Microsoft

sqlbits

SQL and AI

Robin Lester

Toolings

Microsoft SQL Server

Visual Studio

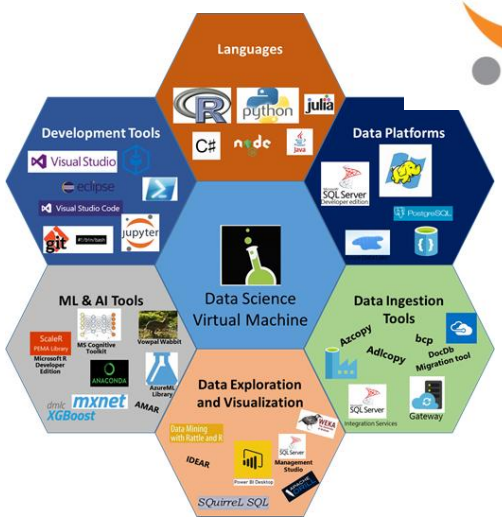
Microsoft CNTK

Bot Framework

Azure ML

jupyter

Microsoft Machine Learning Server



Azure SDK

Cognitive Services

Power BI

HD Insight

databricks

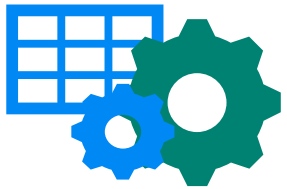
Spark MLlib

Developer | Data Professional | Data Scientist

Challenges in Machine Learning today

Data Movement
Moving data from the database to the R/Python runtimes becomes painful as data volumes grow and carries security and compliance risks

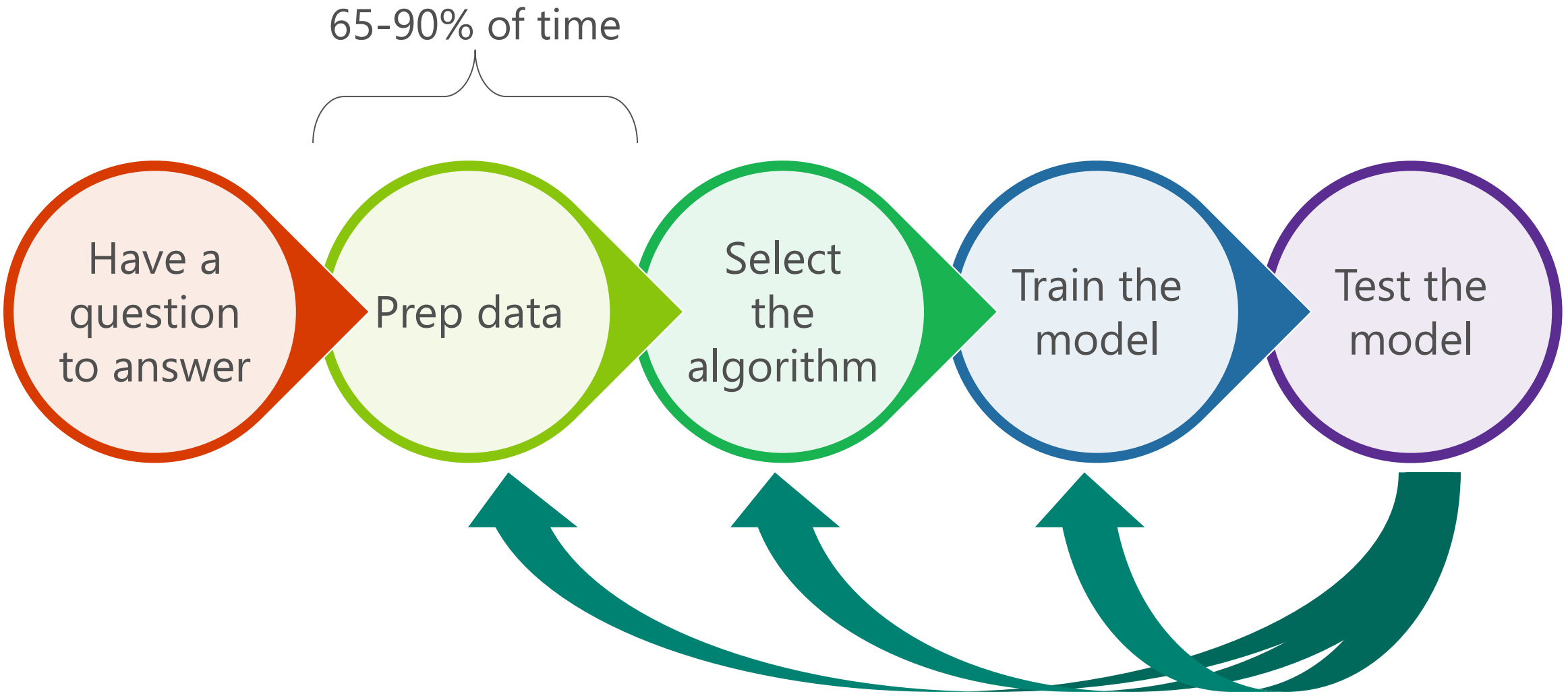
Data Transformation
Preparing and transforming data for cleansing and feature engineering is time consuming



Deployment
How do I call a machine learning script from my production application and how do I operationalize my models?

Scale & Performance
Many R/Python functions run single threaded and only accommodate datasets that fit into available memory.

Machine Learning Project Flow



SQL Server Machine learning Services on prem and cloud

SQL Server

- Extensibility Framework
  - R (2016)
  - Python (2017)
  - Java (2019)
- Native Scoring using PREDICT
- In-database Package Management
- Microsoft Machine Learning Server

Azure SQL Database

- Native scoring using PREDICT
- R Support in West Central US (Preview)
  - Base R packages
  - RevoScaleR package
  - Trivial parallelism & streaming support

SQL and Machine Learning Services

SQL 2016

- R Support (3.2.2)
- Microsoft R Server
- Realtime Scoring sp_rxPredict

SQL 2017

- Python Support (3.5.2)
- R Support (3.3.3)
- Native Scoring (PREDICT) for Windows and Linux
- In-database Package Management (CREATE EXTERNAL LIBRARY / remote deployment with rxInstallPackages)
- Libraries MicrosoftML and revoscalepy
- Pre-trained Models
- SQL Server Machine Learning Server (Standalone)

