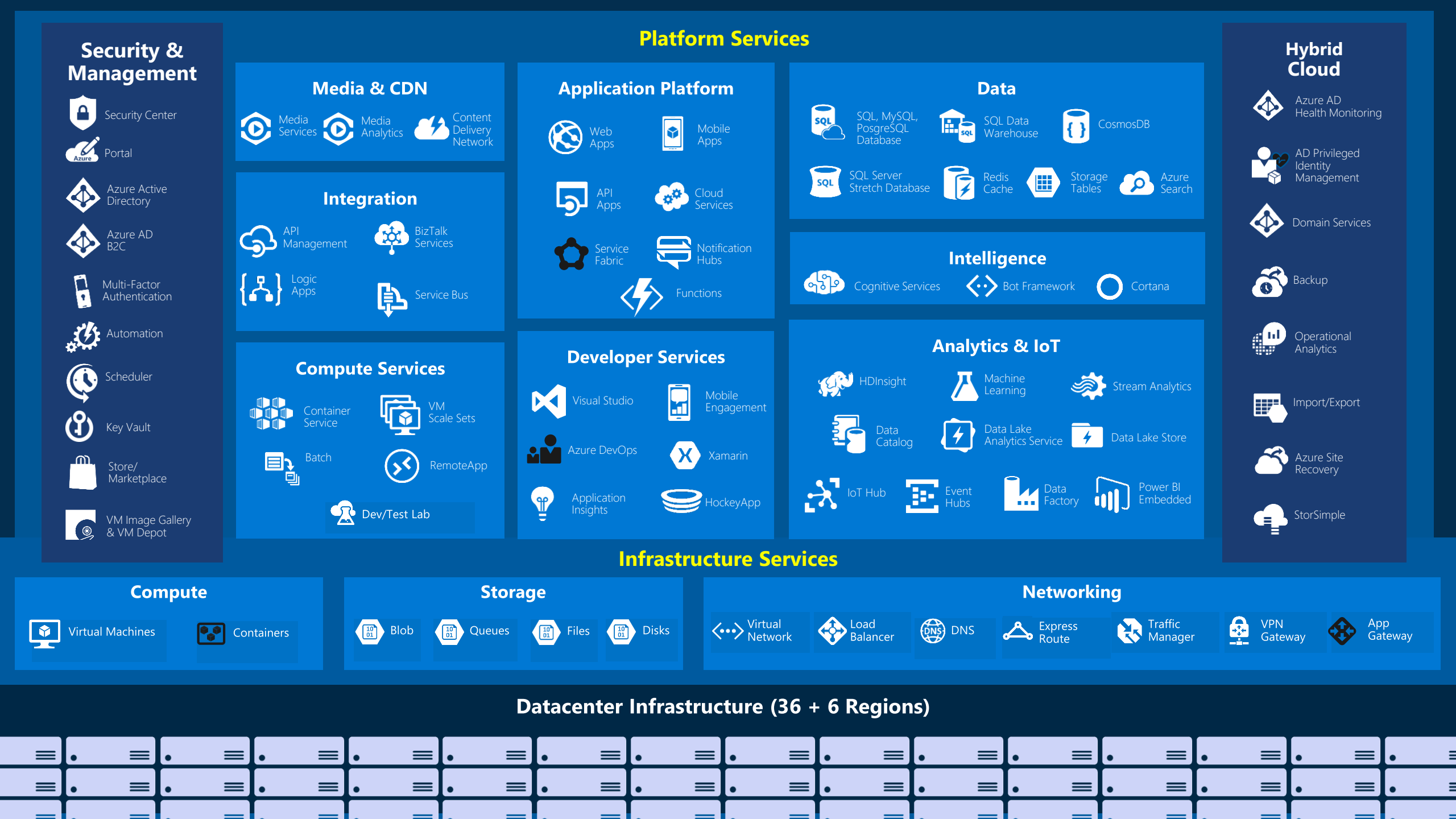
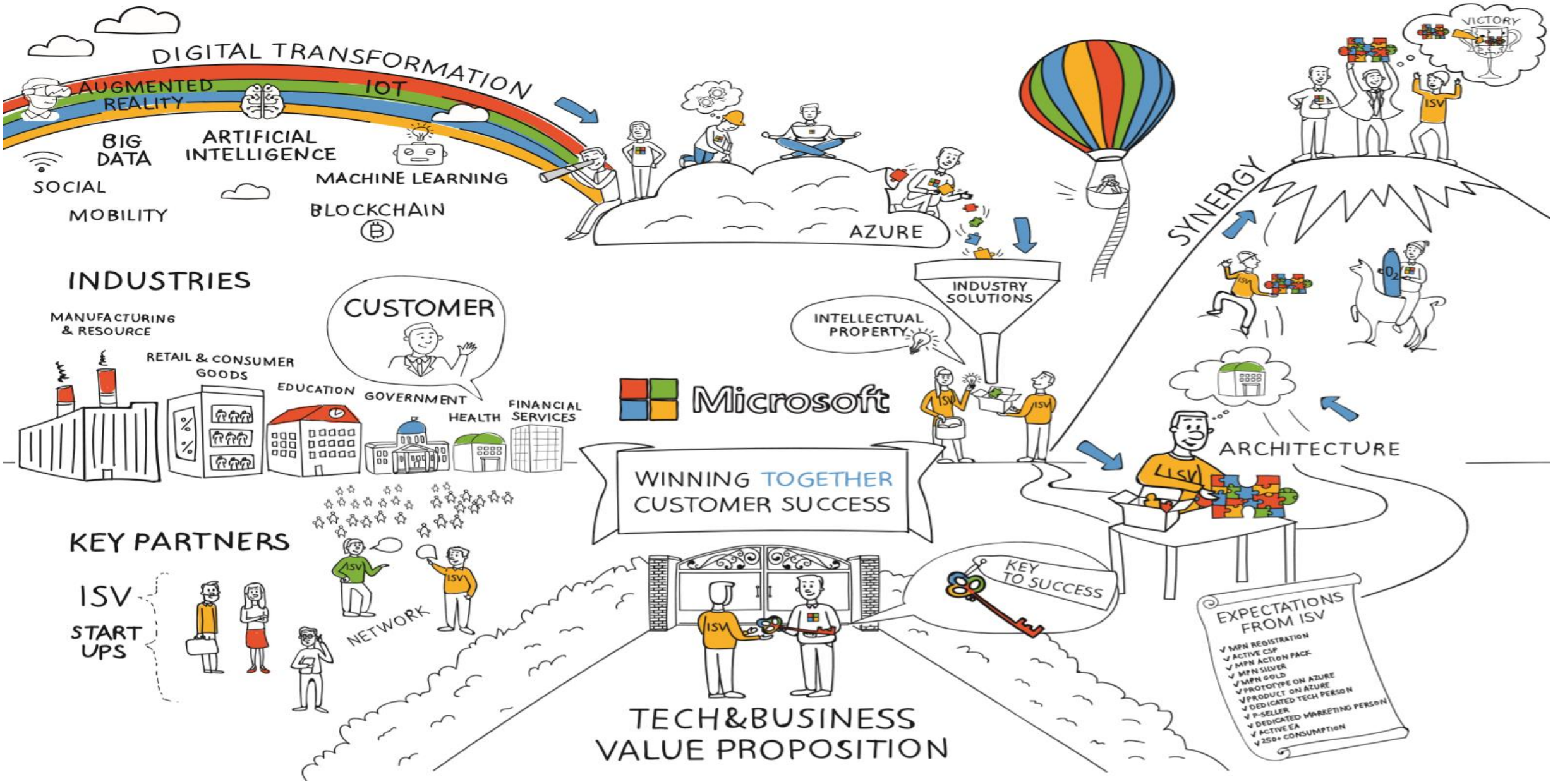
Microsoft

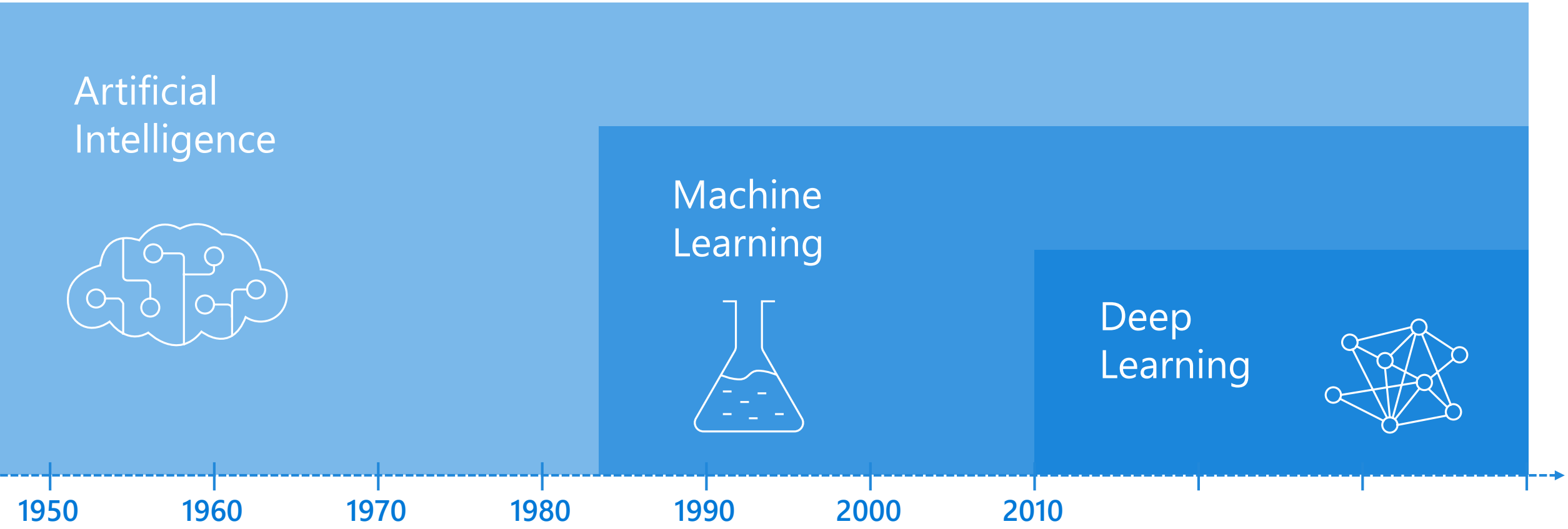
Różne sposoby tworzenia modeli Machine Learning w Azure – od graficznych po zarządzanie w modelami w języku Python. W tle klastry obliczeniowe (z GPU).

Tomasz Kopacz
CEE ISV Technical Recruit Lead
Microsoft


Security & Management
Security Center
Portal
Azure Active Directory
Azure AD B2C
Multi-Factor Authentication
Automation
Scheduler
Key Vault
Store/ Marketplace
VM Image Gallery & VM Depot
Platform Services
Media & CDN
Media Services
Media Analytics
Content Delivery Network
Integration
API Management
BizTalk Services
Logic Apps
Service Bus
Compute Services
Container Service
VM Scale Sets
Batch
RemoteApp
Dev/Test Lab
Application Platform
Web Apps
Mobile Apps
API Apps
Cloud Services
Service Fabric
Notification Hubs
Functions
Developer Services
Visual Studio
Mobile Engagement
Azure DevOps
Xamarin
Application Insights
HockeyApp
Data
SQL, MySQL, PosgreSQL Database
SQL Data Warehouse
CosmosDB
SQL Server Stretch Database
Redis Cache
Storage Tables
Azure Search
Intelligence
Cognitive Services
Bot Framework
Cortana
Analytics & IoT
HDInsight
Machine Learning
Stream Analytics
Data Catalog
Data Lake Analytics Service
Data Lake Store
IoT Hub
Event Hubs
Data Factory
Power BI Embedded
Hybrid Cloud
Azure AD Health Monitoring
AD Privileged Identity Management
Domain Services
Backup
Operational Analytics
Import/Export
Azure Site Recovery
StorSimple
Infrastructure Services
Compute
Virtual Machines
Containers
Storage
Blob
Queues
Files
Disks
Networking
Virtual Network
Load Balancer
DNS
Express Route
Traffic Manager
VPN Gateway
App Gateway
Datacenter Infrastructure (36 + 6 Regions)


DIGITAL TRANSFORMATION
AUGMENTED REALITY
IOT
BIG DATA
ARTIFICIAL INTELLIGENCE
MACHINE LEARNING
SOCIAL
MOBILITY
BLOCKCHAIN
AZURE
INDUSTRIES
MANUFACTURING & RESOURCE
RETAIL & CONSUMER GOODS
EDUCATION
GOVERNMENT
HEALTH
FINANCIAL SERVICES
CUSTOMER
INDUSTRY SOLUTIONS
INTELLECTUAL PROPERTY
Microsoft
WINNING TOGETHER
CUSTOMER SUCCESS
KEY PARTNERS
ISV
START UPS
NETWORK
KEY TO SUCCESS
ARCHITECTURE
SYNERGY
VICTORY
EXPECTATIONS FROM ISV
✓ MPN REGISTRATION
✓ ACTIVE CSP
✓ MPN ACTION PACK
✓ MPN SILVER
✓ MPN GOLD
✓ PROTOTYPE ON AZURE
✓ PRODUCT ON AZURE
✓ DEDICATED TECH PERSON
✓ P-SELLER
✓ DEDICATED MARKETING PERSON
✓ ACTIVE EA
✓ 250+ CONSUMPTION
TECH&BUSINESS
VALUE PROPOSITION

AI, Machine Learning and Deep Learning



Why Learn?

Learn it when you can't code it

(e.g. Recognizing Speech/image/gestures)

Learn it when you can't scale it

(e.g. Recommendations, Spam & Fraud detection)

Learn it when you have to adapt/personalize

(e.g. Predictive typing)

Learn it when you can't track it

(e.g. AI gaming, robot control)

Learn it when build formal, classic algorithm is too complicated


Loaded settings from C:\Users\akapoor\Documents\AirSim\settings.json
Default config: PhysXCar4x4
Press F1 to see help
API server started at (default)
Reverse: (API)
Throttle: (API)
Steering: (API)
Footbrake: (API)
Speed: 1 km/h
Gear: 1

Future of Software

Program Logic

+

Artificial Intelligence

SW Development

Data Schemas

Functions & Code

Debugger to process

Use telemetry to inform

AI Development

Data Curation + Data Labeling

Machine Learning Models

A/B Testing, Experimentation

Telemetry driven learning

Mathematics used in AI (PART OF)

Linear Algebra

- Vectors: definition, scalars, addition, scalar multiplication, inner product(dot product)…
- Matrices: definition, addition, transpose, scalar multiplication, matrix multiplication…
- Eigenvalues & eigenvectors: concept, intuition, significance, how to find
- Principle component analysis, Singular value decomposition

Calculus

- Functions, Scalar derivative: intuition, common rules of differentiation, chain rule, partial derivatives
- Gradient: concept, intuition, properties, directional derivative, Vector and matrix calculus: how to find derivative of …
- Gradient algorithms: local/global maxima and minima, saddle point, convex functions, …

Probability

- Basic rules and axioms: events, sample space, frequentist approach, dependent and independent events, conditional …
- Random variables: continuous and discrete, expectation, variance, distributions- joint and conditional
- Bayes' Theorem, MAP, MLE, Popular distributions: binomial, bernoulli, poisson, exponential, gaussian, Conjugate priors

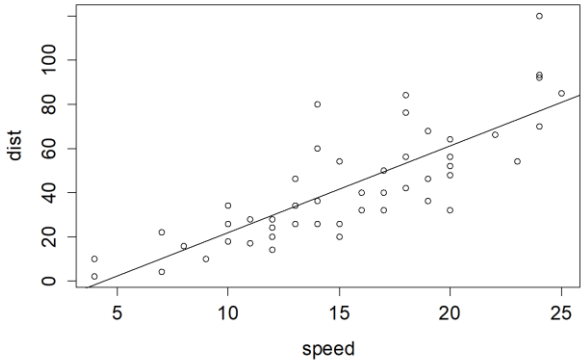
Miscellaneous

- Information theory: entropy, cross-entropy, KL divergence, mutual information
- Markov Chain: definition, transition matrix, stationarity

AI / Machine learning techniques

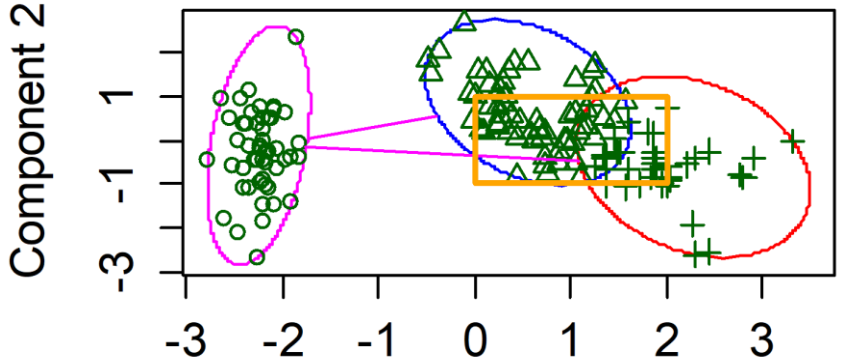
Supervised learning (task driven)

- Training data includes examples of correct solutions to the problem
- Find/generalize relationship between the input data and solution
- Example: regression, classification



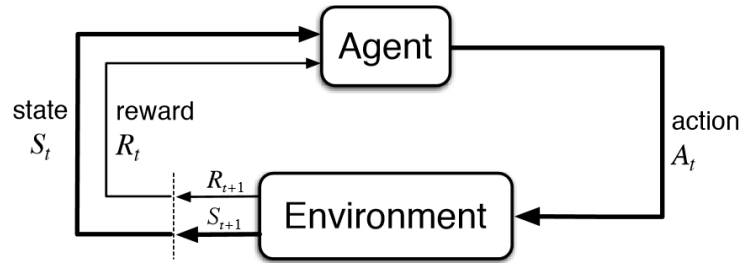
Unsupervised learning (data driven)

- Points are not associated with known output values
- Creates a model that learns inherent structure in training data
- Example: clustering, time-series



Reinforcement learning (react)

- Observations gathered from the interaction with the environment to take actions that would maximize the reward or minimize the risk
- Example: Q-Learning, SARSA, DQN



What are the “neural networks”

This is just a special kind of algorithm!

We can do clustering using DNN or regression or time series …

“Classification”

- Artificial Neural Network
  - Deep Neural Network (generic type neural network with many hidden layers)
  - Deep Belief networks
  - Recurrent neural networks (robot control, music composition, grammar learning, time series)
  - Convolutional neural network (mainly cognitive vision services; object detection)
  - …

See: https://en.wikipedia.org/wiki/Types_of_artificial_neural_networks

DNN - „Background”

The activating function sums up the "input signals" and reports the output signal

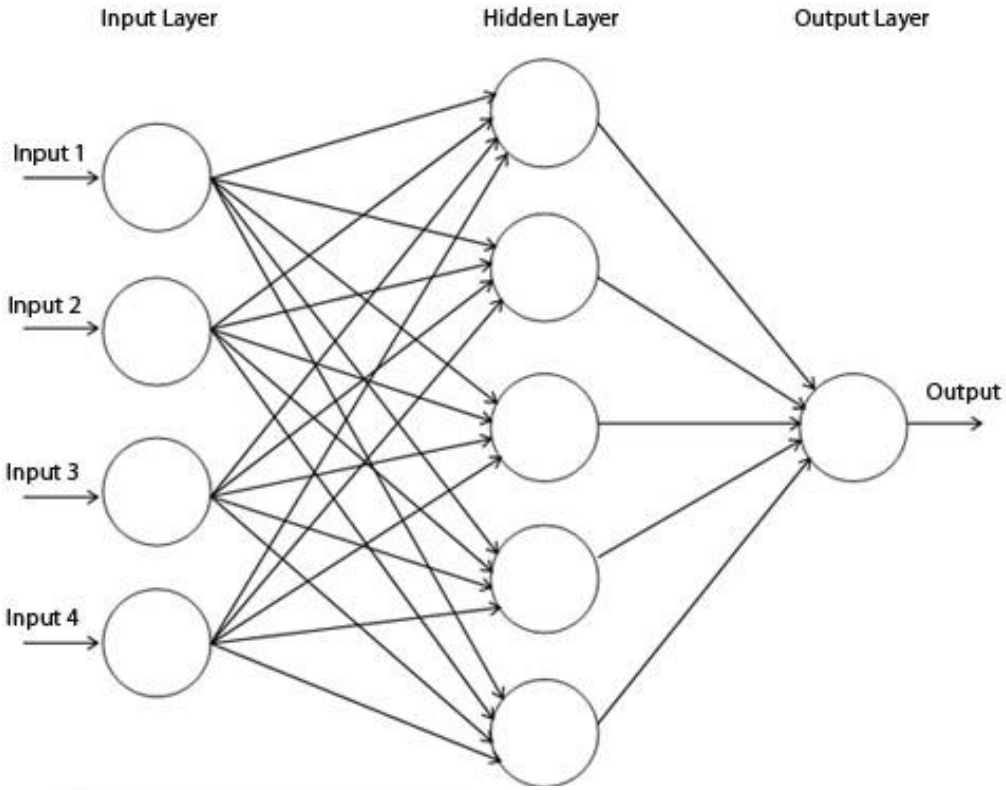
- The initial weights are random

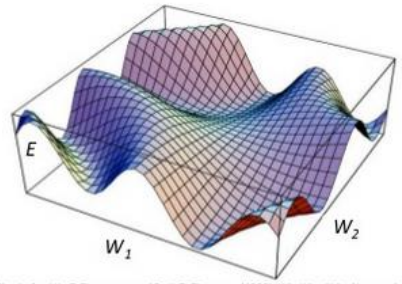
After a given learning phase, the weights are corrected to reduce the error

