PYTHON FOR DATA ANALYTICS

Association Rules Mining from Transactional Data

by Stan Smoltis

AGENDA

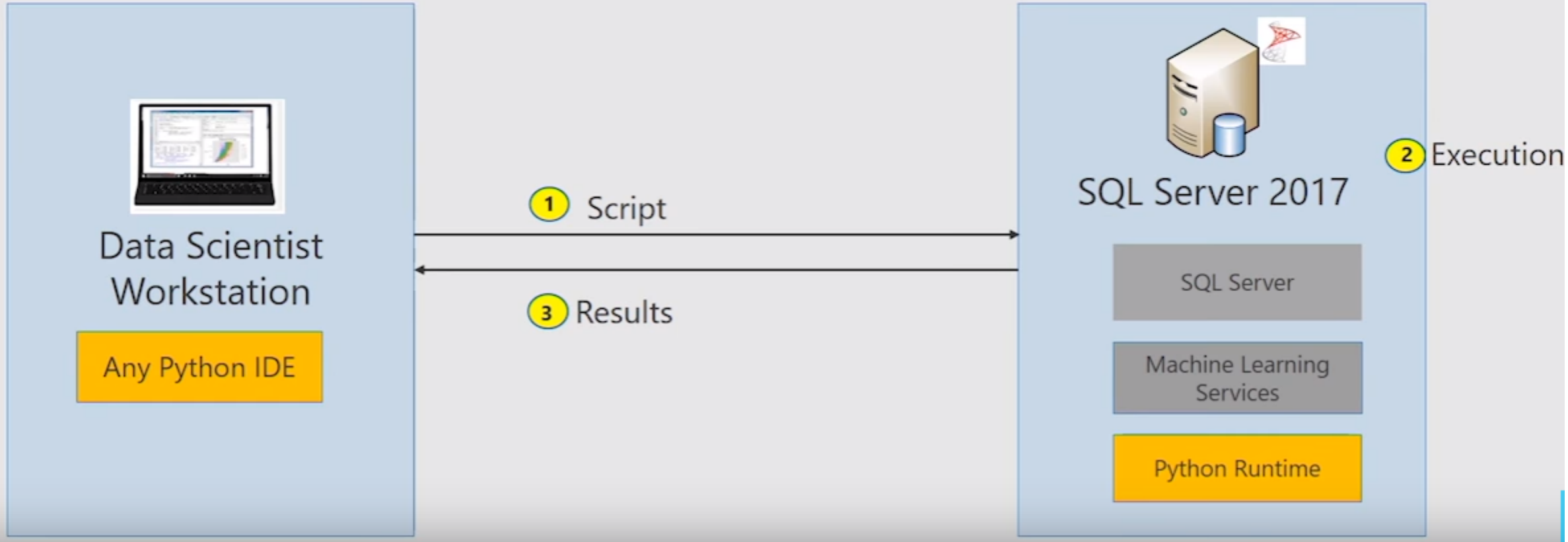
- Python comes to SQL Server
- Core Python Packages
  - numpy
  - Pandas
- Association Rules
- Demo

INTRODUCTION

- SQL Server 2017: Python integration for in-database analytics and computation
- Supports both Python and R
- Deploy Python ML and AI apps in PROD
- Apps don't have to be Python aware
- Helps with memory-bound algorithms
- Provides remote execution context (***revoscalepy***)
- Improved ML algorithms and pre-trained models (***microsoftml***)
- BYO Python FW into SQL Server (i.e. ***tensorflow***)

