

Machine Learning in SQL using PyCaret

Ship your ML code to data by integrating PyCaret in SQL Server

This post is a **step-by-step tutorial** on how to train and deploy an Unsupervised Machine Learning Clustering model in SQL Server using [PyCaret](#)(a low-code ML library in Python).

Things we will cover in this article:

1. How to download and install SQL Server for free
2. How to create a new database and importing data into a database
3. How to enable and use Python scripting in database
4. How to train a clustering algorithm in order to assign cluster labels to each observation in the dataset

I. Bringing Code to Data — The case for using Database for ML

The go-to tools/ environments for performing ML experiments are Command-Line, IDEs, or Notebooks. However, such tools/environments may pose limitations when the data size gets very large, or when the ML model is required to be put in production. There has been a dire need to have the ability to programme and train models where data reside. MS SQL Server introduced this capability in their SQL Server version 2019. The distinct advantages of using SQL Server for Machine Learning are:

- i. Extracting a large amount of data from the system is tedious and time-consuming. Conducting ML experiments on a server brings the code to data, rather than taking data to the code
- ii. ML experiments are executed mostly in computer/cpu memory. Most of the machines hit a performance ceiling when training an ML algorithm on large data sets. ML on the SQL Server database avoids this
- iii. It is easy to integrate and deploy ML Pipelines along with other ETL processes


II. SQL Server

SQL Server is a Microsoft relational database management system. As a database server, it performs the primary function of storing and retrieving data as requested by different applications. In this tutorial, we will use [SQL Server 2019 Developer](#) for machine learning by importing PyCaret library into SQL Server.

III. Download Software

If you have used SQL Server before, it is likely that you have it installed and have access to the database. If not, [click here](#) to download SQL Server 2019 Developer or other edition.


Try SQL Server on-premises or in the cloud



SQL Server 2019 on-premises

Build intelligent, mission-critical applications using a scalable, hybrid data platform for demanding workloads. Get started with a 180-day free trial of SQL Server 2019 on Windows.

[Download free trial ↴](#)



SQL Server 2019 on Azure

Get started with SQL Server 2019 on Azure Virtual Machines in minutes with preconfigured images on Linux and Windows. Take advantage of unique built-in security and manageability to automate tasks like patching and backups, and save with Azure Hybrid Benefit by reusing your existing on-premises licenses.

[Get started in Azure >](#)

Or, download a free specialized edition



Developer

SQL Server 2019 Developer is a full-featured free edition, licensed for use as a development and test database in a non-production environment.

[Download now ↴](#)



Express

SQL Server 2019 Express is a free edition of SQL Server, ideal for development and production for desktop, web, and small server applications.


[Download now ↴](#)

IV. Setting up the Environment

Before using PyCaret functionality into SQL Server, you'll need to install SQL Server and PyCaret. This is a multi-step process:

Step 1 — Install SQL Server


Download the SQL Server 2019 Developer Edition file “**SQL2019-SSEI-Dev.exe**”



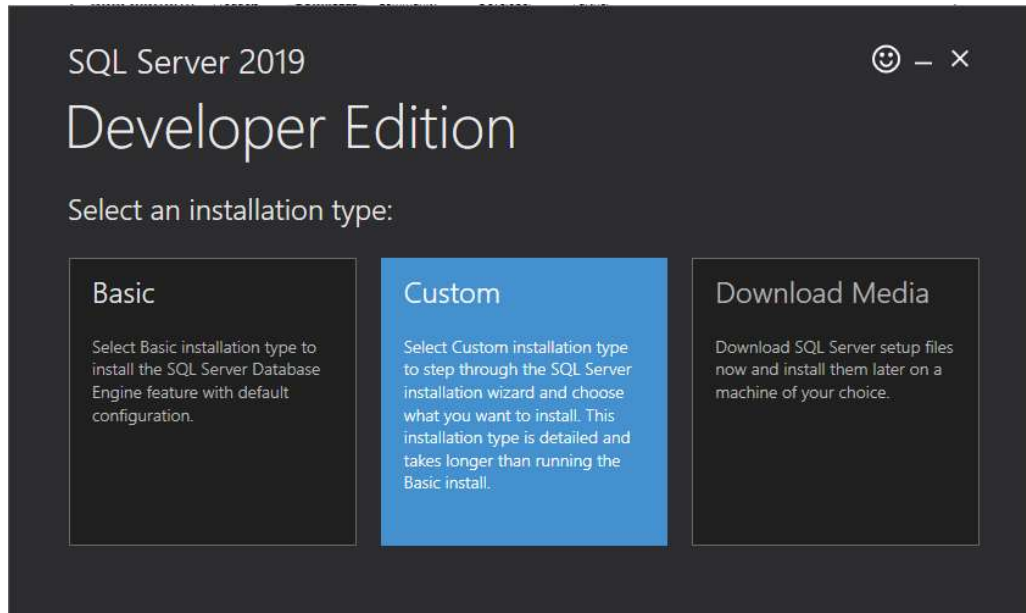
Developer

SQL Server 2019 Developer is a full-featured free edition, licensed for use as a development and test database in a non-production environment.