SQL 2019

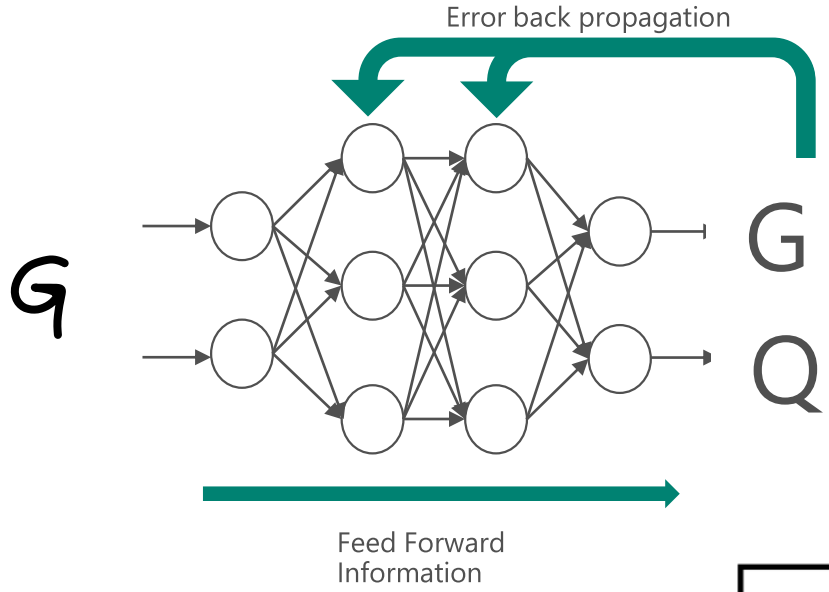
- Linux support for R and Python

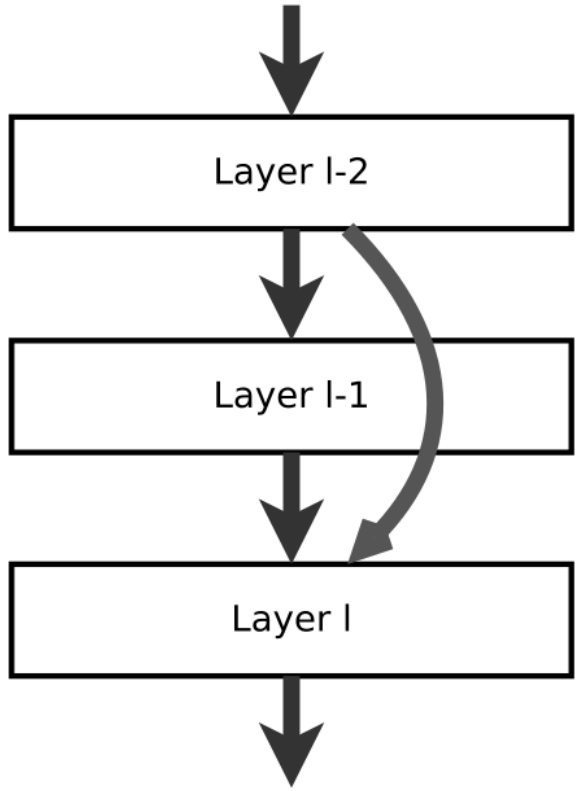
Azure SQL Database

- Preview of R Services (vCore model)
- Native Scoring using PREDICT

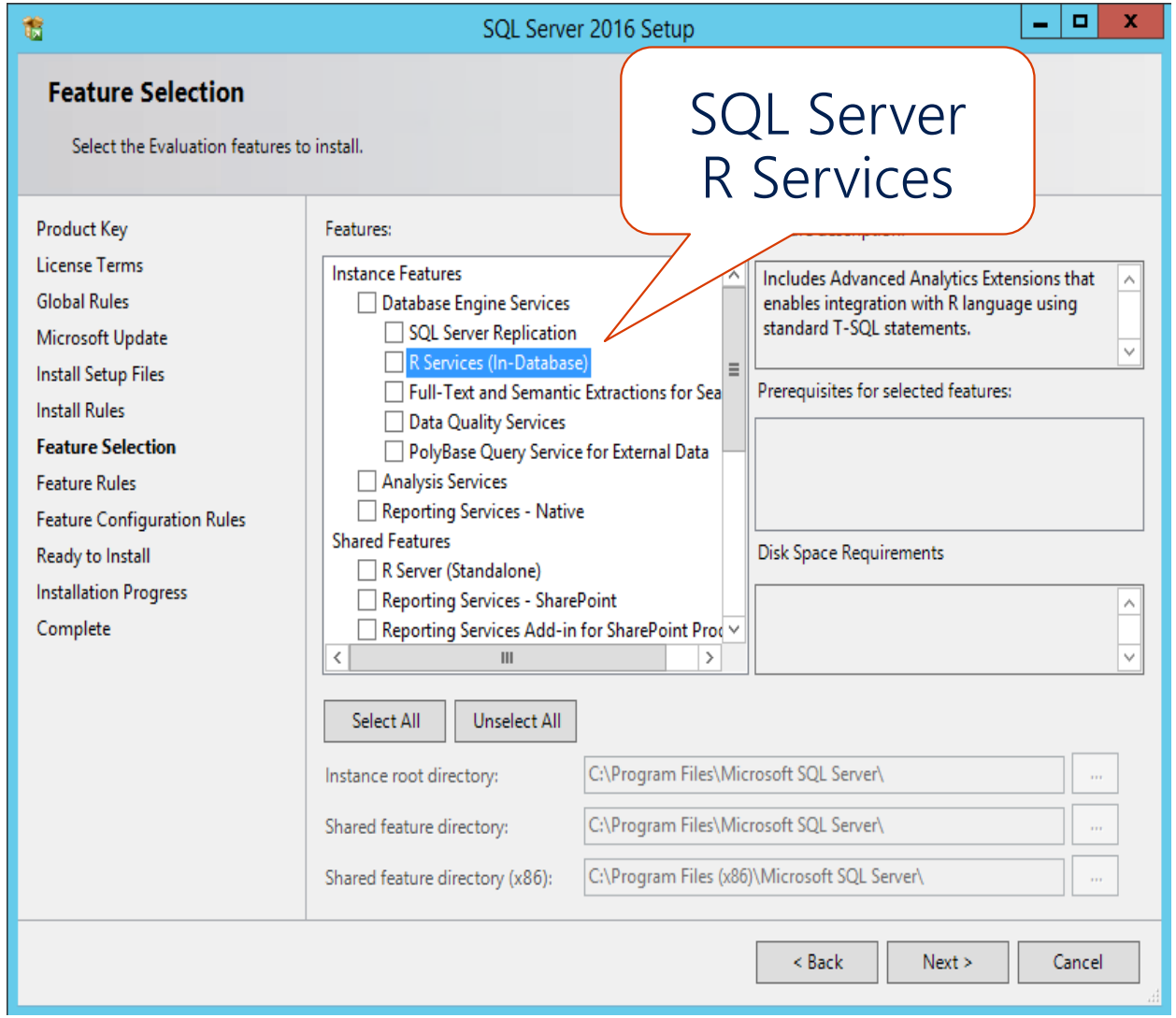
DNN Models in SQL

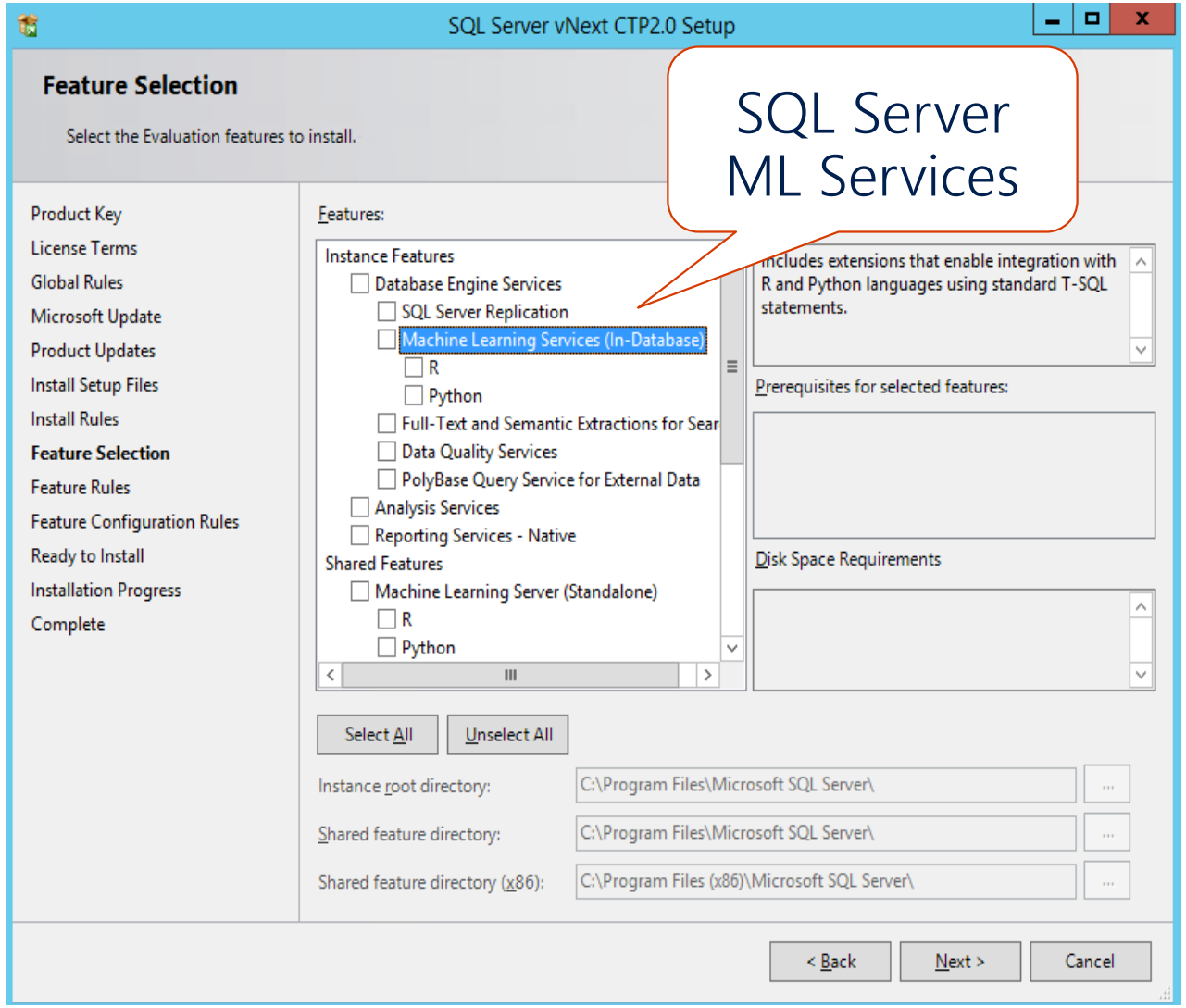
- Pretrained ML Models
  - Convolutional neural network
    - AlexNet
  - Residual neural network
    - ResNet-18
    - ResNet-50
    - ResNet-101
  - https://docs.microsoft.com/en-us/sql/advanced-analytics/r/install-pretrained-models-sql-server
- Use for
  - Sentiment Analysis / Image Featurisation
- Demo





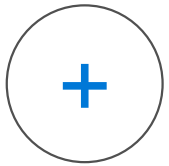
SQL Server Machine Learning Services





Microsoft R Server

Distributed R Algorithms
Parallel Execution
Enterprise Grade Operationalization
Real-time and Batch Scoring
R Client

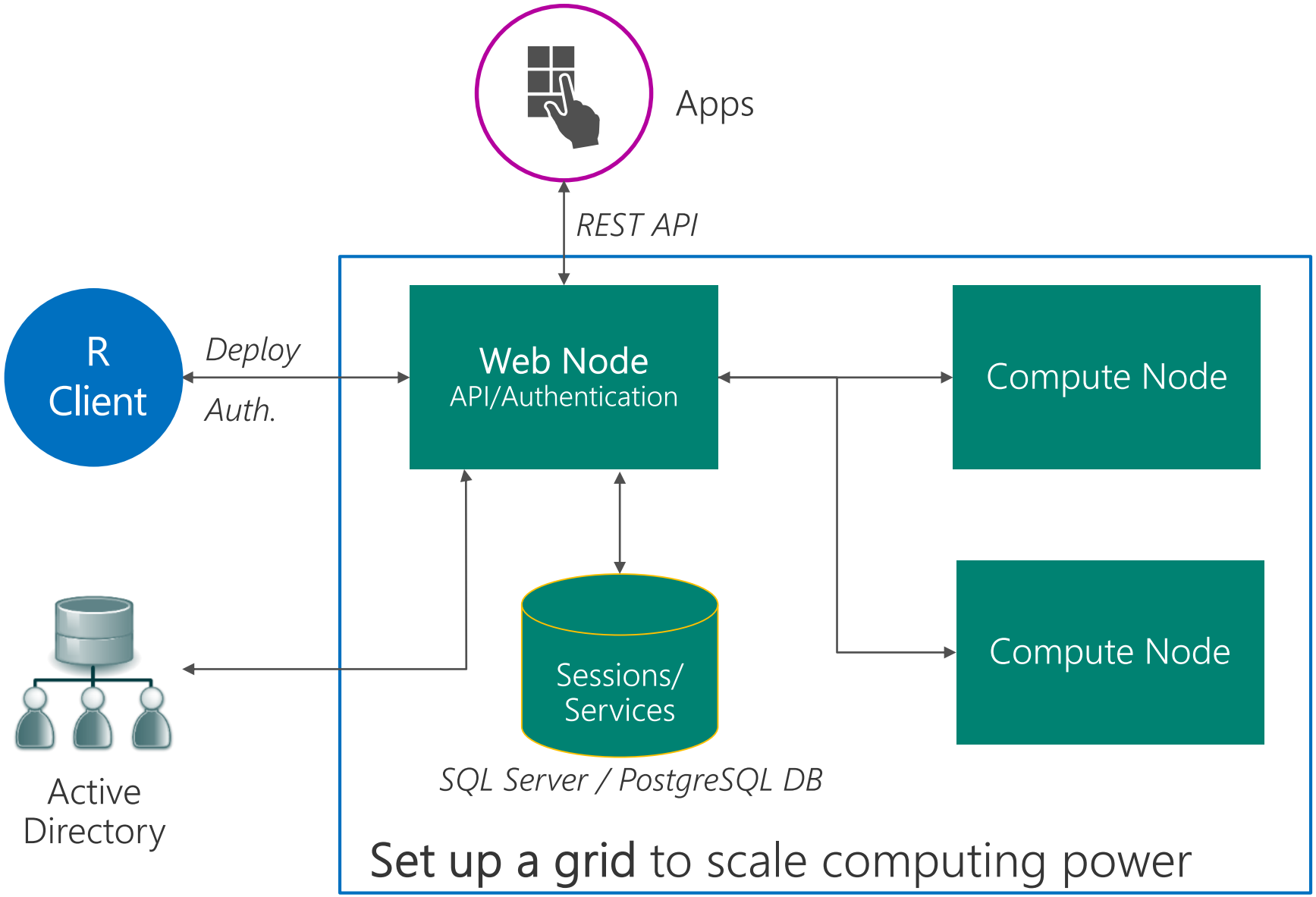


AI with Python

Full Support for AI toolkits
Distributed Python algorithms
Pre-trained cognitive models
Hadoop licensing changes