Data Exploration and Model Development

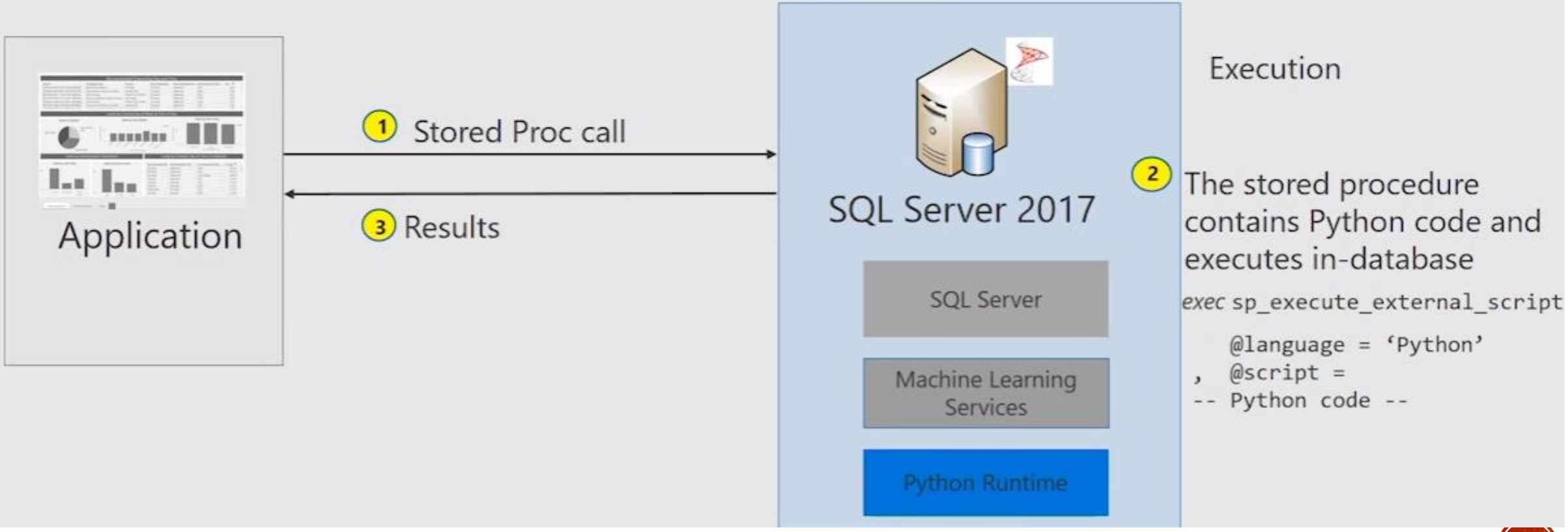
Working from IDE on a local workstation, execute Python code that runs in-database on remote SQL server, and get the results back.