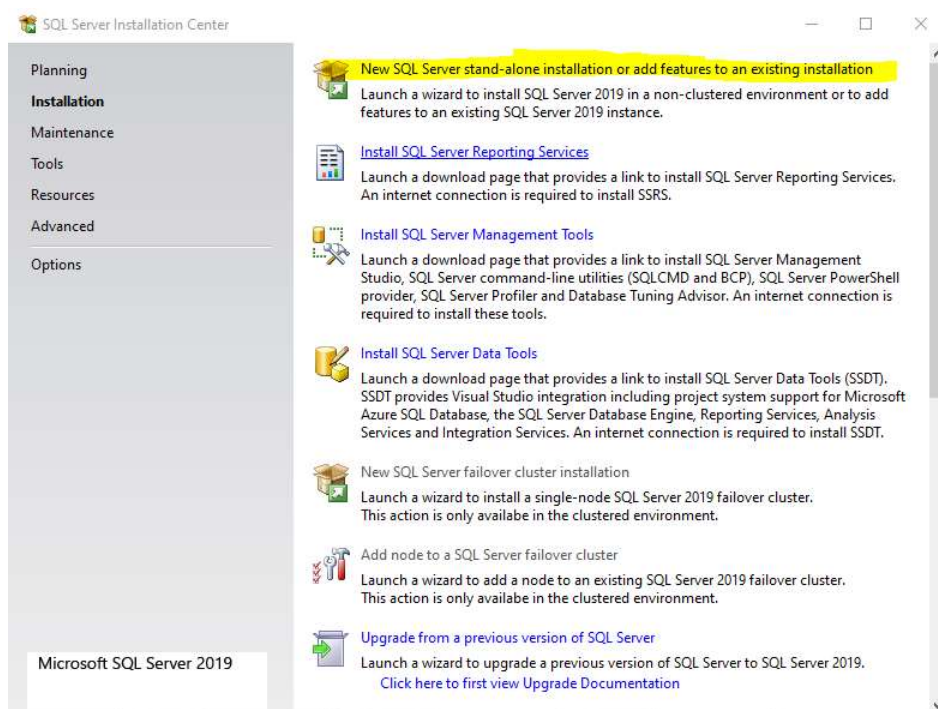
[Download now ↴](#)

 SQL2019-SSEI-Dev.exe ^

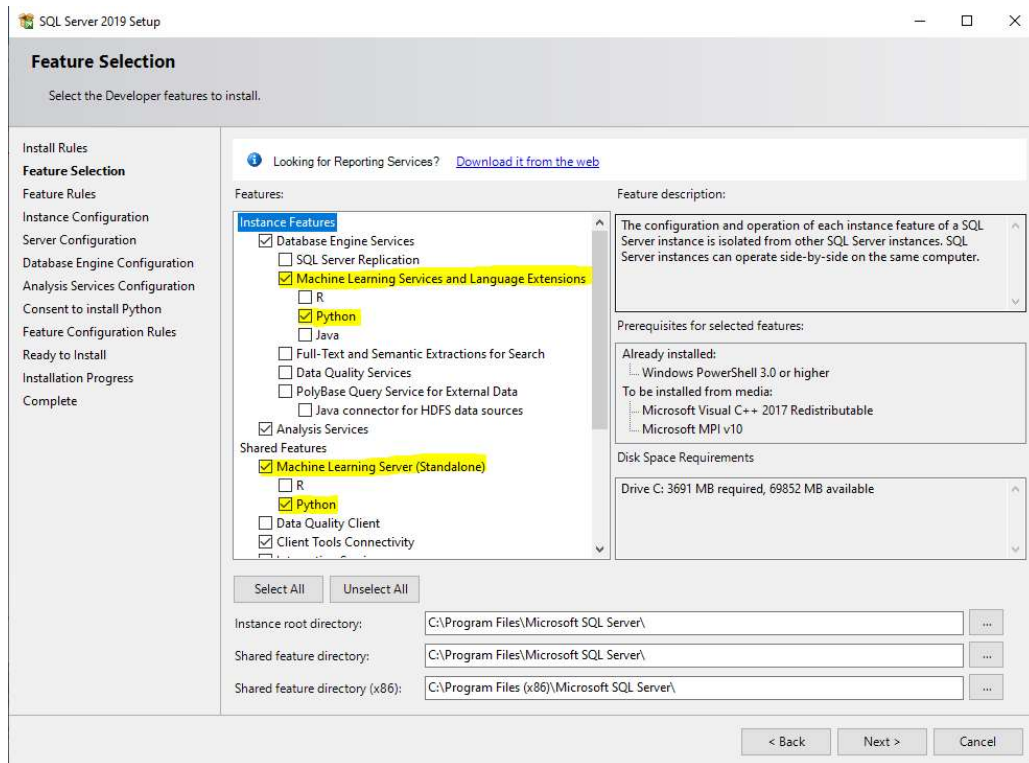
Open the file and follow the instructions to install (recommended to use Custom install option)



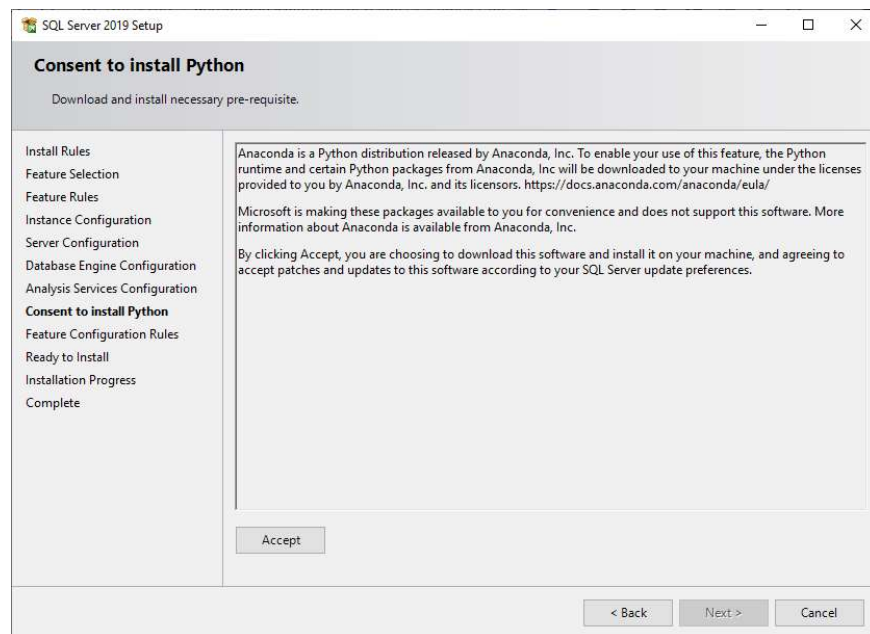
Choose New SQL Server stand-alone installation



