Ship Machine Learning Model to Data Using PyCaret — Part II

Binary Classification

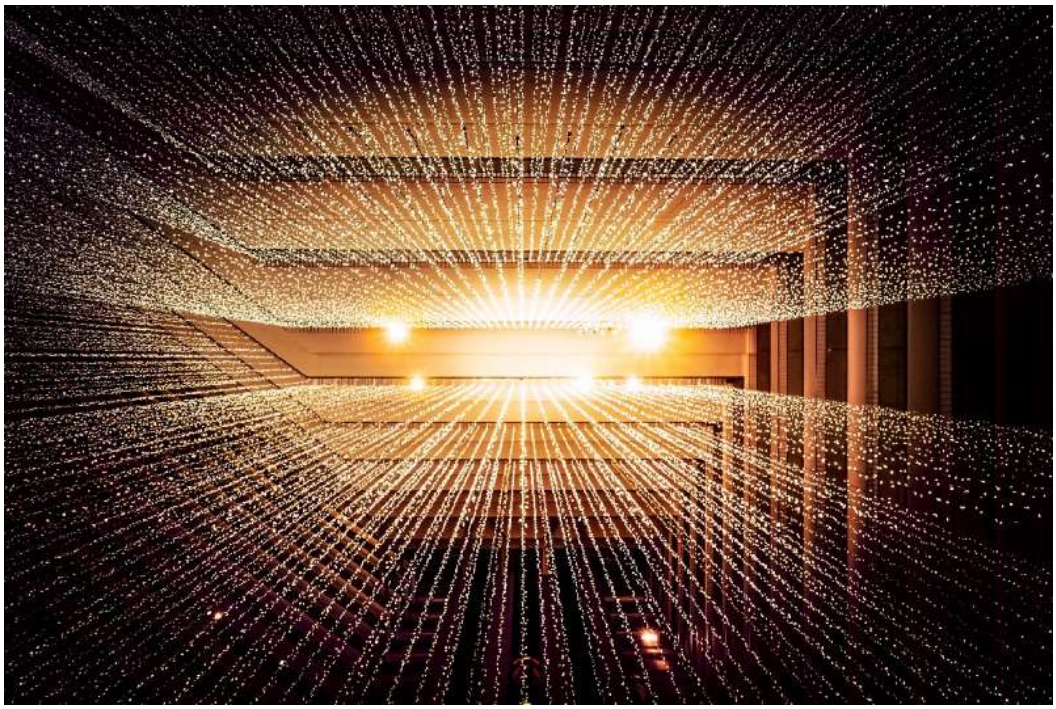


Photo by Joshua Sortino on Unsplash

My previous post **Machine Learning in SQL using PyCaret 1.0** provided details about integrating **PyCaret** with **SQL Server**. In this article, I will provide step-by-step details on how to train and deploy a Supervised Machine Learning Classification model in SQL Server using **PyCaret 2.0** (PyCaret is a low-code ML library in Python).

Things to be covered in this article:

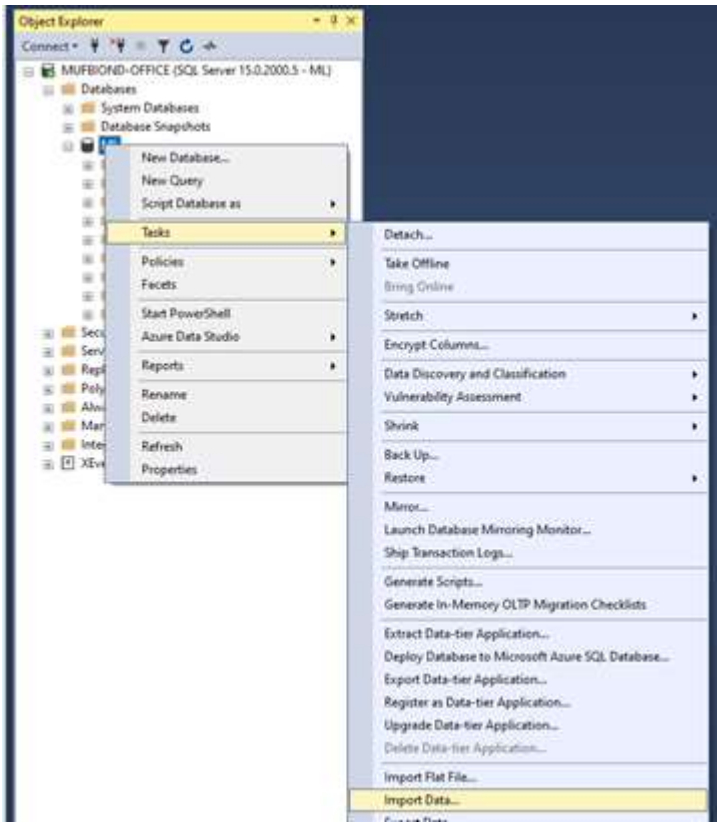
1. How to load data into SQL Server table
2. How to create and save a model in SQL Server table
3. How to make model predictions using the saved model and store results in the table

