Machine Learning and AI

UNIVERSITY OF NORTH CAROLINA CHARLOTTE

DSBA 6190-U90 | Colby T. Ford, Ph.D.

Overview

Machine Learning Options in the Cloud

Azure Machine Learning Studio

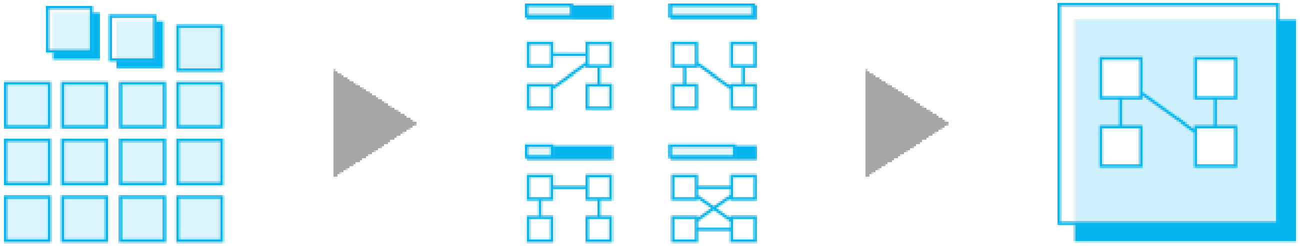
- AutoML

Azure AI Studio

- Open-Source Models vs. OpenAI

Cognitive Services

Intro to Machine Learning



Prepare Data

Connect to various sources to ingest data

Build & Train

Train with the data to establish a model

Deploy

Deploy the model and track performance

Machine Learning and AI Options in the Azure Cloud

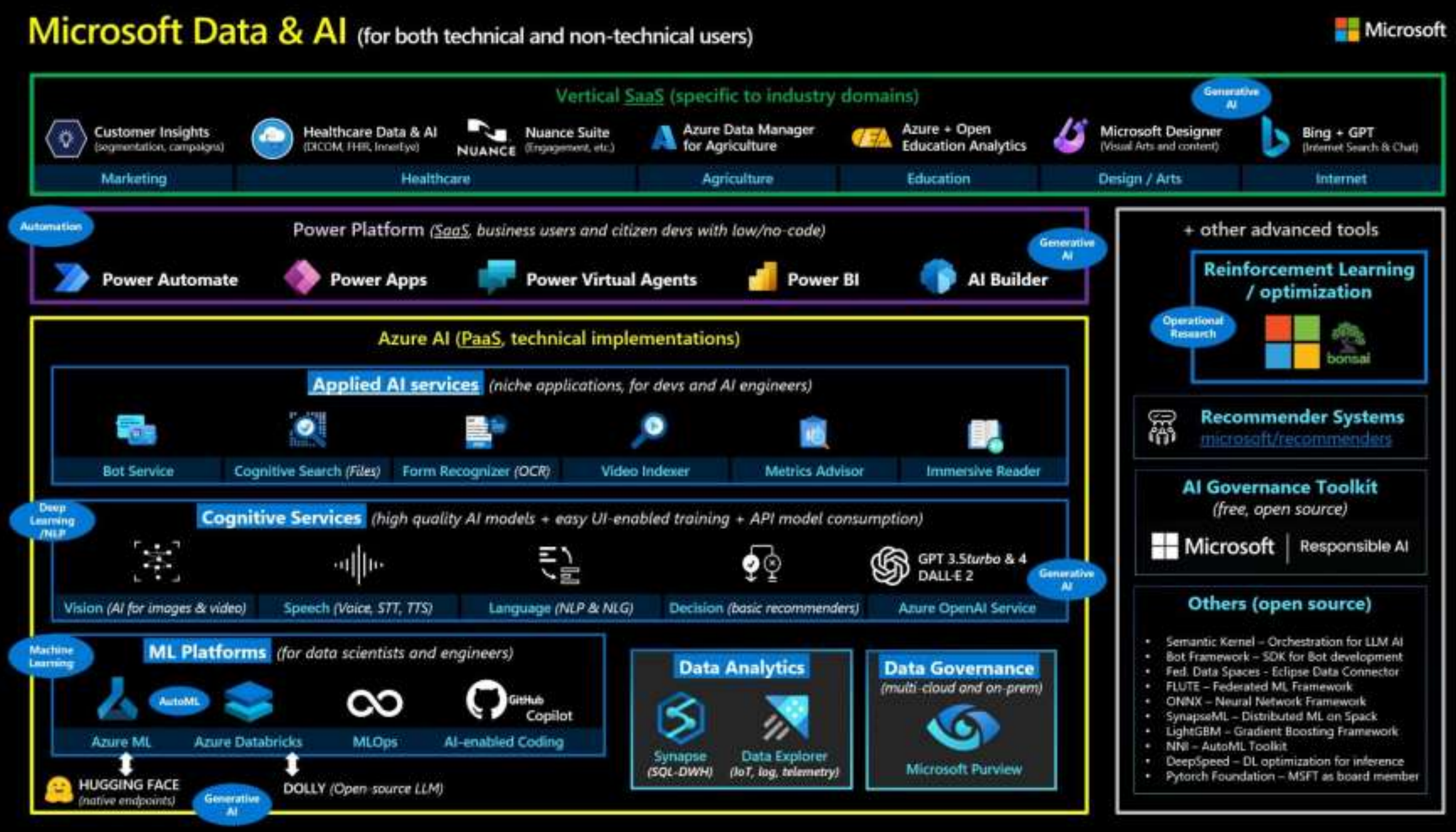
DSVM

Azure AI Services / Azure AI Studio

Databricks

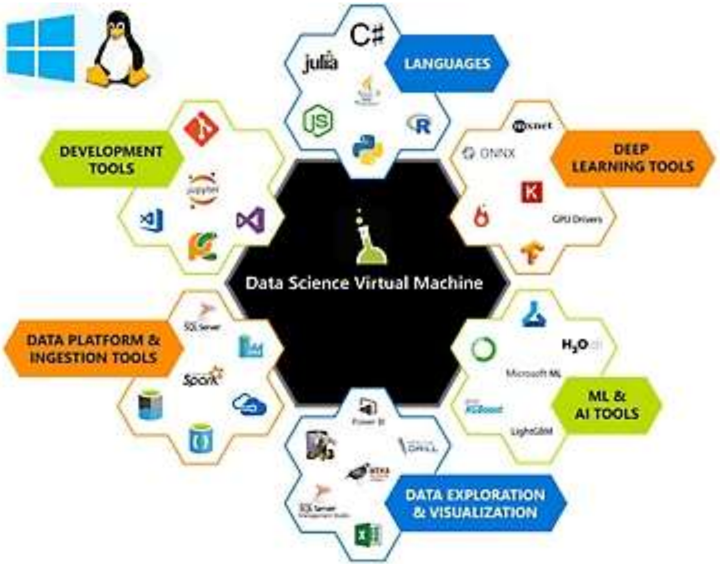
Machine Learning Studio

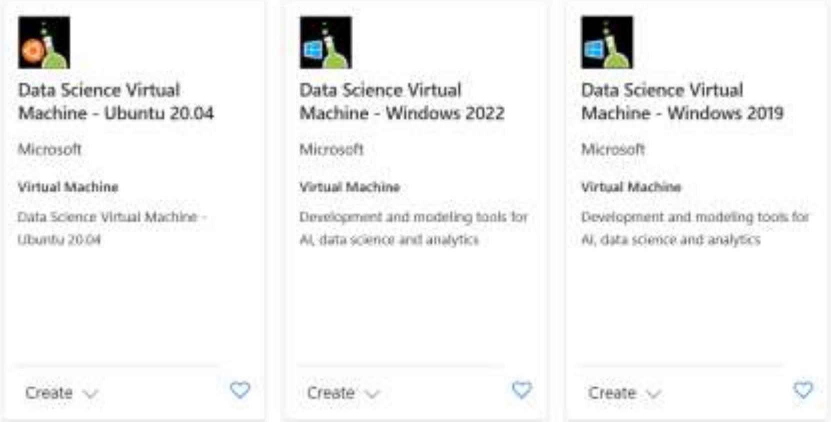
AI Studio


Microsoft Data & AI (for both technical and non-technical users)
Microsoft
Vertical SaaS (specific to industry domains)
Generative AI
Customer Insights (segmentation, campaigns)
Healthcare Data & AI (DICOM, FHIR, InnerEye)
NUANCE
Nuance Suite (Engagement, etc.)
Azure Data Manager for Agriculture
Azure + Open Education Analytics
Microsoft Designer (Visual Arts and content)
Bing + GPT (Internet Search & Chat)
Marketing
Healthcare
Agriculture
Education
Design / Arts
Internet
Automation
Power Platform (SaaS, business users and citizen devs with low/no-code)
Generative AI
Power Automate
Power Apps
Power Virtual Agents
Power BI
AI Builder
+ other advanced tools
Reinforcement Learning / optimization
Operational Research
bonsai
Azure AI (PaaS, technical implementations)
Applied AI services (niche applications, for devs and AI engineers)
Bot Service
Cognitive Search (Files)
Form Recognizer (OCR)
Video Indexer
Metrics Advisor
Immersive Reader
Recommender Systems
microsoft/recommenders
Deep Learning /NLP
Cognitive Services (high quality AI models + easy UI-enabled training + API model consumption)
GPT 3.5turbo & 4
DALL-E 2
Generative AI
Vision (AI for images & video)
Speech (Voice, STT, TTS)
Language (NLP & NLG)
Decision (basic recommenders)
Azure OpenAI Service
AI Governance Toolkit
(free, open source)
Microsoft | Responsible AI
Machine Learning
ML Platforms (for data scientists and engineers)
AutoML
GitHub Copilot
Azure ML
Azure Databricks
MLOps
AI-enabled Coding
Data Analytics
Synapse (SQL-DWH)
Data Explorer (IoT, log, telemetry)
Data Governance (multi-cloud and on-prem)
Microsoft Purview
HUGGING FACE (native endpoints)
Generative AI
DOLLY (Open-source LLM)
Others (open source)
Semantic Kernel – Orchestration for LLM AI
Bot Framework – SDK for Bot development
Fed. Data Spaces – Eclipse Data Connector
FLUTE – Federated ML Framework
ONNX – Neural Network Framework
SynapseML – Distributed ML on Spark
LightGBM – Gradient Boosting Framework
NNI – AutoML Toolkit
DeepSpeed – DL optimization for inference
Pytorch Foundation – MSFT as board member