In the Instance Features option, select the features including “**Python**” under **Machine Learning Services** and **Language Extensions** and **Machine Learning Server (Standalone)**



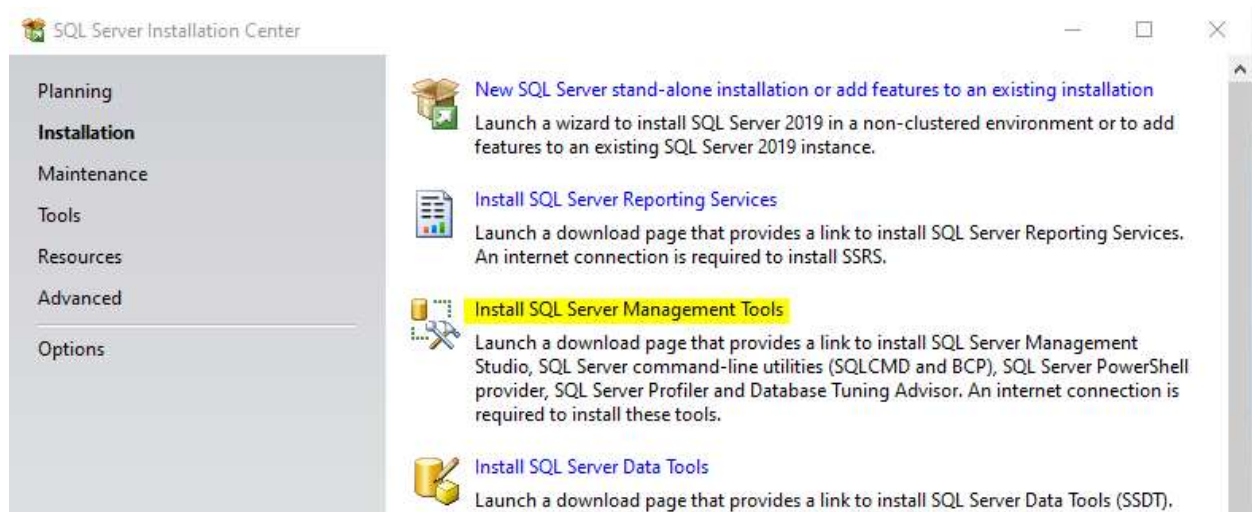
Click **“Accept”** to provide consent to install Python



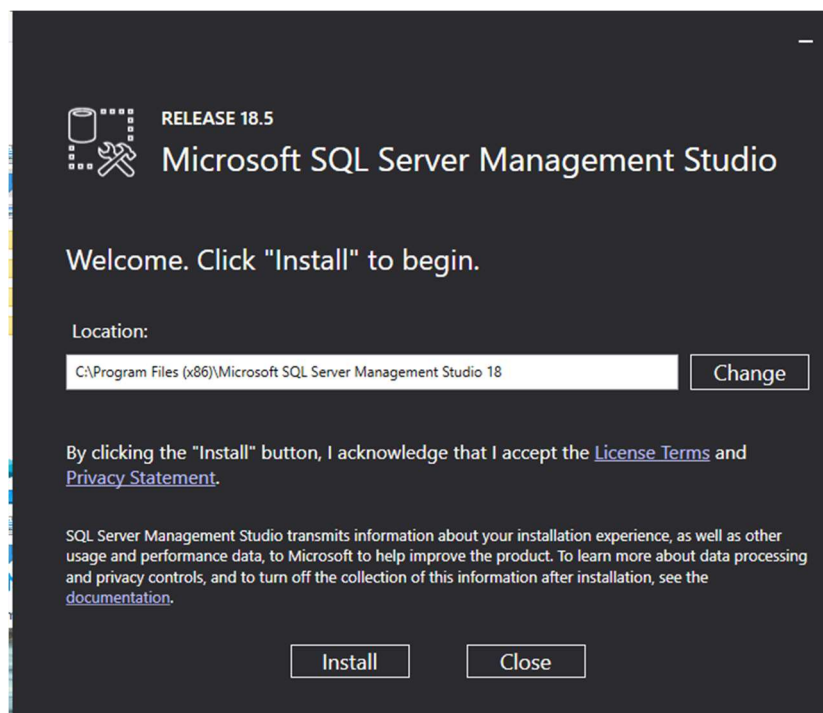
Installation may take 15–20 minutes

Step 2 — Install Microsoft SQL Server Management Studio (SSMS)

[Click here](#) or Open SQL Server Installation Center to download “SQL Server Management Tools” file “SSMS-Setup-ENU.exe”