I. Import/Load Data

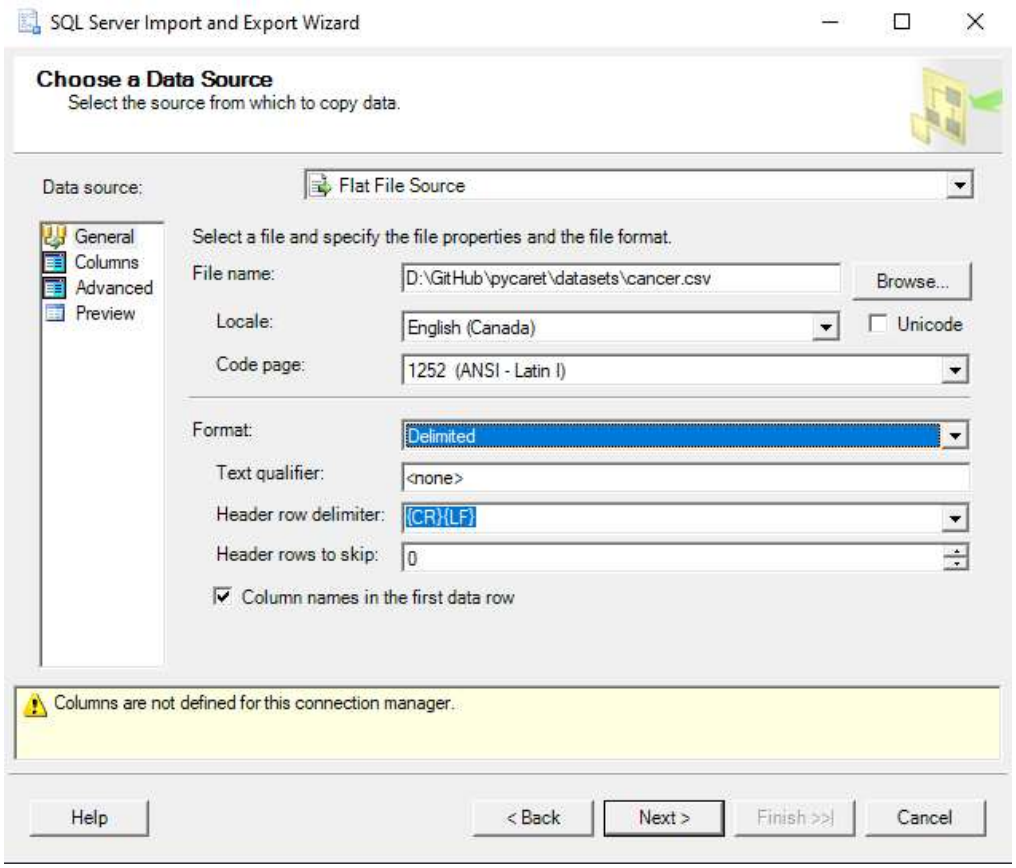
You will now have to import CSV file into a database using SQL Server Management Studio.

