SQL Server ML Services in Production

How to Use This thing

Agenda

- Faster time to insight
- The Wheel of Data Science

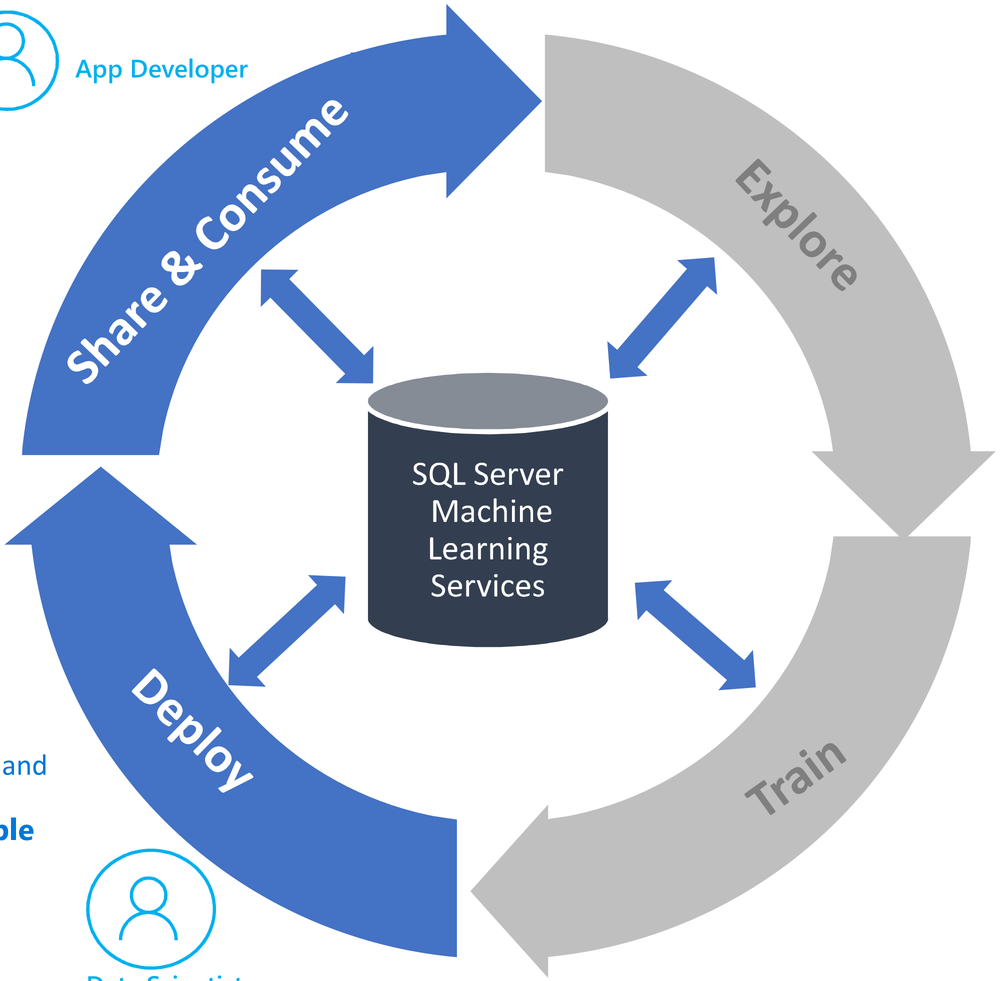
Faster Time to Insights

- Integration with SQL query execution
  - Parallel query pushing data to multiple external processes / threads
  - Use in-memory technology and Columnstore Indexes alongside your ML scripts
- Streaming mode execution
  - Stream data in batches to the R/Python process to scale beyond available memory
- Train and Predict using parallelism
  - Leverage RevoScaleR/revoscalepy and scale your R and Python scripts using multi-threading and parallel processing
- Native scoring for faster real-time predictions (New in 2017)

The Wheel of Data Science

App Developer

Make apps intelligent - Consume machine learning models from any app with a **simple stored proc call**



Explore and experiment with SQL Server data in your favorite IDE - use R, Python, T-SQL.

Data Scientist

Train models remotely on SQL Server from your IDE

Train models using R/Python code embedded in T-SQL stored proc.

SQL Developer/DBA

Deploy and schedule training and prediction scripts in-DB - make models and predictions **accessible to any application**

Manage model version in DB

Data Scientist

Data Scientist

- Works against the database.
- Explores, trains models.
- Come up with a great model.
- Now What?
  - how to deploy, and where?

Deployment - I

- Serialise the model
  - CRANR - `serialise`
  - Python - `pickle.dumps`
  - RevoScaleR - `rxSerializeModel`
  - revoscalepy - `rx_serialize_model`
- Save the model to a table as `varbinary(max)`
  - ODBC - insert into a table via stored procedure
  - RevoScaleR - `rxWriteObject`
  - revoscalepy - `rx_write_object`
- Model created through SPEES can be directly inserted in T-SQL