Microsoft Machine Learning Server

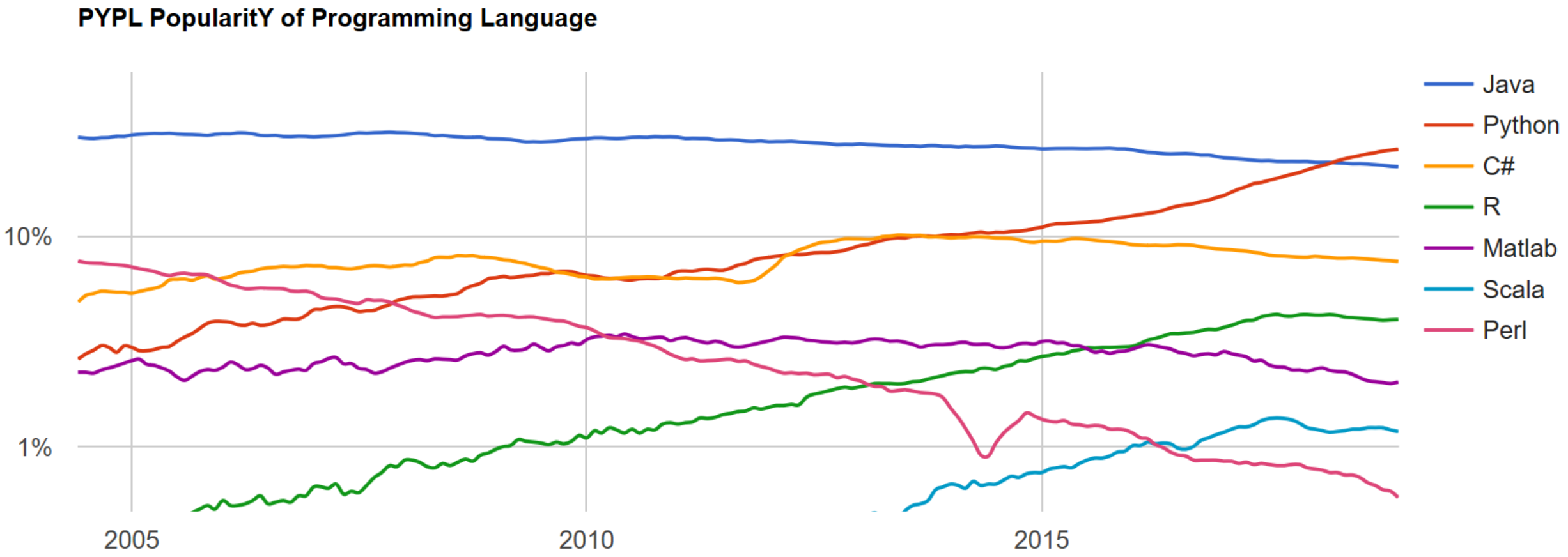
Microsoft Machine Learning Server



- Easily scale up a single server to a grid to handle more concurrent requests
- Load balancing cross compute nodes
- A shared pool of warmed up R shells to improve scoring performance.

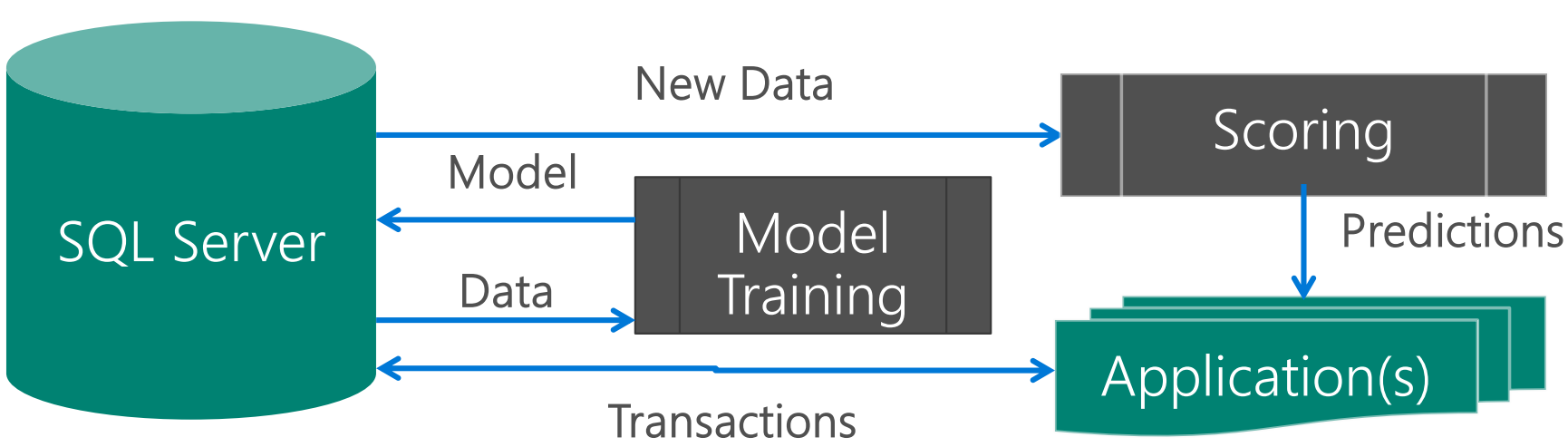
Why R and Python?