1. Create a table "**cancer**" in the database

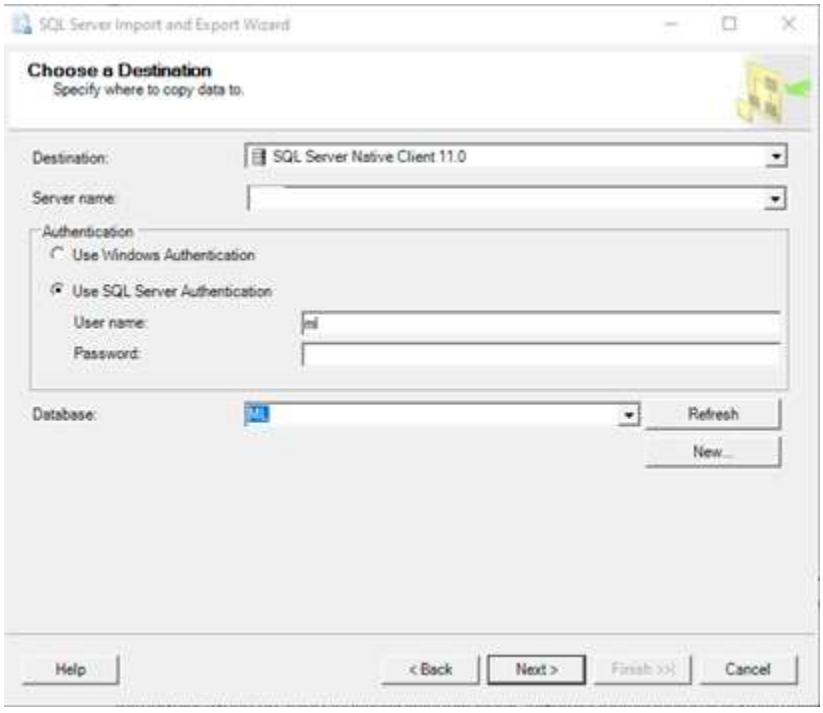
2. Right-click the database and select **Tasks -> Import Data**



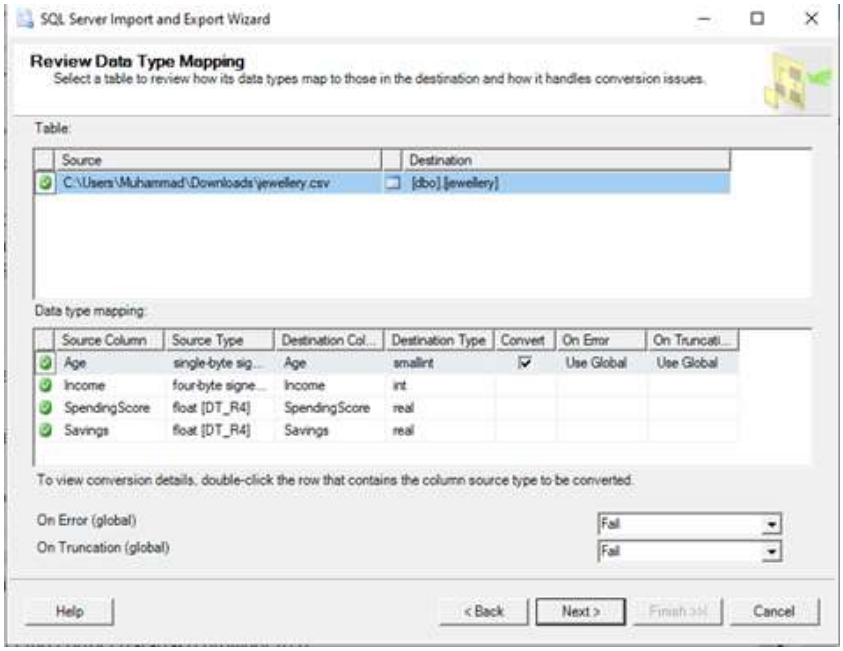
3. For Data Source, select **Flat File Source**. Then use the **Browse** button to select the CSV file. Spend some time configuring the data import before clicking the **Next** button.