Data Science Virtual Machine

- A virtual machine that comes with lots of data science-relevant software preinstalled.
  - Programming languages: Python, R, Julia
  - IDEs: Jupyter, RStudio, VS Code
  - ML/DL frameworks: Scikit-Learn, PyTorch, Tensorflow
  - Other software: Azure Storage Explorer, Azure CLI, Database drivers, Office, Power BI, Git





Azure AI Services

Formerly called “Cognitive Services”, AI Services is a set of APIs that serve state-of-the-art models for various tasks. This includes computer vision, NLP, Open AI LLMs, etc.

Phi-3 open models

Build with the most capable and cost-effective small language models (SLMs) available.

Azure OpenAI Service

Build your own copilot and generative AI applications with cutting-edge language and vision models.

Azure AI Search

Retrieve the most relevant data using keyword, vector, and hybrid search.

Azure AI Content Safety

Monitor text and images to detect offensive or inappropriate content.

Azure AI Translator

Translate documents and text in real time across more than 100 languages.

Azure AI Speech

Use industry-leading AI services such as speech-to-text, text-to-speech, speech translation, and speaker recognition.

Azure AI Vision

Read text, analyze images, and detect faces with optical character recognition (OCR) and machine learning.

Azure AI Language

Build conversational interfaces, summarize documents, and analyze text using prebuilt AI-powered features.

Azure AI Document Intelligence

Apply advanced machine learning to extract text, key-value pairs, tables, and structures from documents.

https://azure.microsoft.com/en-us/products/ai-services

Azure Machine Learning Studio

Simplify and accelerate the building, training, and deployment of your machine learning models. Use automated machine learning to identify suitable algorithms and tune hyperparameters faster. Improve productivity and reduce costs with autoscaling compute and DevOps for machine learning. Seamlessly deploy to the cloud and the edge with one click.

Azure Machine Learning service capabilities

Automated machine learning

Identify suitable algorithms and hyperparameters faster.

Managed compute

Train models with ease and reduce costs by autoscaling powerful GPU clusters.

DevOps for machine learning

Increase productivity with experiment tracking, model management and monitoring, integrated CI/CD, and machine learning pipelines.

Simple deployment

Deploy models on-premises, to the cloud, and at the edge with a few lines of code.

Tool-agnostic Python SDK

Azure Machine Learning service integrates with any Python environment, including Visual Studio Code, Jupyter notebooks, and PyCharm.

Support for open-source frameworks

Use your favorite machine learning frameworks and tools, such as PyTorch, TensorFlow, and scikit-learn.