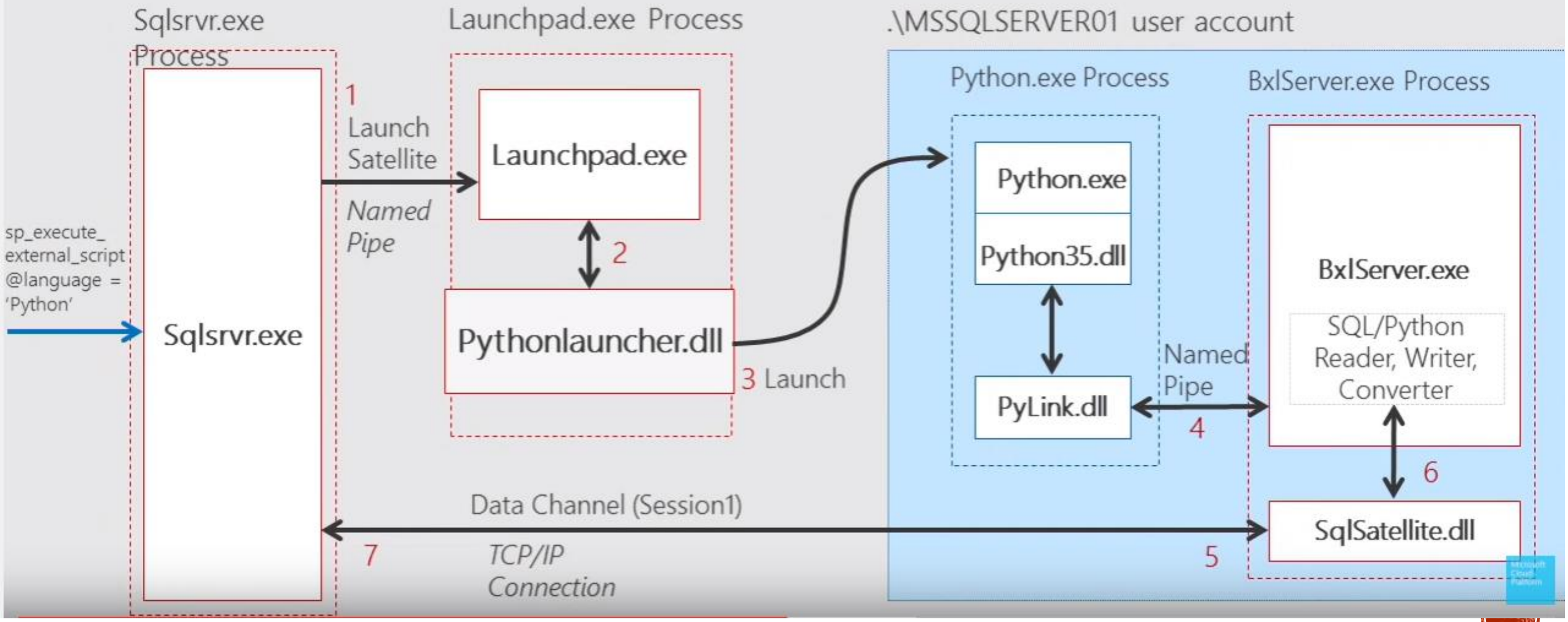
Model Operationalization

Python Code->T-SQL Stored Proc

Call T-SQL Stored Proc from your application to get the results from Python code execution (predictions, scores, plots etc.) in the applications



Python in SQL Server



https://www.youtube.com/watch?v=FcoY795jTcc

EXTERNAL PYTHON LIBRARIES

C:\Program Files\Microsoft SQL Server\MSSQL14.SQL2017\PYTHON_SERVICES\Scripts>
pip install tensorflow

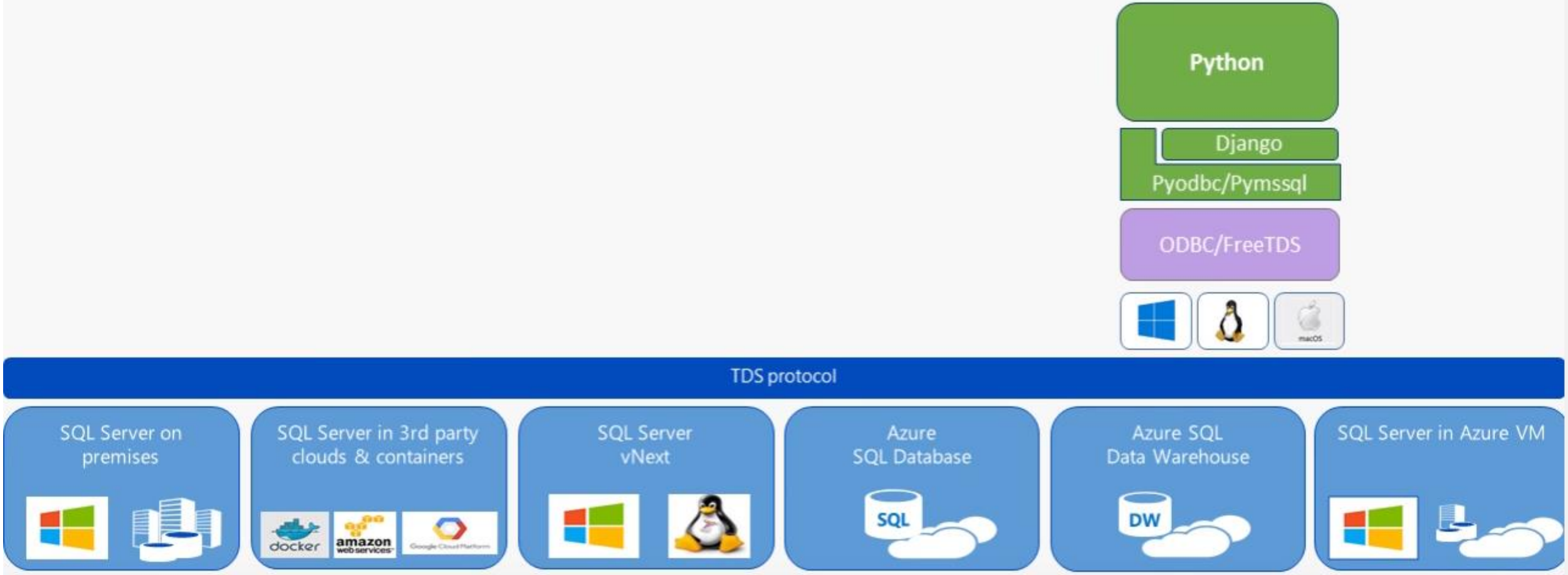
```
EXEC sp_execute_external_script
     @language = N'Python'
    ,@script = N'
import os
os.environ["TF_CPP_MIN_LOG_LEVEL"]="3"

import tensorflow as tf
hello = tf.constant("Hello, Tensorflow!")
sess = tf.Session()
print(sess.run(hello))
'
```

