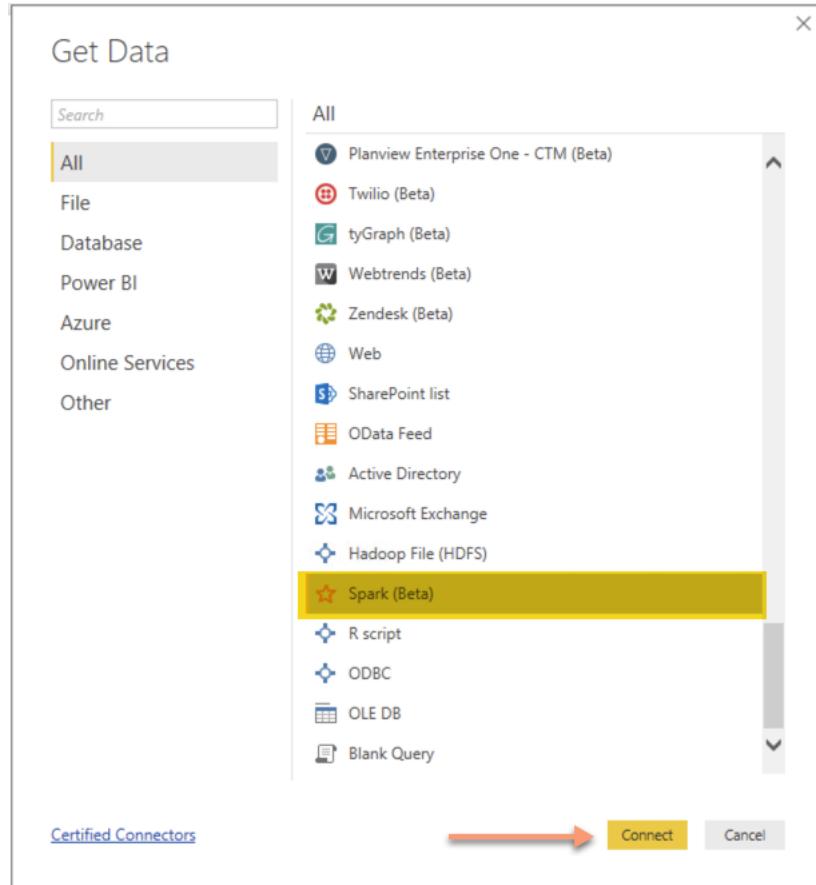
8. Select **Power BI** as the **Business Intelligence tool** of choice for connecting to Databricks as seen in the following screenshot:

The screenshot shows the Azure Databricks User Guide page for Business Intelligence Tools. The left sidebar contains a navigation menu with sections like Getting Started Guide, User Guide, and Business Intelligence Tools. Under Business Intelligence Tools, there's a sub-section for Connecting BI Tools. The main content area has a yellow header "Business Intelligence Tools". Below it, a paragraph explains that BI tools can connect to Azure Databricks clusters to query data in tables. A list of supported tools includes Tableau, Power BI (which is highlighted in yellow), Alteryx, Looker, and SQL Workbench/J. Navigation buttons < PREVIOUS > and < NEXT > are at the bottom.

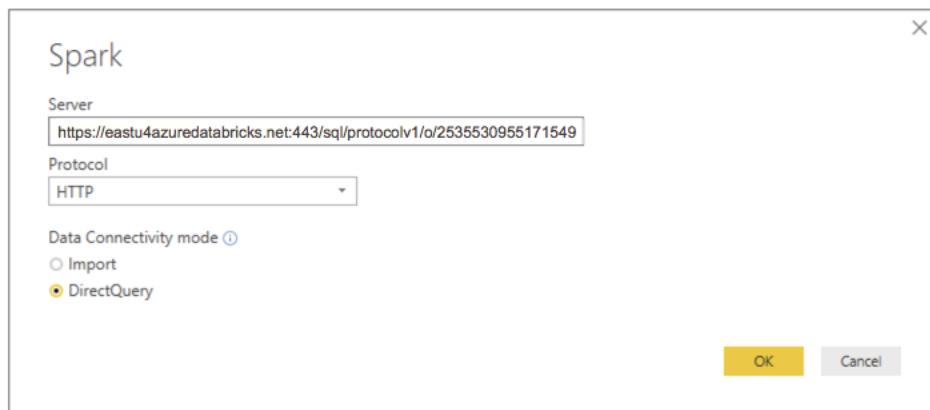
9. Databricks should have step-by-step directions for connecting Tables from Spark to Power BI as found in the following link: <https://docs.azuredatabricks.net/user-guide/bi/power-bi.html>
10. There should be a Power BI desktop installation already available in our Data Science Virtual machine that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