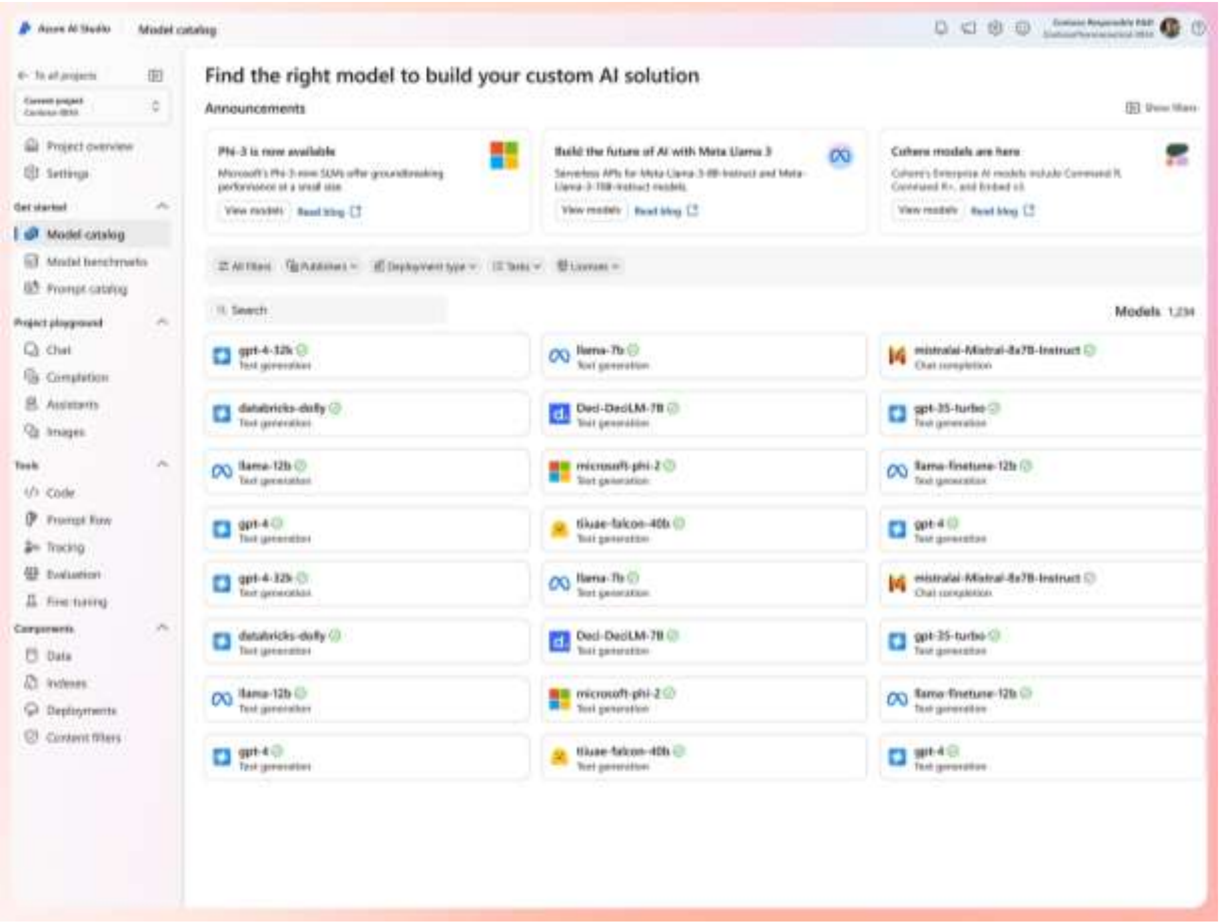
https://azure.microsoft.com/en-us/services/machine-learning-service/

Azure AI Studio

The “generative AI development hub”

- Allows you to test and deploy various models, including LLMs and image models
  - Open-source, OpenAI, and others

https://ai.azure.com/



Automated Machine Learning

(When you're too ~~lazy~~ busy to do it yourself…)

AutoML

- Automatically test various algorithms, hyperparameters, and scaling techniques
- Three Experiment Types:
  - Classification
  - Regression
  - Forecasting
- Steps:
  - Preprocessing & Scaling
  - Featurization
  - Cross-Validation
  - Ensembling

Classification

[Logistic Regression]
[Light GBM]
[Gradient Boosting]
[Decision Tree]
[K Nearest Neighbors]
[Linear SVC]
[C-Support Vector Classification (SVC)]
[Random Forest]
[Extremely Randomized Trees]
[XGboost]
[DNN Classifier]
[DNN Linear Classifier]
[Naive Bayes]
[Stochastic Gradient Descent (SGD)]

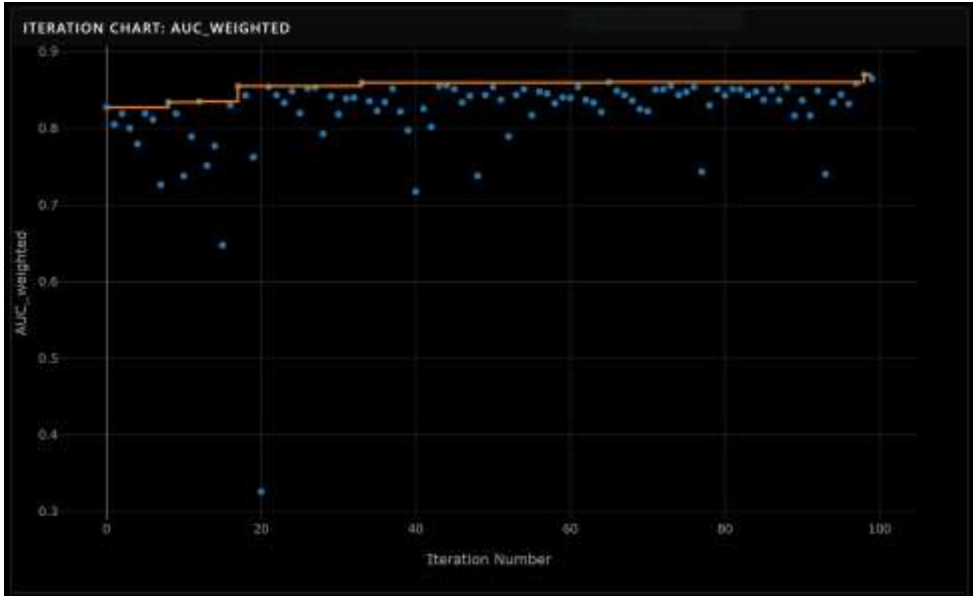
Regression & Time Series Forecasting

[Elastic Net]
[Light GBM]
[Gradient Boosting]
[Decision Tree]
[K Nearest Neighbors]
[LARS Lasso]
[Stochastic Gradient Descent (SGD)]
[Random Forest]
[Extremely Randomized Trees]
[XGboost]
[DNN Regressor]
[Linear Regressor]

https://docs.microsoft.com/en-us/azure/machine-learning/service/how-to-configure-auto-train