100 %

Messages

```
STDOUT message(s) from external script:
b'Hello, Tensorflow!'
```

Connectors for Python



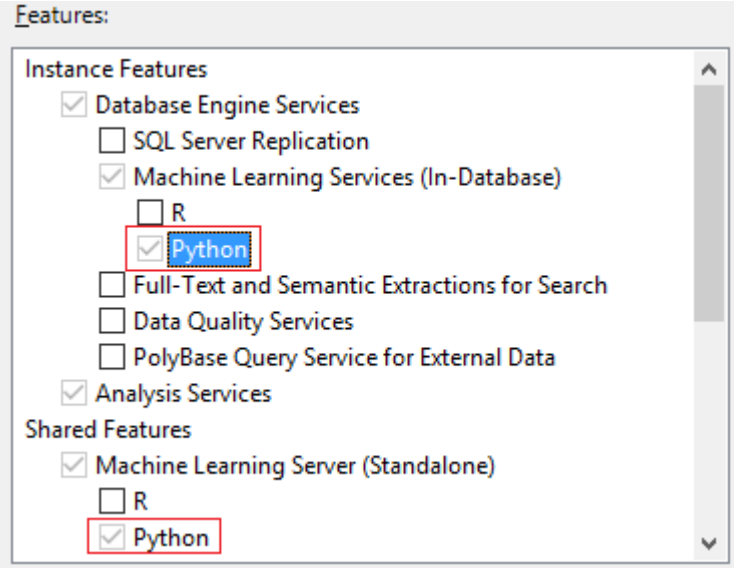
https://www.microsoft.com/en-us/sql-server/developer-get-started/python/windows/