Open **SSMS-Setup-ENU.exe** file to start the installation

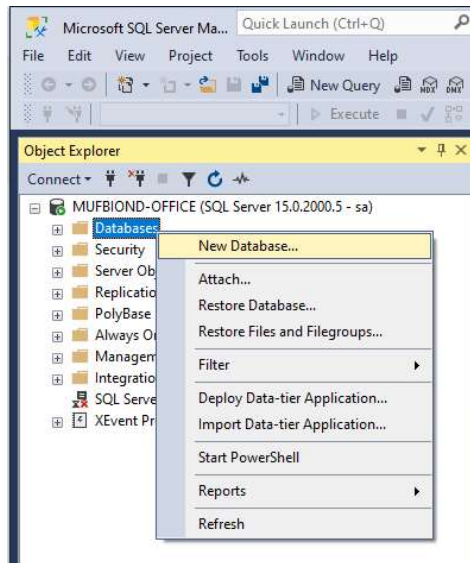


Installation may take 5–10 minutes

Step 3 — Create a database for Machine Learning

Once you have everything installed, you will need to start an instance of the server. To do so, start the SSMS. At the login stage you'll be asked to the name of the SQL Server that you can choose from the drop-down menu. Once a connection is established, you can see all the objects from the server. If you have downloaded SQL Server for the first time and you do not have a database to work with, you will need to create a new database first.

In the Object Explorer panel, right-click on Databases, and choose New Database



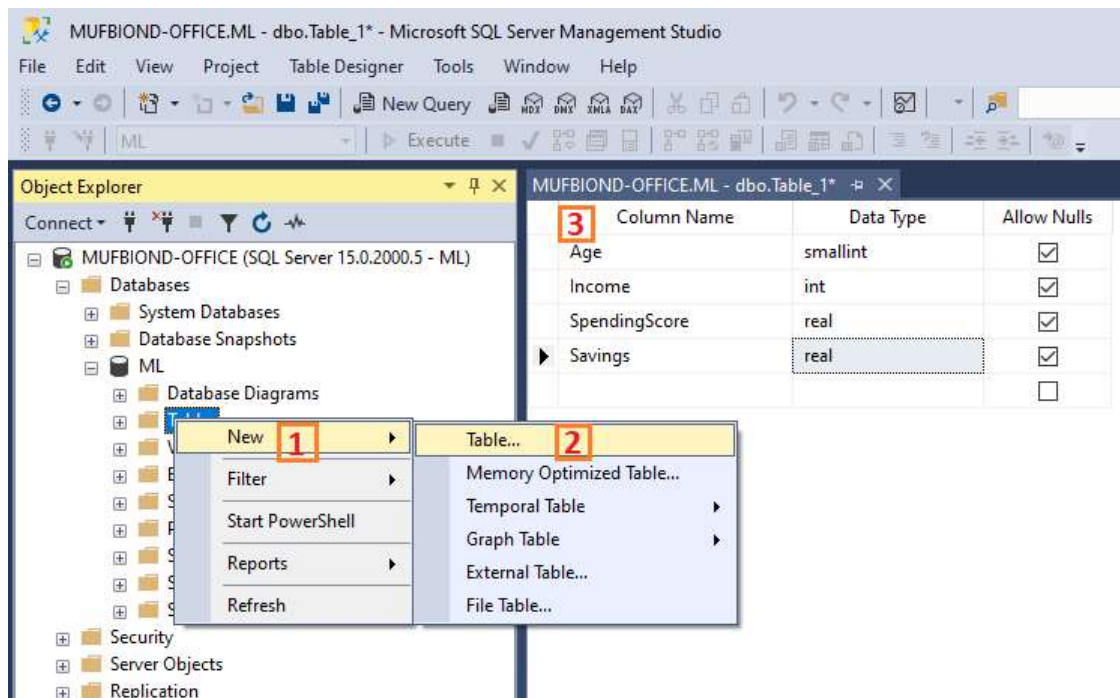
Enter the database name and other information

The setup may take 2–3 minutes including creating a database, user and setting ownership

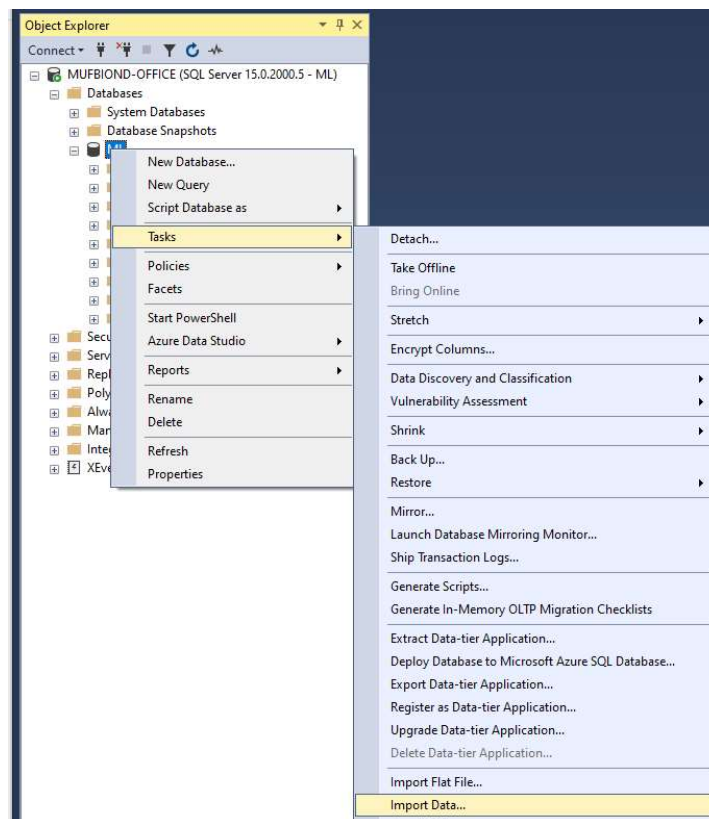
Step 4 — Import CSV File

You will now have to import a CSV file into a database using SQL Server Management Studio.