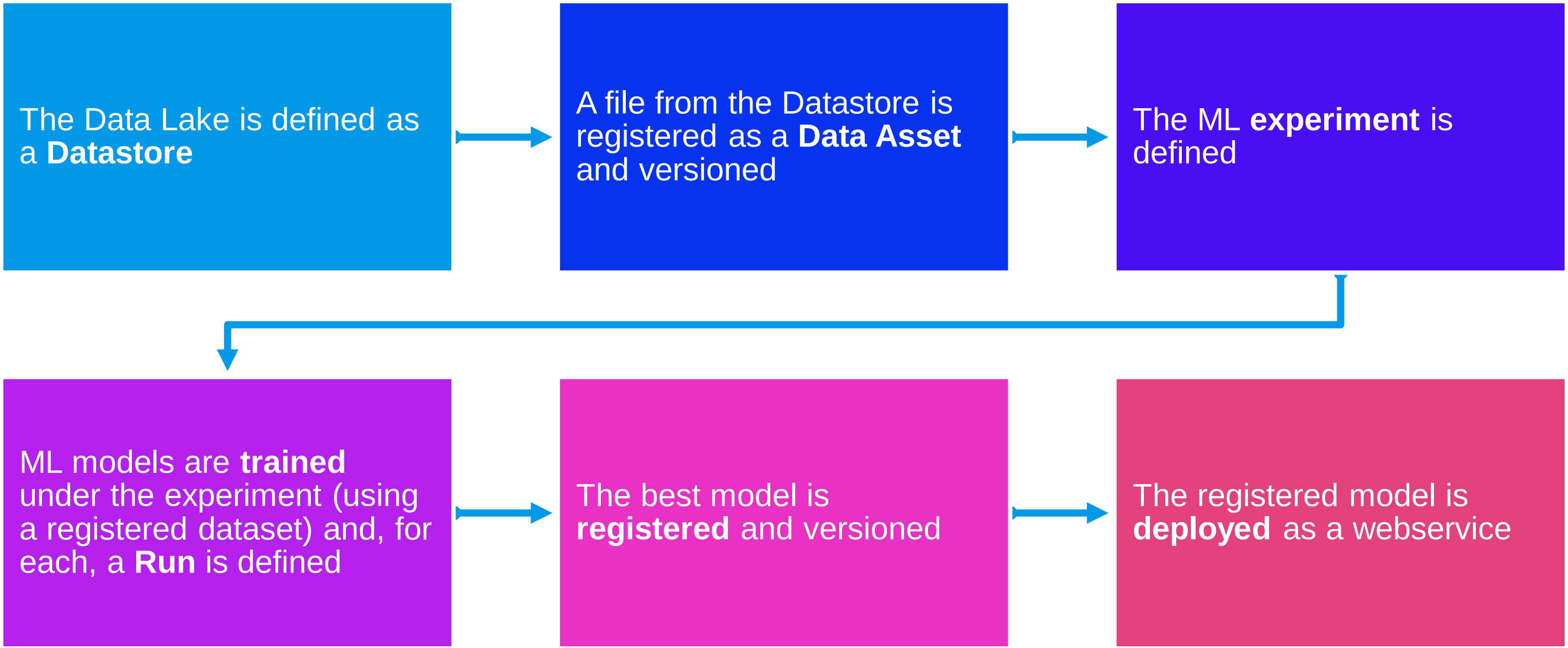
Picking the Best Model

- Picks a best model(s) based on a Primary Metric
  - In Classification Problems:
    - Accuracy, Weighted AUC, Average Weighted Precision Score, Normalized Macro Recall, Weighted Precision Score
  - In Regression/Forecasting Problems:
    - Spearman Correlation, Normalized RMSE, R^2, Normalized MAE



https://docs.microsoft.com/en-us/azure/machine-learning/service/how-to-understand-automated-ml

MLOps Lifecycle in Azure Machine Learning



Machine Learning Lab Assignment

Part 1: Train and Deploy an AutoML Model in Azure Machine Learning Studio as an API.

1. Find a ML dataset and upload it to the data lake
2. Using either the Python SDK or GUI, create an AutoML experiment and train a series of models
3. Evaluate the models and pick the best one
4. Deploy it as an API endpoint

Part 2: Play with an Azure OpenAI model and compare it to an open-source model.

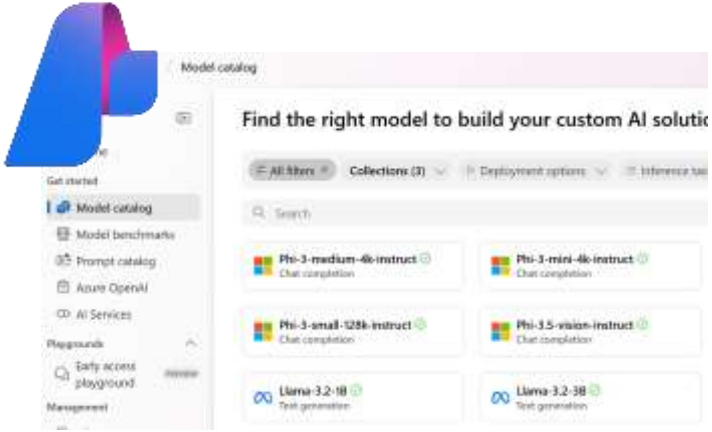
1. Using the class Azure AI Studio, play with the deployed OpenAI model, testing its performance and limitations
2. Pick an open-source model and play with it as well, testing its performance and limitations
3. Compare the two

MLOps and Model Deployment

“Now that my client has an ML model that they’re happy with, how do we go about using it in production?”