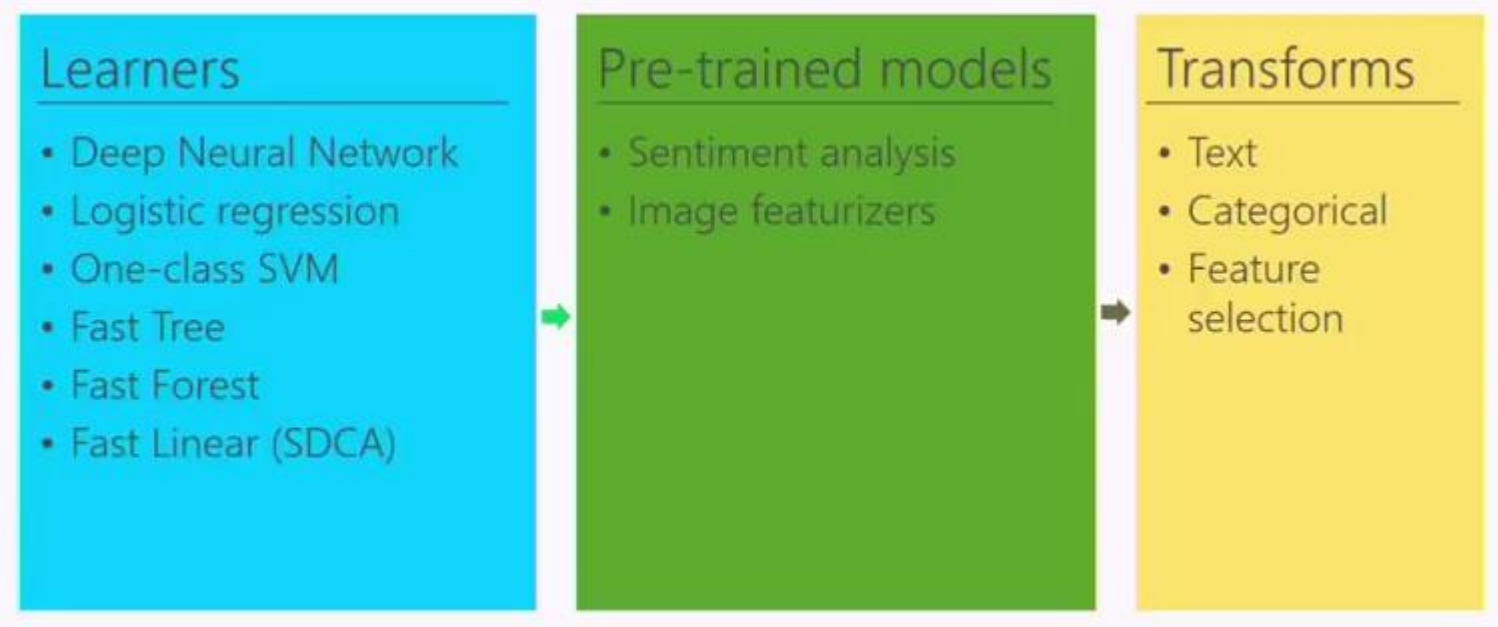
SUPPORT FOR PYTHON SERVICES

- Officially Supported by Microsoft Engineers
- Contact Microsoft Customer Support with questions via email, phone, forums, twitter

QUICK CONFIGURATION

- sp_configure ‘external scripts enabled’, 1
- Built-in ***revoscalepy*** to manage execution context and parallelize it
- Built-in ***microsoftml*** (proprietary collection of functions for ML)





MicrosoftML: Algorithm Cheat Sheet

This cheat sheet helps you choose the best MicrosoftML algorithm for your predictive analytics solution. Your decision is driven by both the nature of your data and the question you're trying to answer.

START

Predicting categories

Two

TWO-CLASS CLASSIFICATION (type = "binary")

- Accuracy, fast training, large memory footprint → **rxFastTrees()**
- Fast training, large memory footprint → **rxFastForest()**
- Linear model, fast training large data sets → **rxFastLinear()**
- Accuracy, long training time, large data sets → **rxNeuralNet()**
- Linear model, fast training large data sets → **rxLogisticRegression()**
- Highly unbalanced classes → **rxOneClassSvm()**

Three or more

MULTI-CLASS CLASSIFICATION (type="multiClass")

- Accuracy, long training time, large data sets → **rxNeuralNet()**
- Linear model, fast training, large data sets → **rxLogisticRegression()**

Predicting values

REGRESSION (type = "regression")

- Accuracy, fast training, large memory footprint → **rxFastTrees()**
- Fast training, large memory footprint → **rxFastForest()**
- Linear model, fast training, large data sets → **rxFastLinear()**
- Accuracy, long training time, large data sets → **rxNeuralNet()**