Changes in weights are controlled by the current learning rate

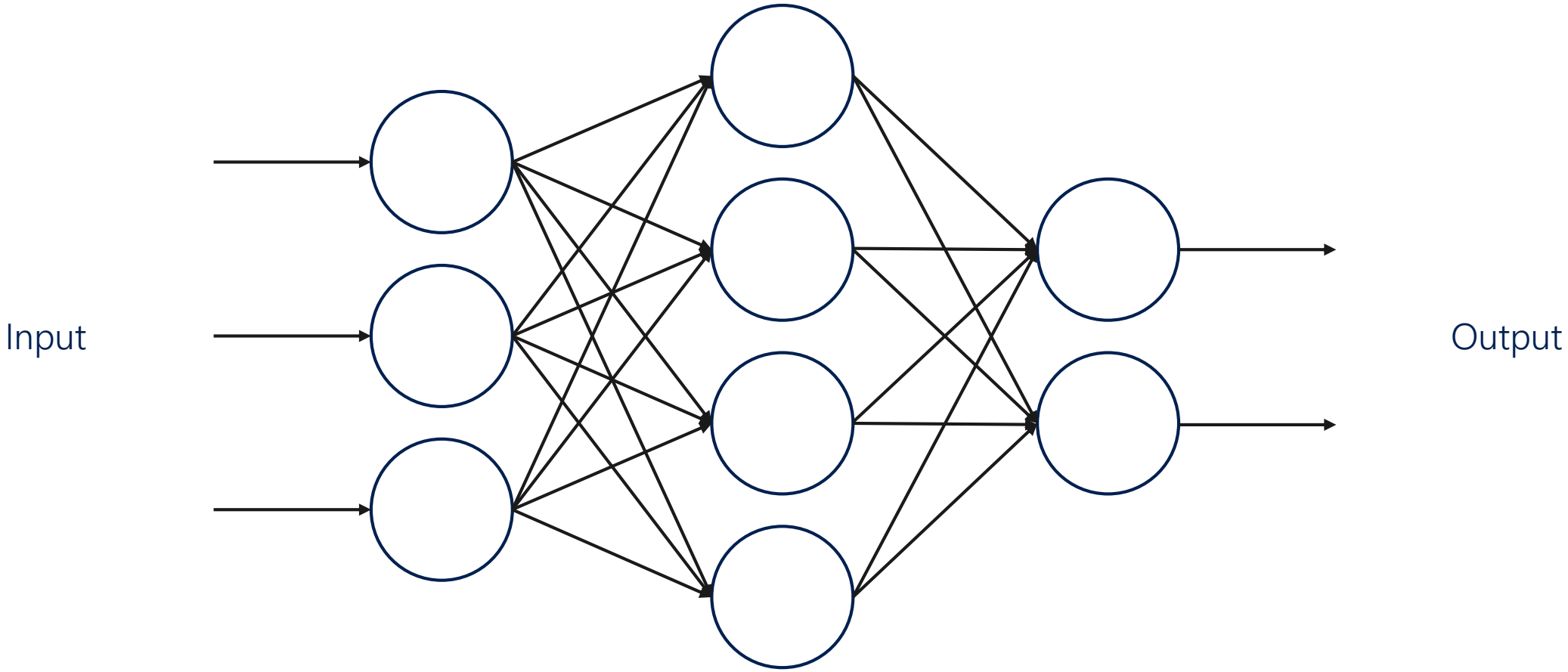
- CNTK: Stochastic Gradient Descent, optimization alghoritm, change LR
  - (1 bit to scale out)
- There are many local maximums!

For many weight attributes they create a multidimensional space





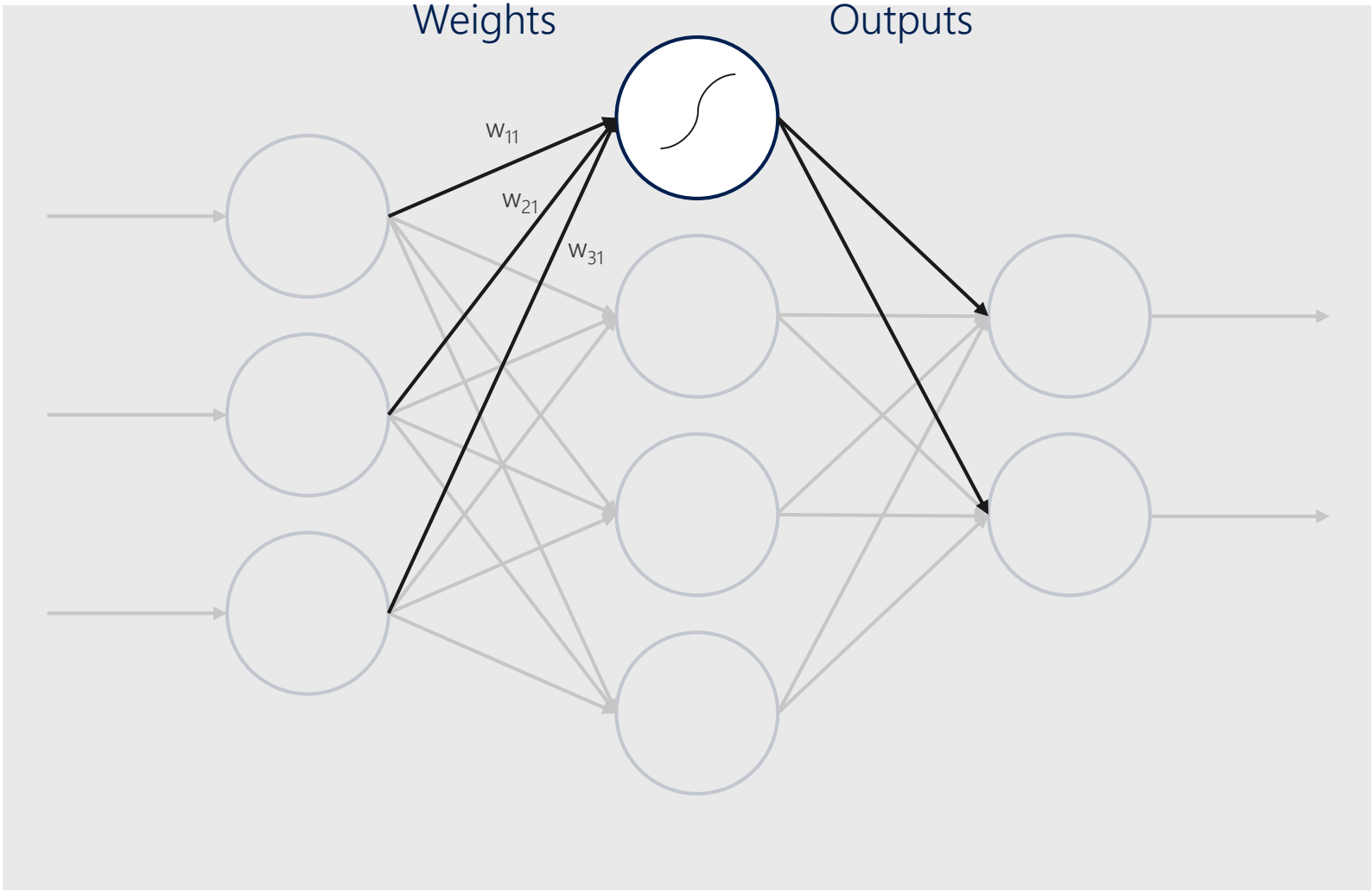
Source: Oberhofer, W., T. Zimmerer, and D.-K. T. Zimmerer (1996). Wie Künstliche Neuronale Netze lernen, p. 16


Input
Output

Activation

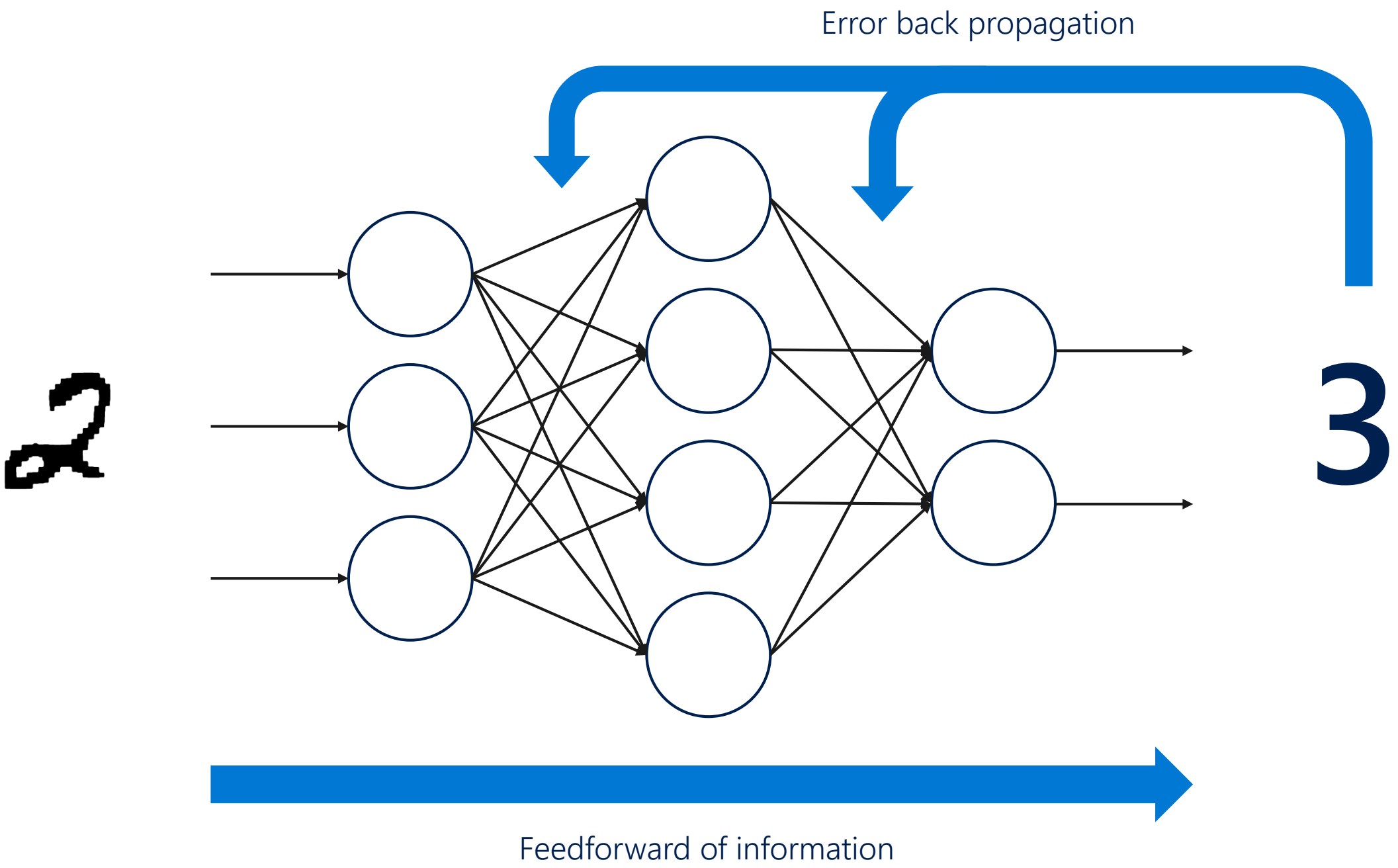
Weights

Outputs



Input

Output


Error back propagation
2
3
Feedforward of information

Can Machine Learning Help Me?

- Automated prediction is core
- Lots of past data already available
- Magic numbers in current prediction system

Yes

- Prediction is small part of experience
- No past data available
- Many business-rules govern the experience
- Predictions do not have a predictable pattern

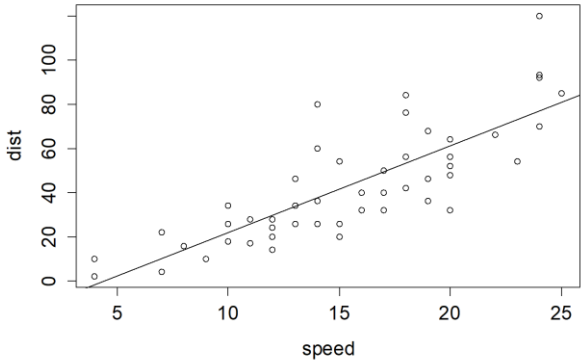
No

Choose algorithm?

Machine learning techniques

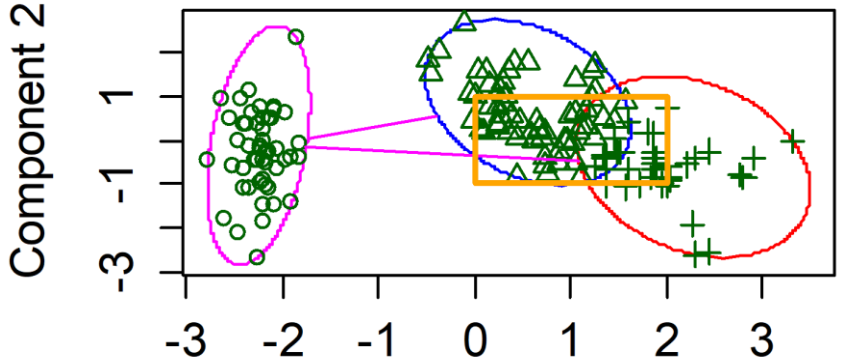
Supervised learning (task driven)

- Training data includes examples of correct solutions to the problem
- Find/generalize relationship between the input data and solution
- Example: regression, classification



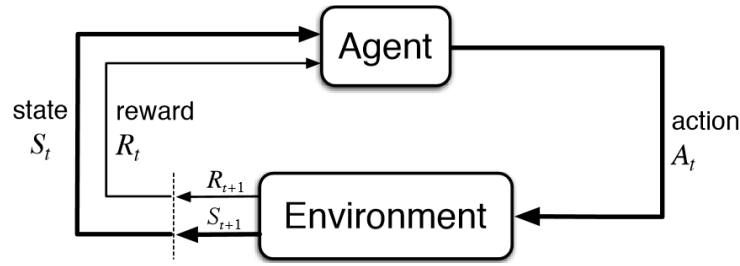
Unsupervised learning (data driven)

- Points are not associated with known output values
- Creates a model that learns inherent structure in training data
- Example: clustering, time-series



Reinforcement learning (react)

- Observations gathered from the interaction with the environment to take actions that would maximize the reward or minimize the risk
- Example: Q-Learning, SARSA, DQN



Develop a Model - Classification

Classification

Scenarios:

Which customer are more likely to buy, stay, leave (churn analysis)

Which transactions/actions are fraudulent

Which quotes are more likely to become orders

Recognition of patterns: speech, speaker, image, movement, etc.

Algorithms: Boosted Decision Tree, Decision Forest, Decision Jungle, Logistic Regression, SVM, ANN

Develop a Model - Regression

Regression

Scenarios:

- Stock prices prediction
- Sales forecasts
- Premiums on insurance based on different factors
- Quality control: number of complaints over time based on product specs, utilization, etc.
- Workforce prediction
- Workload prediction

Algorithms: Bayesian Linear, Linear Regression, Ordinal Regression, ANN
Boosted Decision Tree, Decision Forest

Develop a Model - Clustering

Clustering

Scenarios:

- Customer segmentation: divide a customer base into groups of individuals that are similar in specific ways relevant to marketing, such as age, gender, interests, spending habits, etc.
- Market segmentation
- Quantization of all sorts, such as, data compression, color reduction, etc.
- Pattern recognition

Algorithms: K-means

Sometimes: First, Principal Component Analysis to reduce number of dimensions

Develop a Model - Recommendations

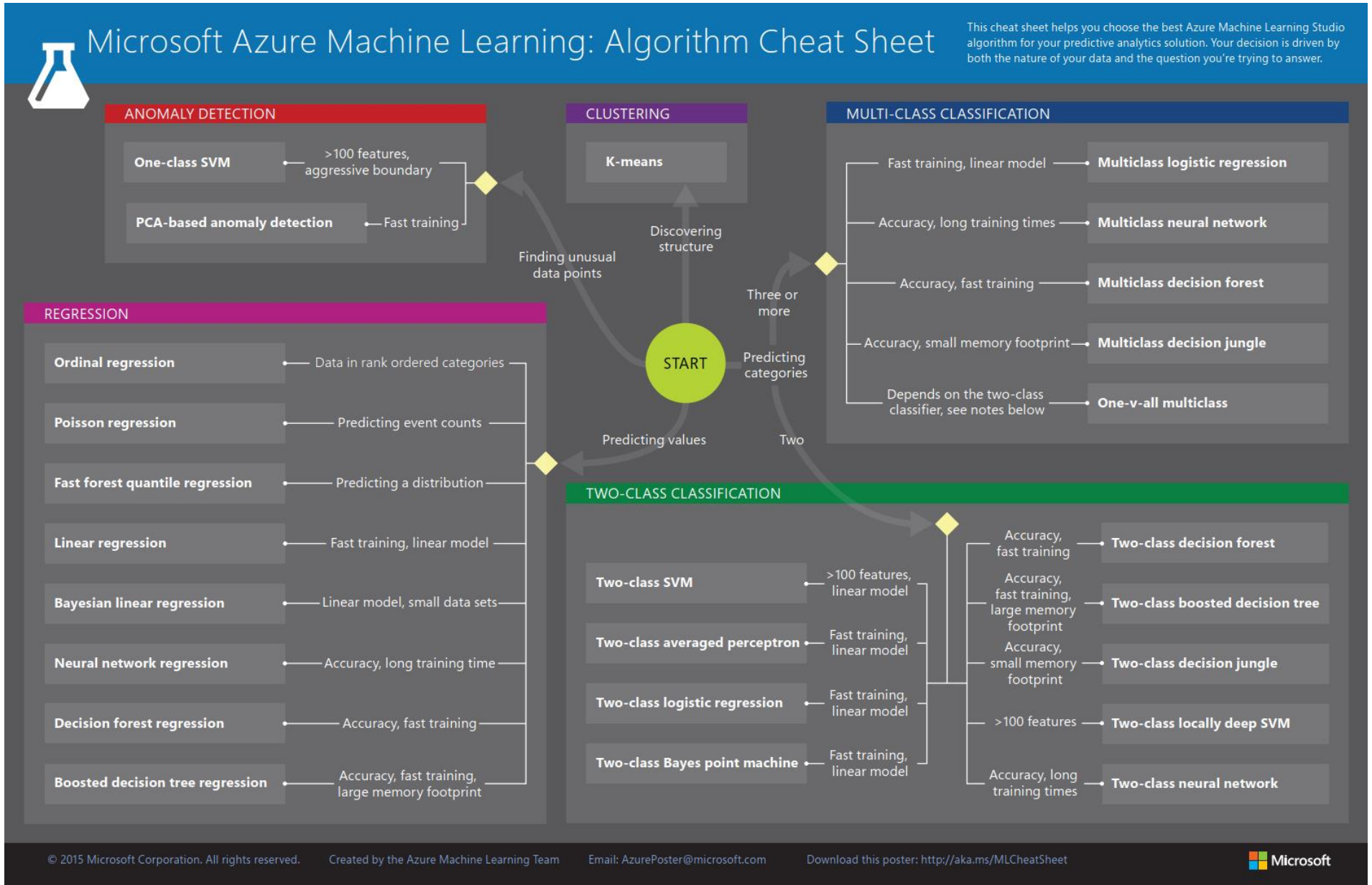
Recommendations

Scenarios:

- Product recommendations
- Netflix video recommendations

Algorithms: Matchbox

http://aka.ms/MLCheatSheet

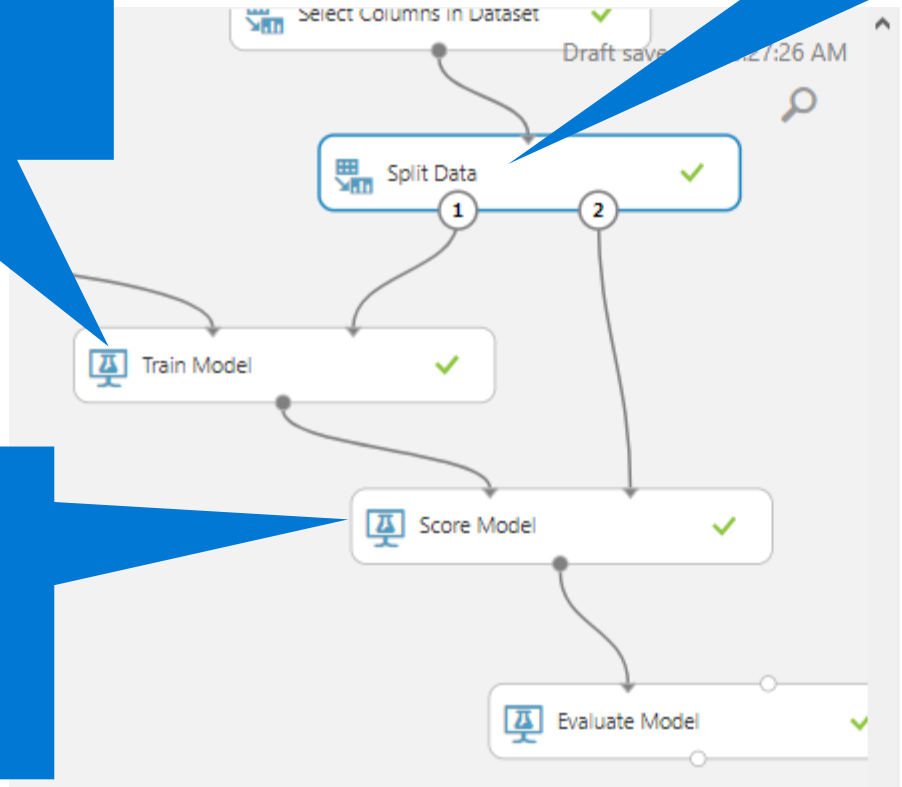


Validate model

Testing

Split data to training and testing set

Train model (using attached algorithm)



Check how model work on testing set

Split Data

Splitting mode

Split Rows

Fraction of rows in the firs...