Deployment - II

```
sqlServerCtxString <- "Driver=SQL Server;server=.\\sqlsat; database=SqlSatDb; uid=sa;pwd=sapwd"
# use compute context
sqlCtx <- RxInSqlServer(connectionString = sqlServerCtxString, numTasks = 4)
# set the compute context to be the sql context
rxSetComputeContext(sqlCtx)

mydata <- RxSqlServerData(sqlQuery = …, connectionString = sqlServerCtxString);

logitObj <- rxLogit(tipped ~ passenger_count + trip_distance + trip_time_in_secs +
                    direct_distance, data = mydata)
modelbin <- serialize(logitObj, NULL)
modelbinstr = paste(modelbin, collapse = "")

library("RODBC")
#this is for persisting the model to disk in SQL Server
conn <- "Driver={SQL Server native Client 11.0}; server=.\\sqlsat;database=SqlSatDb;uid=sa;pwd=sapwd"

conn <- odbcDriverConnect(connection = conn)
q <- paste("EXEC dbo.pr_UpsertModel @ModelName = 'TestModel',  @Model = '", modelbinstr, "'", sep = "")
sqlQuery(conn, q)
```

Scoring / Predicting

- Load model from table.
- Call score / predict method passing in data and model
  - CRANR - `predict`
  - RevoScaleR - `rxPredict`
  - revoscalepy - `rx_predict`
- fdasf
- sdfsdf

Score / Predict - II

```
DECLARE @inData nvarchar(max) = 'select TOP(10000) passenger_count, trip_distance,
                                                    trip_time_in_secs,
dbo.fn_CalculateDistance(pickup_latitude, pickup_longitude,  dropoff_latitude, dropoff_longitude)
as direct_distance
from dbo.tb_NYCityTaxi tablesample (1 percent) repeatable (98052)'

DECLARE @lmodel2 varbinary(max) = (SELECT TOP 1 ModelBin  FROM dbo.tb_Model);
  EXEC sp_execute_external_script @language = N'R',
     @script = N'
       mod <- unserialize(as.raw(model));
       OutputDataSet<-rxPredict(modelObject = mod, data = InputDataSet, outData = NULL,
         predVarNames = "Score", type = "response", writeModelVars = FALSE, overwrite = TRUE);',
  @input_data_1 = @inData,
  @params = N'@model varbinary(max)',
  @model = @lmodel2
  WITH RESULT SETS ((Score float));
GO
```

Operatioanalize Scoring / Predicting

- To operationalize you wrap the call to SPEES in a store procedure.
- You pass in the values to score into the procedure.
- You can score both in batch as well as single event.
- Best practice is to store the score / prediction in a table for later analysis.

Single Event Proc

```
CREATE PROCEDURE dbo.pr_PredictTip
                        @passenger_count int, @trip_distance float, @trip_time_in_secs int,
                        @pickup_latitude varchar(30), @pickup_longitude varchar(30),
                        @dropoff_latitude varchar(30), @dropoff_longitude varchar(30)
AS
BEGIN
SET NOCOUNT ON;

DECLARE @direct_distance float = (SELECT dbo.fn_CalculateDistance(@pickup_latitude, @pickup_longitude,  @dropoff_latitude,
@dropoff_longitude));

DECLARE @inData nvarchar(max) = 'SELECT @passenger_count as passenger_count, …'
DECLARE @model varbinary(max) = (SELECT TOP 1 ModelBin  FROM dbo.tb_Model);

 EXEC sp_execute_external_script @language = N'R',
     @script = N'
       mod <- unserialize(as.raw(model));
       …',
  @input_data_1 = @inData,
  @params = N'@model varbinary(max), @passenger_count int, @trip_distance float, @trip_time_in_secs int, @direct_distance
float',
  @model = @model, @passenger_count = @passenger_count, @trip_distance = @trip_distance,
  @trip_time_in_secs = @trip_time_in_secs,@direct_distance = @direct_distance
  WITH RESULT SETS ((Score float))
END
```

http://nielsberglund.com

Execute Proc

```
EXEC dbo.pr_PredictTip  @passenger_count = 4,
                        @trip_distance = 7.37,
                        @trip_time_in_secs = 6000,
                        @pickup_latitude = '40.758606999999998',
                        @pickup_longitude = '-73.991602',
                        @dropoff_latitude = '40.713977',
                        @dropoff_longitude = '-73.979772999999994'
```

Real Time Scoring

- Real Time Scoring (RTS) introduced in SQL Server 2016 (after release)
- Scoring via a SQLCLR procedure: `sp_RxPredict`.
- Supports models from certain RevoScaleR, revoscalepy and Microsoft ML algorithms.
- RTS does not require the external engine to be installed, only the model.

```
DECLARE @irismodel varbinary(max)
SELECT @irismodel = [native_model_object] from [ml_models]
WHERE model_name = 'iris.dtree'
AND model_version = 'v1'

EXEC sp_rxPredict @model = @irismodel,
                  @inputData = N'SELECT * FROM iris_rx_data'
```

Native Scoring

- Native Scoring (NS) introduced in SQL Server 2017 via T-SQL PREDICT.
- Uses native C++ libraries.
- Reads the binary model and scores without the overhead of R or Python.

```
DECLARE @model varbinary(max) = (
  SELECT native_model_object
  FROM ml_models
  WHERE model_name = 'iris.dtree' AND model_version = 'v1');

SELECT d.*, p.*
  FROM PREDICT(MODEL = @model, DATA = dbo.iris_rx_data as d)
  WITH(setosa_Pred float, versicolor_Pred float, virginica_Pred float) as p;
GO
```

Summary

- Creating a model.
- Storing the model in a table.
- When scoring retrieve the model and pass data to the model.
- Wrap the call to SPEES in an outer procedure.
- Real Time Scoring via SQLCLR procedure
- Native Scoring via T-SQL Predict.