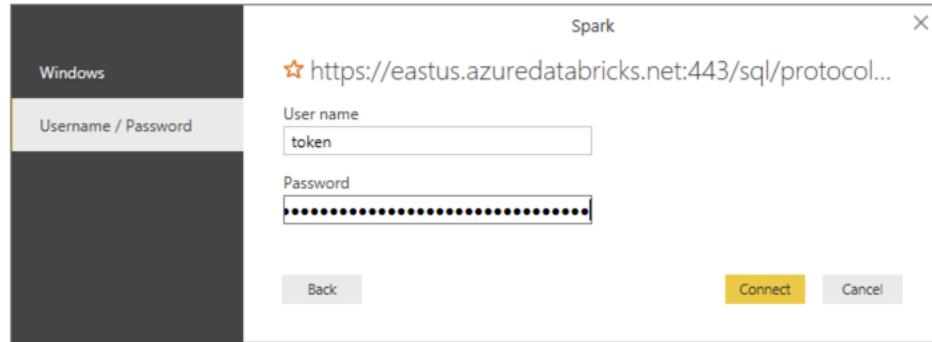
11. Select **Spark (Beta)** and **Connect** as seen in the following screenshot:



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



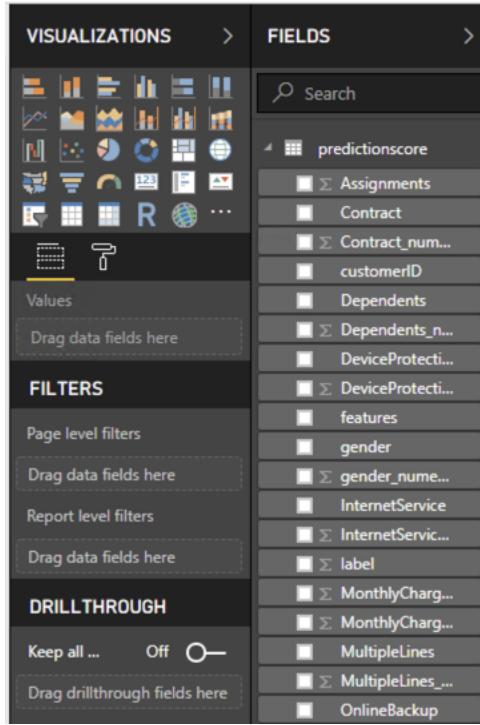
13. Next, Enter the User Name and Password credentials as seen in the following screenshot:



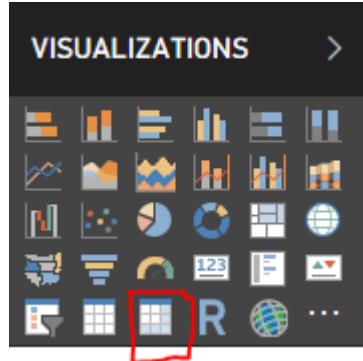
14. We can now view our data from the **predictionscore** table in Spark now viewed directly within Power BI as seen in the following screenshot:

| customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService     | MultipleLines | InternetService | OnlineSecurity      | OnlineBackup        |
|------------|--------|---------------|---------|------------|--------|------------------|---------------|-----------------|---------------------|---------------------|
| 0064-SUDOG | Female | 0 Yes         | Yes     | 12         | Yes    | No               | No            | No              | No internet service | No internet service |
| 0164-XAIRP | Female | 0 No          | No      | 24         | Yes    | No               | No            | No              | No internet service | No internet service |
| 0230-UBYPQ | 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-UXFAP | Female | 0 Yes         | No      | 51         | Yes    | Yes              | Fiber optic   | No              | No                  |                     |
| 0644-QQMDK | Male   | 1 No          | No      | 4          | Yes    | No               | Fiber optic   | No              | No                  |                     |
| 0657-DOGUM | Female | 0 Yes         | No      | 48         | Yes    | Yes              | DSL           | Yes             | No                  |                     |
| 0674-EYYZV | Female | 0 No          | No      | 1          | Yes    | No               | DSL           | No              | No                  |                     |
| 0872-CASZU | Male   | 0 Yes         | No      | 59         | Yes    | Yes              | DSL           | Yes             | No                  |                     |
| 0886-QGENL | Female | 1 Yes         | No      | 27         | Yes    | No               | Fiber optic   | Yes             | No                  |                     |
| 0902-RFHOF | Male   | 0 No          | No      | 38         | Yes    | No               | No            | No              | No internet service | No internet service |

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 build a heatmap showing how accurate our model is within different segments of the population. Select the Matrix visualization from the Visualizations bar.



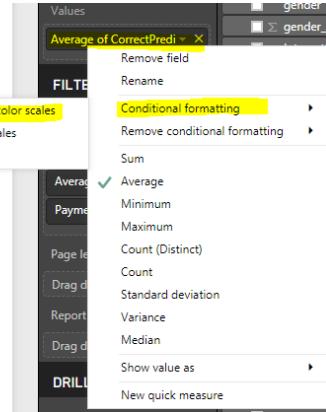
17. Drag **Assignments** into the Rows area. These **Assignments** represent the 10 customer segments we created in Azure Machine Learning Studio. Next, Drag **PaymentMethod** into the Columns area. Drag **CorrectPrediction** into the Values area.

The screenshot shows the Power BI Fields pane. On the left, under 'VALUES', there is a section labeled 'CorrectPrediction'. On the right, under 'FIELDS', there is also a section labeled 'CorrectPrediction'. A red arrow points from the 'CorrectPrediction' in the 'VALUES' section to the 'CorrectPrediction' in the 'FIELDS' section, highlighting the selection process.

18. Right click on **CorrectPrediction** in the Values area and select Average.

The screenshot shows the Power BI Values pane with a context menu open over the 'Average of CorrectPrediction' entry. The 'Average' option is highlighted with a yellow box. Other options in the menu include Remove field, Rename, Conditional formatting, Remove conditional formatting, Sum, Minimum, Maximum, Count (Distinct), Count, Standard deviation, Variance, Median, Show value as, and New quick measure.

19. Right click on **Average of CorrectPrediction**, which now appears in the Values area, but this time select Conditional formatting and Background color scales.



20. We now have a heatmap showing how accurate our predictions are within each customer segment and for customers of each payment type.

| Assignments | Bank transfer (automatic) | Credit card (automatic) | Electronic check | Mailed check | Total |
|-------------|---------------------------|-------------------------|------------------|--------------|-------|
| 0           | 0.86                      | 0.77                    | 0.68             | 0.92         | 0.80  |
| 1           | 0.86                      | 0.91                    | 0.62             | 0.89         | 0.82  |
| 2           | 0.90                      | 0.93                    | 0.64             | 1.00         | 0.85  |
| 3           | 0.79                      | 0.86                    | 0.60             | 0.92         | 0.75  |
| 4           | 0.86                      | 0.78                    | 0.84             | 1.00         | 0.84  |
| 5           | 0.88                      | 0.98                    | 0.69             | 0.93         | 0.85  |
| 6           | 0.73                      | 0.84                    | 0.64             | 0.78         | 0.73  |
| 7           | 0.94                      | 0.81                    | 0.70             | 0.89         | 0.81  |
| 8           | 0.64                      | 0.88                    | 0.62             | 0.85         | 0.71  |
| 9           | 0.95                      | 0.94                    | 0.81             | 1.00         | 0.90  |
| Total       | 0.84                      | 0.87                    | 0.67             | 0.85         | 0.79  |

21. We have successfully completed this workshop!! Feel free to play with other visualizations in Power BI while fellow participants finish up.