0.8

Randomized split

Random seed

123

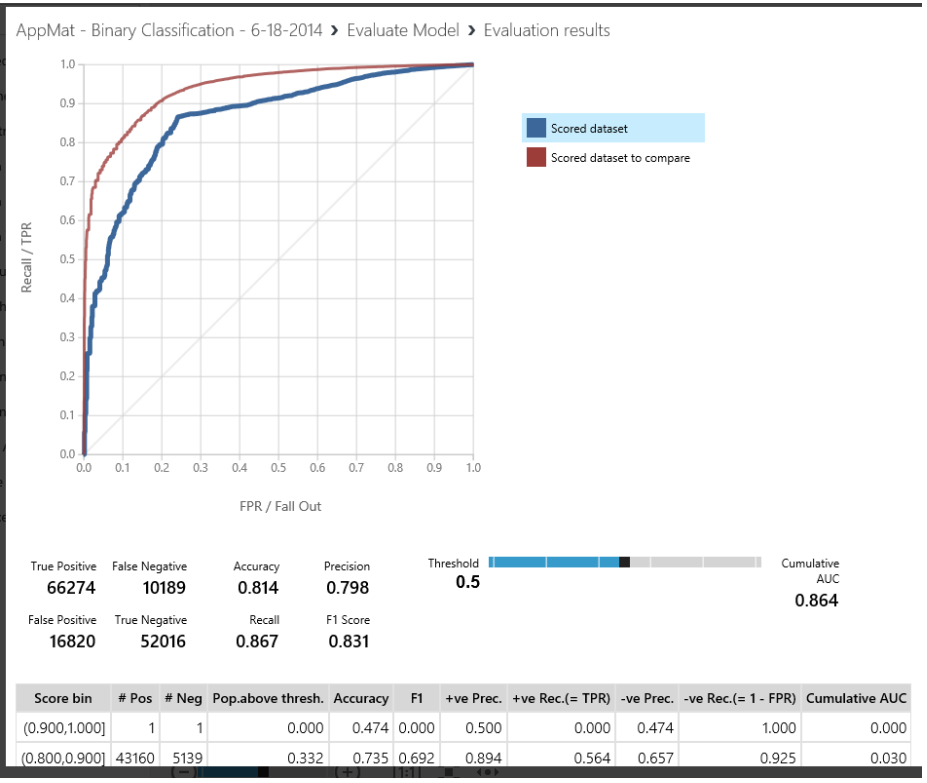
Stratified split

False

START TIME	6/28/2018 ...
END TIME	6/28/2018 ...
ELAPSED TIME	0:00:00.000
STATUS CODE	Finished

Model Experiment and Evaluate

Classification



Score bin	# Pos	# Neg	Pop.above thresh.	Accuracy	F1	+ve Prec.	+ve Rec.(= TPR)	-ve Prec.	-ve Rec.(= 1 - FPR)	Cumulative AUC
(0.900,1.000]	1	1	0.000	0.474	0.000	0.500	0.000	0.474	1.000	0.000
(0.800,0.900]	43160	5139	0.332	0.735	0.692	0.894	0.564	0.657	0.925	0.030

- Accuracy $= \frac{TP + TN}{P+N}$
- Precision $= \frac{TP}{TP+FP}$
- Recall $= \frac{TP}{TP+FN} = \frac{TP}{P}$
- $F_\beta = \frac{(1+\beta^2) \cdot precision \cdot recall}{\beta^2 \cdot precision + recall}$

Regression

fy18-car-automobile-price › Evaluate Model ›

Metrics

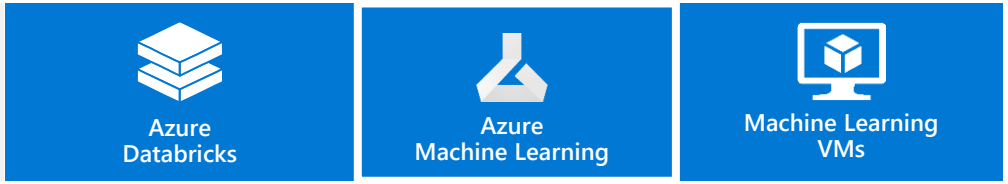
Mean Absolute Error	3808.687462
Root Mean Squared Error	5114.95604
Relative Absolute Error	0.720506
Relative Squared Error	0.643097
Coefficient of Determination	0.356903

Machine Learning on Azure

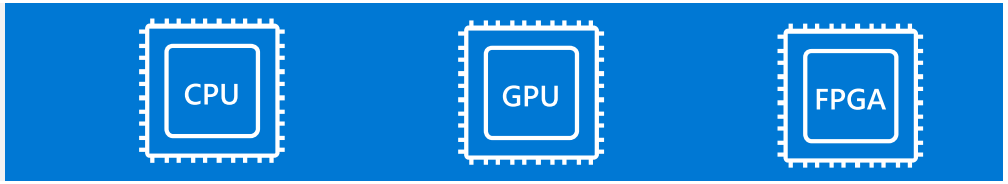
Sophisticated pretrained models
To simplify solution development

Popular frameworks
To build advanced deep learning solutions

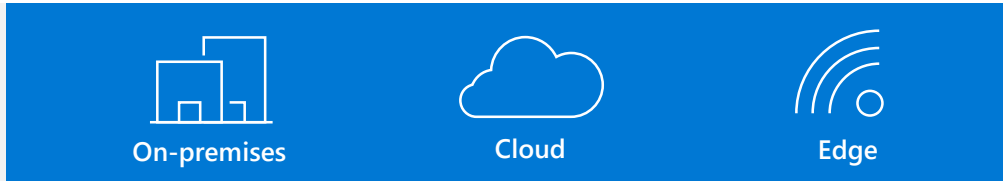
Productive services
To empower data science and development teams



Powerful infrastructure
To accelerate deep learning



Flexible deployment
To deploy and manage models on intelligent cloud and edge



Machine Learning & AI Portfolio

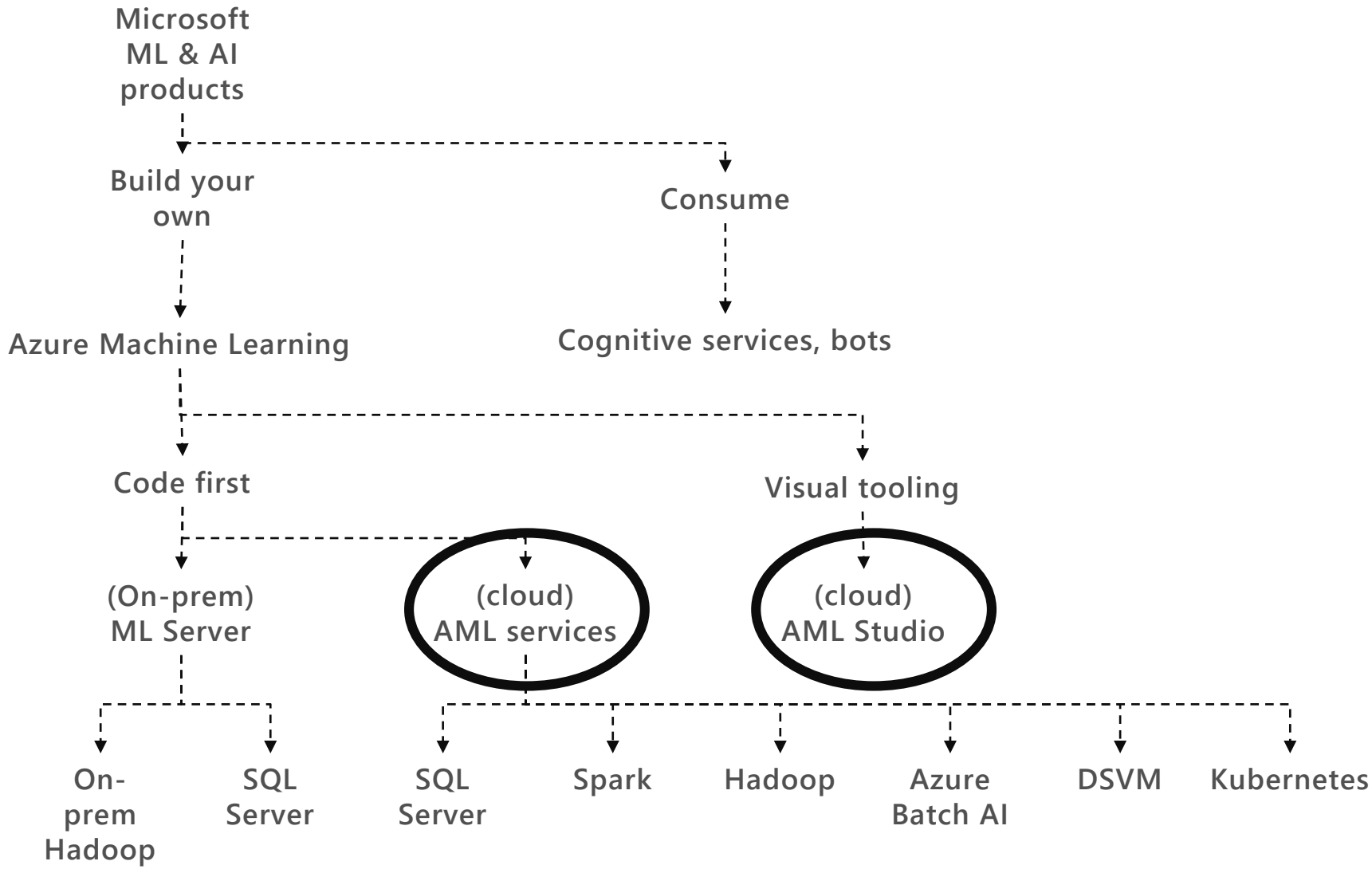
When to use what?

Build your own or consume pre-trained models?

Which experience do you want?

Deployment target

What engine(s) do you want to use?



Learn – basic ML

https://docs.microsoft.com/en-us/azure/machine-learning

Watch/Read: Data Science for beginers

https://docs.microsoft.com/en-us/azure/machine-learning/studio/data-science-for-beginners-the-5-questions-data-science-answers

For more: Machine Learning & AI Foundations: Recommendations on LinkedIn (https://www.linkedin.com/learning/machine-learning-ai-foundations-recommendations?u=3322)

Python basics: Python Essential Training (https://www.linkedin.com/learning/python-essential-training-2?u=3322)

Python for Machine Learning using jupyter notebooks (https://www.linkedin.com/learning/python-for-data-science-essential-training?u=3322)

Materials – step 2

https://msropendata.com/ - many datasets from different industries!

http://www.misjaazure.pl
https://gallery.azure.ai/
https://aischool.microsoft.com/en-us/home

https://www.microsoft.com/en-us/research/lab/microsoft-research-ai/
https://github.com/MSRConnections/Azure-training-course
https://www.ailab.microsoft.com/experiments

Demo

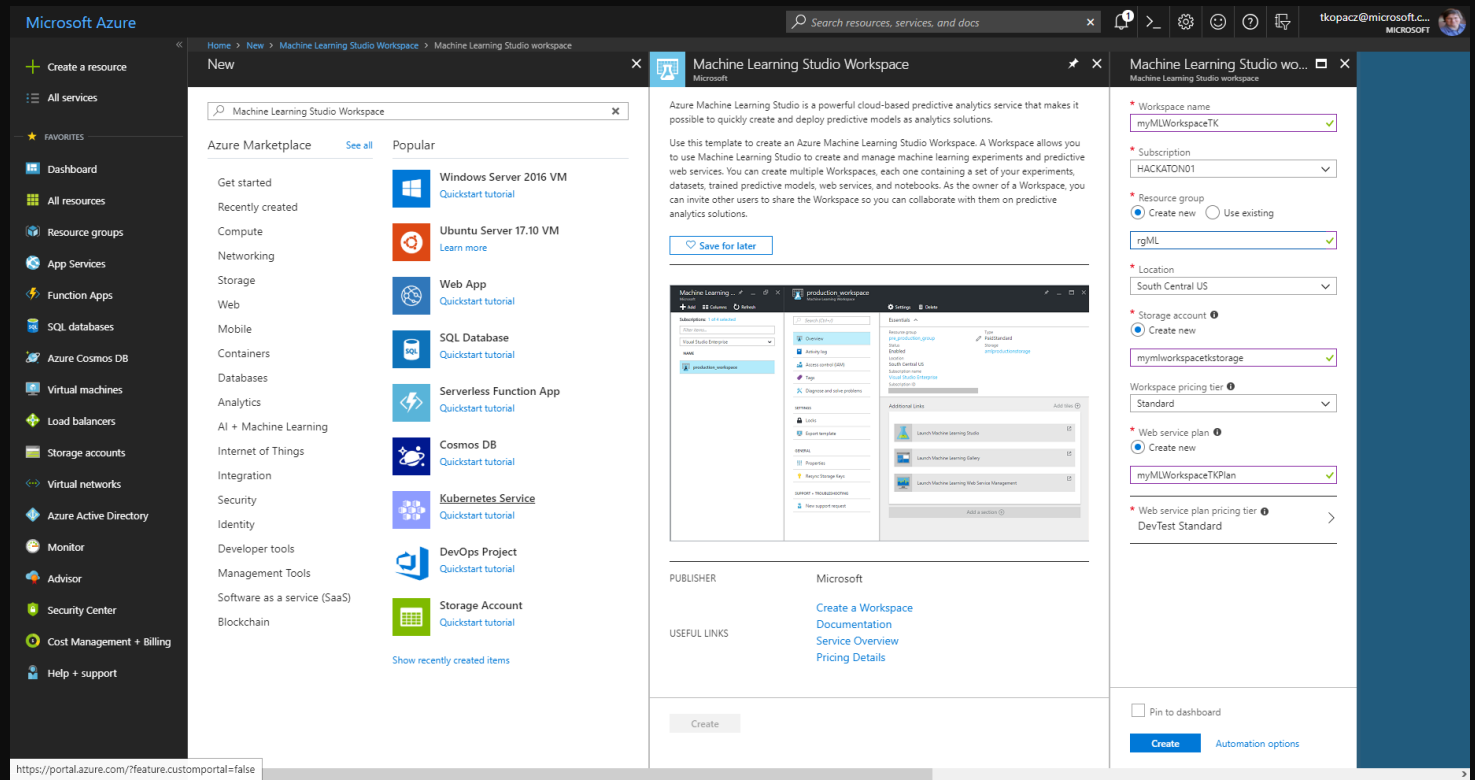
How to start with Machine Learning Studio?