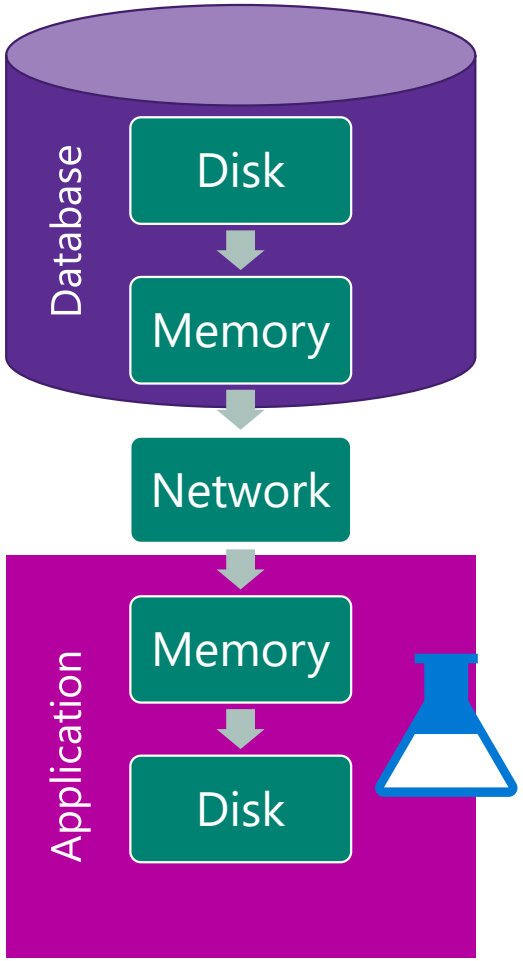
Popularity of Programming Languages



http://pypl.github.io/PYPL.html

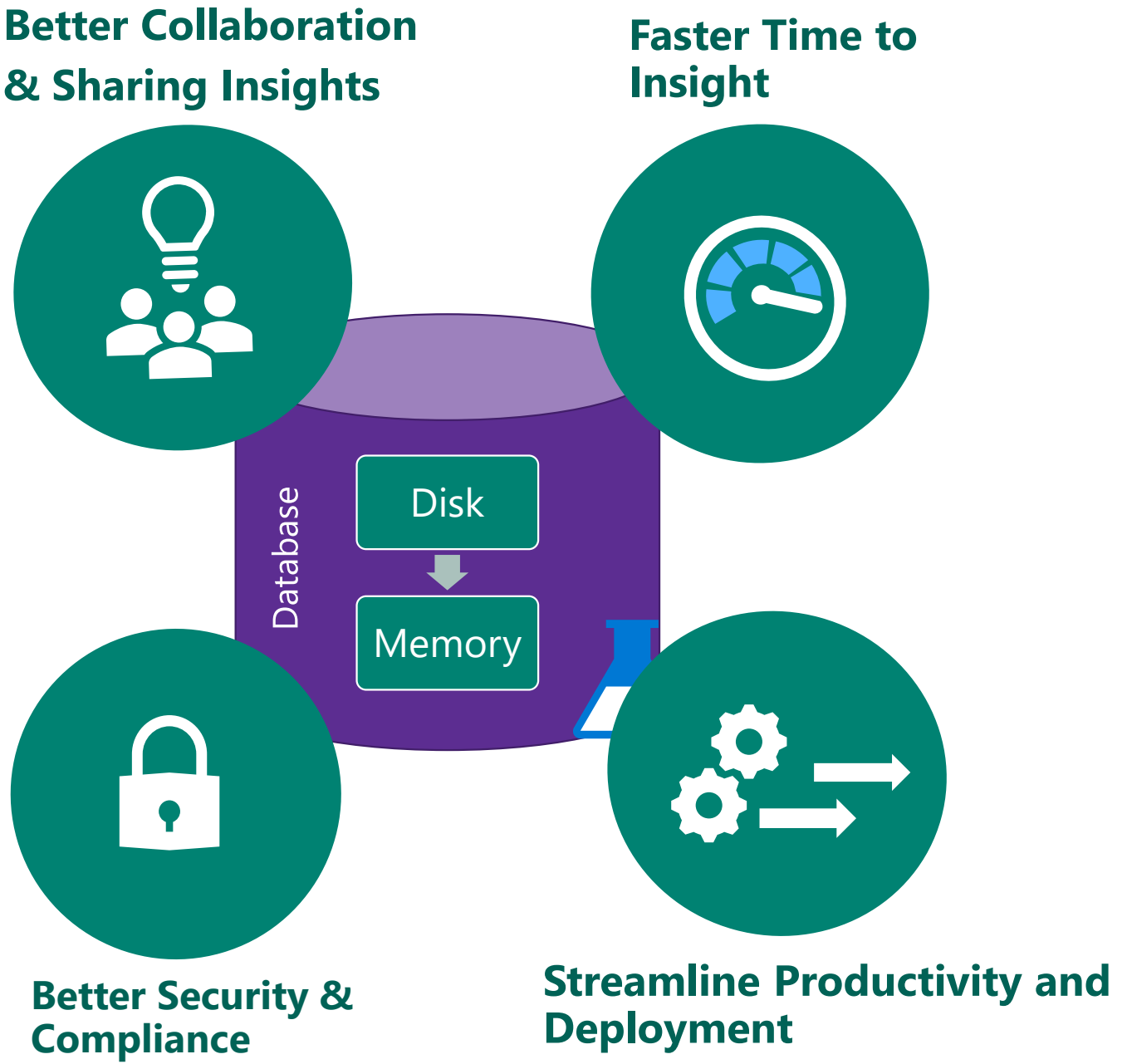
Traditional Approach to ML



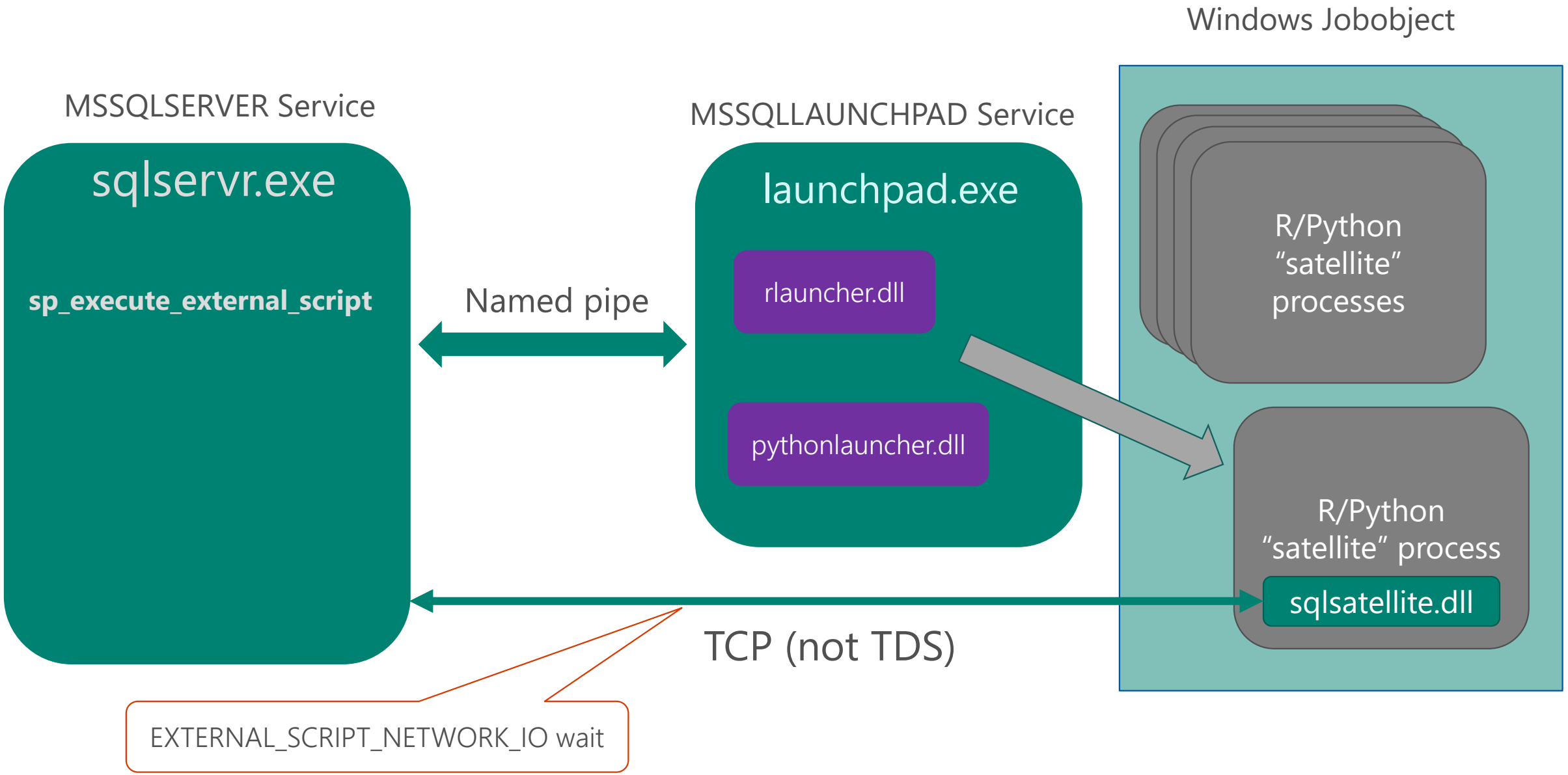


Data movement is slow!