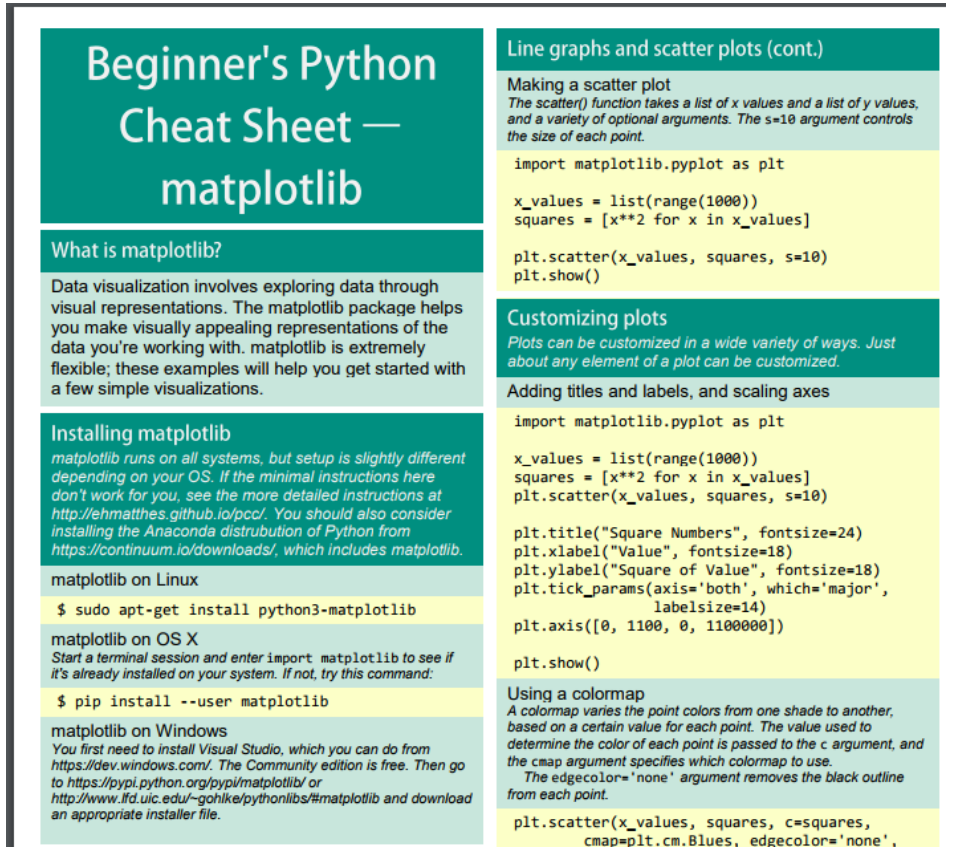
Finding unusual data points

ANOMALY DETECTION

- >100 features, aggressive boundary → **rxOneClassSvm()**

PYTHON LANGUAGE

- Everything is an object
- Collections and Lambdas
- Negative indexing
- List comprehensions
- Generators
- Shallow copies
- Tuples
- Classes and packages
- More general purpose than R



Beginner's Python Cheat Sheet — matplotlib

What is matplotlib?

Data visualization involves exploring data through visual representations. The matplotlib package helps you make visually appealing representations of the data you're working with. matplotlib is extremely flexible; these examples will help you get started with a few simple visualizations.

Installing matplotlib

matplotlib runs on all systems, but setup is slightly different depending on your OS. If the minimal instructions here don't work for you, see the more detailed instructions at http://ehmatthes.github.io/pcc/. You should also consider installing the Anaconda distrubution of Python from https://continuum.io/downloads/, which includes matplotlib.

matplotlib on Linux

```
$ sudo apt-get install python3-matplotlib
```

matplotlib on OS X

Start a terminal session and enter `import matplotlib` *to see if it's already installed on your system. If not, try this command:*

```
$ pip install --user matplotlib
```

matplotlib on Windows

You first need to install Visual Studio, which you can do from https://dev.windows.com/. The Community edition is free. Then go to https://pypi.python.org/pypi/matplotlib/ or http://www.lfd.uic.edu/~gohlke/pythonlibs/#matplotlib and download an appropriate installer file.

Line graphs and scatter plots (cont.)

Making a scatter plot

The scatter() function takes a list of x values and a list of y values, and a variety of optional arguments. The `s=10` *argument controls the size of each point.*

```
import matplotlib.pyplot as plt

x_values = list(range(1000))
squares = [x**2 for x in x_values]

plt.scatter(x_values, squares, s=10)
plt.show()
```

Customizing plots

Plots can be customized in a wide variety of ways. Just about any element of a plot can be customized.