Create Machine Learning Studio Workspace

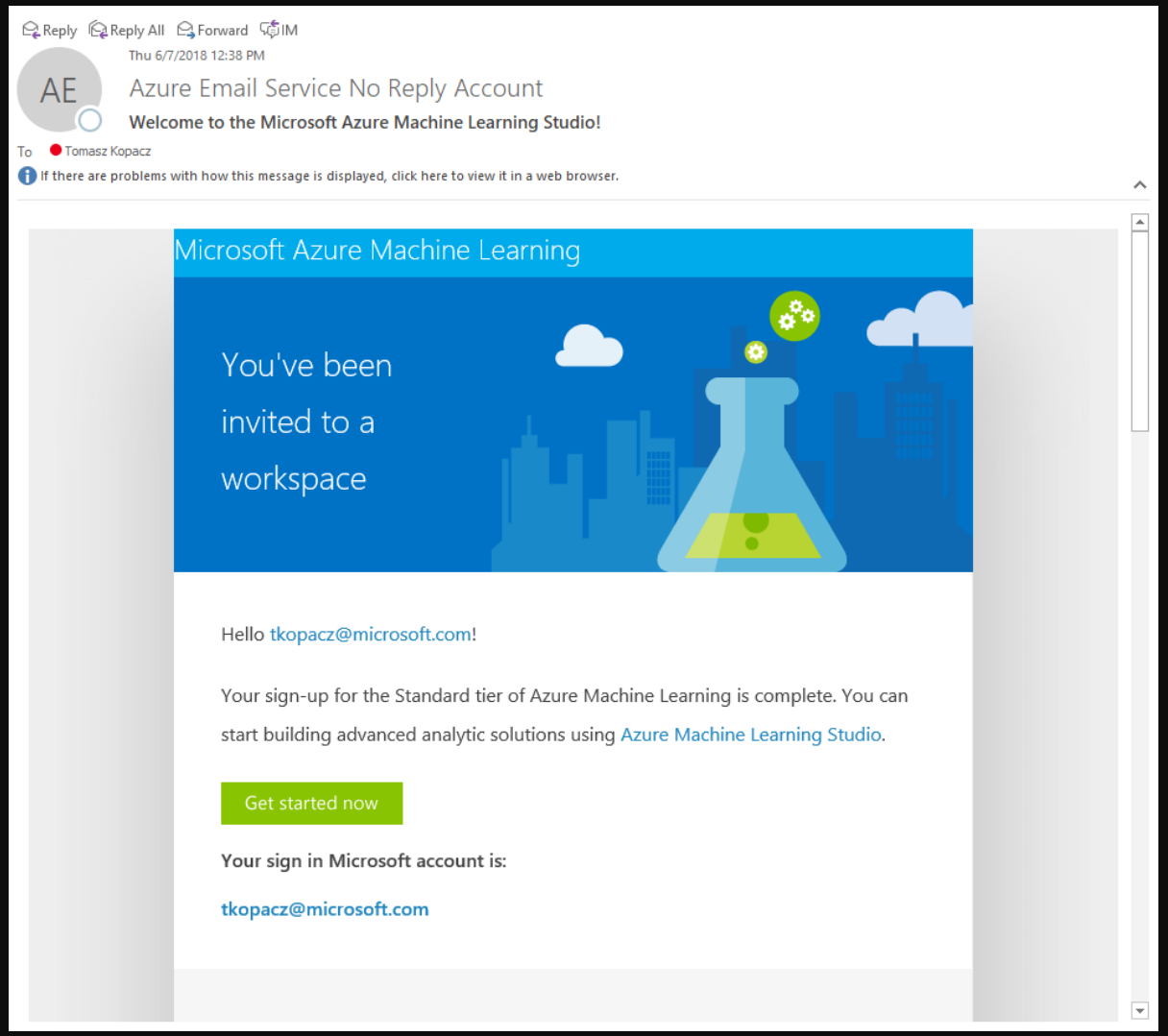
http://portal.azure.com

Create a Resource

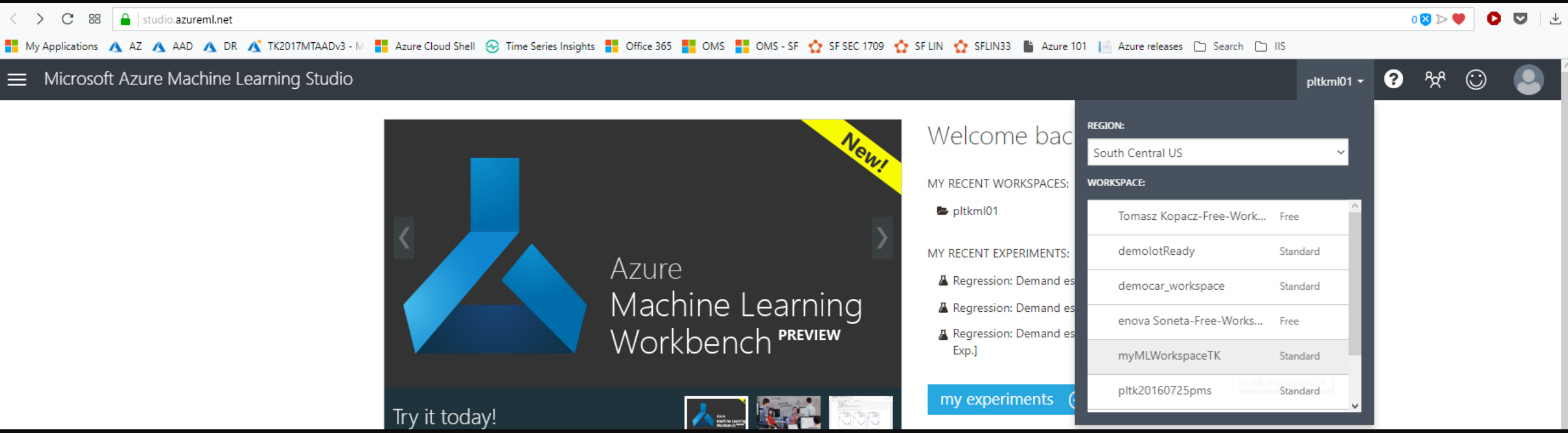
Machine Learning Studio Workspace



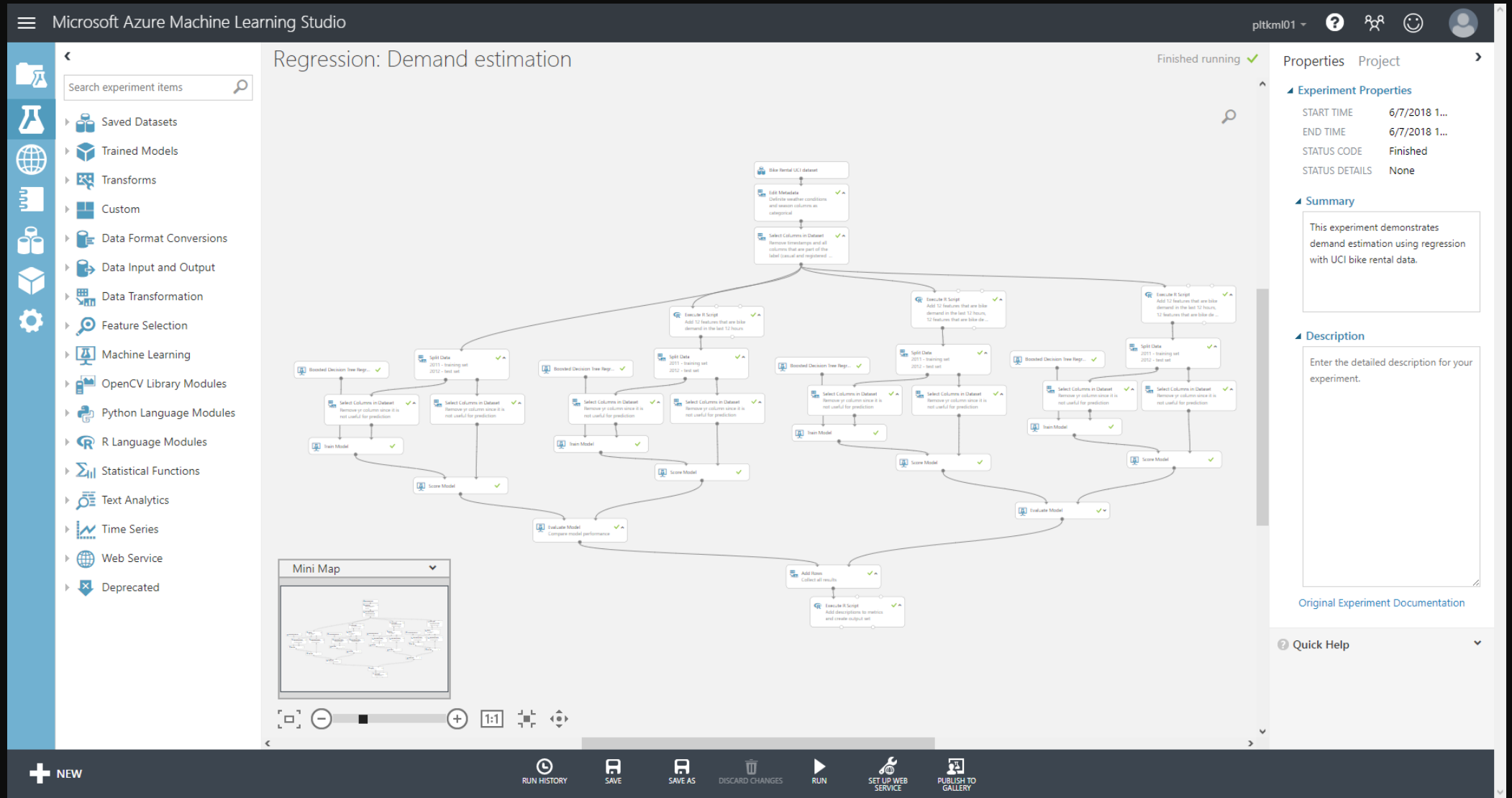
You should receive mail like this



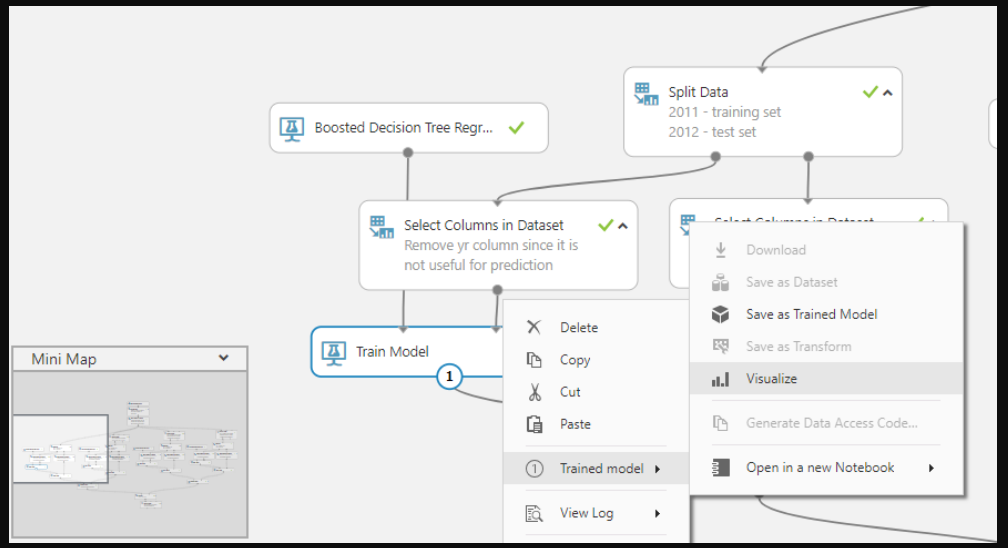
Go & login to studio.azure.ml

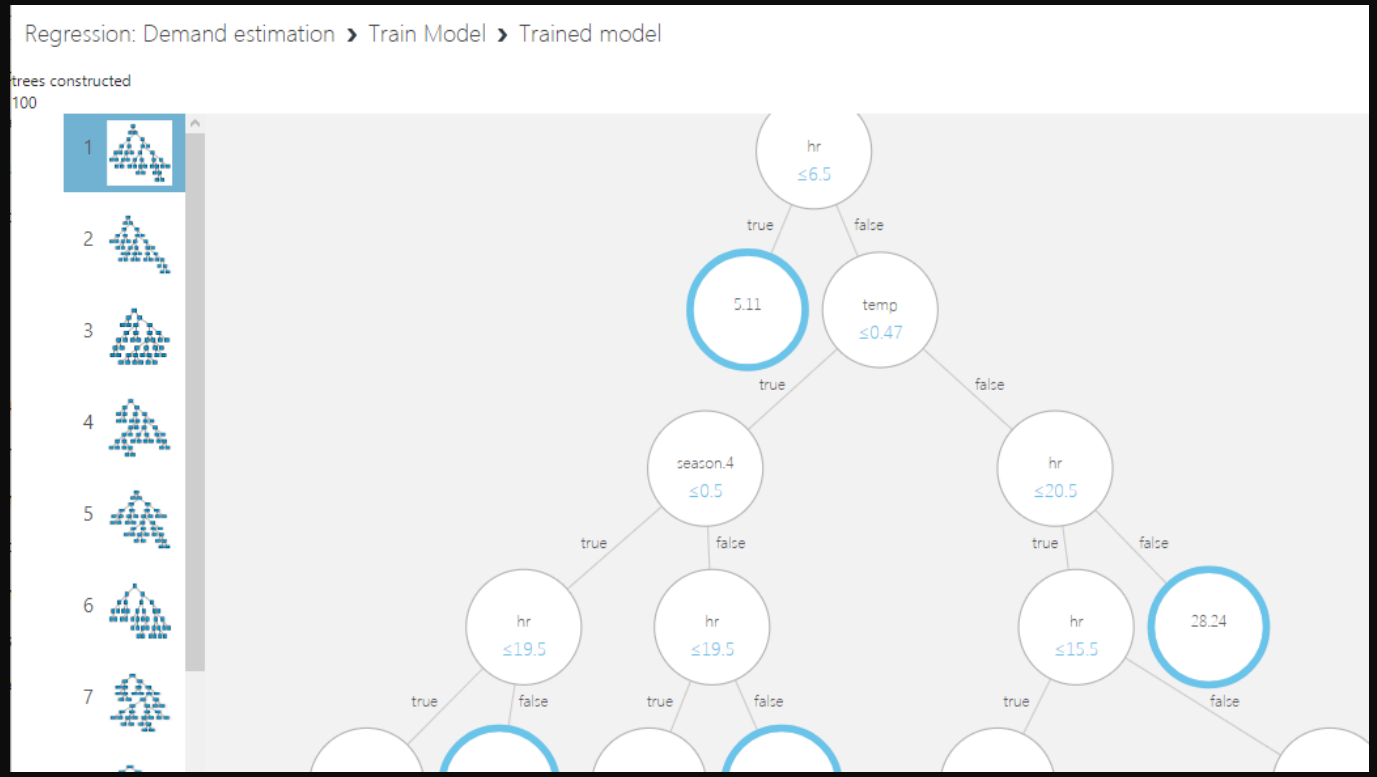


Sample UI

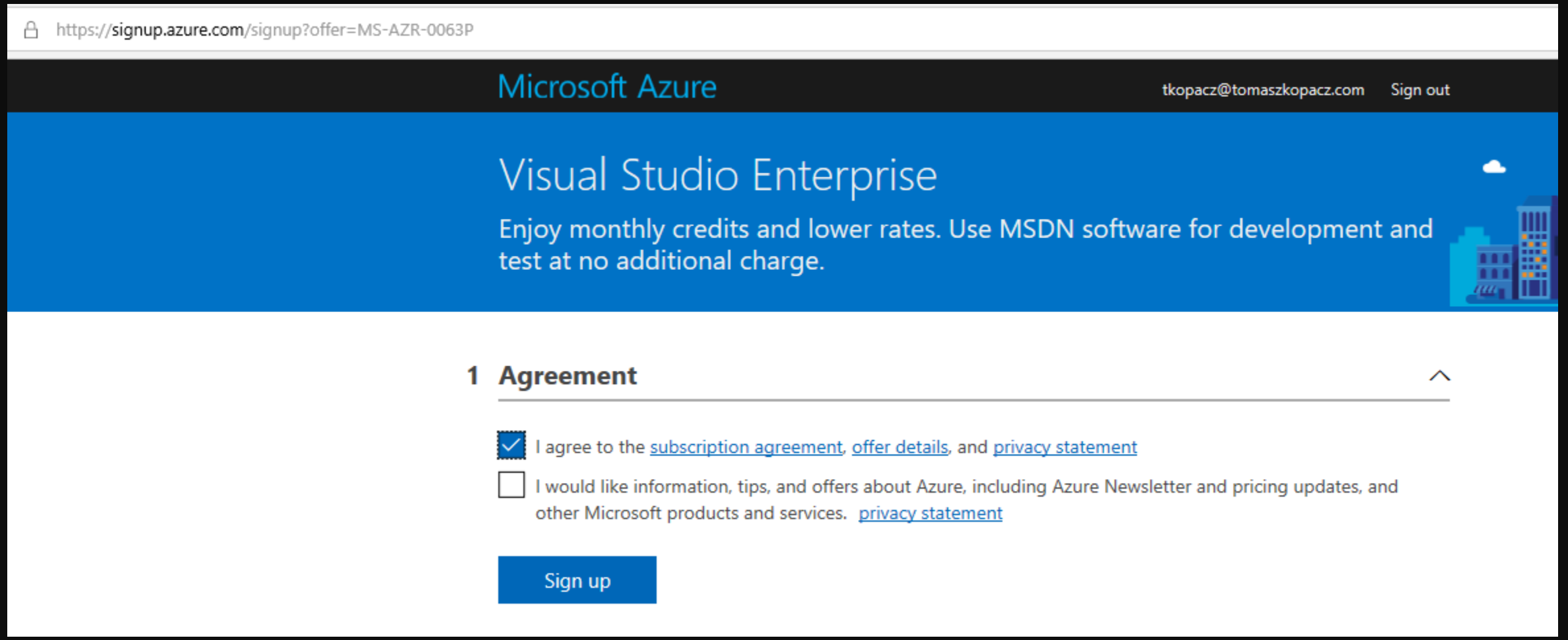

Microsoft Azure Machine Learning Studio
pltkml01
Regression: Demand estimation
Finished running
Search experiment items
Saved Datasets
Trained Models
Transforms
Custom
Data Format Conversions
Data Input and Output
Data Transformation
Feature Selection
Machine Learning
OpenCV Library Modules
Python Language Modules
R Language Modules
Statistical Functions
Text Analytics
Time Series
Web Service
Deprecated
Mini Map
Properties
Project
Experiment Properties
START TIME 6/7/2018 1...
END TIME 6/7/2018 1...
STATUS CODE Finished
STATUS DETAILS None
Summary
This experiment demonstrates demand estimation using regression with UCI bike rental data.
Description
Enter the detailed description for your experiment.
Original Experiment Documentation
Quick Help
NEW
RUN HISTORY
SAVE
SAVE AS
DISCARD CHANGES
RUN
SET UP WEB SERVICE
PUBLISH TO GALLERY

See the model

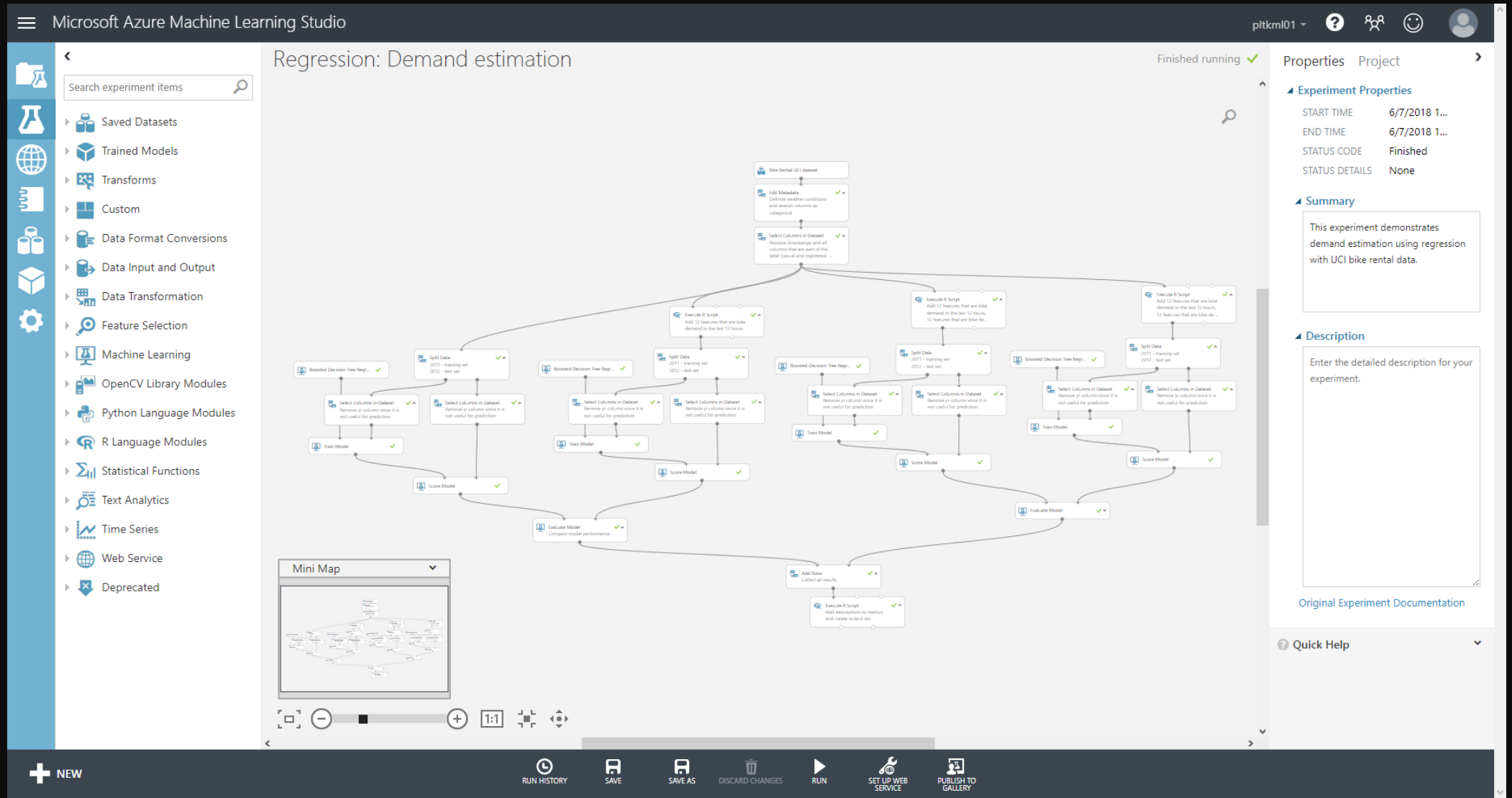




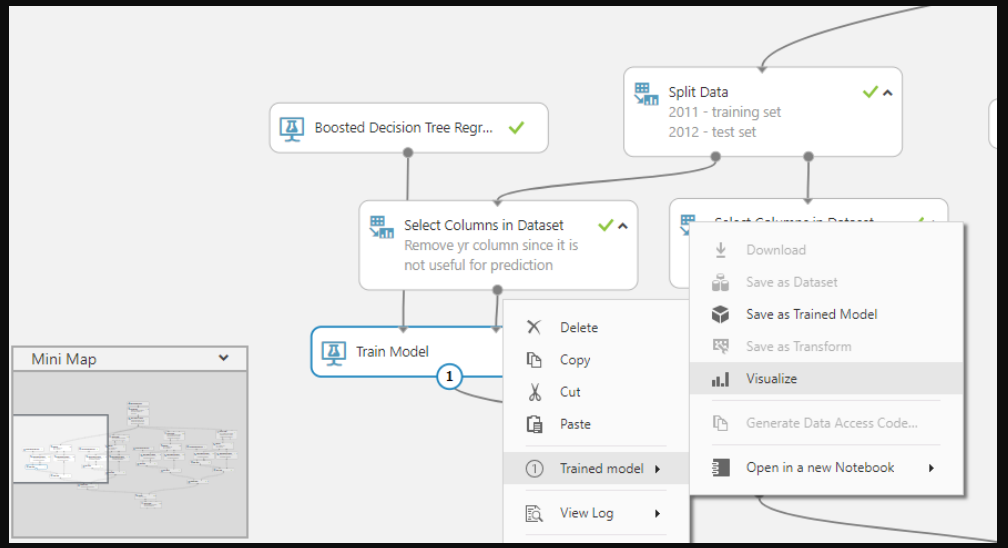
Sign

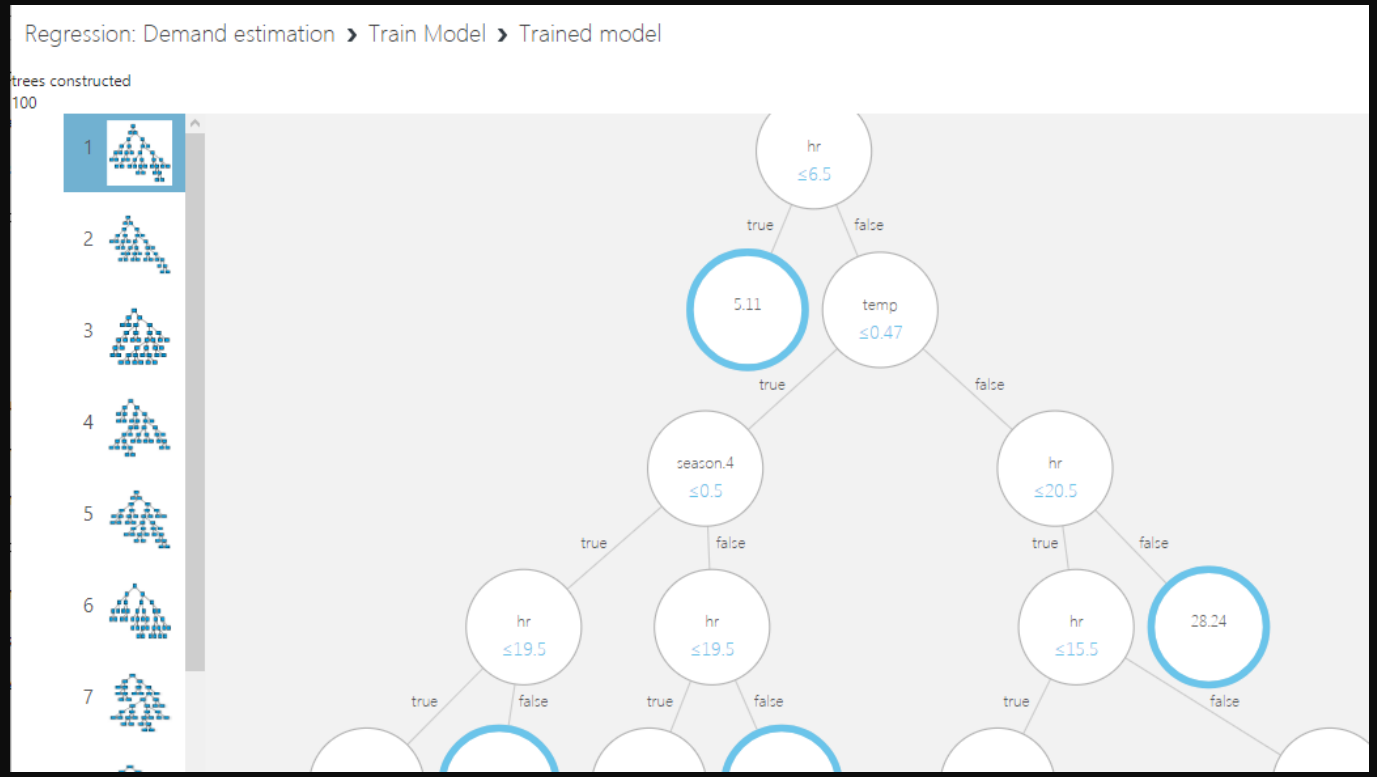

https://signup.azure.com/signup?offer=MS-AZR-0063P
Microsoft Azure
tkopacz@tomaszkopacz.com
Sign out
Visual Studio Enterprise
Enjoy monthly credits and lower rates. Use MSDN software for development and test at no additional charge.
1 Agreement
I agree to the subscription agreement, offer details, and privacy statement
I would like information, tips, and offers about Azure, including Azure Newsletter and pricing updates, and other Microsoft products and services. privacy statement
Sign up