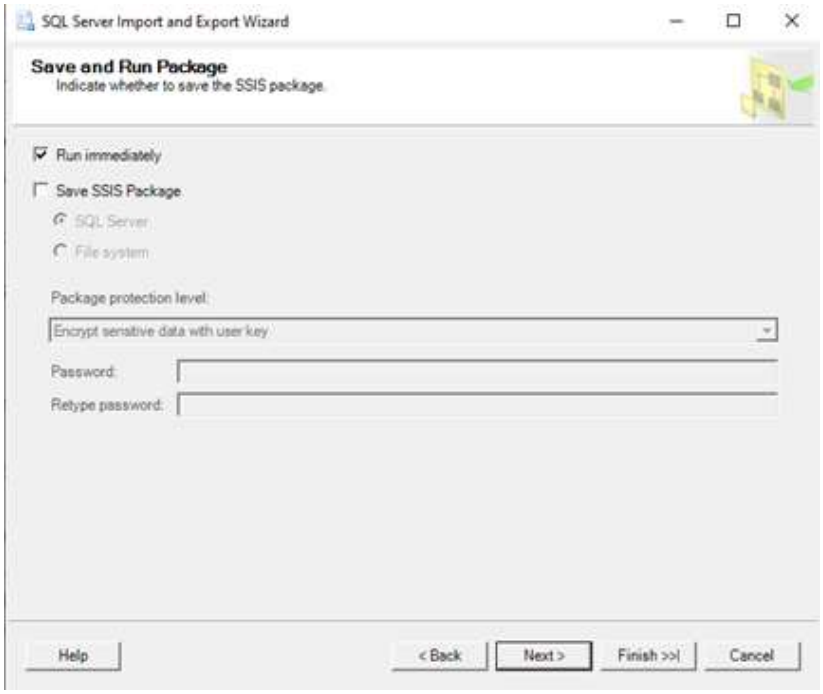
4. For Destination, select the correct database provider (e.g. SQL Server Native Client 11.0). Enter the **Server name**; check **Use SQL Server Authentication**, enter the **Username**, **Password**, and **Database** before clicking the **Next** button.



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



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



7. Click the Finish button to run the package

II. Create ML Model & Save in Database Table

Classification is a type of supervised machine learning to predict the categorical class **labels** which are discrete and unordered. The module available in the **PyCaret** package can be used for **binary** or **multiclass** problems.

In this example, we will be using a '**Brest Cancer Dataset**'. Creating and saving a model in a database table is a multi-step process. Let's go by them step by step:

i. Create a stored procedure to create a trained model in this case an Extra Trees Classifier algorithm. The procedure will read data from the cancer table created in the previous step.

Below is the code used to create the procedure:

```
-- Stored procedure that generates a PyCaret model using the cancer data using Extra Trees Classifier
Algorithm
DROP PROCEDURE IF EXISTS generate_cancer_pycaret_model;
Go
CREATE PROCEDURE generate_cancer_pycaret_model (@trained_model varbinary(max) OUTPUT) AS
BEGIN
EXECUTE sp_execute_external_script
@language = N'Python'
, @script = N'
import pycaret.classification as cp
import pickle
trail1 = cp.setup(data = cancer_data, target = "Class", silent = True, n_jobs=None)
# Create Model
et = cp.create_model("et", verbose=False)
#To improve our model further, we can tune hyper-parameters using tune_model function.
#We can also optimize tuning based on an evaluation metric. As our choice of metric is F1-score, lets
optimize our algorithm!
tuned_et = cp.tune_model(et, optimize = "F1", verbose=False)
#The finalize_model() function fits the model onto the complete dataset.
#The purpose of this function is to train the model on the complete dataset before it is deployed in
production
final_model = cp.finalize_model(tuned_et)

# Before saving the model to the DB table, convert it to a binary object
trained_model = []
prep = cp.get_config("prep_pipe")
trained_model.append(prep)
trained_model.append(final_model)
trained_model = pickle.dumps(trained_model)'
, @input_data_1 = N'select "Class", "age", "menopause", "tumor_size", "inv_nodes", "node_caps",
"deg_malig", "breast", "breast_quad", "irradiat" from dbo.cancer'
, @input_data_1_name = N'cancer_data'
, @params = N'@trained_model varbinary(max) OUTPUT'
, @trained_model = @trained_model OUTPUT;
END;
GO
```