Push ML compute to the database

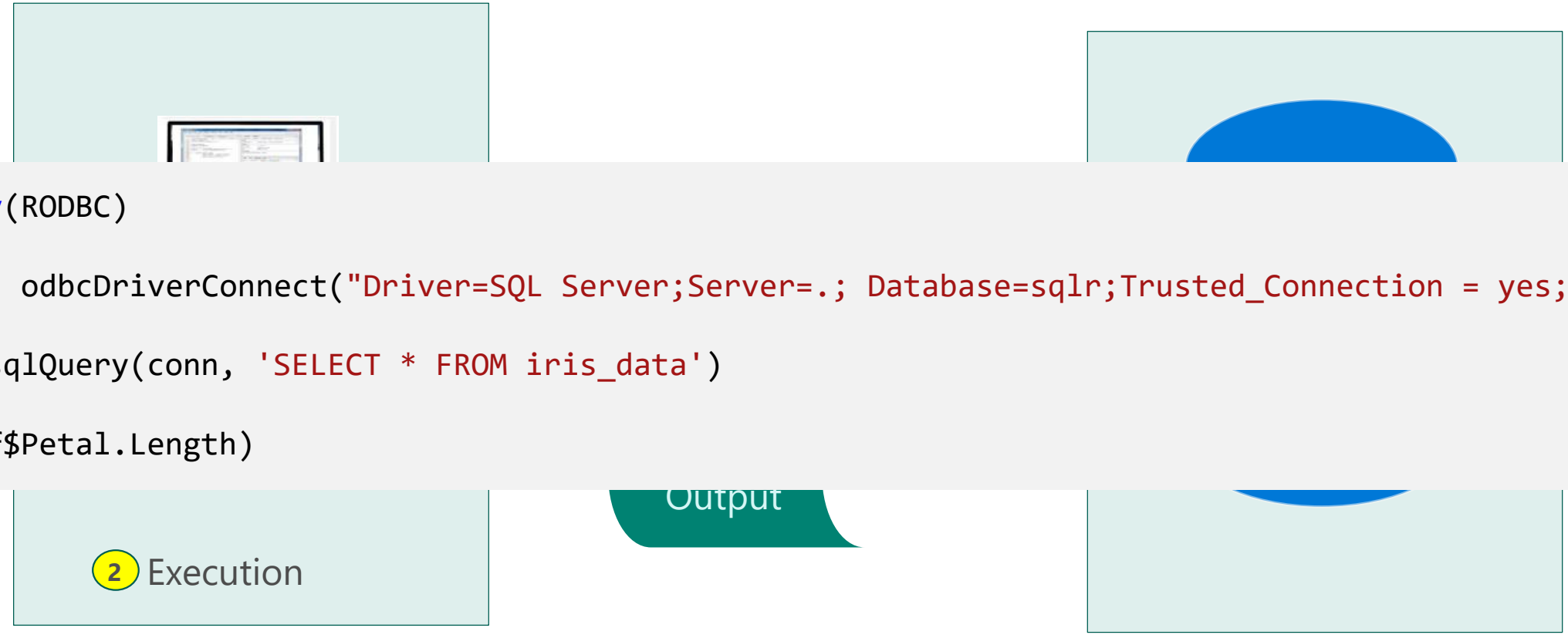


The SQL Extensibility Architecture



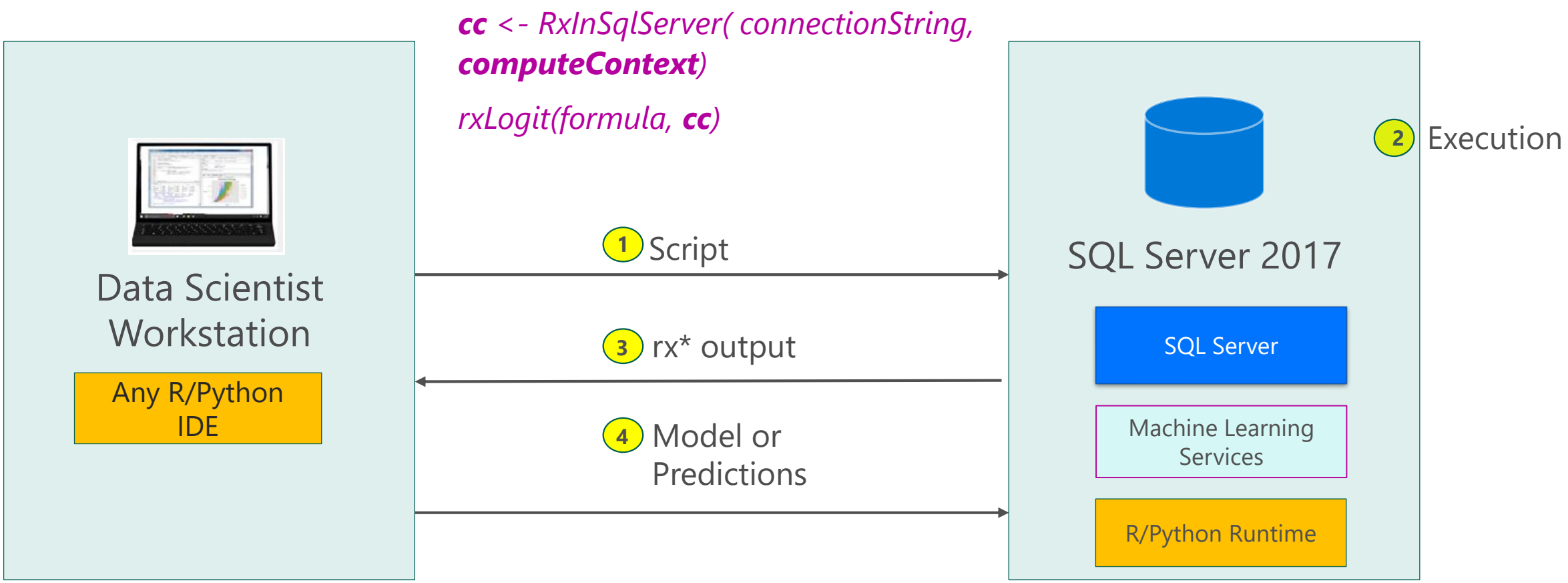
Ways of Executing R Against SQL Data

1) Remote Client via RODBC




```
library(RODBC)

conn <- odbcDriverConnect("Driver=SQL Server;Server=.; Database=sqlr;Trusted_Connection = yes;")

df <- sqlQuery(conn, 'SELECT * FROM iris_data')

hist(df$Petal.Length)
```

2) Remote Client via compute context

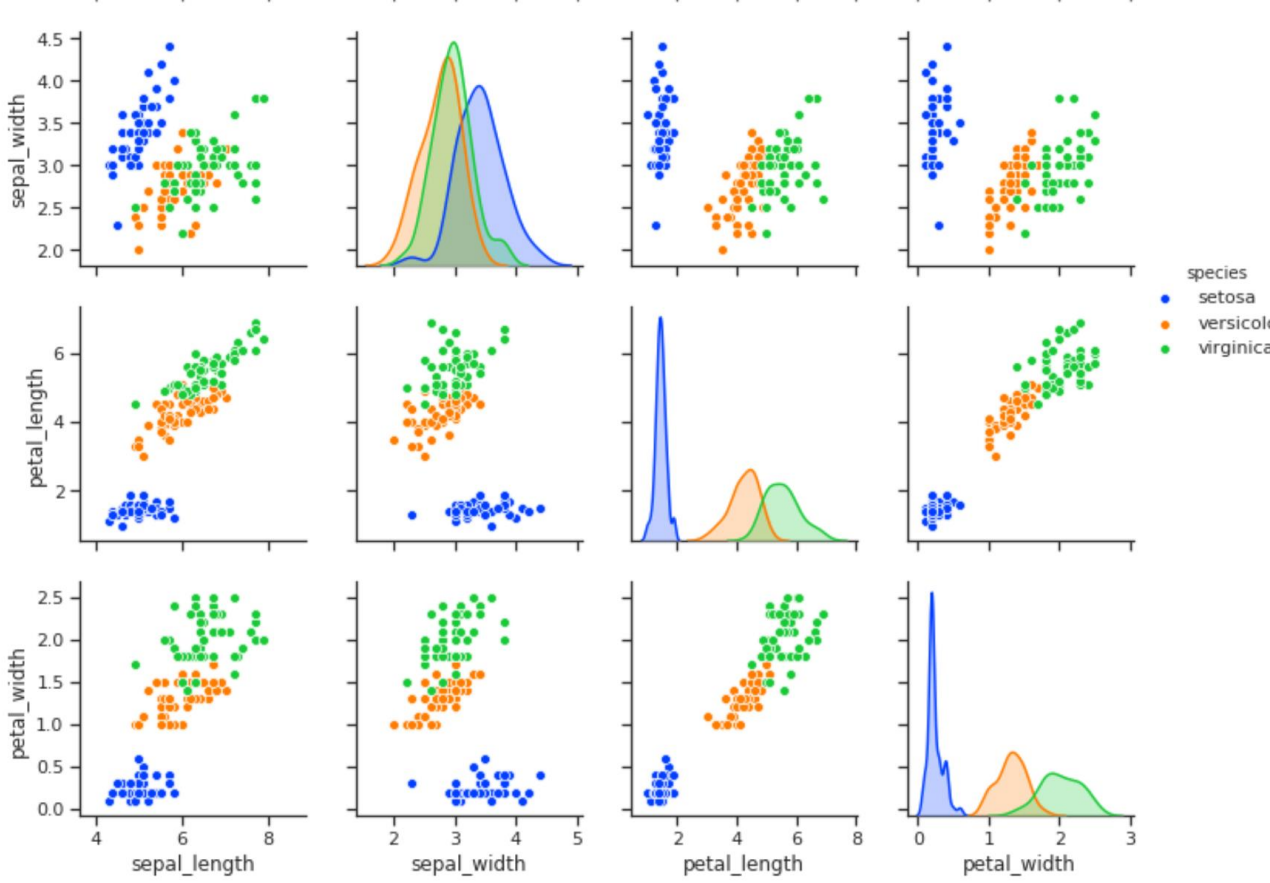


SQL Compute Context from remote client

- Requirement
  - Use rx* functions (revoscale or microsoftml)
- Key Benefits
  - Push compute to server from client
  - Eliminate data movement from server to client
  - Use server resources for ML script execution

3) On SQL Server via sp_execute_external_script

```
sp_execute_external_script
    @language = N'language,
    @script = N'script'
    [ , @input_data_1 = N'input_data_1' ]
    [ , @input_data_1_name = N'input_data_1_name' ]
    [ , @output_data_1_name = N'output_data_1_name' ]
    [ , @parallel = 0 | 1 ]
    [ , @params = N'@parameter_name data_type [ OUT | OUTPUT ] [
,...n ]' ]
    [ , @parameter1 = 'value1' [ OUT | OUTPUT ] [ ,...n ] ]
```



Iris Demo

Predicting the species from the sepal and the petal

sp_execute_external_script Demo

Basics to test we have R running on the server

```
EXEC   sp_execute_external_script
       @language = N'R'
     , @script = N'print("hello world")'

EXEC   sp_execute_external_script
       @language = N'R'
     , @script = N'x <- c(1,2,3,4,5);
dfx <- data.frame(x);'
     , @output_data_1_name = N'dfx'
WITH RESULT SETS ((ID INT));

DROP TABLE IF EXISTS #Tmp
CREATE TABLE #Tmp (ID INT)
INSERT INTO #Tmp VALUES (1), (2), (3), (4), (5)

EXEC   sp_execute_external_script
       @language = N'R'
     , @script = N'x <- intarray;
dfx <- data.frame(x);'
    , @output_data_1_name = N'dfx'
, @input_data_1 = N'SELECT * FROM #Tmp'
, @input_data_1_name = N'intarray'
WITH RESULT SETS UNDEFINED;
```

sp_execute_external_script Demo

Create iris into a table

```
CREATE TABLE iris_data ("Sepal.Length" float not null,
"Sepal.Width" float not null,
"Petal.Length" float not null,
"Petal.Width" float not null,
"Species" varchar(100))

INSERT INTO iris_data
EXEC   sp_execute_external_script
       @language = N'R'
     , @script = N'iris_data <- iris;'
     , @input_data_1 = N''
     , @output_data_1_name = N'iris_data'

ALTER TABLE iris_data ADD ID INT PRIMARY KEY NOT NULL IDENTITY (1,1)

SELECT TOP 6 * FROM iris_data
```

sp_execute_external_script - Create Model

```
BEGIN TRY
CREATE TABLE [dbo].[iris_models] (
[model_name] [varchar](30) NOT NULL,
[model] [varbinary](max) NOT NULL)
END TRY
BEGIN CATCH
print ERROR_MESSAGE()
END CATCH
```

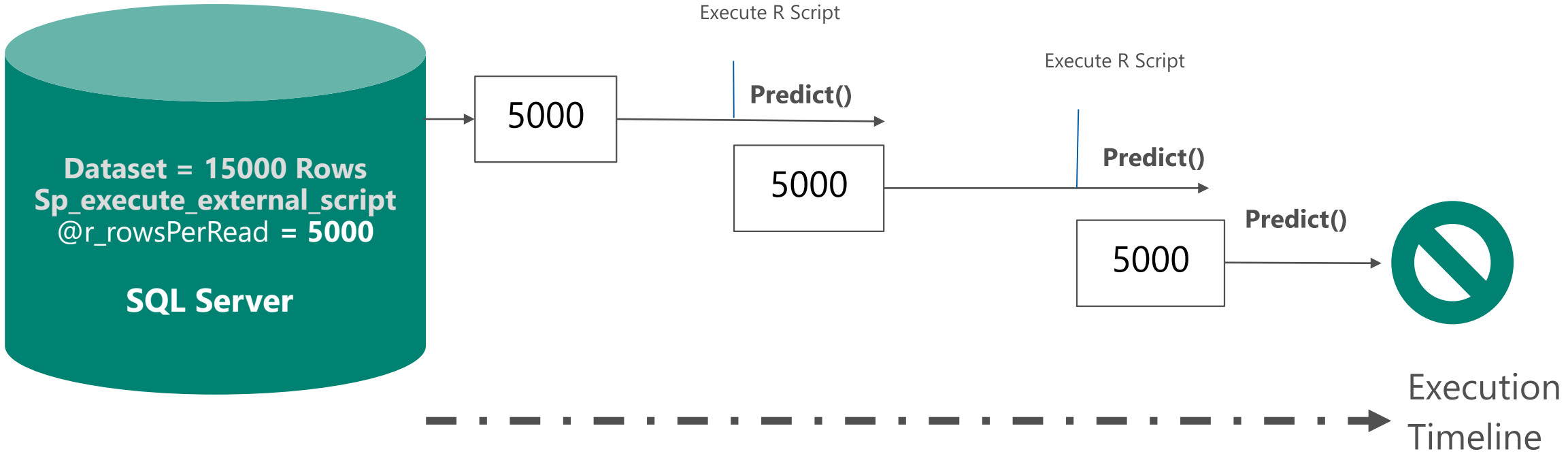
```
CREATE PROCEDURE [dbo].[generate_iris_rxBTrees_model]
AS
BEGIN
DELETE FROM [dbo].[iris_models] WHERE model_name = 'iris_rxBTrees_model'

DECLARE @model varbinary(max);
EXECUTE sp_execute_external_script
  @language = N'R'
  , @script = N'
iris.sub <- c(sample(1:50, 25), sample(51:100, 25), sample(101:150, 25))
iris.dtree <- rxDTree(Species ~ Sepal.Length + Sepal.Width + Petal.Length +
Petal.Width, data = iris[iris.sub, ])
model <- rxSerializeModel(iris.dtree, realtimeScoringOnly = FALSE)
#realtimeScoringOnly - Setting this flag could reduce the model size but
rxUnserializeModel can no longer retrieve the RevoScaleR model
rxUnserializeModel(model)
cat(paste0("R Process ID = ", Sys.getpid()))
cat("\n")
    '
  , @params = N'@model varbinary(max) OUTPUT'
  , @model = @model OUTPUT
  INSERT [dbo].[iris_models]
  VALUES('iris_rxBTrees_model', @model) ;
END;

GO

EXEC [generate_iris_rxBTrees_model]

GO

SELECT * FROM [iris_models]
```