Sample UI

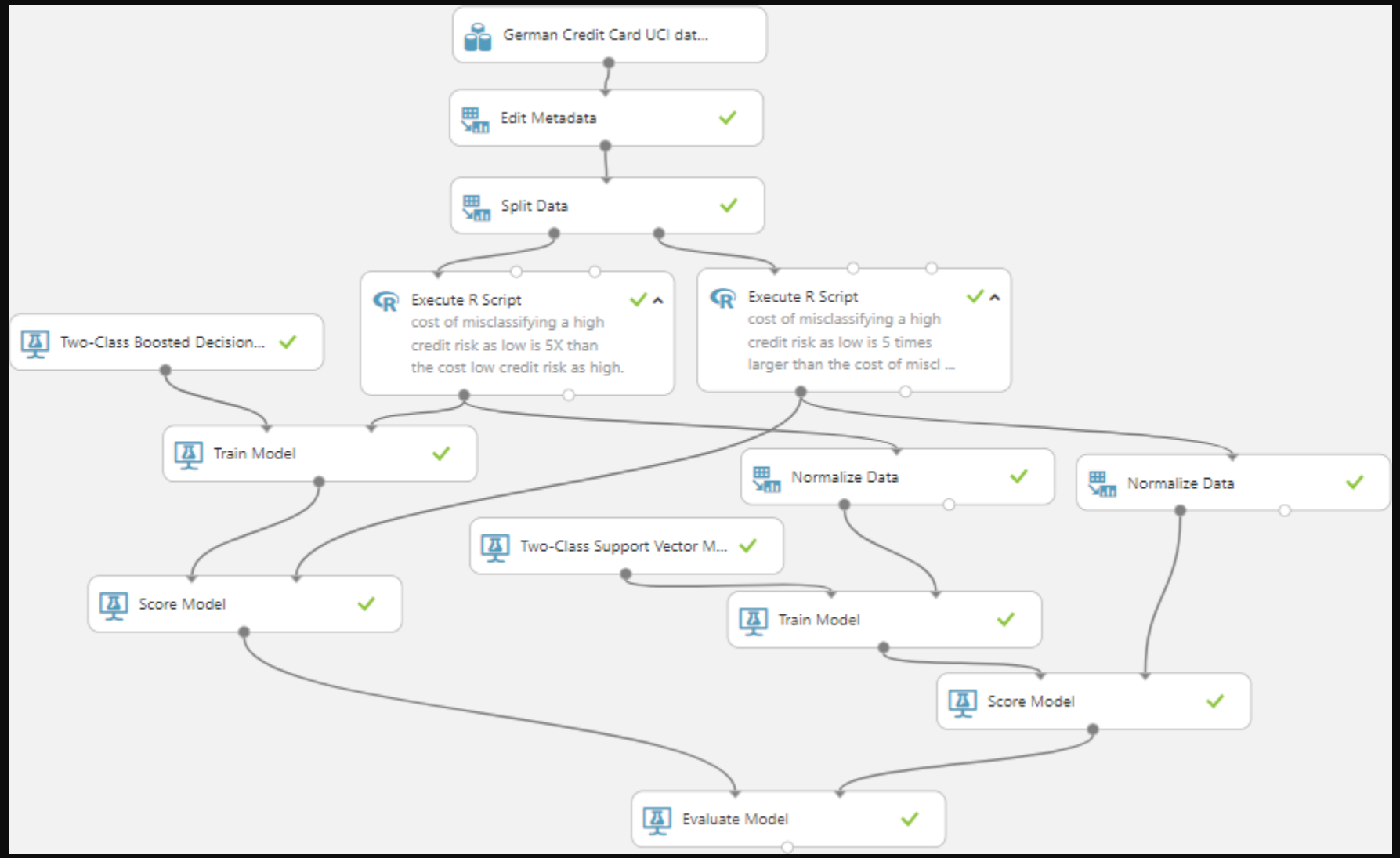

Microsoft Azure Machine Learning Studio
pltkml01
Regression: Demand estimation
Finished running
Search experiment items
Saved Datasets
Trained Models
Transforms
Custom
Data Format Conversions
Data Input and Output
Data Transformation
Feature Selection
Machine Learning
OpenCV Library Modules
Python Language Modules
R Language Modules
Statistical Functions
Text Analytics
Time Series
Web Service
Deprecated
Mini Map
Properties
Project
Experiment Properties
START TIME 6/7/2018 1...
END TIME 6/7/2018 1...
STATUS CODE Finished
STATUS DETAILS None
Summary
This experiment demonstrates demand estimation using regression with UCI bike rental data.
Description
Enter the detailed description for your experiment.
Original Experiment Documentation
Quick Help
NEW
RUN HISTORY
SAVE
SAVE AS
DISCARD CHANGES
RUN
SET UP WEB SERVICE
PUBLISH TO GALLERY

See the model



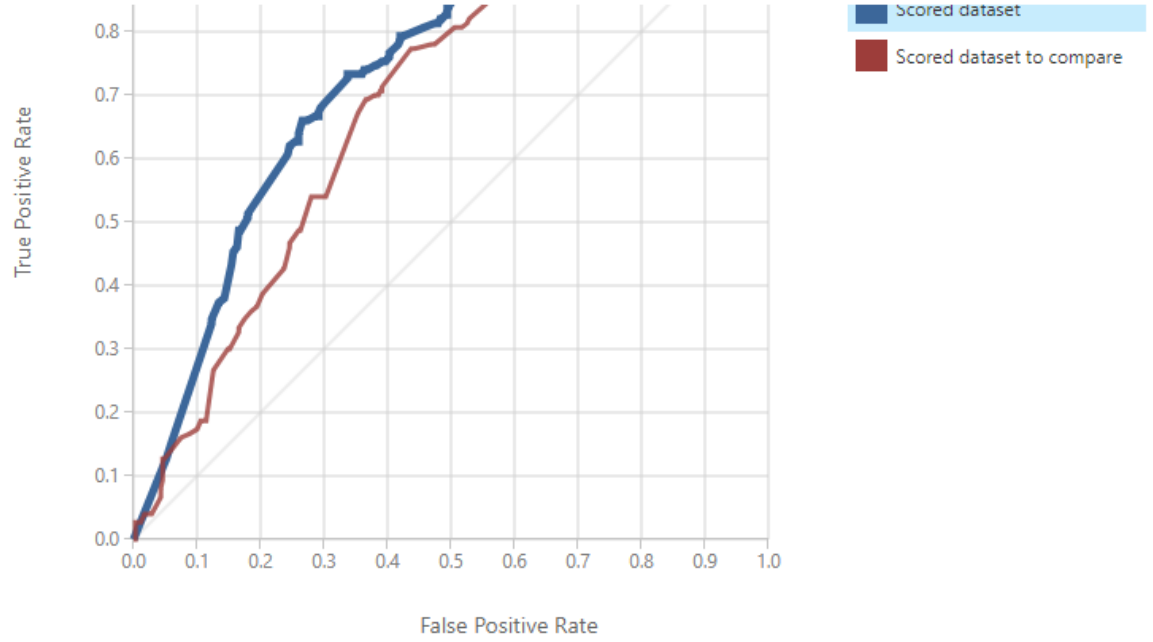


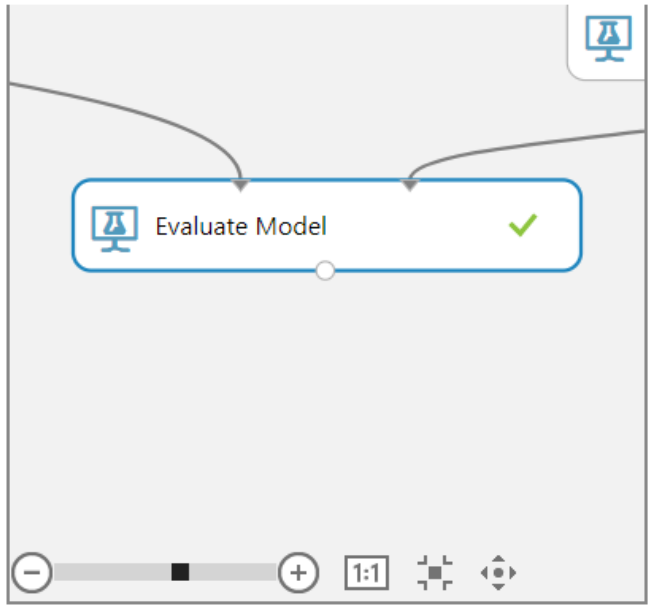
Final


German Credit Card UCI dat...
Edit Metadata
Split Data
Execute R Script
cost of misclassifying a high credit risk as low is 5X than the cost low credit risk as high.
Execute R Script
cost of misclassifying a high credit risk as low is 5 times larger than the cost of miscl ...
Two-Class Boosted Decision...
Train Model
Normalize Data
Normalize Data
Two-Class Support Vector M...
Score Model
Train Model
Score Model
Evaluate Model

Results

fy18-GermanCredit › Evaluate Model › Evaluation results





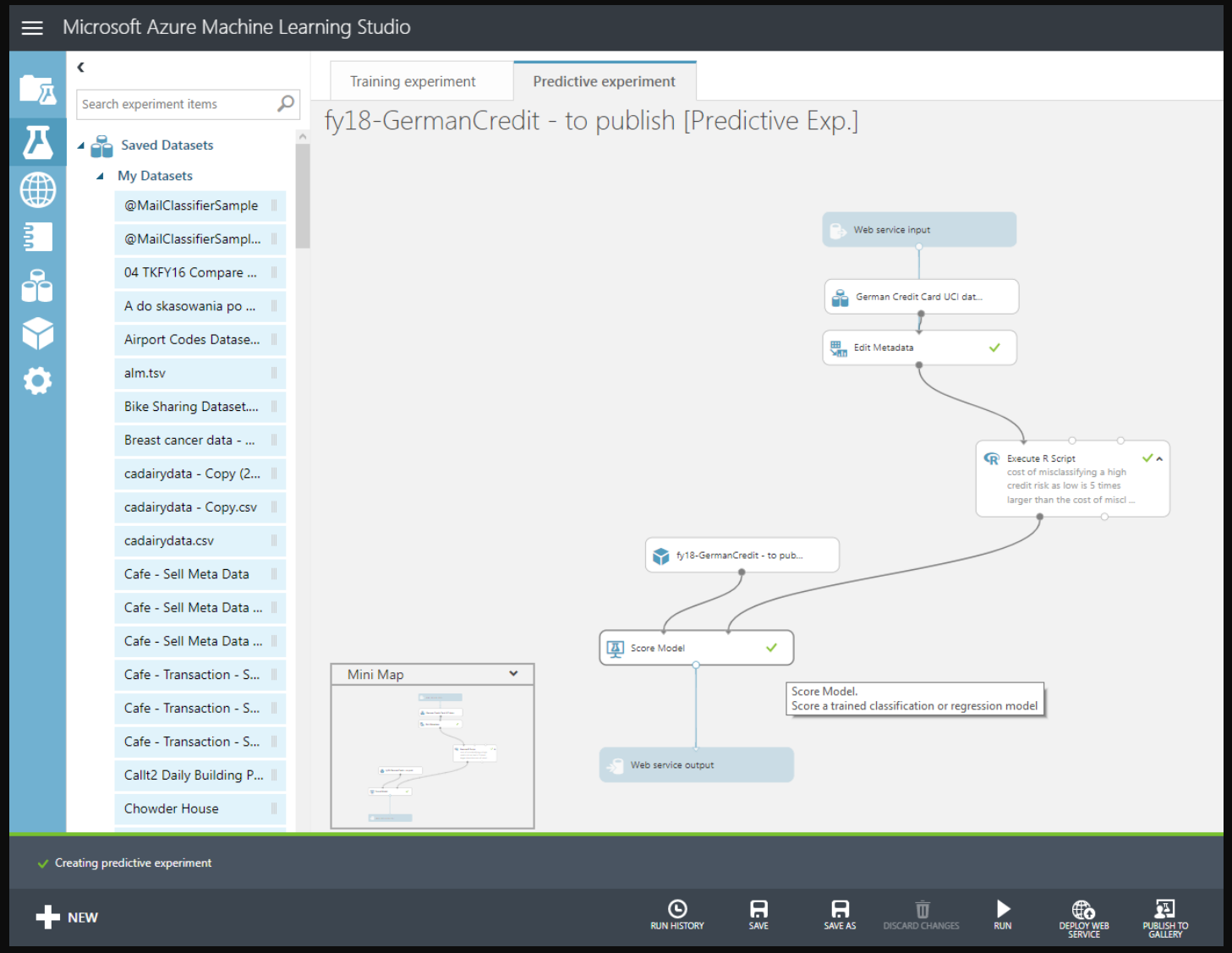
True Positive	False Negative	Accuracy	Precision	Threshold	AUC
620	130	0.725	0.783	0.5	0.739

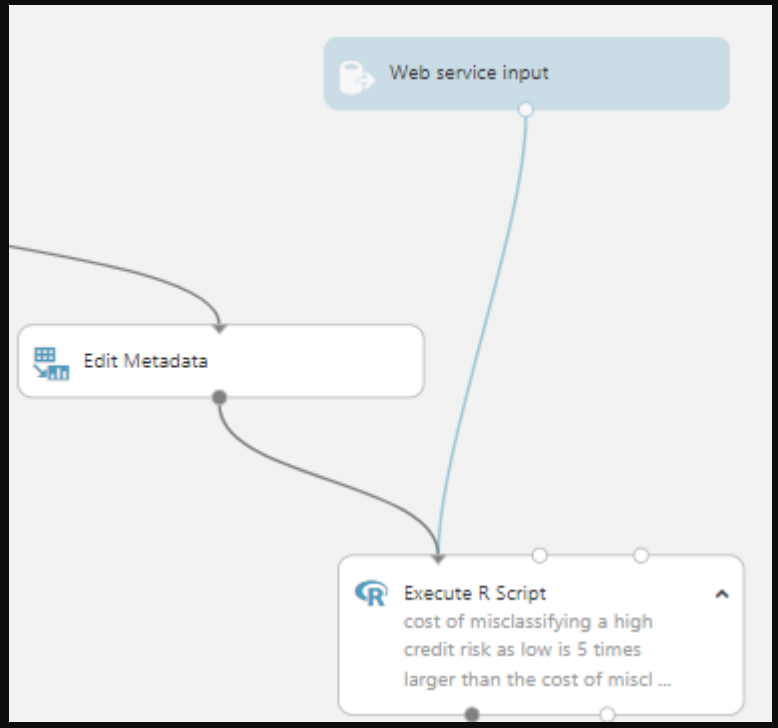
False Positive	True Negative	Recall	F1 Score
172	178	0.827	0.804

Positive Label	Negative Label
2	1

Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	95	18	0.103	0.388	0.220	0.841	0.127	0.336	0.949	0.003
(0.800,0.900]	375	71	0.508	0.665	0.718	0.841	0.627	0.482	0.746	0.087

Publish better model as web service





Microsoft Cognitive Toolkit (CNTK)

1st-class on Linux and Windows and Docker

Rich API support

- Mostly implemented in C++ (train and eval)
- Low level + high level Python API
- R and C# API for train and eval, UWP, Java and Spark support
- Built-in readers for distributed learning
- Keras backend support (Beta) (Keras = high-level neural networks API, for HUMANS)
- Model compression (Fast binarized evaluation)

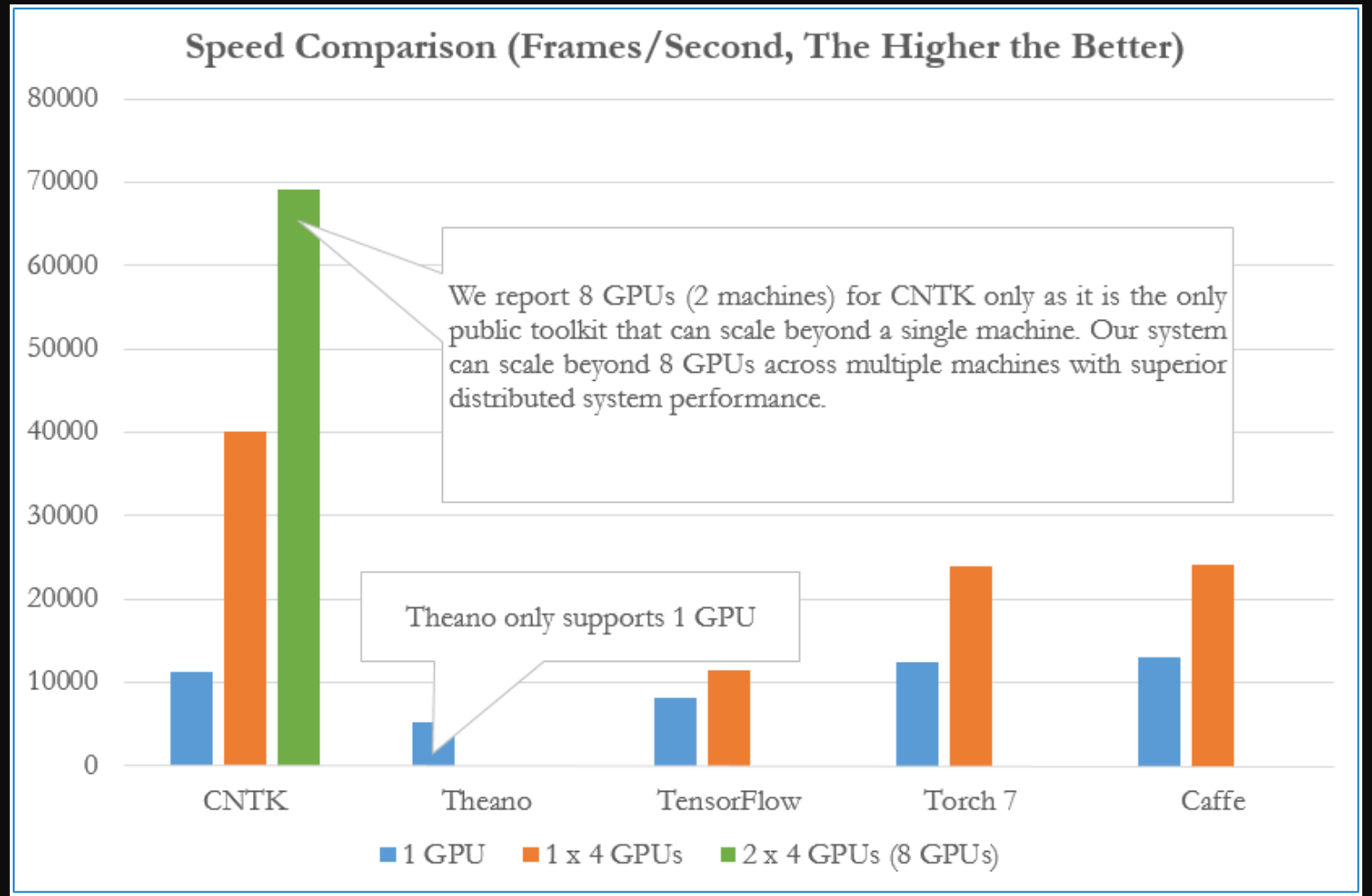
Latest version: v2.2 (Sep. 15, 2017)

Open Neural Network Exchange (ONNX)

- (beta) Share model across frameworks.
- Calculate on CPU | GPU | FPGA | Silicon
- Use / Train: CNTK / Caffe 2, PyTorch / Keras ….

By the way – why CNTK ?

It can scale ACROSS MACHINES! Docs here



SimpleNet.ipynb

jupyter SimpleNet Last Checkpoint: 14 hours ago (autosaved) — Azure Notebooks — My Libraries — 2018Athens

File Edit View Insert Cell Kernel Data Widgets Help — Not Trusted — Python 3.6

Run | Markdown | Enter/Exit RISE Slideshow

```
    progress_printer = ProgressPrinter(0)
    trainer = C.Trainer(z, (ce, pe), [sgd(z.parameters, lr=lr_per_minibatch)], [progress_printer])

    # Get minibatches of training data and perform model training
    minibatch_size = 25
    num_minibatches_to_train = 1024

    aggregate_loss = 0.0
    for i in range(num_minibatches_to_train):
        train_features, labels = generate_random_data(minibatch_size, inputs, outputs)
        # Specify the mapping of input variables in the model to actual minibatch data to be trained with
        trainer.train_minibatch({features : train_features, label : labels})
        sample_count = trainer.previous_minibatch_sample_count
        aggregate_loss += trainer.previous_minibatch_loss_average * sample_count

    last_avg_error = aggregate_loss / trainer.total_number_of_samples_seen

    test_features, test_labels = generate_random_data(minibatch_size, inputs, outputs)
    avg_error = trainer.test_minibatch({features : test_features, label : test_labels})
    print(' error rate on an unseen minibatch: {}'.format(avg_error))
    return last_avg_error, avg_error
```

In [15]:
```
ffnet()
```

```
 average      since    average      since      examples
    loss       last     metric       last
 ------------------------------------------------------
Learning rate per minibatch: 0.125
    0.917      0.917       0.56       0.56            25
    0.788      0.723       0.52        0.5            75
    0.734      0.693      0.463       0.42           175
    0.651      0.579      0.395      0.335           375
    0.579      0.511      0.277      0.168           775
      0.5      0.423      0.202      0.129          1575
    0.413      0.329      0.149     0.0963          3175
     0.34      0.268      0.119     0.0903          6375
    0.285      0.229     0.0988     0.0783         12775
     0.25      0.216     0.0887     0.0786         25575
 error rate on an unseen minibatch: 0.0
```

Out[15]:
```
(0.25024763371795417, 0.0)
```

The example above sets up a 2-layer fully connected deep neural network with 50 hidden dimensions per layer. We first setup two input variables, one for the input data and one for the labels. We then called the fully connected classifier network model function which simply sets up the required weights, biases, and activation functions for each layer.

We set two root nodes in the network: ce is the cross entropy which defined our model's loss function, and pe is the classification error. We set up a trainer object with the root nodes of the network and a learner. In this case we pass in the standard SGD learner with default parameters and a learning rate of 0.02.