ii. Create a table that is required to store the trained model object

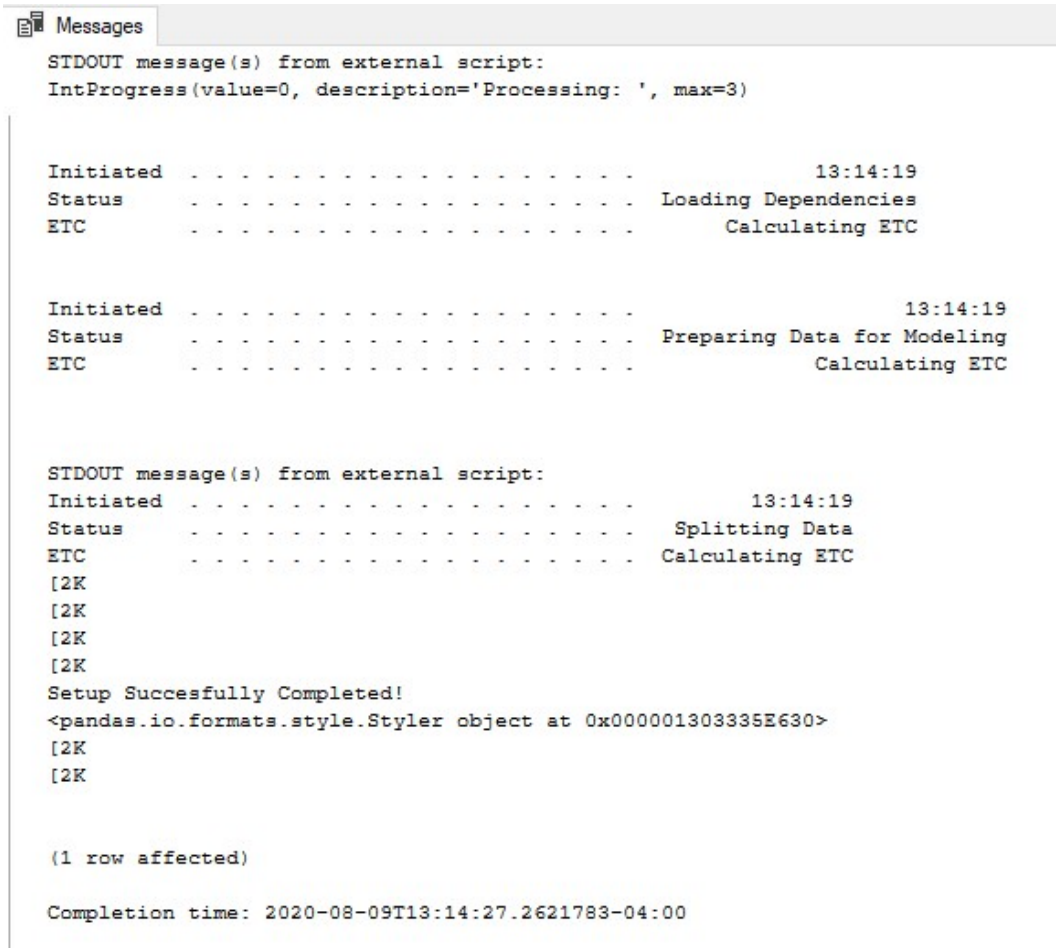
```
DROP TABLE IF EXISTS dbo.pycaret_models;
GO
```

```
CREATE TABLE dbo.pycaret_models (
model_id  INT NOT NULL PRIMARY KEY,
dataset_name VARCHAR(100) NOT NULL DEFAULT('default dataset'),
model_name  VARCHAR(100) NOT NULL DEFAULT('default model'),
model   VARBINARY(MAX) NOT NULL
);
GO
```

iii. Invoke stored procedure to create a model object and save into a database table

```
DECLARE @model VARBINARY(MAX);
EXECUTE generate_cancer_pycaret_model @model OUTPUT;
INSERT INTO pycaret_models (model_id, dataset_name, model_name, model) VALUES(2, 'cancer', 'Extra
Trees Classifier algorithm', @model);
```

The output of this execution is:



Output from Console

The view of table results after saving model

	model_id	dataset_name	model_name	model
▶	1	cancer	Logistic Regression	<Binary data>
	2	cancer	Extra Trees Classifier algorithm	<Binary data>
*	NULL	NULL	NULL	NULL

SQL Server Table Results

III. Running Predictions

The next step is to run the prediction for the test dataset based on the saved model. This is again a multi-step process. Let's go through all the steps again.

i. Create a stored procedure that will use the test dataset to detect cancer for a test datapoint