sp_execute_external_script - Predict

```
CREATE PROCEDURE [dbo].[predict_species] (@model
varchar(100))
AS
BEGIN
declare @nb_model varbinary(max) = (
SELECT model
FROM iris_models
WHERE model_name = @model);
-- Predict species based on the specified model:
EXEC sp_execute_external_script
@language = N'R'
  , @script = N'
require("RevoScaleR");
irismodel<-rxUnserializeModel(nb_model)
species <- rxPredict(irismodel, iris_data[,2:5]);
OutputDataSet <- cbind(iris_data[1], species,
iris_data[6]);

cat(paste0("R Process ID = ", Sys.getpid()))
cat("\n")

OutputDataSet <- OutputDataSet;
'
, @input_data_1 = N'SELECT id,
"Sepal.Length",
"Sepal.Width",
"Petal.Length",
"Petal.Width",
"Species"
FROM
iris_data'
, @input_data_1_name = N'iris_data'
, @params = N'@nb_model varbinary(max)'
, @nb_model = @nb_model
with result sets ((
ID INT,
Setosa Float,
Versicolor Float,
Virginica Float,
ActualSpecies Varchar(150)));
END;

GO

EXEC [predict_species] 'iris_rxBTrees_model'
```

The memory bound problem

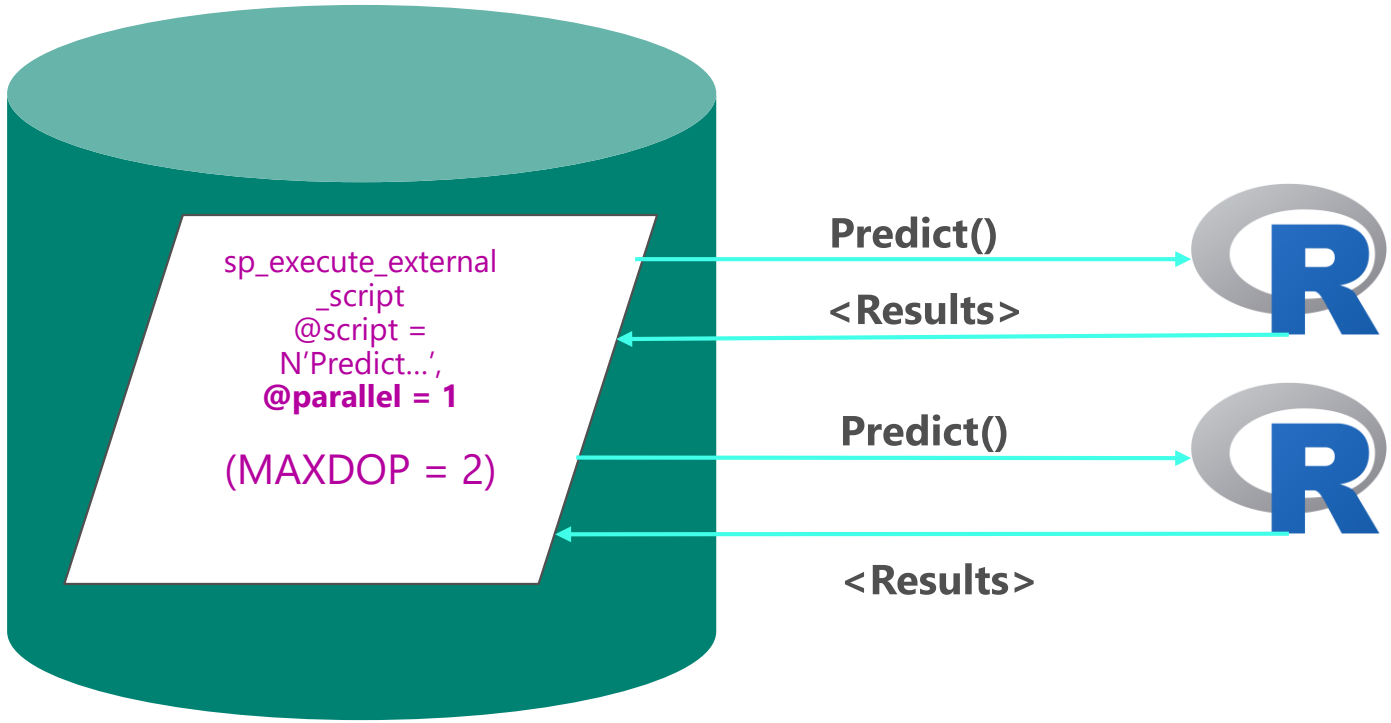
- Open source R can be both thread bound and memory bound
- Solution
  - SQL streaming allows these R packages to predict on much smaller sets of data so accommodates memory limitations



Streaming Demo

```
CREATE procedure [dbo].[predict_species_stream] (@model varchar(100))
as
begin
declare @nb_model varbinary(max) = (select model from iris_models where model_name = @model);
-- Predict species based on the specified model:
exec sp_execute_external_script
@language = N'R'
  , @script = N'
require("RevoScaleR");
irismodel<-rxUnserializeModel(nb_model)
species<-rxPredict(irismodel, iris_data[,2:5]);
OutputDataSet <- cbind(iris_data[1], species, iris_data[6]);
colnames(OutputDataSet) <- c("id", "Species.Actual", "Species.Expected");

cat(paste0("R Process ID = ", Sys.getpid()))
cat("\n")

OutputDataSet <- OutputDataSet;
'
, @input_data_1 = N'
select id, "Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width", "Species"
  from iris_data'
, @input_data_1_name = N'iris_data'
, @params = N'@nb_model varbinary(max), @r_rowsPerRead int = 4'
, @nb_model = @nb_model
with result sets UNDEFINED;
end;

GO

EXEC [predict_species_stream] 'iris_rxBTrees_model'
```

Parallel Processing – Trivial Parallelism



Parallel Processing – Trivial Parallelism

- Requirements:
  - Parallel query plan for the input SELECT statement
  - No dependency between rows (ex: scoring) – Trivial Parallelism
- Key Benefits:
  - Scales to large data sets
  - Leverage multiple CPUs
  - Integrates with SQL Server parallel query execution
  - Can use for standard R packages

Parallel Processing Demo – Prep data

```
--Create Fake Data
SET NOCOUNT ON
SELECT TOP 0 * INTO iris_data_big  FROM iris_data

GO
INSERT INTO iris_data_big ([Sepal.Length], [Sepal.Width],
[Petal.Length], [Petal.Width])
VALUES (
RIGHT(ABS(CHECKSUM(NEWID())), 2)/10.0,
RIGHT(ABS(CHECKSUM(NEWID())), 2)/10.0,
RIGHT(ABS(CHECKSUM(NEWID())), 2)/10.0,
RIGHT(ABS(CHECKSUM(NEWID())), 2)/10.0
)
GO 1000

INSERT INTO iris_data_big ([Sepal.Length], [Sepal.Width],
[Petal.Length], [Petal.Width])
SELECT
RIGHT(ABS(CHECKSUM(NEWID())), 2)/10.0,
RIGHT(ABS(CHECKSUM(NEWID())), 2)/10.0,
RIGHT(ABS(CHECKSUM(NEWID())), 2)/10.0,
RIGHT(ABS(CHECKSUM(NEWID())), 2)/10.0
FROM iris_data_big
GO 10

CREATE INDEX [IXiris_data_big] ON iris_data_big(ID)
```

Parallel Processing Demo

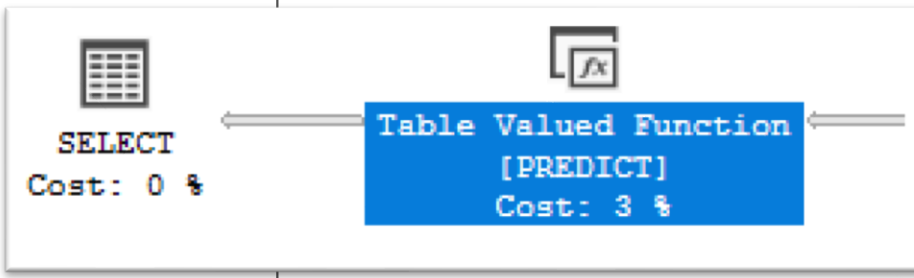
```
CREATE PROCEDURE [dbo].[predict_species_parallel] (@model varchar(100))
as
begin
declare @nb_model varbinary(max) = (select model from iris_models where model_name = @model);
-- Predict species based on the specified model:
exec sp_execute_external_script
@language = N'R'
  , @script = N'
require("RevoScaleR");
irismodel<-rxUnserializeModel(nb_model)
species<-rxPredict(irismodel, iris_data[,2:5]);
OutputDataSet <- cbind(iris_data[1], species, iris_data[6]);
colnames(OutputDataSet) <- c("id", "Species.Actual", "Species.Expected");

cat(paste0("R Process ID = ", Sys.getpid()))
cat("\n")
cat("-----------------")
cat("\n")

OutputDataSet <- head(OutputDataSet);
'
, @parallel = 1
, @input_data_1 = N'select id, "Sepal.Length", "Sepal.Width", "Petal.Length", "Petal.Width", "Species"
  from iris_data_big WHERE LEFT(ID,1) BETWEEN 1 AND 5 /*Add this bit to make parallel*/
   OPTION(MAXDOP 3)'
, @input_data_1_name = N'iris_data'
, @params = N'@nb_model varbinary(max), @r_rowsPerRead int = 100000'
, @nb_model = @nb_model
with result sets (("ID" INT, "Setosa" INT, "versicolor" INT, "virginica" INT, "Species" varchar(150)));
END;

GO

EXEC [predict_species_parallel] 'iris_rxBTrees_model'
```