CNTK Toolkit – „HOL"

+GPU

Logistic Regression

A cancer hospital has provided data and wants us to determine if a patient has a fatal malignant cancer vs. a benign growth. To help classify each patient, we are given their age and the size of the tumor. Intuitively, one can imagine that younger patients and/or patient with small tumor size are less likely to have malignant cancer.

https://cntk.ai/pythondocs/CNTK_101_LogisticRegression.html

LTSM Time Series (based on SIN(x))

Time Series Analysis – trying to predict „future" value of f(x) = sin(x) without knowing what is „sinus" function

https://cntk.ai/pythondocs/CNTK_106A_LSTM_Timeseries_with_Simulated_Data.html

Logistic Regression (malignant cancer)

jupyter CNTK_101_LogisticRegression Last Checkpoint: 15 hours ago (autosaved) Azure Notebooks My Libraries 2018Athens

File Edit View Insert Cell Kernel Data Widgets Help Not Trusted Python 3.6

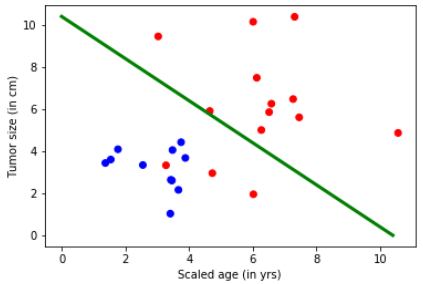
Run Code Enter/Exit RISE Slideshow

Visualization

It is desirable to visualize the results. In this example, the data is conveniently in two dimensions and can be plotted. For data with higher dimensions, visualization can be challenging. There are advanced dimensionality reduction techniques that allow for such visualizations t-sne.

In [27]:
```
# Model parameters
print(mydict['b'].value)

bias_vector   = mydict['b'].value
weight_matrix = mydict['w'].value

# Plot the data
import matplotlib.pyplot as plt

# given this is a 2 class
colors = ['r' if l == 0 else 'b' for l in labels[:,0]]
plt.scatter(features[:,0], features[:,1], c=colors)
plt.plot([0, bias_vector[0]/weight_matrix[0][1]],
         [ bias_vector[1]/weight_matrix[0][0], 0], c = 'g', lw = 3)
plt.xlabel("Scaled age (in yrs)")
plt.ylabel("Tumor size (in cm)")
plt.show()
```

```
[ 7.991415 -7.991414]
```



Exploration Suggestions

- Try exploring how the classifier behaves with different data distributions - suggest changing the `minibatch_size` parameter from 25 to say 64. Why is the error increasing?
- Try exploring different activation functions
- Try exploring different learners
- You can explore training a multiclass logistic regression classifier.

LTSM Time Series (based on SIN(x))

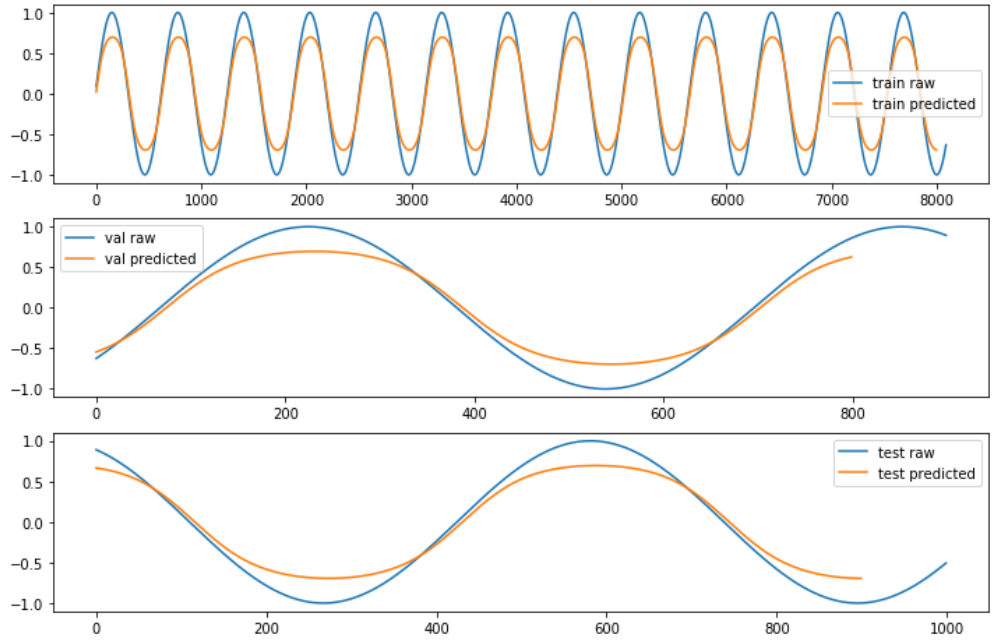
jupyter CNTK_106A_LSTM_Timeserie... Last Checkpoint: 15 hours ago (autosaved)

File Edit View Insert Cell Kernel Data Widgets Help

Run Code Enter/Exit RISE Slideshow

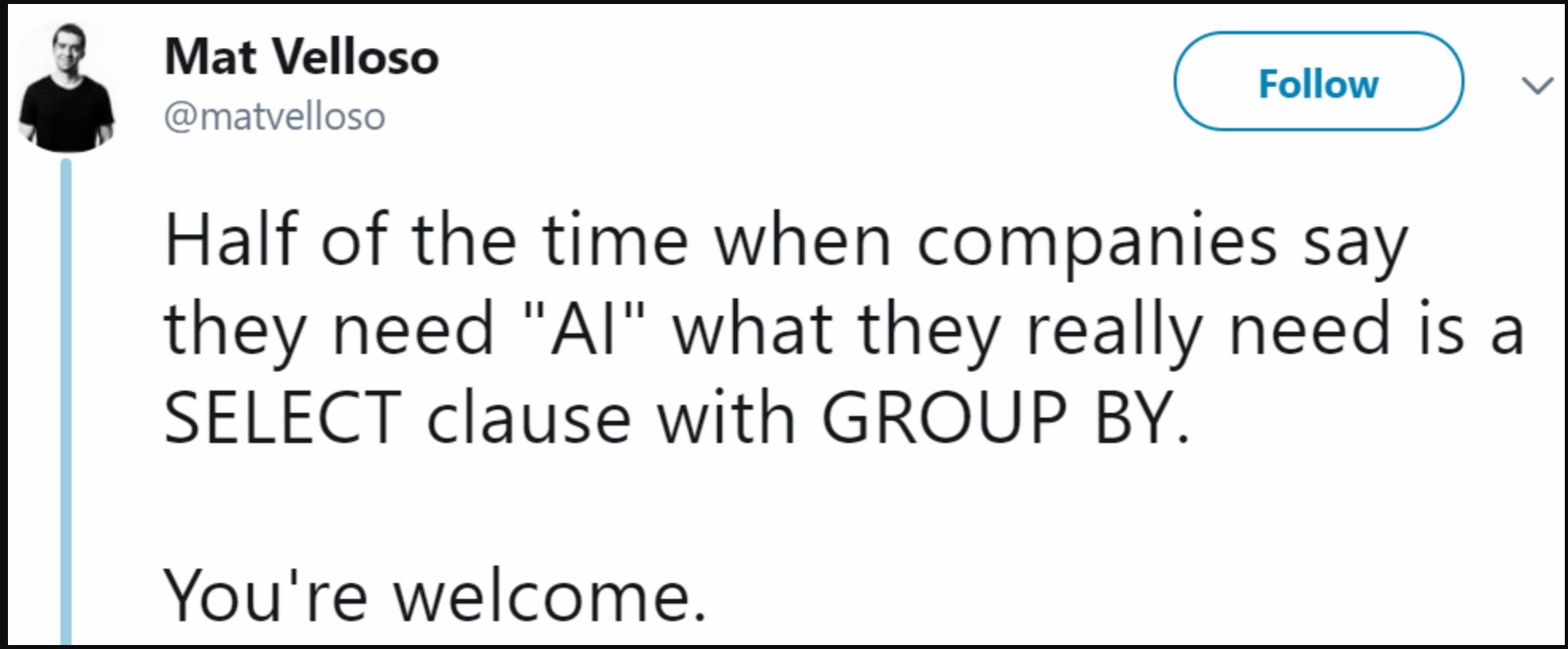
course. We also plot the expected output (Y) and the prediction our model made to shows how well the simple LSTM approach worked.

In [51]:
```
# predict
f, a = plt.subplots(3, 1, figsize = (12, 8))
for j, ds in enumerate(["train", "val", "test"]):
    results = []
    for x1, y1 in next_batch(X, Y, ds):
        pred = z.eval({x: x1})
        results.extend(pred[:, 0])
    a[j].plot(Y[ds], label = ds + ' raw');
    a[j].plot(results, label = ds + ' predicted');
[i.legend() for i in a];
```



Not perfect but close enough, considering the simplicity of the model.

And, remember



Mat Velloso: Technical Advisor, MS CTO

Demo

How to start with Machine Learning Services?

Why Python? And Code

Customers have told us they love the convenience

Customers have told us they need:

- Greater control over compute & data
- More options for model deployment
- Which frameworks? ALL OF THEM!

Top 20 Python libraries for data science in 2018

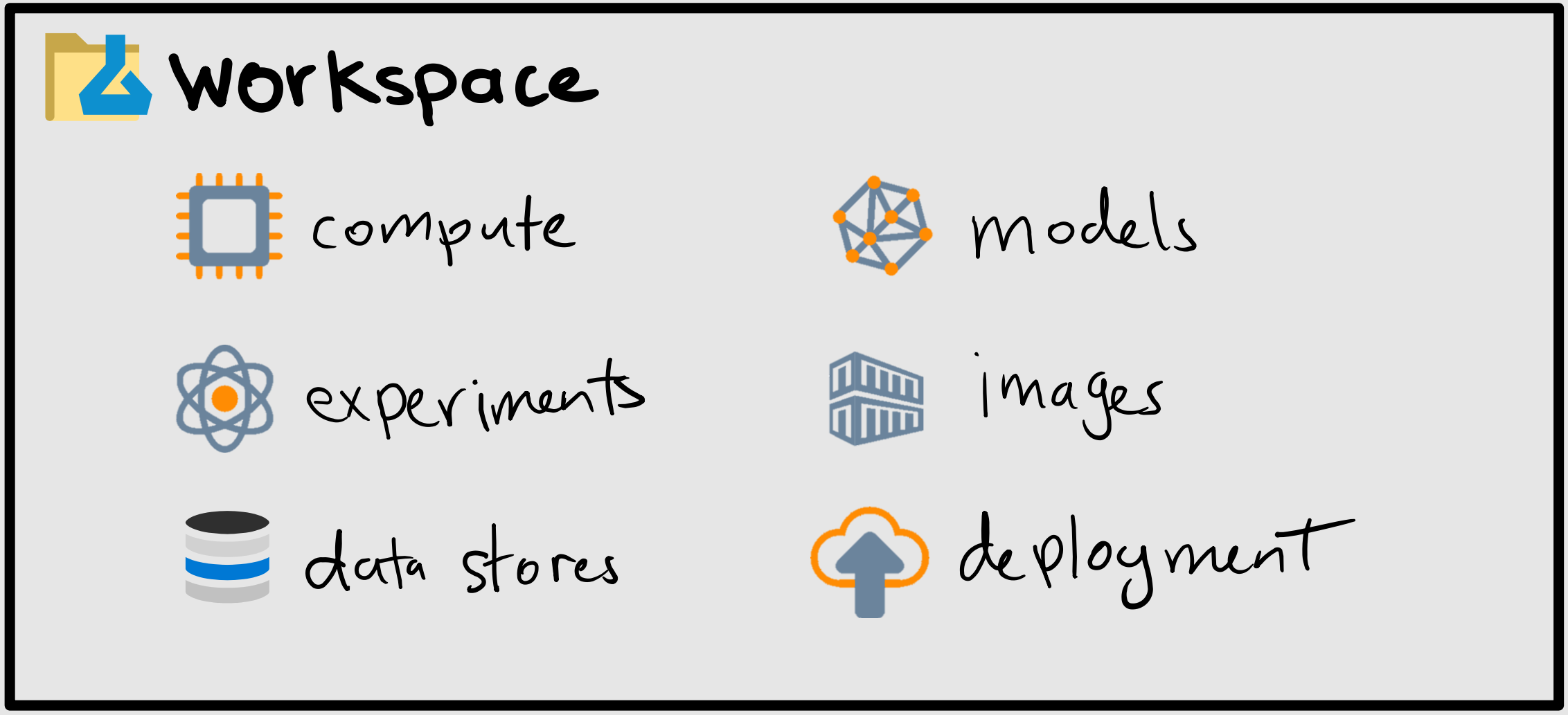
NumPy (Commits: 17911, Contributors: 641)
SciPy (Commits: 19150, Contributors: 608)
Pandas (Commits: 17144, Contributors: 1165)
StatsModels (Commits: 10067, Contributors: 153)
Matplotlib (Commits: 25747, Contributors: 725)
Seaborn (Commits: 2044, Contributors: 83)
Plotly (Commits: 2906, Contributors: 48)
Bokeh (Commits: 16983, Contributors: 294)
Pydot (Commits: 169, Contributors: 12)
Scikit-learn (Commits: 22753, Contributors: 1084)
XGBoost / LightGBM / CatBoost (Commits: 3277 / 1083 / 1509, Contributors: 280 / 79 / 61)
Eli5 (Commits: 922, Contributors: 6)
TensorFlow (Commits: 33339, Contributors: 1469)
PyTorch (Commits: 11306, Contributors: 635)
Keras (Commits: 4539, Contributors: 671)
Dist-keras / elephas / spark-deep-learning(Commits: 1125 / 170 / 67, Contributors: 5 / 13 / 11)
NLTK (Commits: 13041, Contributors: 236)
SpaCy (Commits: 8623, Contributors: 215)
Gensim (Commits: 3603, Contributors: 273)
Scrapy (Commits: 6625, Contributors: 281)

Microsoft Cognitive Toolkit (CNTK)
Caffe2

<100 more libraries ☺>

Azure Machine Learning Service

Team Workspace - Logical



Team Workspace - physical

Workspace

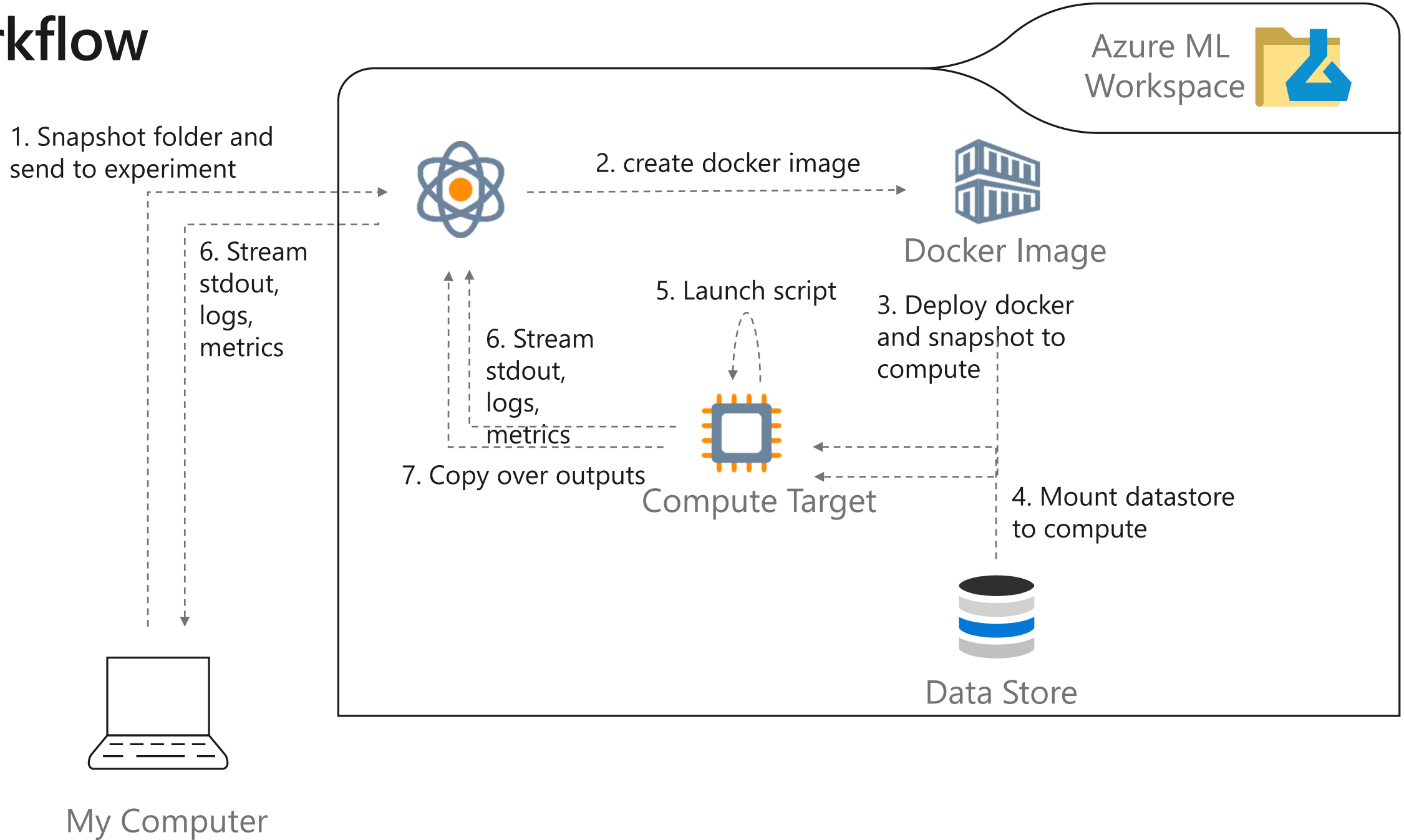
storage

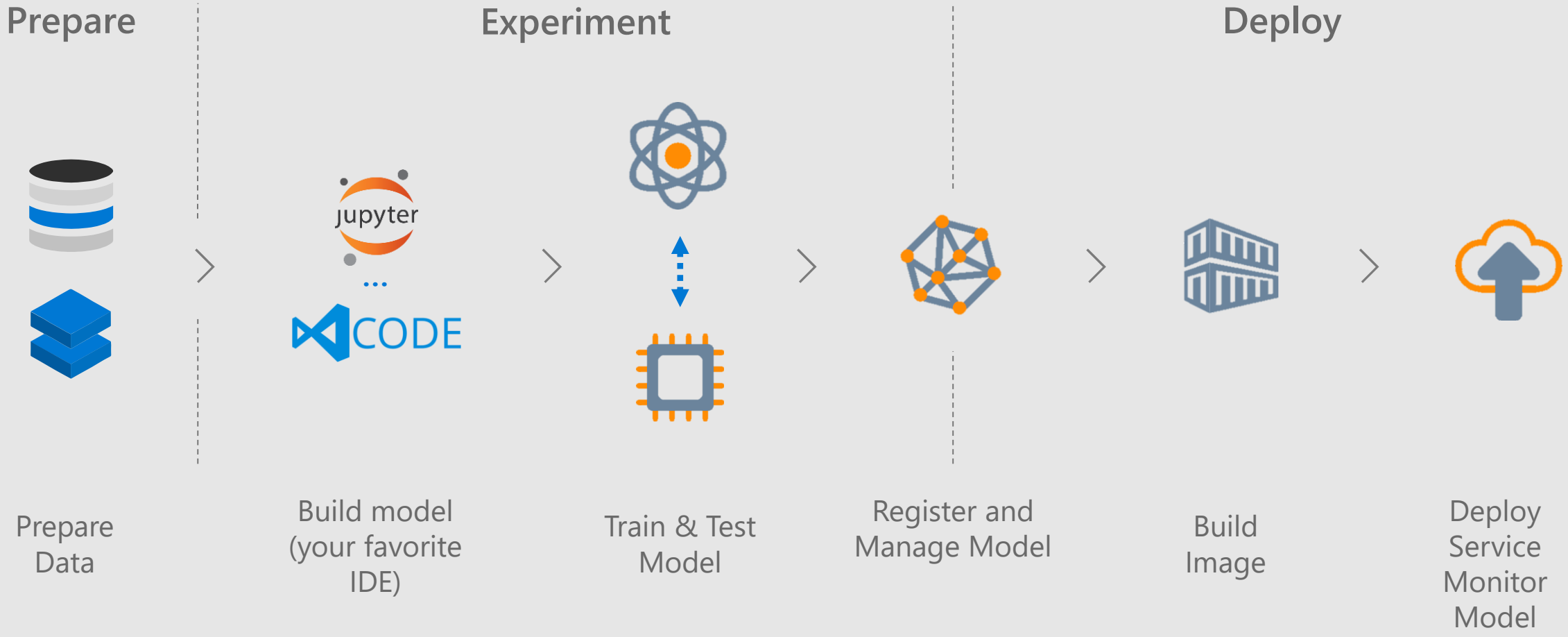
container registry

key vault

application insights

Workflow


Azure ML Workspace
1. Snapshot folder and send to experiment
2. create docker image
Docker Image
6. Stream stdout, logs, metrics
5. Launch script
3. Deploy docker and snapshot to compute
6. Stream stdout, logs, metrics
7. Copy over outputs
Compute Target
4. Mount datastore to compute
Data Store
My Computer


Prepare
Experiment
Deploy
jupyter
CODE
Prepare Data
Build model (your favorite IDE)
Train & Test Model
Register and Manage Model
Build Image
Deploy Service Monitor Model

Types of Compute Target

Virtual Machine

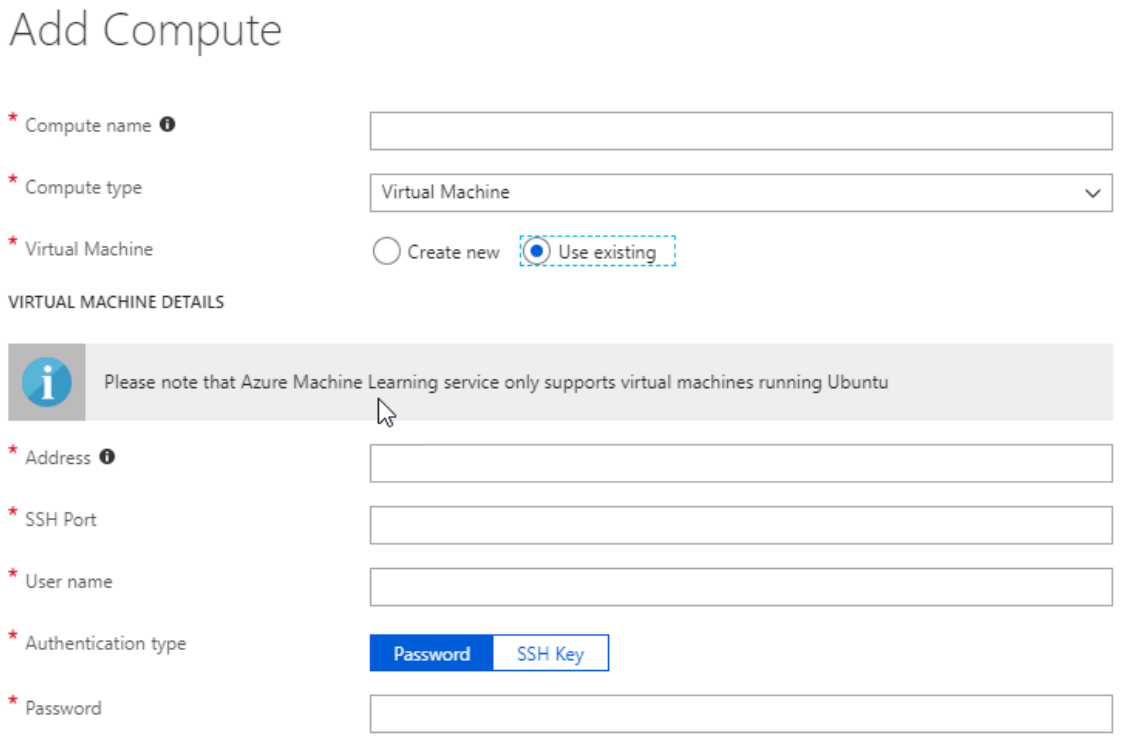
- New in Azure | Existing in Azure
- Anywhere (on prem as well!)