Below is the code to create a database procedure:

```
DROP PROCEDURE IF EXISTS pycaret_predict_cancer;
GO
CREATE PROCEDURE pycaret_predict_cancer (@id INT, @dataset varchar(100), @model varchar(100))
AS
BEGIN
DECLARE @py_model varbinary(max) = (select model
from pycaret_models
where model_name = @model
and dataset_name = @dataset
and model_id = @id
);
EXECUTE sp_execute_external_script
@language = N'Python',
@script = N'
# Import the scikit-learn function to compute error.
import pycaret.classification as cp
import pickle
cancer_model = pickle.loads(py_model)

# Generate the predictions for the test set.
predictions = cp.predict_model(cancer_model, data=cancer_score_data)

OutputDataSet = predictions
print(OutputDataSet)
'
, @input_data_1 = N'select "Class", "age", "menopause", "tumor_size", "inv_nodes", "node_caps",
"deg_malig", "breast", "breast_quad", "irradiat" from dbo.cancer'
, @input_data_1_name = N'cancer_score_data'
, @params = N'@py_model varbinary(max)'
, @py_model = @py_model
with result sets (("Class" INT, "age" INT, "menopause" INT, "tumor_size" INT, "inv_nodes" INT,
"node_caps" INT, "deg_malig" INT, "breast" INT, "breast_quad" INT,
"irradiat" INT, "Class_Predict" INT, "Class_Score" float ));
END;
GO
```

ii. Create a table to save the predictions along with the dataset

```
DROP TABLE IF EXISTS [dbo].[pycaret_cancer_predictions];
GO
CREATE TABLE [dbo].[pycaret_cancer_predictions](
[Class_Actual] [nvarchar] (50) NULL,
[age] [nvarchar] (50) NULL,
[menopause] [nvarchar] (50) NULL,
[tumor_size] [nvarchar] (50) NULL,
[inv_nodes] [nvarchar] (50) NULL,
[node_caps] [nvarchar] (50) NULL,
[deg_malig] [nvarchar] (50) NULL,
[breast] [nvarchar] (50) NULL,
[breast_quad] [nvarchar] (50) NULL,
[irradiat] [nvarchar] (50) NULL,
[Class_Predicted] [nvarchar] (50) NULL,
[Class_Score] [float] NULL
) ON [PRIMARY]
GO
```

iii. Call pycaret_predict_cancer procedure to save predictions result into a table

```
--Insert the results of the predictions for test set into a table
INSERT INTO [pycaret_cancer_predictions]
EXEC pycaret_predict_cancer 2, 'cancer', 'Extra Trees Classifier algorithm';
```

iv. Execute the SQL below to view the result of the prediction

```
-- Select contents of the table
SELECT * FROM [pycaret_cancer_predictions];
```

Results | Messages

	Class_Actual	age	menopause	tumor_size	inv_nodes	node_caps	deg_malig	breast	breast_quad	irradiat	Class_Predicted	Class_Score
1	0	5	1	1	1	2	1	3	1	1	0	0.0117
2	0	5	4	4	5	7	10	3	2	1	1	0.7782
3	0	3	1	1	1	2	2	3	1	1	0	0.0545
4	0	6	8	8	1	3	4	3	7	1	1	0.6151
5	0	4	1	1	3	2	1	3	1	1	0	0.039
6	1	8	10	10	8	7	10	9	7	1	1	0.9692
7	0	1	1	1	1	2	10	3	1	1	0	0.2131
8	0	2	1	2	1	2	1	3	1	1	0	0.0286
9	0	2	1	1	1	2	1	1	1	5	0	0.0816
10	0	4	2	1	1	2	1	2	1	1	0	0.0466
11	0	1	1	1	1	1	1	3	1	1	0	0.0222
12	0	2	1	1	1	2	1	2	1	1	0	0.0084
13	1	5	3	3	3	2	3	4	4	1	1	0.5765
14	0	1	1	1	1	2	3	3	1	1	0	0.0521
15	1	8	7	5	10	7	9	5	5	4	1	0.8917
16	1	7	4	6	4	6	1	4	3	1	1	0.6809
17	0	4	1	1	1	2	1	2	1	1	0	0.0107

Predictions Result

IV. Conclusion

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

Photo by Tobias Fischer on Unsplash

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 other supervised & unsupervised learning techniques using Python and **PyCaret** within a S**QL Server**.

V. Important Links

PyCaret

My LinkedIn Profile