Create a table “**jewellery**” in the database



Right-click the database and select **Tasks -> Import Data**



For Data Source, select **Flat File Source**. Then use the **Browse** button to select the CSV file. Spend some time configuring the data import before clicking the **Next** button.

The screenshot shows the 'Choose a Data Source' step of the SQL Server Import and Export Wizard. The 'Data source' dropdown is set to 'Flat File Source'. The 'General' tab is selected in the left-hand pane. The 'File name' field contains 'C:\Users\Muhammad\Downloads\jewellery.csv', with a 'Browse...' button to its right. The 'Locale' is set to 'English (Canada)' and the 'Code page' is '1252 (ANSI - Latin I)'. The 'Format' is 'Delimited'. The 'Text qualifier' is '<none>'. The 'Header row delimiter' is '{CR}{LF}'. The 'Header rows to skip' is '0'. The checkbox 'Column names in the first data row' is checked. At the bottom, there are buttons for 'Help', '< Back', 'Next >', 'Finish >>', and 'Cancel'.

For Destination, select the correct database provider (e.g. SQL Server Native Client 11.0). Enter the **Server name**; check **Use SQL Server Authentication**, enter the **Username**, **Password**, and **Database** before clicking the **Next** button.

The screenshot shows the 'Choose a Destination' step of the SQL Server Import and Export Wizard. The 'Destination' dropdown is set to 'SQL Server Native Client 11.0'. The 'Server name' field is empty. Under the 'Authentication' section, 'Use SQL Server Authentication' is selected with a radio button. The 'User name' field contains 'ml' and the 'Password' field is empty. The 'Database' dropdown is set to 'ML', with 'Refresh' and 'New...' buttons to its right. At the bottom, there are buttons for 'Help', '< Back', 'Next >', 'Finish >>', and 'Cancel'.

In the Select Source Tables and Views window, you can Edit Mappings before clicking the **Next** button.

The screenshot shows the 'Review Data Type Mapping' window. It has a title bar 'SQL Server Import and Export Wizard' and a subtitle 'Review Data Type Mapping'. Below the subtitle is a description: 'Select a table to review how its data types map to those in the destination and how it handles conversion issues.' There is a 'Table:' section with a table showing the source and destination tables. The source table is 'C:\Users\Muhammad\Downloads\jewellery.csv' and the destination table is '[dbo].[jewellery]'. Below this is a 'Data type mapping:' section with a table showing the mapping of source columns to destination columns. The columns are: Source Column, Source Type, Destination Col..., Destination Type, Convert, On Error, and On Truncati... The rows are: Age (single-byte sig... to smallint, Convert checked, Use Global), Income (four-byte signe... to int, Convert checked, Use Global), SpendingScore (float [DT_R4] to real, Convert checked, Use Global), and Savings (float [DT_R4] to real, Convert checked, Use Global). Below the table is a note: 'To view conversion details, double-click the row that contains the column source type to be converted.' There are two dropdown menus for 'On Error (global)' and 'On Truncation (global)', both set to 'Fail'. At the bottom are buttons for 'Help', '< Back', 'Next >', 'Finish >>', and 'Cancel'.

Source	Destination
C:\Users\Muhammad\Downloads\jewellery.csv	[dbo].[jewellery]

Source Column	Source Type	Destination Col...	Destination Type	Convert	On Error	On Truncati...
Age	single-byte sig...	Age	smallint	<input checked="" type="checkbox"/>	Use Global	Use Global
Income	four-byte signe...	Income	int	<input checked="" type="checkbox"/>	Use Global	Use Global
SpendingScore	float [DT_R4]	SpendingScore	real	<input checked="" type="checkbox"/>	Use Global	Use Global
Savings	float [DT_R4]	Savings	real	<input checked="" type="checkbox"/>	Use Global	Use Global

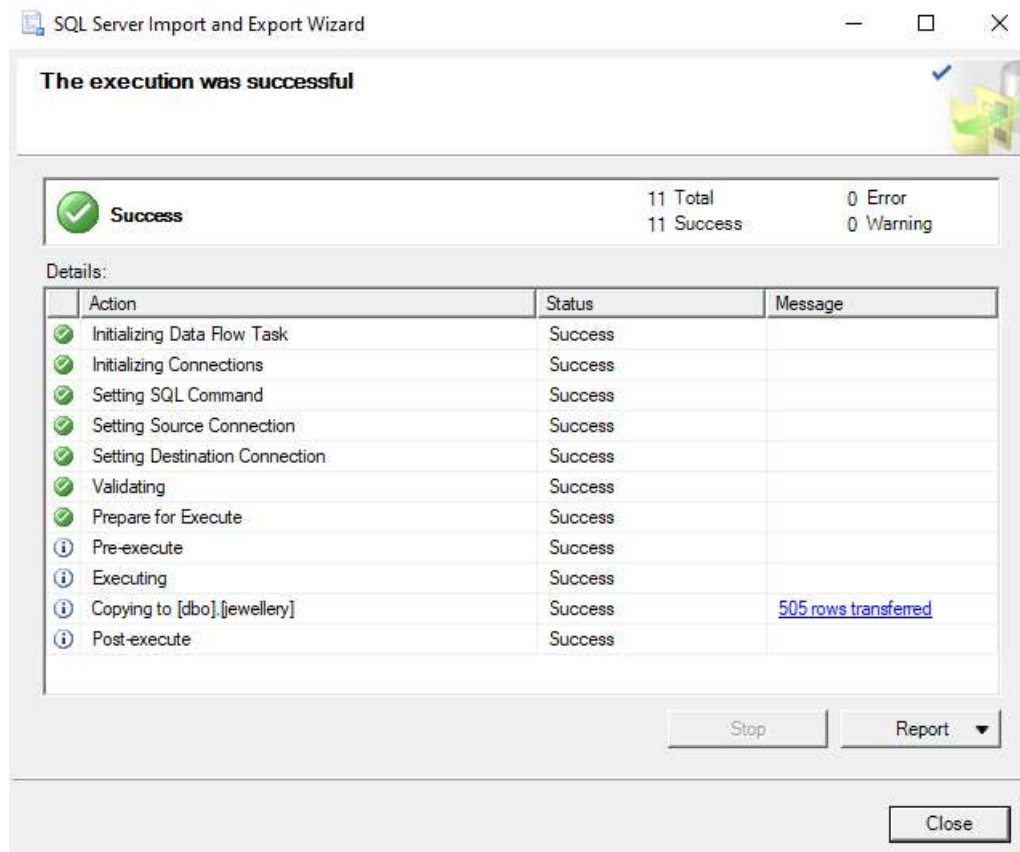
On Error (global): Fail
On Truncation (global): Fail