Azure Batch AI

- Best choice regarding price/performance
- + Low priority instances

Azure Kubernetes Service (AKS)

- Maybe for CPU bound? Many parallel calculations? Existing „free” cluster?



VM on Azure

Type	Sizes	Description
General purpose	B, Dsv3, Dv3, DSv2, Dv2, Av2, DC	Balanced CPU-to-memory ratio. Ideal for testing and development, small to medium databases, and low to medium traffic web servers.
Compute optimized	Fsv2, Fs, F	High CPU-to-memory ratio. Good for medium traffic web servers, network appliances, batch processes, and application servers.
Memory optimized	Esv3, Ev3, M, GS, G, DSv2, Dv2	High memory-to-CPU ratio. Great for relational database servers, medium to large caches, and in-memory analytics.
Storage optimized	Ls	High disk throughput and IO. Ideal for Big Data, SQL, and NoSQL databases.
GPU	NV, NVv2, NC, NCv2, NCv3, ND	Specialized virtual machines targeted for heavy graphic rendering and video editing, as well as model training and inferencing (ND) with deep learning. Available with single or multiple GPUs.
High performance compute	H	Our fastest and most powerful CPU virtual machines with optional high-throughput network interfaces (RDMA).

NC_v2 – Pascal Generation GPU

- Pascal generation GPU instances - NVIDIA Tesla P100 GPUs
- Targeted for accelerating machine training jobs and HPC
- Specs:
  - FP64 – 4.7 TFLOPS of double precision floating point performance
  - FP32 – 9.3 TFLOPS of single precision performance
  - FP16 – 18.7 TFLOPS of half-precision performance
  - GPU Memory 16 GB

	NC6s_v2	NC12s_v2	NC24s_v2	NC24rs_v2
Cores (Broadwell 2.6Ghz)	6	12	24	24
GPU	1 x P100	2 x P100	4 x P100	4 x P100
Memory	112 GB	224 GB	448 GB	448 GB
Local Disk	~700 GB SSD	~1.4 TB SSD	~3 TB SSD	~3 TB SSD
Network	Azure Network	Azure Network	Azure Network	Azure Network + InfiniBand

Batch Services

Azure Batch Services

- Cloud-scale job scheduling and compute management
- Linux or Windows, any VM size
- BatchLab – alternative UI to manage

Flow:

- Upload data files and application files
- Create Pool of compute nodes
- Create Job
- Add task(s) to Job, pointing into
- Start computation (send job)

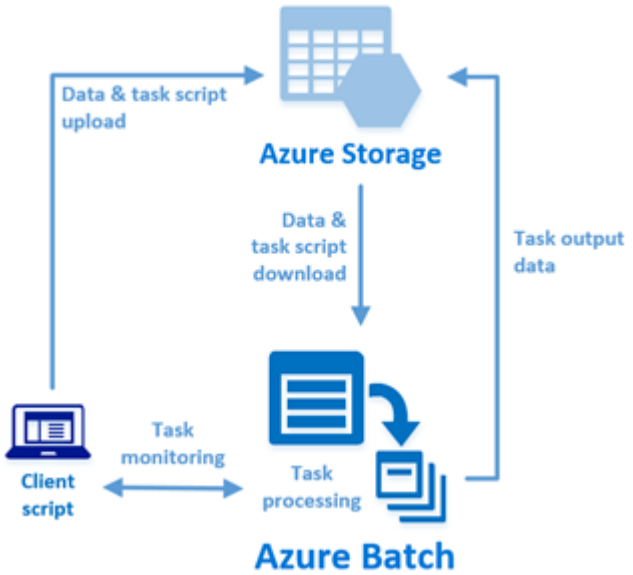
Two general types of task

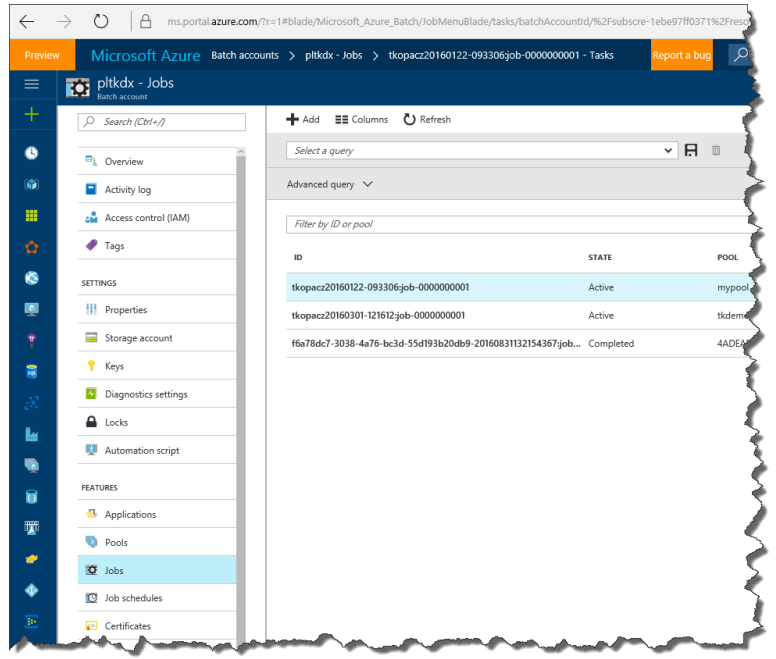
- Parameter sweep (for each row, for set of number)
- Message Passing Interface – HPC style, multi-instance task

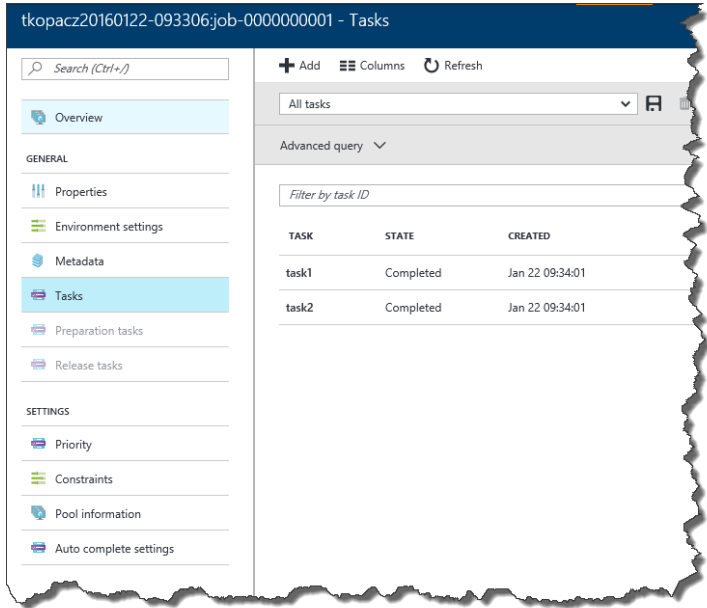
Work with clusters of GPUs

- Autodesk, Chaos Group V-Ray, custom

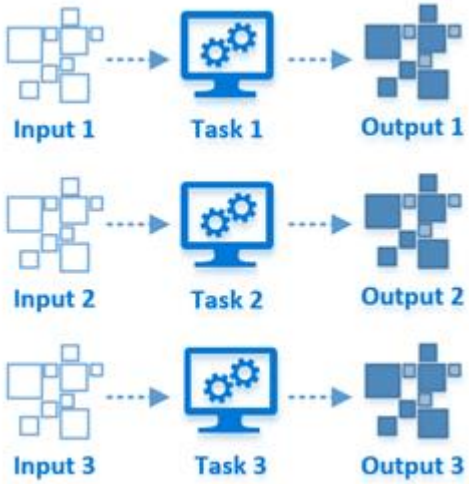
Shipyard (containers; see also later)







Azure Batch AI



To sum up: Batch Processing

Pool – Job – Task (MPI, dependences etc)

Batch AI – specialized service

- Fire and forget jobs, auto-scaling clusters, resource sharing
  - Batch or interactive use, VM images, containers, or setup script
- Framework-specific configuration
  - Stream output and return logs, Export trained models and configuration
- Azure Files, Blob FUSE, NFS, or parallel file system
  - Lustre, Gluster, BeeGFS, …
- Azure CLI, Python, C#, Java, and REST API's, RBAC
- Support for any Deep Learning or machine learning framework: Microsoft Cognitive Toolkit (CNTK), TensorFlow, and Chainer

Standard and low priority VMs

Train model

Run (ACI, Batch AI)

```
from azureml.core.runconfig import RunConfiguration
from azureml.core.conda_dependencies import CondaDependencies

# create a new runconfig object
run_config = RunConfiguration()

# signal that you want to use ACI to execute script.
run_config.target = "containerinstance"

# ACI container group is only supported in certain regions, which can be different t
run_config.container_instance.region = 'eastus2'

# set the ACI CPU and Memory
run_config.container_instance.cpu_cores = 1
run_config.container_instance.memory_gb = 2

# enable Docker
run_config.environment.docker.enabled = True

# set Docker base image to the default CPU-based image
run_config.environment.docker.base_image = azureml.core.runconfig.DEFAULT_CPU_IMAGE

# use conda_dependencies.yml to create a conda environment in the Docker image for e
run_config.environment.python.user_managed_dependencies = False

# auto-prepare the Docker image when used for execution (if it is not already prepar
run_config.auto_prepare_environment = True

# specify CondaDependencies obj
run_config.environment.python.conda_dependencies = CondaDependencies.create(conda_pa
```

```
compute_config = BatchAiCompute.provisioning_configuration(vm_size="STANDARD_NC6", # GPU-based VM
                                                           #vm_priority='lowpriority', # optional
                                                           autoscale_enabled=True,
                                                           cluster_min_nodes=0,
                                                           cluster_max_nodes=4)

# create the cluster
compute_target = ComputeTarget.create(ws, batchai_cluster_name, compute_config)
```

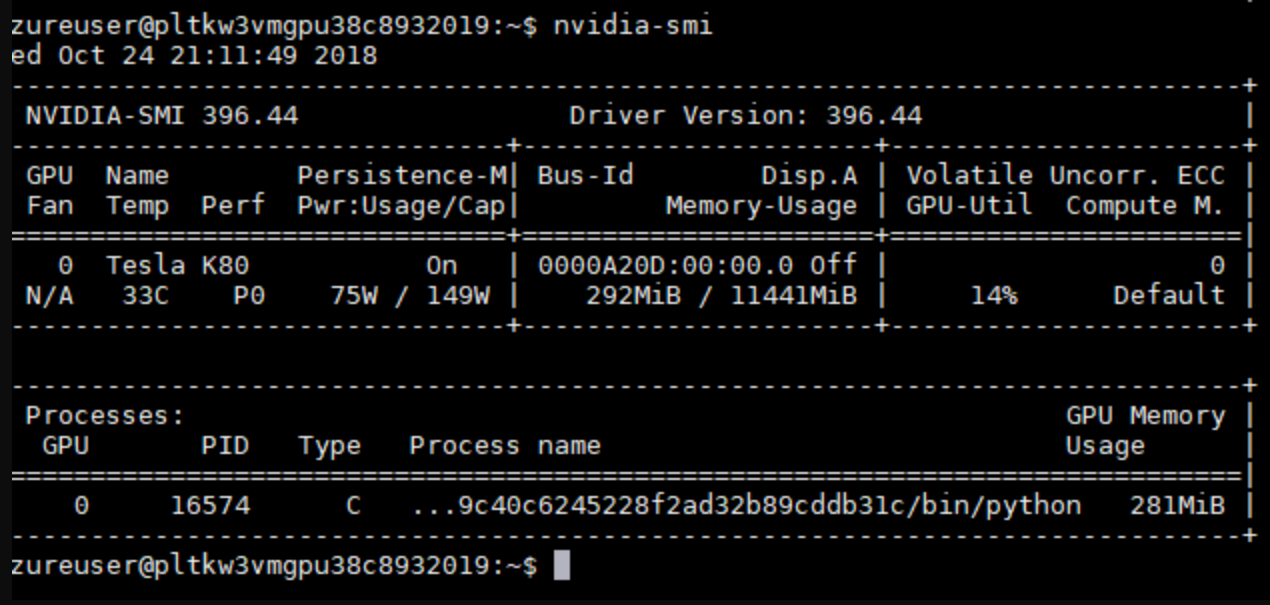
```
%%time
from azureml.core.script_run_config import ScriptRunConfig

script_run_config = ScriptRunConfig(source_directory='./',
                                    script='train.py',
                                    run_config=run_config)

run = experiment.submit(script_run_config)
```

```
CPU times: user 344 ms, sys: 109 ms, total: 453 ms
Wall time: 5.2 s
```

```
# Show run details
run
```

```
%%time
# Shows output of the run on stdout.
run.wait_for_completion(show_output=True)
```

```
libstdcxx-ng-8.2.0   | 2.9 MB    | ########## | 100%

libedit-3.1          | 171 KB    |            |   0%
libedit-3.1          | 171 KB    | ########## | 100%

mkl_fft-1.0.6        | 150 KB    |            |   0%
mkl_fft-1.0.6        | 150 KB    | ########## | 100%
Downloading and Extracting Packages
Preparing transaction: ...working... done
Verifying transaction: ...working... done
Executing transaction: ...working... done

Collecting azureml-defaults==0.1.68 (from -r /azureml-setup/c
  Downloading https://files.pythonhosted.org/packages/46/ad/6
azureml_defaults-0.1.68-py2.py3-none-any.whl
Collecting azureml-core==0.1.68.* (from azureml-defaults==0.1
ine 1))
```



Demo

Training on:
Local – end 2 end
VM z ACI
VM z GPU / Tensorboard
Batch AI

Register Model

register model

```
# register model from the best run
model = best_run.register_model(model_name = 'mnist_tf_model', model_path = 'outputs/model')
```

Note the registered model automatically gets an auto-increasing version number.

```
print(model.name, model.id, model.version, sep = '\t')
```

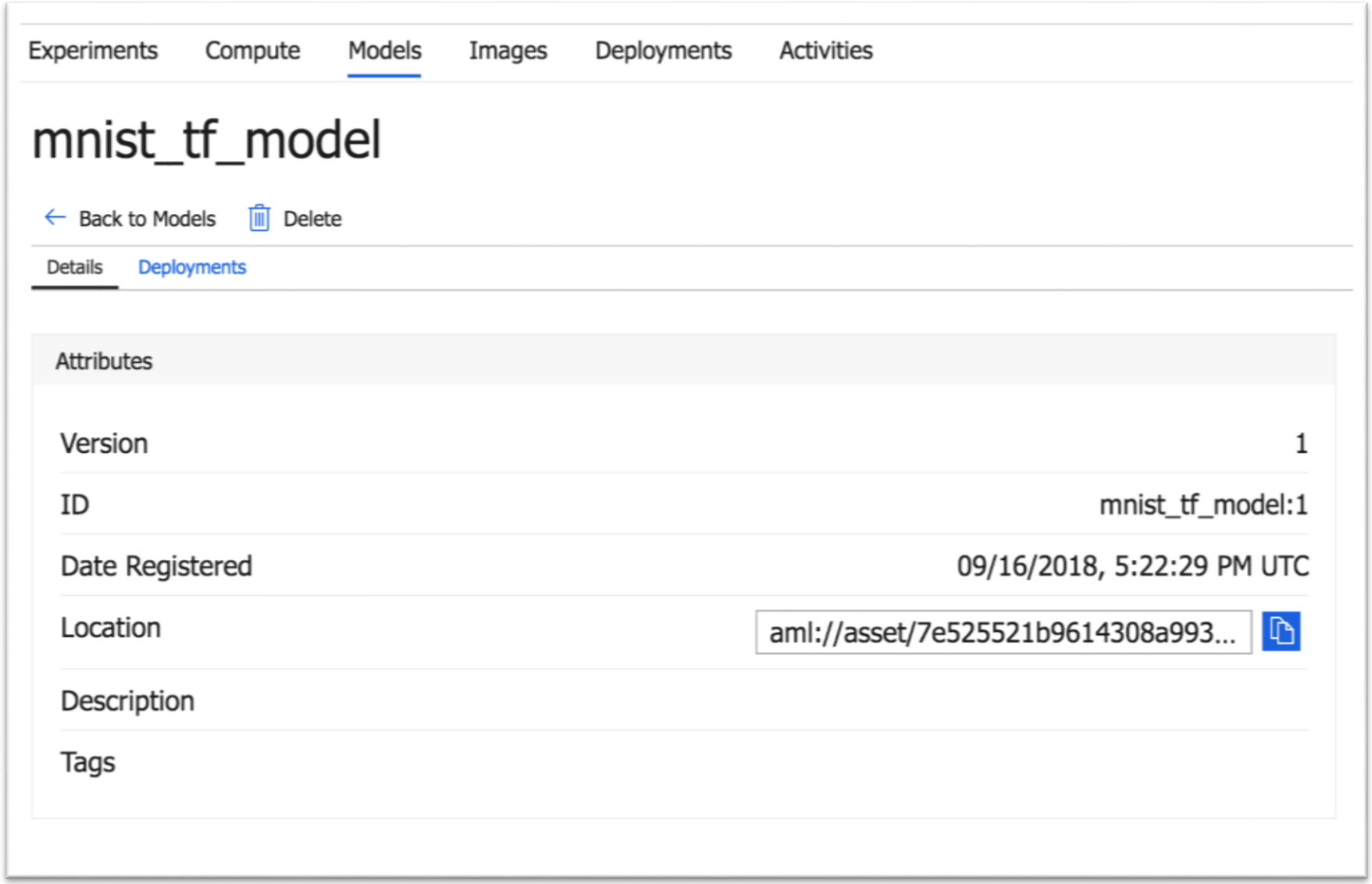
```
mnist_tf_model  mnist_tf_model:2        2
```

Download the `model` folder locally and inspect the files in it.

```
import os
model.download('./')
os.listdir('./model/')
```

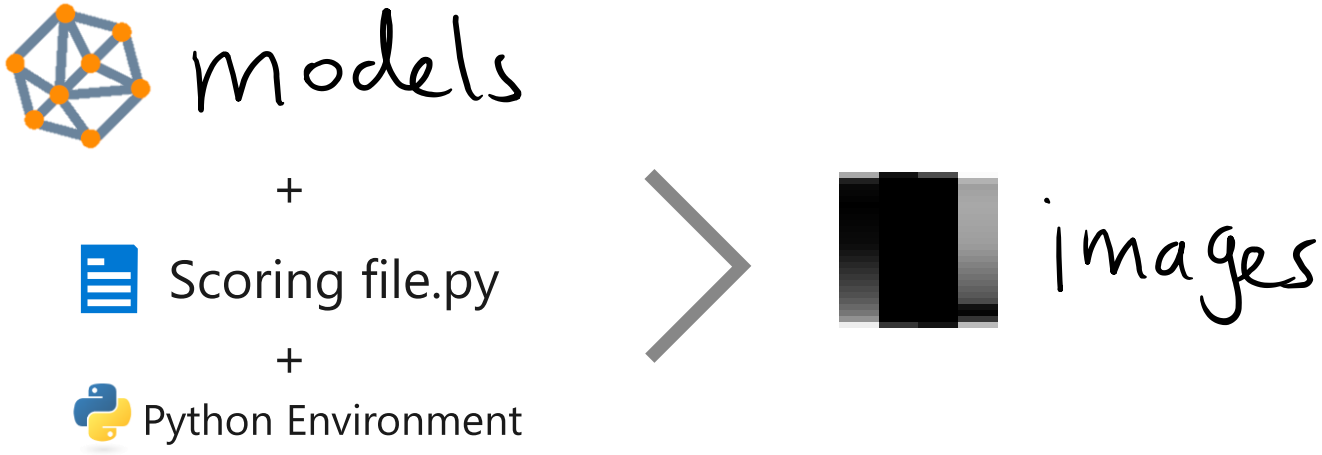
```
['mnist-tf.model.meta',
 'checkpoint',
 'mnist-tf.model.index',
 'mnist-tf.model.data-00000-of-00001']
```

register model



Create Image

Create images



Scoring File

```python
import json
import numpy as np
import os
import tensorflow as tf

#from azureml.assets.persistence.persistence import get_model_path
from azureml.core.model import Model

def init():
    global X, output, sess
    tf.reset_default_graph()
    # retreive the local path to the model using the model name
    model_root = Model.get_model_path('mnist_tf_model')
    saver = tf.train.import_meta_graph(os.path.join(model_root, 'mnist-tf.model.meta'))
    X = tf.get_default_graph().get_tensor_by_name("network/X:0")
    output = tf.get_default_graph().get_tensor_by_name("network/output/MatMul:0")

    sess = tf.Session()
    saver.restore(sess, os.path.join(model_root, 'mnist-tf.model'))

def run(raw_data):
    data = np.array(json.loads(raw_data)['data'])
    # make prediction
    out = output.eval(session = sess, feed_dict = {X: data})
    y_hat = np.argmax(out, axis = 1)
    return json.dumps(y_hat.tolist())
```