Large Language Models, Prompts, and RAG

Places to Play with Open-Source Models

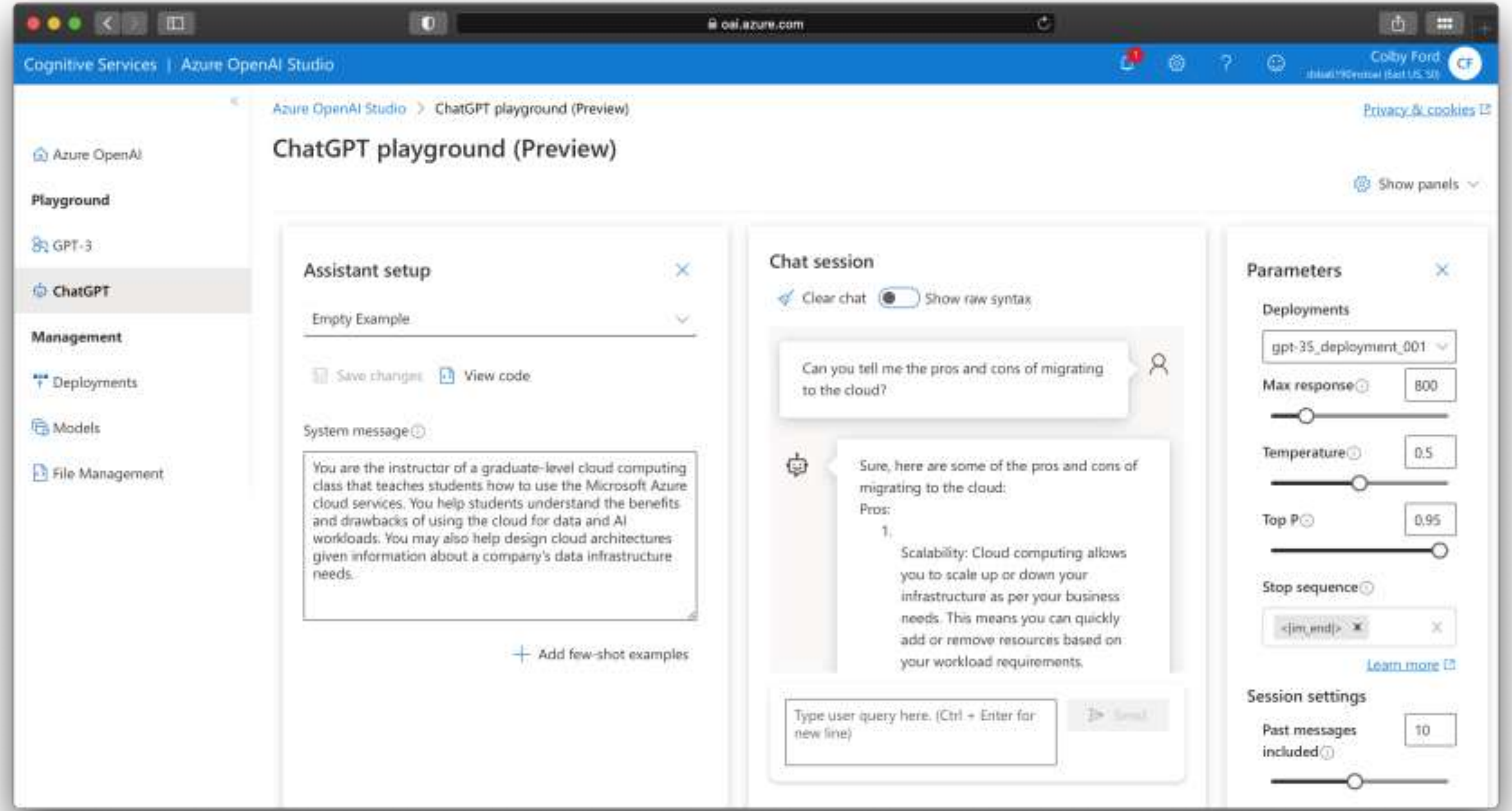


Azure AI Studio - Model Catalog
https://ai.azure.com/explore/models

Hugging Face Models
https://huggingface.co/models

Ollama
https://ollama.com/

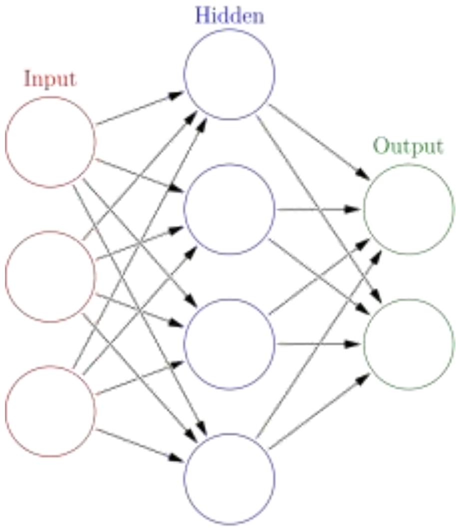
Can ChatGPT Replace Me?!?



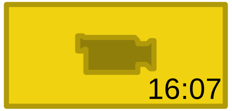
Link

Deep Learning + GPU Acceleration

Overview of Neural Networks

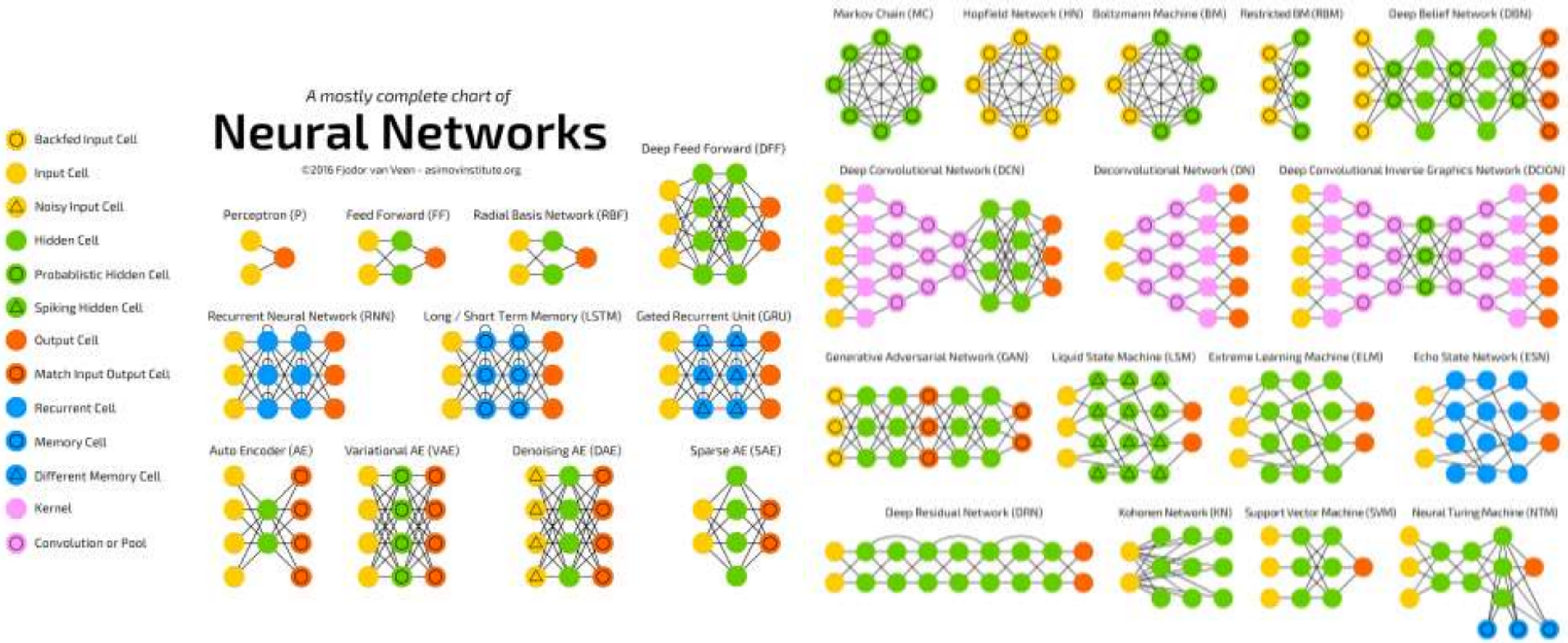


https://commons.wikimedia.org/w/index.php?title=File:Colored_neural_network.svg&oldid=279111871