Check Run immediately and click the **Next** button

The screenshot shows the 'Save and Run Package' window. It has a title bar 'SQL Server Import and Export Wizard' and a subtitle 'Save and Run Package'. Below the subtitle is a description: 'Indicate whether to save the SSIS package.' There are two checkboxes: 'Run immediately' (checked) and 'Save SSIS Package' (unchecked). Below 'Save SSIS Package' are two radio buttons: 'SQL Server' (selected) and 'File system'. Below these is a 'Package protection level:' section with a dropdown menu set to 'Encrypt sensitive data with user key'. There are two password fields: 'Password:' and 'Retype password:'. At the bottom are buttons for 'Help', '< Back', 'Next >', 'Finish >>', and 'Cancel'.

Run immediately: ☒
Save SSIS Package: ☐
SQL Server: ☒
File system: ☐
Package protection level: Encrypt sensitive data with user key
Password:
Retype password:

Click the Finish button to run the package



Step 5 — Enable SQL Server for Python Scripts

We will run Python “inside” the SQL Server by using the **sp_execute_external_script** system stored procedure. To begin, you need to open a ‘**New Query**’. Execute the following query in your instance to enable the use of the procedure for remote script execution:

```
EXEC sp_configure 'external scripts enabled', 1
```

```
RECONFIGURE WITH OVERRIDE
```

Note: Restart the instance before proceeding to the next steps.

Following SQL Statements can be executed to check the Python path and list installed packages.

Check Python Path:

```
EXECUTE sp_execute_external_script
@language =N'Python',
@script=N'import sys; print("\n".join(sys.path))'
```

```
EXECUTE sp_execute_external_script
@language = N'Python',
@script=N'import sys; print("\n".join(sys.path))'
```

100 %

Messages

STDOUT message(s) from external script:

```
C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\PYTHON_SERVICES\python37.zip
C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\PYTHON_SERVICES\DLLs
C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\PYTHON_SERVICES\lib
C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\PYTHON_SERVICES
C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\PYTHON_SERVICES\lib\site-packages
C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\PYTHON_SERVICES\lib\site-packages\win32
C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\PYTHON_SERVICES\lib\site-packages\win32\lib
C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\PYTHON_SERVICES\lib\site-packages\Pythonwin
C:\ProgramData\MSSQLSERVER\Temp-PY\Appcontainer1\448BA1F4-B82D-4F3C-BDD8-734084220323\rxLibs
C:\ProgramData\MSSQLSERVER\Temp-PY\Appcontainer1\276D11DD-9540-4F65-A44A-161EF891D9E6
```

Completion time: 2020-05-07T16:21:36.5230435-04:00

List Installed Packages:

```
EXECUTE sp_execute_external_script
@language = N'Python',
@script = N'
import pkg_resources
import pandas as pd
installed_packages = pkg_resources.working_set
installed_packages_list = sorted(["%s==%s" % (i.key, i.version) for i in installed_packages])
df = pd.DataFrame(installed_packages_list)
OutputDataSet = df
'

WITH RESULT SETS (( PackageVersion nvarchar (150) ))
```

```
EXECUTE sp_execute_external_script
    @language = N'Python',
    @script = N'
import pkg_resources
import pandas as pd
installed_packages = pkg_resources.working_set
installed_packages_list = sorted(["%s==%s" % (i.key, i.version) for i in installed_packages])
df = pd.DataFrame(installed_packages_list)
OutputDataSet = df
'
WITH RESULT SETS (( PackageVersion nvarchar (150) ))
```

PackageVersion
adal==1.2.1
alabaster==0.7.12
asn1crypto==0.24.0
astropy==4.0.1.post1
atomicwrites==1.3.0
attrs==19.1.0
awscli==1.18.54
babel==2.6.0
backcall==0.1.0
backports.os==0.1.1

Step 6 — Adding PyCaret Python Package to SQL Server

To install PyCaret package, open a command prompt and browse to the location of Python packages where SQL Server is installed. The default location is:

C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\PYTHON_SERVICES

Navigate to “Scripts” directory and use pip command to install **PyCaret** package

pip.exe install pycaret

```
Microsoft Windows [Version 10.0.18363.815]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Muhammad>cd C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\PYTHON_SERVICES
C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\PYTHON_SERVICES>cd Scripts
C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\PYTHON_SERVICES\Scripts>pip.exe install pycaret

Successfully uninstalled Cython-0.29.2
Successfully installed Cython-0.29.14 DateTime-4.3 astropy-4.0.1.post1 awscli-1.18.54 blis-0.4.1 boto-2.49.0 boto3-1.13.4 botocore-1.16.4 catalogue-1.0.0 catboost-0.20.2 chart-studio-1.1.0 colorlover-0.3.0 combo-0.1.0 confuse-1.1.0 cufflinks-0.17.0 cymem-2.0.3 datefinder-0.7.0 funcy-1.14 future-0.18.2 gensim-3.8.3 graphviz-0.14 htmlmin-0.1.12 importlib-metadata-1.6.0 jmespath-0.9.5 joblib-0.14.1 kmodes-0.10.1 lightgbm-2.3.1 missingno-0.4.2 mlxtend-0.17.2 murmurhash-1.0.2 pandas-profiling-2.3.0 phik-0.9.11 plac-1.1.3 plotly-4.4.1 preshed-3.0.2 pyLDAvis-2.1.2 pyasn1-0.4.8 pycaret-1.0.0 pyod-0.7.9 regex-2020.5.7 retrying-1.3.3 rsa-3.4.2 s3transfer-0.3.3 scikit-learn-0.22 scipy-1.4.1 shap-0.32.1 singledispatch-3.4.0.3 smart-open-2.0.0 spacy-2.2.4 srsly-1.0.2 suod-0.0.4 tbb-2019.0 textblob-0.15.3 thinc-7.4.0 tqdm-4.46.0 umap-learn-0.4.2 wasabi-0.6.0 wordcloud-1.7.0 xgboost-0.90 yellowbrick-1.0.1 zipp-3.1.0 zope.interface-5.1.0

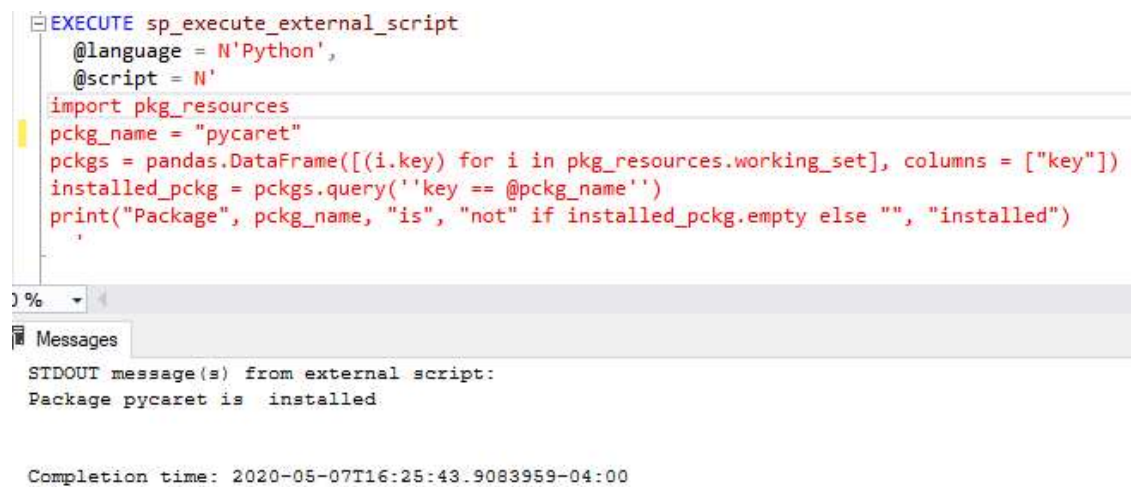
C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\PYTHON_SERVICES\Scripts>
```

Note: Make sure, you have access to the SQL Server directory to install package and/or change configurations. Otherwise, the package installation will fail.

Installation may take 5–10 minutes

Execute the following SQL to verify the PyCaret installation from SQL Server:

```
EXECUTE sp_execute_external_script
    @language = N'Python',
    @script = N'
import pkg_resources
pkg_name = "pycaret"
pckgs = pandas.DataFrame([(i.key) for i in pkg_resources.working_set], columns = ["key"])
installed_pckg = pckgs.query('key == @pckg_name')
print("Package", pkg_name, "is", "not" if installed_pckg.empty else "", "installed")'
```



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a T-SQL query that executes a Python script using the `sp_execute_external_script` stored procedure. The script imports `pkg_resources` and `pandas` to check if the 'pycaret' package is installed. The bottom pane, titled 'Messages', shows the output of the script: 'Package pycaret is installed'. At the very bottom, the completion time is noted as '2020-05-07T16:25:43.9083959-04:00'.

```
EXECUTE sp_execute_external_script
    @language = N'Python',
    @script = N'
import pkg_resources
pkg_name = "pycaret"
pckgs = pandas.DataFrame([(i.key) for i in pkg_resources.working_set], columns = ["key"])
installed_pckg = pckgs.query('key == @pckg_name')
print("Package", pkg_name, "is", "not" if installed_pckg.empty else "", "installed")'
```

Messages

STDOUT message(s) from external script:
Package pycaret is installed

Completion time: 2020-05-07T16:25:43.9083959-04:00

V. ML Experiment Example — Clustering in SQL Server

Clustering is a machine learning technique that groups data points with similar characteristics. These groupings are useful for exploring data, identifying patterns, and analyzing a subset of data. Some common business use cases for clustering are:

- ✓ Customer segmentation for the purpose of marketing.
- ✓ Customer purchasing behaviour analysis for promotions and discounts.
- ✓ Identifying geo-clusters in an epidemic outbreak such as COVID-19.