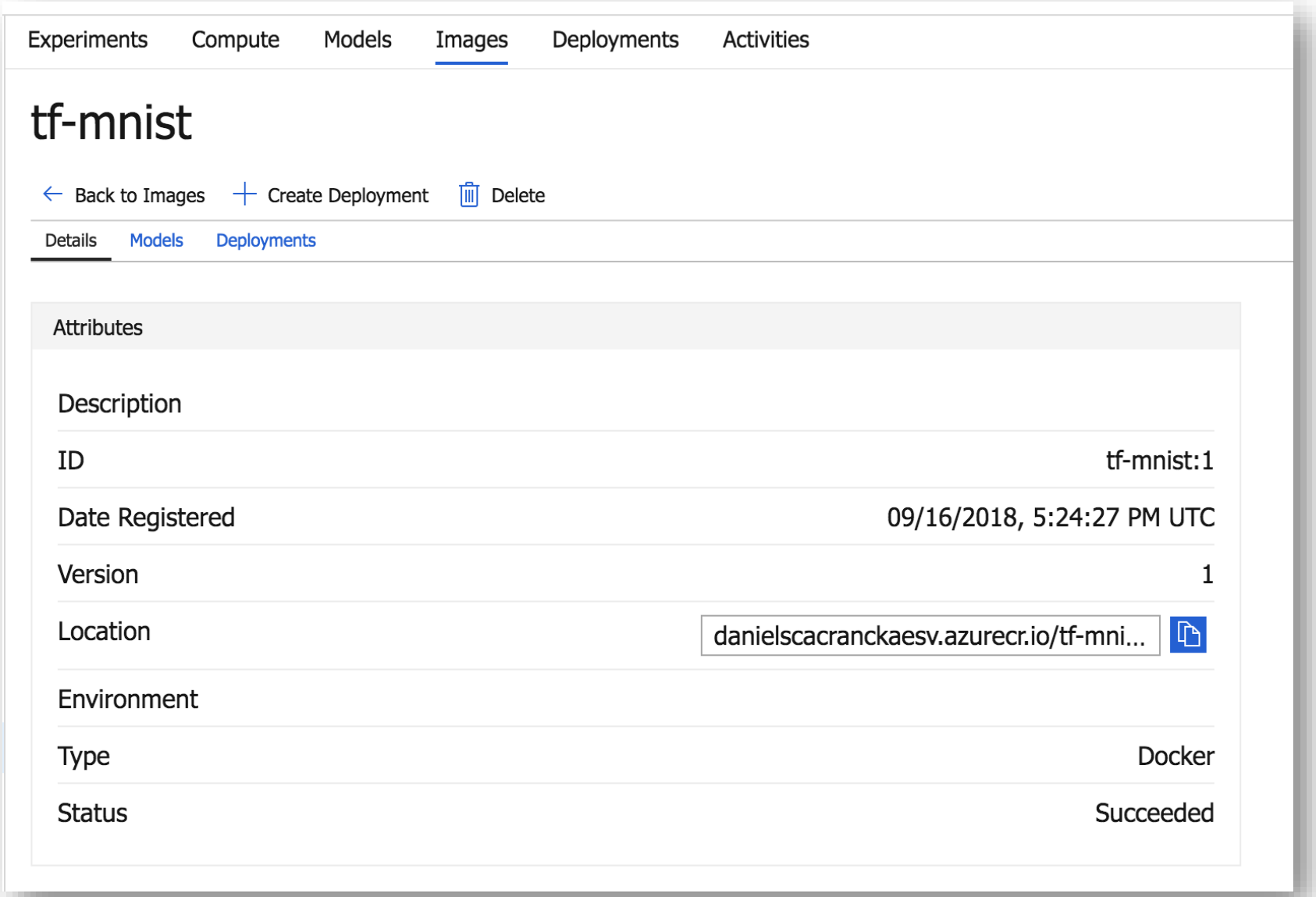
Environment File

```
# Conda environment specification. The dependencies defined in this file will
# be automatically provisioned for runs with userManagedDependencies=False.

# Details about the Conda environment file format:
# https://conda.io/docs/user-guide/tasks/manage-environments.html#create-env-file-manually

name: project_environment
dependencies:
  # The python interpreter version.
  # Currently Azure ML only supports 3.5.2 and later.
- python=3.6.2

- pip:
    # Required packages for AzureML execution, history, and data preparation.
  - azureml-defaults
  - tensorflow==1.9.0
```

Create the Image

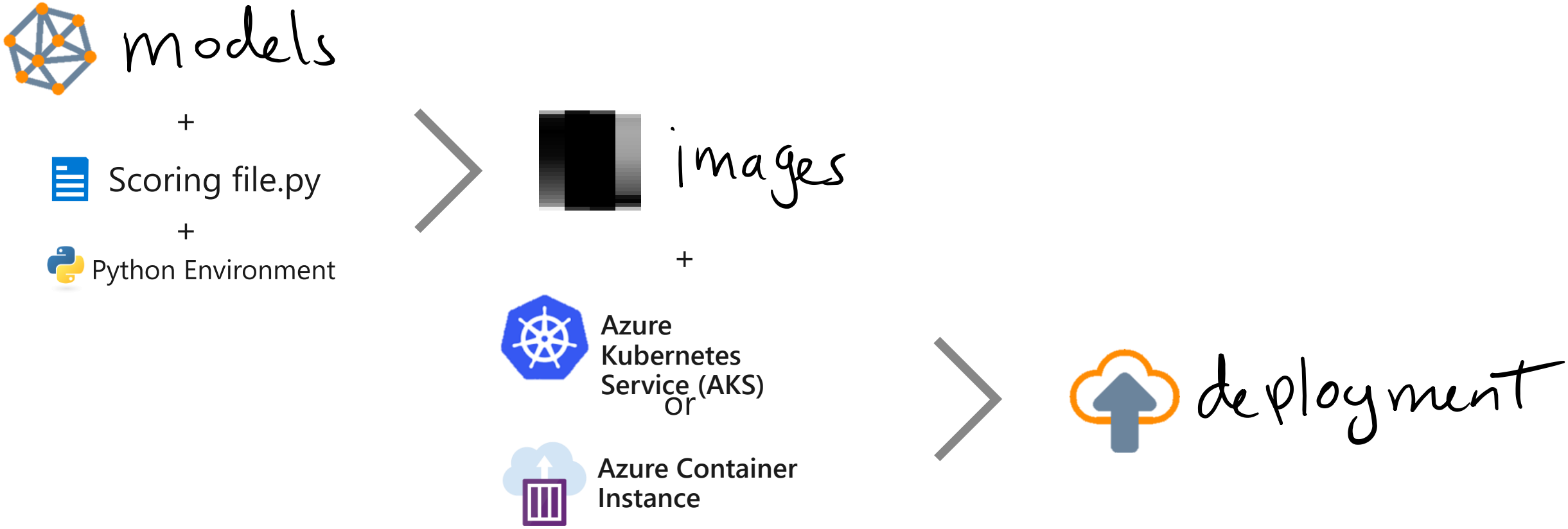
```
from azureml.core.image import Image, ContainerImage

image_config = ContainerImage.image_configuration(runtime= "python",
                                 execution_script="score.py",
                                 conda_file="myenv.yml")

image = Image.create(name = "mnist_image",
                     models = [model],                    # this is the model object
                     image_config = image_config,
                     workspace = ws)
```

Images


Experiments
Compute
Models
Images
Deployments
Activities
tf-mnist
Back to Images
Create Deployment
Delete
Details
Models
Deployments
Attributes
Description
ID
tf-mnist:1
Date Registered
09/16/2018, 5:24:27 PM UTC
Version
1
Location
danielscacranckaesv.azurecr.io/tf-mni...
Environment
Type
Docker
Status
Succeeded

Deploy image

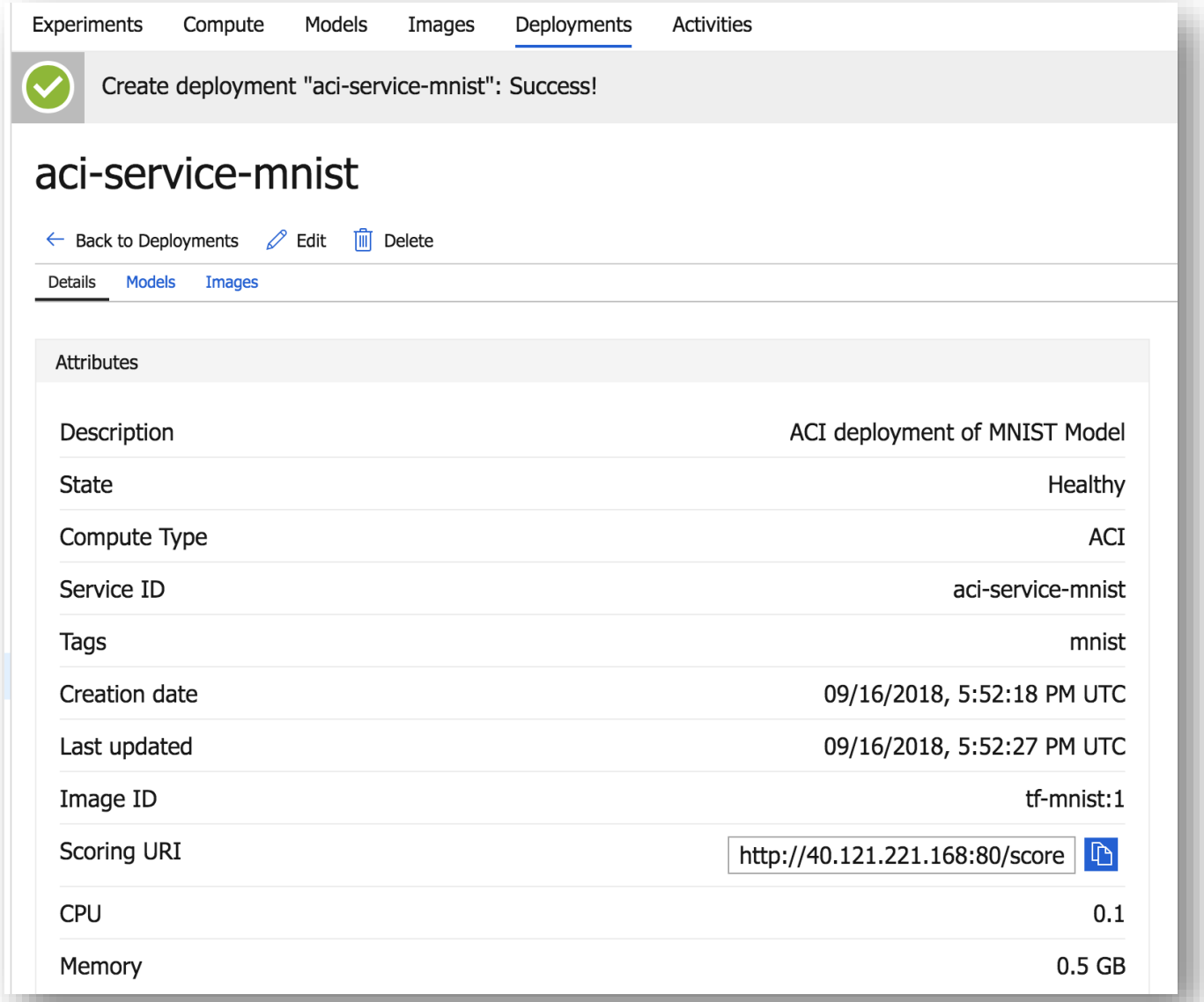
Deploy image



Deploy image

```
from azureml.core.webservice import Webservice
from  azureml.core.webservice import AciWebservice

aciconfig = AciWebservice.deploy_configuration(cpu_cores = 1,
                                               memory_gb = 2)

aci_service = Webservice.deploy_from_image(deployment_config = aciconfig,
                                           image = image,
                                           name = 'aci_service_name',
                                           workspace = ws)
```

Deploy image



Deploy image

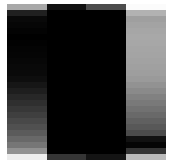
models

+

Scoring file.py

+

Python Environment



images

+

Azure Kubernetes Service (AKS)

or

Azure Container Instance

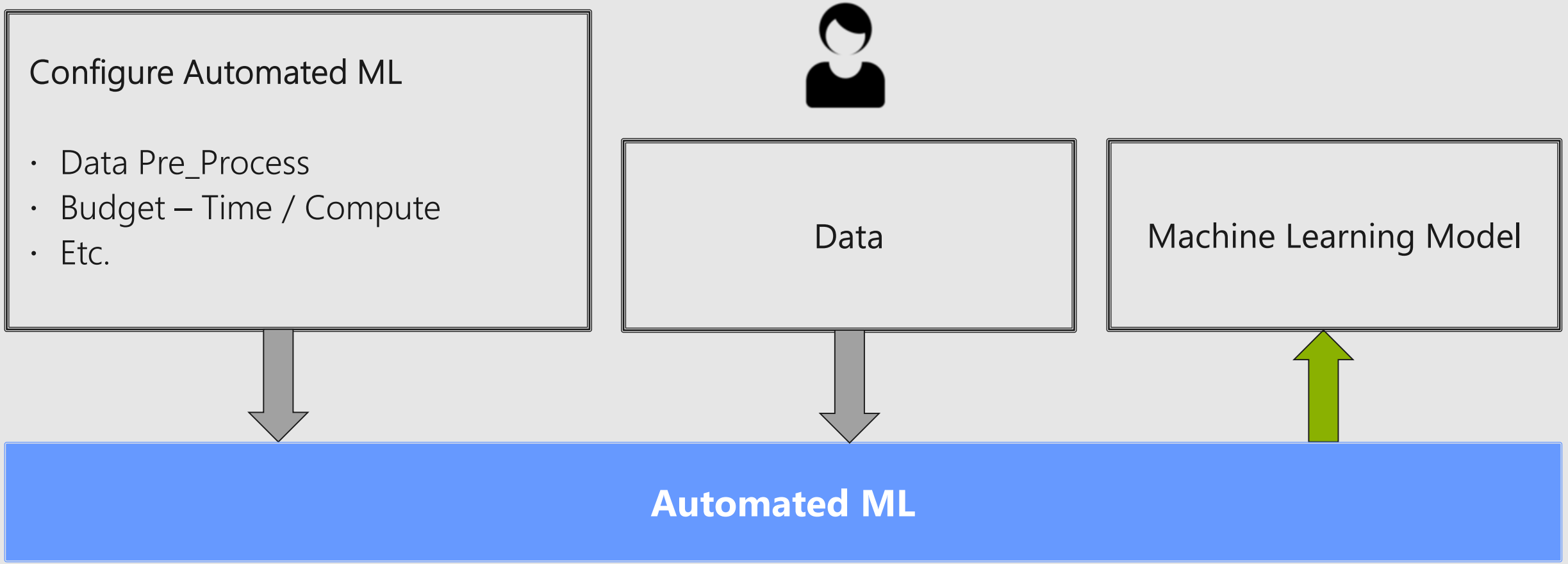
deployment

Demo

Model Deployment

Hyperparameter tuning

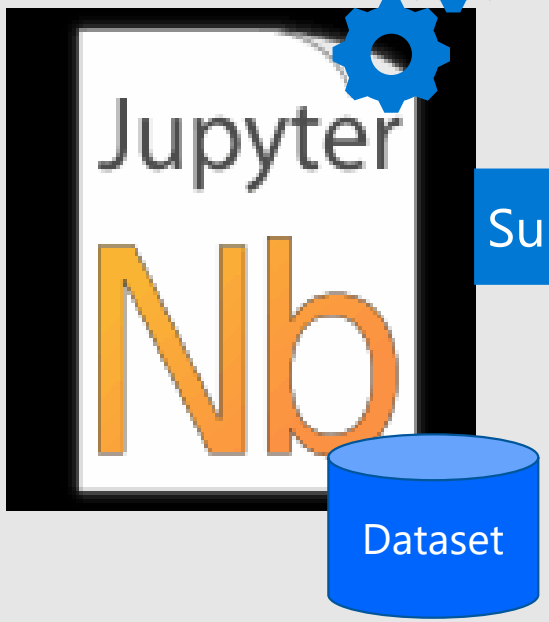
User inputs & system outputs



Automated ML – User Experience

Python Script

Configuration



Submit

Automated ML

Generate Algorithms & Hyperparameter values

Train

Models (User Compute – Local or Cloud)

Iteration	Pipeline	Iteration metric	Best metric
0	SparseNormalizer, LogisticRegression	0.99755788	0.99755788
1	StandardScalerWrapper, KNeighborsClassifier	0.99788041	0.99788041
2	MaxAbsScaler, LightGBMClassifier	0.99827043	0.99827043
3	MaxAbsScaler, DecisionTreeClassifier	0.8321242	0.99827043
4	SparseNormalizer, LightGBMClassifier	0.99794397	0.99827043
5	StandardScalerWrapper, KNeighborsClassifier	0.9983558	0.9983558
6	StandardScalerWrapper, LightGBMClassifier	0.99883517	0.99883517
7	StandardScalerWrapper, SGDClassifierWrapper	0.99455258	0.99883517
8	MaxAbsScaler, LightGBMClassifier	0.99753115	0.99883517
9	StandardScalerWrapper, KNeighborsClassifier	0.99863863	0.99883517

Pages: 1 2 3 4 Next Last 10 per page

Output

High Quality Machine Learning Model

Easy to Use

jupyter 102.auto-ml-regression (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Not Trusted | Python [default]

Run Code

Instantiate Auto ML Regressor

Instantiate a AutoML Object This creates an Experiment in Azure ML. You can reuse this objects to trigger multiple runs. Each run will be part of the same experiment.

Property	Description
primary_metric	This is the metric that you want to optimize. Auto ML Regressor supports the following primary metrics *spearman_correlation* *normalized_root_mean_squared_error* *r2_score*
max_time_sec	Time limit in seconds for each iterations
iterations	Number of iterations. In each iteration Auto ML Classifier trains the data with a specific pipeline
num_cross_folds	Cross Validation split

In [5]:
```
from azureml.train.automl import AutoMLConfig

automl_config = AutoMLConfig(task = 'regression',
                             debug_log = 'automl_errors.log',
                             primary_metric = 'spearman_correlation',
                             max_time_sec = 12000,
                             iterations = 10,
                             n_cross_validations = 3,
                             verbosity = logging.INFO,
                             X = X,
                             y = y,
                             path=project_folder)
```

Training the Model

You can call the fit method on the AutoML instance and pass the run configuration. For Local runs the execution is synchronous. Depending on the data and number of iterations this can run for while. You will see the currently running iterations printing to the console.

fit method on Auto ML Regressor triggers the training of the model. It can be called with the following parameters

Parameter	Description
X	(sparse) array-like, shape = [n_samples, n_features]
y	(sparse) array-like, shape = [n_samples,], [n_samples, n_classes] Multi-class targets. An indicator matrix turns on multilabel classification.
compute_target	Indicates the compute used for training. *local* indicates train on the same compute which hosts the jupyter notebook. For DSVM and Batch AI please refer to the relevant notebooks.
show_output	True/False to turn on/off console output

In [6]:
```
local_run = experiment.submit(automl_config, show_output=True)
```

```
Parent Run ID: AutoML_e7a4236e-8935-4e93-888d-1ea8310a6b22
********************************************************************************************
ITERATION: The iteration being evaluated.
PIPELINE: A summary description of the pipeline being evaluated.
DURATION: Time taken for the current iteration.
METRIC: The result of computing score on the fitted pipeline.
BEST: The best observed score thus far.
********************************************************************************************

 ITERATION     PIPELINE                                       DURATION               METRIC      BEST
         0      Normalize extra trees regressor              0:00:12.069893           0.688     0.688
         1      Normalize lightGBM regressor                 0:00:11.192919           0.597     0.688
         2      Normalize Elastic net                        0:00:09.866233           0.689     0.689
         3      Scale 0/1 lightGBM regressor                 0:00:10.069764           0.656     0.689
         4      Robust Scaler kNN regressor                  0:00:09.090668           0.598     0.689
         5      Normalize lightGBM regressor                 0:00:12.562876           0.649     0.689
         6      Robust Scaler kNN regressor                  0:00:09.361137           0.600     0.689
         7      Normalize SGD regressor                      0:00:09.010672           0.070     0.689
         8      Scale 0/1 extra trees regressor              0:00:10.442752           0.685     0.689
         9      Robust Scaler Gradient boosting regres0:00:09.567582                  0.651     0.689
```

Demo

AutoML on remote VM

Demo

Hyperparameter tuning