Adding titles and labels, and scaling axes

```
import matplotlib.pyplot as plt

x_values = list(range(1000))
squares = [x**2 for x in x_values]
plt.scatter(x_values, squares, s=10)

plt.title("Square Numbers", fontsize=24)
plt.xlabel("Value", fontsize=18)
plt.ylabel("Square of Value", fontsize=18)
plt.tick_params(axis='both', which='major',
                labelsize=14)
plt.axis([0, 1100, 0, 1100000])

plt.show()
```

Using a colormap

A colormap varies the point colors from one shade to another, based on a certain value for each point. The value used to determine the color of each point is passed to the `c` *argument, and the* `cmap` *argument specifies which colormap to use.*

The `edgecolor='none'` *argument removes the black outline from each point.*

```
plt.scatter(x_values, squares, c=squares,
        cmap=plt.cm.Blues, edgecolor='none',
```

https://ehmatthes.github.io/pcc/cheatsheets/README.html

NUMPY FOR DATA STRUCTURES

- Fast operations on arrays
- Main object is **homogeneous** multidimensional array (**ndarray)**
- Array shaping is metadata operation => instantaneous
- Offers Matlab-ish capabilities within Python
- Input data format for many ML libraries
- Slicing

SLICING ARRAYS

```
# indices:     0   1   2   3   4
>>> a = array([10,11,12,13,14])
# [10,11,12,13,14]
>>> a[1:3]
array([11, 12])

# negative indices work also
>>> a[1:-2]
array([11, 12])
```

SIMPLE ARRAY MATH

```
>>> a = array([1,2,3,4])
>>> b = array([2,3,4,5])
>>> a + b
array([3, 5, 7, 9])
>>> a * b
array([ 2, 6, 12, 20])
>>> a ** b
array([  1,   8,   81,   1024])
```

MULTI-DIMENSIONAL ARRAYS

```
>>> a = array([[ 0, 1, 2, 3],
               [10,11,12,13]])
>>> a
array([[ 0, 1, 2, 3],
       [10,11,12,13]])
```

GET/SET ELEMENTS

```
>>> a[1,3]
13
```

column

row

```
>>> a[1,3] = -1
>>> a
array([[ 0, 1, 2, 3],
       [10,11,12,-1]])
```

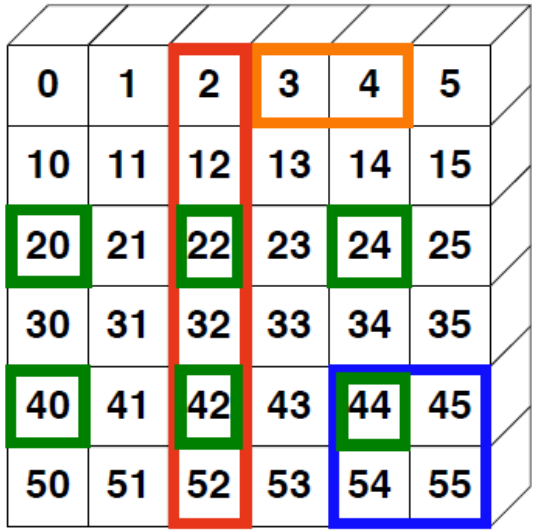
NDARRAY SLICING EXAMPLE

SLICING WORKS MUCH LIKE STANDARD PYTHON SLICING

```
>>> a[0,3:5]
array([3, 4])

>>> a[4:,4:]
array([[44, 45],
       [54, 55]])

>>> a[:,2]
array([2,12,22,32,42,52])
```

STRIDES ARE ALSO POSSIBLE

```
>>> a[2::2,::2]
array([[20, 22, 24],
       [40, 42, 44]])
```



PANDAS FOR DATA EXPLORATION

- Fast and efficient **Series/DataFrame** objects for data sets operations
- High performance Split-Apply-Combine operations on data sets
- Integrates with Visualization library
- Time series-functionality

PYFIM - FREQUENT ITEM SET MINING FOR PYTHON

- http://www.borgelt.net/pyfim.html
- PyFIM is an extension module that makes several frequent item set mining implementations available as functions

ASSOCIATION RULE MINING (1/2)

- rule-based machine learning method for discovering interesting relations between variables in large databases.