Native Scoring using PREDICT function

- Requirements:
- Works with models built using RevoScaleR or revoscalepy:
  - rxLinMod, rxLogit, rxBTrees, rxDTree, rxDForest
- Serialize using rxSerializeModel (R) or rx_serialize_model (Python)
- Key Benefits:
  - No dependency on R or Python runtime = Faster
  - Up to 1000 times faster than traditional R/Python scoring!
  - Ideal for highly concurrent scoring of few rows
  - Can be used in INSERT/UPDATE/MERGE statement directly
  - Enabled on SQL Server in Windows, Linux and Azure SQL DB

SQL PREDICT Demo

```
DECLARE @model varbinary(max) = (SELECT TOP 1 model from
[dbo].[iris_models] WHERE model_name = 'iris_rxBTrees_model');
SELECT @model

DROP TABLE IF EXISTS #TMP
SELECT [Sepal.Length] * .9 AS [Sepal.Length] ,
[Sepal.Width] * .9 AS [Sepal.Width],
[Petal.Length] * .9 AS [Petal.Length] ,
[Petal.Width]* .9 AS [Petal.Width],
Species,
ID
INTO #tmp
FROM iris_data


SELECT *
FROM PREDICT(MODEL = @model,
  DATA = #tmp AS d) WITH (setosa_Pred float, versicolor_Pred float,
virginica_Pred float) AS p;
```



SQL PREDICT – GA on Azure SQL Database

```
DECLARE @model varbinary(max) =
```

0x626C6F62298B1834AEF8DA6B26B89D32475A371DEDE9328FD60DB36F45A5E2C702CB1095010000030140000E472E9C26354AFC8A4474CFD49695C6E8130000000000000000000000090000070000007278445472656500050100020000000010B000003890100000100000000000803F010000000000000010000000000000001000000F4F3F2F1
000000400600000000000400000000000010000000000000001000000000000000300000000000800000000000000803F00000000CDCC8C3F030000000000000000000000001000000000803F010000000500000747265653101000000000000001000000F4F3F2F1000000400000000000004000000000000100000000000000100000000000000
B00000000034000000000000000803F000000000000803F0000080000000800000000500000001000000000803F010000000000000010000000000000001000000F4F3F2F1000000400F000000000400000000000000000000000000001000000000000000700000000034001000000000000803F000000000000803FFF000000FF000000000000
0000100000000000803F010000000000000001000000000000001000000F4F3F2F10000004005000000000040000000000000000000000000001000000000000000800000000140000000000000000803F000000000B010003B5010000010000000000803F010000000000000010000000000000001000000F4F3F2F10000004006000000000040000
000000001000000000000000100000000000000030000000000080000000000000803F00000000CDCC8C3F030000000000000000000000001000000000803F010000000500000747265653101000000000000001000000F4F3F2F100000040040000000000400000000000050000000000000001000000000000012000000006400000000000000
00803F000000000000803FA20700000000F07FA20700000000F07F000000000000000000000F03F0000000000000040000000000000000840000000000000018400000000000001C4001000000000000803F010000000000000010000000000000001000000F4F3F2F1000000400F000000000400000000000000000000000000001000000000000000070000
0000034001000000000000803F000000000000803FFF000000FF000000000000000001000000000803F010000000000000010000000000000001000000F4F3F2F10000004005000000000040000000000000000000000000001000000000000000800000000140000000000000000803F000000000B020003D5010000010000000000803F0100000000
0000000100000000000000001000000F4F3F2F100000040060000000000400000000000010000000000000001000000000000000300000000000800000000000000803F00000000CDCC8C3F030000000000000000000000001000000000803F010000000500000747265653101000000000000001000000F4F3F2F1000000400000000000004000000
000001400000000000000001000000000000000B000000000340000000000000000803F000000000000803F0000080000000800000000004B000000190000003200000001A0000000180000000000000000000000000000000000000000000000000000000000000000000000000000100000000000000020000000000000000000000010000
000000803F010000000000000010000000000000001000000F4F3F2F1000000400F000000000400000000000000000000000000001000000000000000700000000034001000000000000803F000000000000803FFF000000FF000000000000000001000000000803F010000000000000010000000000000001000000F4F3F2F10000004005000000000
040000000000000000000000000001000000000000000800000000140000000000000000803F000000000B030003B5010000010000000000803F010000000000000010000000000000001000000F4F3F2F100000040060000000000400000000000010000000000000001000000000000000300000000000800000000000000803F00000000CDCC8
C3F030000000000000000000000001000000000803F010000000500000747265653101000000000000001000000F4F3F2F100000040040000000000400000000000050000000000000001000000000000012000000006400000000000000803F000000000000803FA20700000000F07FA20700000000F07F000000000000000000000F03F0000000
00000F03F000000000000040000000000000040000000000000008400100000000803F010000000000000010000000000000001000000F4F3F2F1000000400F000000000400000000000000000000000000001000000000000000700000000034001000000000000803F000000000000803FFF000000FF000000000000000001000000000803F010
000000000000001000000000000000100000F4F3F2F10000004005000000000040000000000000000000000000001000000000000000800000000140000000000000000803F000000000B040003CD020000010000000000803F010000000000000010000000000000001000000F4F3F2F1000000400600000000004000000000001000000000000
0001000000000000000300000000000800000000000000803F00000000CDCC8C3F030000000000000000000000001000000000803F010000000500000747265653101000000000000001000000F4F3F2F1000000400400000000004000000000002800000000000000010000000000000120000000064000000000000000803F000000000000803
FA20700000000F07FA20700000000F07F000000000000000000000F03F000000000000F03F0000000000000040000000000000040000000000000084000000000000039400000000000003940000000000000000000000000000000000000000000000000000000000000000000000000000000000000003940000000000000003940000000000
000000000000000039400000000000000000000000000000000039400000000000000F03F000000000003840555555555555D53F000000000000F03F000000000000000000000000000000000000000000000000555555555555D53F000000000000000000000000000E03F4FECC44EECC4EE3F00000000000000000555555555555D53F0000000000000
000000000000000E03F143BB1133BB1A33F000000000000F03F000000000000F03F555555555555D53F555555555555E53F3096FC62C92FD63F7B14AE47E17AD43F010000000000803F010000000000000010000000000000001000000F4F3F2F1000000400F000000000400000000000000000000000000001000000000000000700000000034001
000000000000803F000000000000803FFF000000FF000000000000000001000000000803F010000000000000010000000000000001000000F4F3F2F10000004005000000000040000000000000000000000000001000000000000000800000000140000000000000000803F000000000B0500038D010000010000000000803F010000000000000100
000000000001000000F4F3F2F100000040060000000000400000000000010000000000000001000000000000000300000000000800000000000000803F00000000CDCC8C3F030000000000000000000000001000000000803F010000000500000747265653101000000000000001000000F4F3F2F100000040000000000000400000000000002000
000000000001000000000000000B000000000340000000000000000803F000000000000803F0000080000000800000000002000000050000000100000000803F010000000000000010000000000000001000000F4F3F2F1000000400F000000000400000000000000000000000000001000000000000000700000000034001000000000000803F0
000000000000803FFF000000FF000000000000000001000000000803F010000000000000010000000000000001000000F4F3F2F10000004005000000000040000000000000000000000000001000000000000000800000000140000000000000000803F000000000B0600039C010000010000000000803F010000000000000010000000000000001000
000F4F3F2F100000040060000000000400000000000010000000000000001000000000000000300000000000800000000000000803F00000000CDCC8C3F030000000000000000000000001000000000803F010000000500000747265653101000000000000001000000F4F3F2F100000040050000000000400000000000020000000000000010000
00000000000800000000140000000000000000803F0000000010000000C000000506574616C2E4C656E677468010000000B0000000506574616C2E5769647468010000000000803F010000000000000010000000000000001000000F4F3F2F1000000400F000000000400000000000000000000000000001000000000000000700000000034001000
000000000803F000000000000803FFF000000FF000000000000000001000000000803F010000000000000010000000000000001000000F4F3F2F10000004005000000000040000000000000000000000000001000000000000000800000000140000000000000000803F000000000B070003DD010000010000000000803F010000000000000010000
000000000001000000F4F3F2F100000040060000000000400000000000010000000000000001000000000000000300000000000800000000000000803F00000000CDCC8C3F030000000000000000000000001000000000803F010000000500000747265653101000000000000001000000F4F3F2F1000000400400000000004000000000000A00000
000000001000000000000000120000000064000000000000000803F000000000000803FA20700000000F07FA20700000000F07F000000000000000000000C0524000000000000000494000000000000000F0BF000000000000F0BFF8FFFFFFFFFF38403CB1133BB11337409A999999999905409A9999999999F93F0000000000000000000000000000000010
000000000803F010000000000000010000000000000001000000F4F3F2F1000000400F000000000400000000000000000000000000001000000000000000700000000034001000000000000803F000000000000803FFF000000FF000000000000000001000000000803F010000000000000010000000000000001000000F4F3F2F10000004005000000
0000400000000000000000000000000001000000000000000800000000140000000000000000803F000000000B0800038D010000010000000000803F010000000000000010000000000000001000000F4F3F2F100000040060000000000400000000000010000000000000001000000000000000300000000000800000000000000803F00000000CD
CC8C3F030000000000000000000000001000000000803F010000000500000747265653101000000000000001000000F4F3F2F1000000400000000000004000000000002000000000000000100000000000000B000000000340000000000000000803F000000000000803F00000800000000800000000000000000000000001000000000803F010
000000000000001000000000000000100000F4F3F2F1000000400F000000000400000000000000000000000000001000000000000000700000000034001000000000000803F000000000000803FFF000000FF000000000000000001000000000803F010000000000000010000000000000001000000F4F3F2F10000004005000000000400000000000
0000000000000000100000000000000800000000140000000000000000803F000000000B09000385010000010000000000803F010000000000000010000000000000001000000F4F3F2F100000040060000000000400000000000010000000000000001000000000000000300000000000800000000000000803F00000000CDCC8C3F0300000000000
0000000000000100000000000803F010000000500000747265653101000000000000001000000F4F3F2F1000000400000000000004000000000000000000000000001000000000000000B00000000034000000000000000803F000000000000803F0000080000000800000000001000000000803F010000000000000010000000000000001000
000F4F3F2F1000000400F000000000400000000000000000000000000001000000000000000700000000034001000000000000803F000000000000803FFF000000FF000000000000000001000000000803F010000000000000010000000000000001000000F4F3F2F10000004005000000000040000000000000000000000000001000000000000008
0000000014000000000000000803F000000000B0A00030000000090B000000000100C00020000000500003000000050100030000000B02000900000000B03000900000000B0400080900000000000000000000000000000000000F03F0000000000000F03F000000000000000000000000000000000F03F000000000000F03F00000
0000000F03F000000000000000001090E000500000636C617373000B0F000903000000060000007365746F7361000A0000007665727369636F6C6F72000900000076697267696E696361000B10000901000000070000053706563696573000B110009040000000C000000536570616C2E4C656E677468000B000000536570616C2E5769647468000
C000000506574616C2E4C656E677468000B000000506574616C2E5769647468000B12000905000000070000053706563696573000C000000536570616C2E4C656E677468000B000000536570616C2E5769647468000C000000506574616C2E4C656E677468000B000000506574616C2E5769647468000B130009000000000B140009000000000615
000000000000000000005160001000000051700010000000518000100000000B190008000000000B1A00080100000000000000000000000B1B00090100000007000000537065636965573000100

```
DROP TABLE IF EXISTS #TMP
SELECT * INTO #TMP FROM (
SELECT 5.1 [Sepal.Length], 3.4 [Sepal.Width], 1.5 [Petal.Length], 0.3 [Petal.Width]
UNION SELECT 4.3, 3.1, 1.2, 0.2
UNION SELECT 6.6, 1.4, 5.3, 2.2) AS A

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

Supported Models SQL PREDICT

- revoscalepy models
  - rx_lin_mod
  - rx_logit
  - rx_btrees
  - rx_dtree
  - rx_dforest
- RevoScaleR models
  - rxLinMod
  - rxLogit
  - rxBTrees
  - rxDtree
  - rxDForest

sp_rxPredict

- CLR based real time scoring (SQL Server only)
- Additional installation step
- Runs with 2016 or greater