- Built using interconnected nodes in multiple layers
  - Layers:
    - Input:
      - The set of input features
    - Hidden:
      - Set of mathematical functions to compute activations
    - Output:
      - The predicted class or value
  - Connections:
    - Directional weights

- Examples:
  - Recurrent Neural Networks
    - Useful for NLP
  - Convolutional Neural Networks
    - Useful in Image Recognition
  - Long Short-Term Memory
    - Useful for Time Series
  - …tons more
- “Deep” just means there are multiple layers…



http://www.asimovinstitute.org/neural-network-zoo/

ONNX

- Open Neural Network Exchange
  - An open-source, standardized format for describing neural network models [*.onnx]
  - Helps in usability when the model object is framework-agnostic
  - Model Zoo: a place to publish your trained model for open, external use

Types of Processors

(Well, the ones relevant to ML/DL/AI)

Central Processing Unit

- 1-32 Cores
- up to ~5GHz
- Single precision (32-bit), Double precision (64-bit) and possibly Extended precision (128-bit)

Graphics Processing Unit

- 100s-1000s of slower cores
- ~500-1200MHz
- Single precision to Double precision

Field Programmable Gate Array

- Reconfigurable chips to do a specific task – using programmable logic blocks
- Useful in IoT/Edge scenarios

Tensor Processing Unit

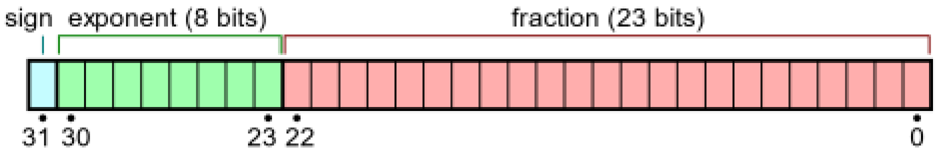
- Developed by Google (especially for use in TensorFlow).
- Proprietary, but can be used on GCP
- High volume, low precision (8-bit)

Precision, Precisely

Single Precision

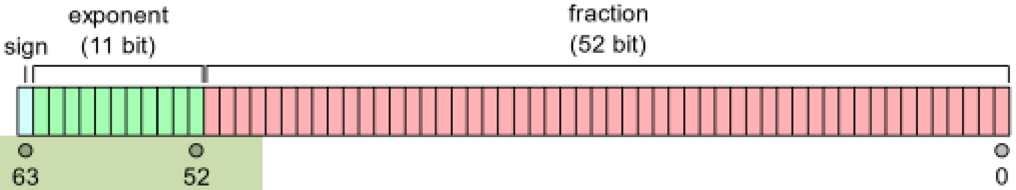
- Max Signed Integer Value: $2^{31} - 1 = 2{,}147{,}483{,}647$
- Max IEEE 754 Value: $(2 - 2^{-23}) \times 2^{127} \approx 3.4028235 \times 10^{38}$
- Significant Digits: up to 9

Binary32 Example
Input: 3.14159265358
Stored as: 3.1415927410125732421875
Error: 8.74325732421875E-8
Binary: 0 10000000 10010010000111111011011



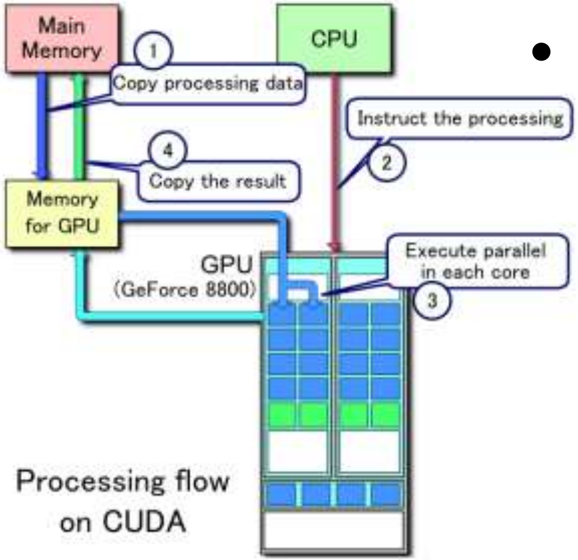
Double Precision

- Max Signed Integer Value: $2^{63} - 1 = 9{,}223{,}372{,}036{,}854{,}775{,}807$
- Max IEEE 754 Value: $(2 - 2^{-52}) \times 2^{1023} \approx 1.7976931 \times 10^{308}$
- Significant Digits: up to 17



Binary64 Example
Input: 3.14159265358
Stored as: 3.14159265358000006074234988773241639137268066640625
Error: 6.074234988773241639137268066640625E-17
Binary: 0 10000000000 1001001000011111101101010100010000111101011011110100

CUDA and OpenCL



CUDA Processing Flow by Tosaka [CC BY 3.0 (https://creativecommons.org/licenses/by/3.0)]

- CUDA
  - Developed by Nvidia
  - For use on Nvidia or CUDA-enable GPUs
  - Parallelized functionality for linear algebra, sparse matrices, Fourier transforms, random numbers, solvers, graph processing, and more.
- OpenCL
  - Originally developed by Apple, maintained by the Khronos Group
  - For use in CPUs, GPUs, and FPGAs (for a variety of vendors)
  - Parallelized functionality for matrix algebra, vectors (especially images)