{A} implies {B}…

- Find frequent sets
- Measure **Support** => proportion where it appears of all transactions
- Measure **Confidence** => how likely B is purchased when A is purchased
- Measure **Lift** => if both A and B are very popular Confidence may be misleading, takes into account popularity of B

ASSOCIATION RULE MINING (2/2)

Transaction 1	🍎 🍺 🍚 🍗
Transaction 2	🍎 🍺 🍚
Transaction 3	🍎 🍺
Transaction 4	🍎 🍐
Transaction 5	🍼 🍺 🍚 🍗
Transaction 6	🍼 🍺 🍚
Transaction 7	🍼 🍺
Transaction 8	🍼 🍐

Support {🍎} = $\frac{4}{8}$

Confidence {🍎 → 🍺} = $\frac{\text{Support } \{🍎, 🍺\}}{\text{Support } \{🍎\}}$

Lift {🍎 → 🍺} = $\frac{\text{Support } \{🍎, 🍺\}}{\text{Support } \{🍎\} \times \text{Support } \{🍺\}}$

http://www.kdnuggets.com/2016/04/association-rules-apriori-algorithm-tutorial.html

TEST DATASET

	StoreCode	StoreName	ProductCode	ItemSeqNo	UnitPrice	Quantity	SaleeValue	Weighted	TimNo	TransactionID	SalesDate
1	248	HOLLETON	47294	5	2.99	1.057	3.16	1	2	447563	2016-01-01 07:45:57.000
2	22	MOUNT HOLLAND	29876	1	4.49	1	4.49	0	1	310938	2016-01-01 07:46:48.000
3	245	HOLLETON	152674	15	2.09	1	2.58	0	2	447564	2016-01-01 07:51:16.000
4	104	BARNEY POINT	59440	1	3	1	3	0	2	625006	2016-01-01 07:54:25.000
5	245	HOLLETON	22081	27	2	1	2	0	3	380213	2016-01-01 07:54:38.000
6	245	HOLLETON	29876	1	4.49	1	4.49	0	3	380215	2016-01-01 07:58:51.000
7	58	WALKERVALE	3022	5	4.49	1	4.49	0	5	241489	2016-01-01 12:24:10.000
8	243	CRESTMEAD	22081	4	2	1	2	0	4	125063	2016-01-01 12:25:00.000
9	22	MOUNT HOLLAND	179084	11	3.59	1	3.59	0	6	234811	2016-01-01 12:25:26.000
10	104	BARNEY POINT	141916	10	4.58	1	4.58	0	6	126140	2016-01-01 12:25:28.000
11	58	WALKERVALE	22081	5	2	1	2	0	8	352118	2016-01-01 12:25:50.000
12	64	RHINE FALLS	57473	2	1.99	1	1.99	0	3	294560	2016-01-01 10:32:04.000

	ProductCode	ProductDescription	DepartmentDescription	CategoryDescription	SubCategoryDescription
1	167299	32/PTUS YES SIM CARD TABLET EACH	NON FOOD	BOOKS/CARDS	TELECOMMUNICATIONS
2	167337	HENDERSON CARDS 499	NON FOOD	BOOKS/CARDS	BTB
3	167341	LINDOR DARK COCONUT GIFT BOX 147GM	GROCERY	CONFECTIONERY	CHOCOLATES BOXED
4	167343	HERB/GARCHIL CHIC DRUMETTES PER KG	MEAT	POULTRY & GAME	POULTRY VALUE ADDED
5	167349	MATILDAS FROZEN BLUEBERRIES 300GM	FROZEN	FRUIT	FRUIT FROZEN
6	167352	KIKKOMAN RICE INS NG CRKY 197GM	GROCERY	COOKING NEEDS	RICE
7	167355	AP PULL CREW SPRT SOCK CMOS/O 11-14	NON FOOD	CLOTHING	MENSWEAR
8	167358	HERB/GARCHIL BEEF PORTERHOUSE PER KG	MEAT	BEEF	BEEF VALUE ADDED
9	167364	GOLDEN CRUMB LAMB CUTLETS PER KG	MEAT	LAMB & MUTTON	LAMB VALUE ADDED
10	167369	HAYWARDS PICCALILLI SPREADABLE 200G	GROCERY	PICKLED PRODUCTS	CONDIMENTS PICKLES V
11	167373	HOOSCAR MACKERAL FILLETS WEQH 118GM	GROCERY	FISH	FISH SUNDRIES

DEMO

- Python examples
- Data transformation
- Association Rules

https://github.com/smoltis/sbiug