In this tutorial, we will use the 'jewellery.csv' file that is available on PyCaret's [Github repository](https://github.com/pycaret/pycaret).

Link to csv File: <https://raw.githubusercontent.com/pycaret/pycaret/master/datasets/jewellery.csv>

Age	Income	SpendingScore	Savings
58	77769	0.7913288	6559.83
59	81799	0.791082	5417.66162
62	74751	0.7026569	9258.993
59	74373	0.765679538	7346.33447
87	17760	0.348777562	16869.5078
29	131578	0.8470341	3535.5144
54	76500	0.785197854	6878.88428
87	42592	0.355289668	18086.2871
83	34384	0.324718684	14783.3789
84	27693	0.367063	17879.5586
85	111389	0.0367953628	16009.2373
36	99780	0.2654326	16398.4
30	99949	0.344679236	13621.64
31	107963	0.290508628	13407.0811
61	71933	0.8441068	8022.2085
92	122879	0.0607240349	13709.67
55	71621	0.7533427	7780.59
87	31481	0.31742397	16180.6885

Sample Data points from jewellery dataset

1. K-Means Clustering

Run the following SQL Code in SQL Server

```
EXECUTE sp_execute_external_script
@language = N'Python',
@script = N'dataset = InputDataSet
import pycaret.clustering as pc
dataset = pc.get_clusters(data = dataset)
OutputDataSet = dataset',
@input_data_1 = N'SELECT [Age], [Income], [SpendingScore], [Savings] FROM [jewellery]'
WITH RESULT SETS(([Age] INT, [Income] INT, [SpendingScore] FLOAT, [Savings] FLOAT, [Cluster]
varchar(15)));
```

2. Output

Results		Messages			
	Age	Income	SpendingScore	Savings	Cluster
1	58	77769	0.79132878780365	6559.830078125	Cluster 1
2	59	81799	0.79108202457428	5417.66162109375	Cluster 1
3	62	74751	0.702656924724579	9258.9931640625	Cluster 1
4	59	74373	0.765679538249969	7346.33447265625	Cluster 1
5	87	17760	0.348777562379837	16869.5078125	Cluster 0
6	29	131578	0.847034096717834	3535.51440429688	Cluster 3
7	54	76500	0.785197854042053	6878.88427734375	Cluster 1
8	87	42592	0.355289667844772	18086.287109375	Cluster 0
9	83	34384	0.324718683958054	14783.37890625	Cluster 0
10	84	27693	0.367062985897064	17879.55859375	Cluster 0
11	85	111389	0.036795362830162	16009.2373046875	Cluster 2
12	36	99780	0.265432596206665	16398.400390625	Cluster 2
13	30	99949	0.344679236412048	13621.6396484375	Cluster 2
14	31	107963	0.290508627891541	13407.0810546875	Cluster 2
15	61	71933	0.844106793403625	8022.20849609375	Cluster 1
16	92	122879	0.0607240349054337	13709.669921875	Cluster 2
17	55	71621	0.753342688083649	7780.58984375	Cluster 1
18	87	31481	0.317423969507217	16180.6884765625	Cluster 0
19	82	33636	0.37178298830986	17866.833984375	Cluster 0
20	88	120678	0.0632728487253189	14264.4736328125	Cluster 2
21	30	101073	0.314387112855911	14324.5556640625	Cluster 2
22	84	122696	0.082186833024025	13809.734375	Cluster 2
23	53	76667	0.760057628154755	5168.2255859375	Cluster 1
24	91	22672	0.271064519882202	15407.646484375	Cluster 0
25	89	119697	0.0916792005300522	16215.3994140625	Cluster 2
26	30	122788	0.872872412204742	5706.1494140625	Cluster 3
27	17	134966	0.907242178916931	4128.044921875	Cluster 3
28	55	78761	0.827174186706543	8376.7490234375	Cluster 1

A new column '**Cluster**' containing the label is attached to the original table.

By default, PyCaret trains a **K-Means** clustering model with 4 clusters (*i.e. all the data points in the table are categorized into 4 groups*). Default values can be changed easily:

To change the number of clusters you can use **num_clusters** parameter within **get_clusters()** function.

To change model type use **model** parameter within **get_clusters()**.

3. K-Modes

See the following code for training **K-Modes** model with **6 clusters**:

```
EXECUTE sp_execute_external_script
@language = N'Python',
@script = N'dataset = InputDataSet
import pycaret.clustering as pc
dataset = pc.get_clusters(data = dataset, model="kmodes", num_clusters = 6)
```

```
OutputDataSet = dataset',  
@input_data_1 = N'SELECT [Age], [Income], [SpendingScore], [Savings] FROM [jewellery]'  
WITH RESULT SETS((([Age] INT, [Income] INT, [SpendingScore] FLOAT, [Savings] FLOAT, [Cluster]  
varchar(15))));
```

Following these steps, you can assign cluster value to every observation point in the jewellery dataset. You can use similar steps on other datasets too, to perform clustering on them.

VI. Conclusion

In this post, we learnt how to build a clustering model using running a Python library (PyCaret) in SQL Server. Similarly, you can build and run other types of supervised and unsupervised ML models depending on the need of the business problem.

You can further check out the [PyCaret](#) website for documentation on other supervised and unsupervised experiments that can be implemented in a similar manner within SQL Server.

My future posts will be tutorials on exploring supervised learning techniques (regression/classification) using Python and Pycaret within a SQL Server.

VII. Important Links

[PyCaret](#)

[PyCaret: User guide and documentation](#)

[PyCaret: Tutorials](#)

[My LinkedIn Profile](#)