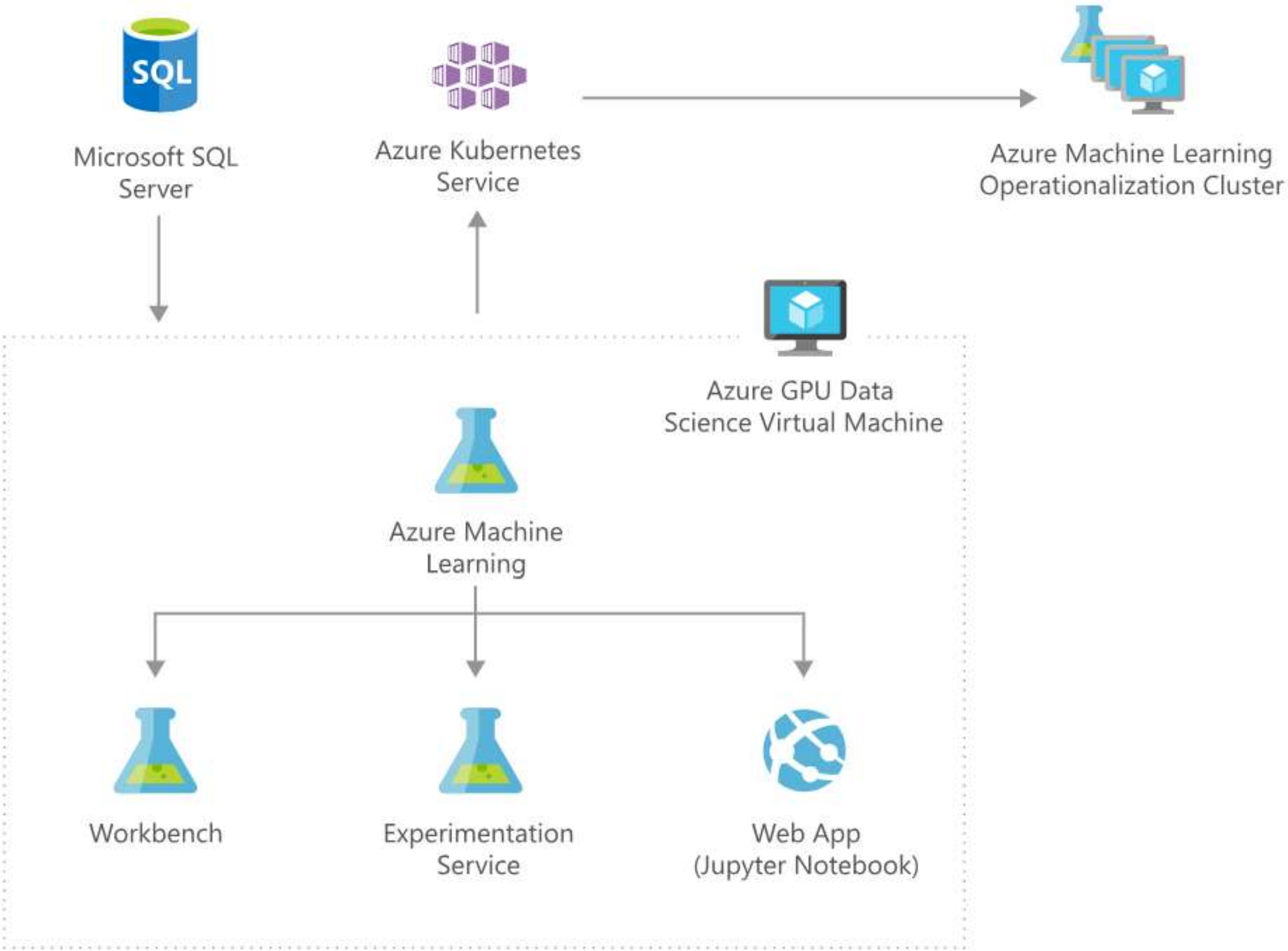
Triton and Metal

```
BLOCK = 512

# This is a GPU kernel in Triton.
# Different instances of this
# function may run in parallel.
@jit
def add(X, Y, Z, N):
    # In Triton, each kernel instance
    # executes block operations on a
    # single thread: there is no construct
    # analogous to threadIdx
    pid = program_id(0)
    # block of indices
    idx = pid * BLOCK + arange(BLOCK)
    mask = idx < N
    # Triton uses pointer arithmetics
    # rather than indexing operators
    x = load(X + idx, mask=mask)
    y = load(Y + idx, mask=mask)
    store(Z + idx, x + y, mask=mask)


...
grid = (ceil_div(N, BLOCK),)
# no thread-block
add[grid](x, y, z, x.shape[0])
```

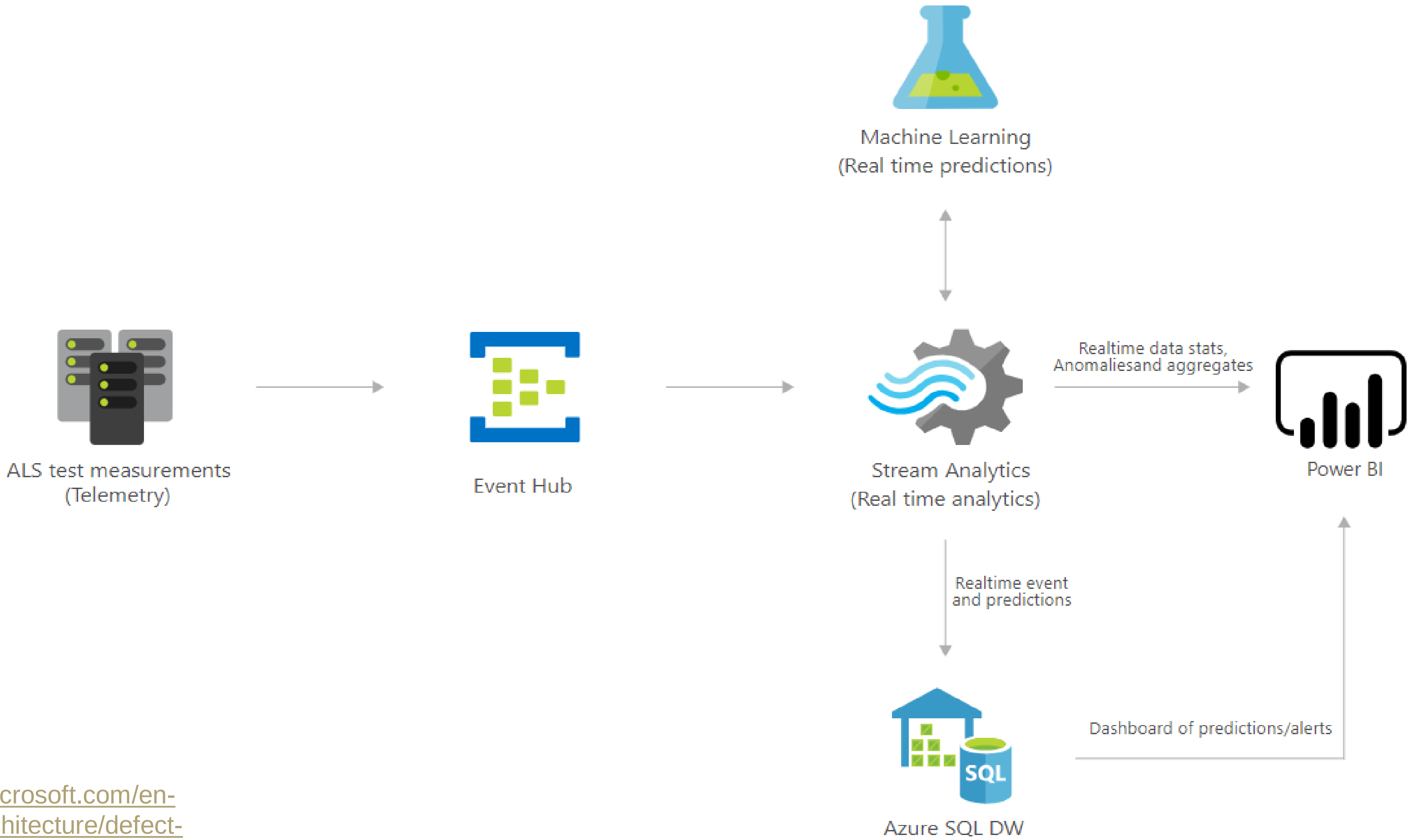
- Triton
    - Developed by OpenAI
    - For use on virtually any GPU
    - Technically a language and compiler
    - Primitives for custom deep learning functionality
- Metal
    - Developed by Apple
    - For use across Apple hardware (integrated M chips with both CPU and GPUs)
    - Mainly for gaming, but now has a backend for PyTorch