```
USE [CLRPredict]
DECLARE @irismodel varbinary(max)
SELECT @irismodel = model FROM
[dbo].[iris_models] WHERE model_name =
'iris_rxBTrees_model'

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

Supported functions for SQL sp_rxPredict

- RevoScaleR models
  - rxLinMod
  - rxLogit
  - rxBTrees
  - rxDtree
  - rxdForest
- MicrosoftML models
  - rxFastTrees
  - rxFastForest
  - rxLogisticRegression
  - rxOneClassSvm
  - rxNeuralNet
  - rxFastLinear
- Transformations supplied by MicrosoftML
  - featurizeText
  - concat
  - categorical
  - categoricalHash
  - selectFeatures

Dynamic Management Views

DMV	Description
sys.dm_exec_requests	New column: **external_script_request_id**
sys.dm_external_script_requests	Returns running external scripts, DOP & assigned user account
sys.dm_external_script_execution_stats	Number of executions for rx* functions in RevoScaleR package
sys.dm_os_performance_counters	New "**External Scripts**" performance counters

Controlling Performance

```
SELECT * FROM sys.resource_governor_external_resource_pools
GO

ALTER EXTERNAL RESOURCE POOL [default]
WITH (
    MAX_CPU_PERCENT = 90
    , AFFINITY CPU = AUTO
    , MAX_MEMORY_PERCENT = 25
);
GO
ALTER RESOURCE GOVERNOR RECONFIGURE;
GO

SELECT * FROM sys.resource_governor_external_resource_pools
```

Demo

View running scripts

```
SELECT r.session_id, r.blocking_session_id, r.status,
DB_NAME(s.database_id) AS database_name
    , s.login_name, r.wait_time, r.wait_type, r.last_wait_type,
r.total_elapsed_time, r.cpu_time
    , r.reads, r.logical_reads, r.writes, er.language,
er.degree_of_parallelism, er.external_user_name
FROM sys.dm_exec_requests AS r
INNER JOIN sys.dm_external_script_requests AS er
ON r.external_script_request_id = er.external_script_request_id
INNER JOIN sys.dm_exec_sessions AS s
ON s.session_id = r.session_id;
```

View Packages Installed

```
EXEC sp_execute_external_script @language = N'R'
, @script = N'
OutputDataSet <- data.frame(installed.packages()[,c("Package", "Version",
"Depends", "License", "LibPath")]);'
WITH result sets((Package NVARCHAR(255), Version NVARCHAR(100), Depends
NVARCHAR(4000)
    , License NVARCHAR(1000), LibPath NVARCHAR(2000)));

EXEC sp_execute_external_script @language = N'Python'
, @script = N'
import pip
OutputDataSet = pandas.DataFrame([(i.key, i.version, i.location) for i in
pip.get_installed_distributions()])'
WITH result sets((Package NVARCHAR(128), Version NVARCHAR(128), Location
NVARCHAR(1000)));
```

Documented Wait Stats

- EXTERNAL_SCRIPT_NETWORK_IO (2017 only)
- EXTERNAL_SCRIPT_PREPARE_SERVICE
- EXTERNAL_SCRIPT_SHUTDOWN

Packages Supported Features

RevoScaleR/revoscalepy	Microsoftml	Open Source
• Streaming • Trivial Parallelism (for scoring) • SQL Compute context • Parallelism for training & scoring • Native scoring with sp_rxPredict • Native scoring PREDICT	• Streaming • Trivial Parallelism (for scoring) • SQL Compute context • Parallelism for training and scoring • Native scoring with sp_rxPredict	• Streaming • Trivial Parallelism (for scoring)

Don't

- Run R / Python script without error handling etc
- Embed secrets in scripts
- Do data transformations that can be achieved in SQL
- Access network resources
- Process/transform files as part of the stored procedure call
- Embed the R/Python code directly in applications

Do

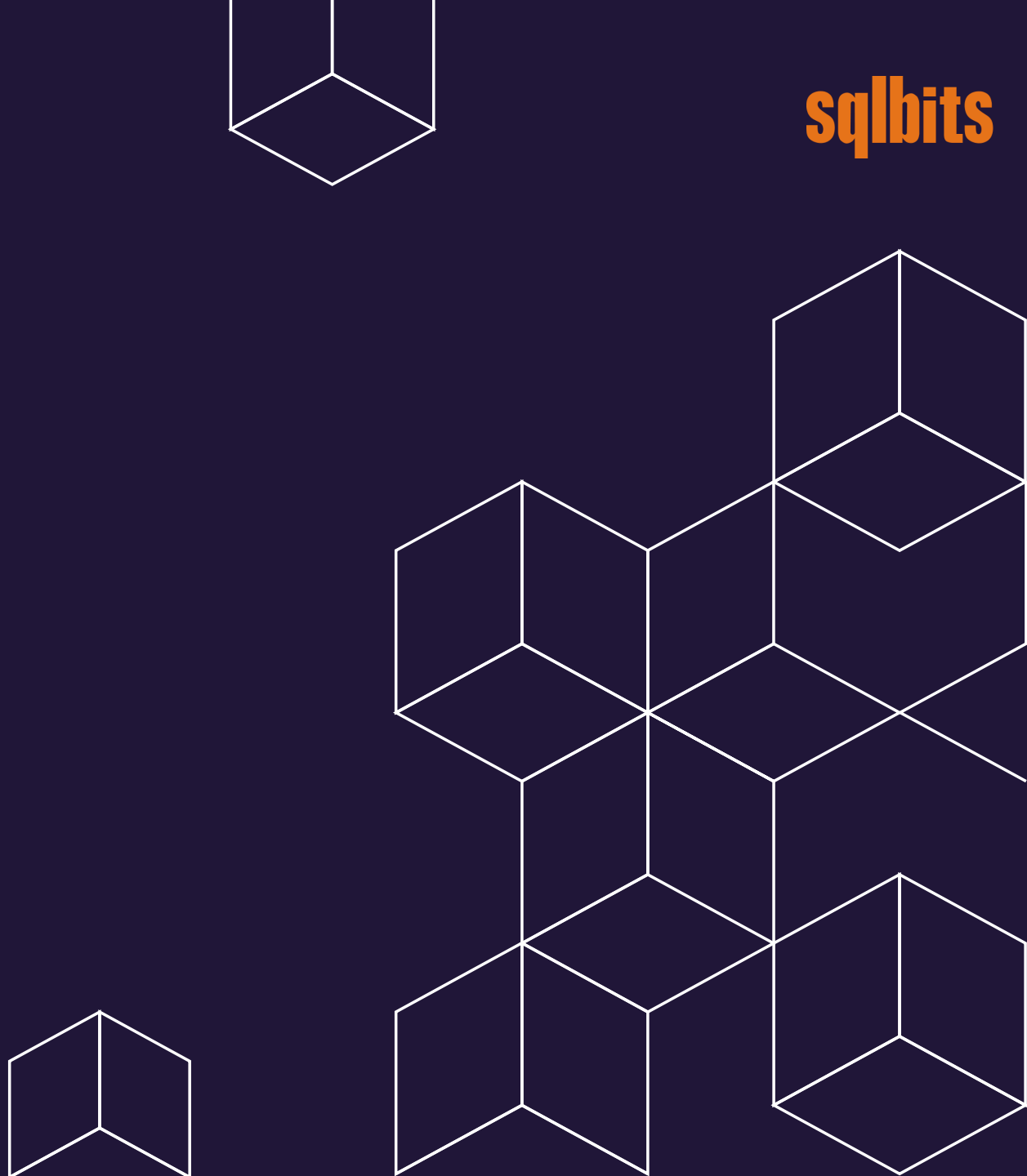
- Develop/Test from RTVS, PTVS, RStudio or other IDE
- SQL Compute Context from client
- Data processing & transformations in SQL Server
- Data integration using SQL Server features
- Model management in database

Leverage best of T-SQL & R / Python. Use the right tool!

Microsoft

sqlbits

Questions?



Microsoft

sqlbits

We'd love your feedback!

aka.ms/SQLBits19