"Triton makes it possible to reach peak hardware performance with relatively little effort; for example, it can be used to write FP16 matrix multiplication kernels that match the performance of cuBLAS—something that many GPU programmers can't do—in under 25 lines of code. Our researchers have already used it to produce kernels that are up to 2x more efficient than equivalent Torch implementations, and we're excited to work with the community to make GPU programming more accessible to everyone."

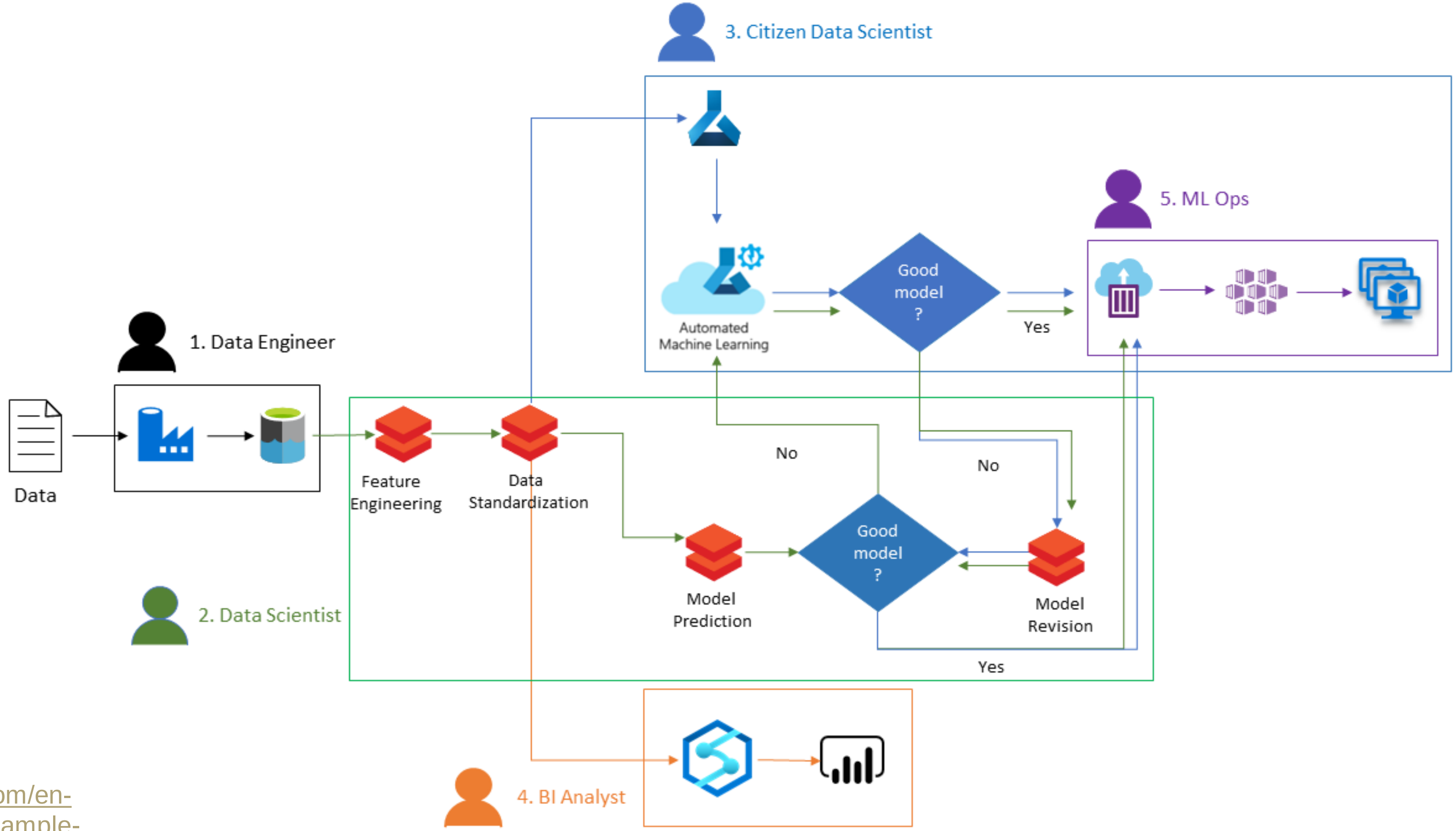
APPENDIX



https://azure.microsoft.com/en-us/solutions/architecture/information-discovery-with-deep-learning-and-nlp/



https://azure.microsoft.com/en-us/solutions/architecture/defect-prevention-with-predictive-maintenance/



https://docs.microsoft.com/en-us/azure/architecture/example-scenario/ai/predict-hospital-readmissions-machine-learning

Machine Learning Deep Dive

- Overview of Algorithms:
  - Regression
  - Tree-Based Classification
  - Clustering
- Computational Complexity

Linear Regression

- Normally in the form:

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + \varepsilon$$

or, in matrix notation…

$$y = X\beta + \varepsilon$$

$$\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nk} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_k \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

- To solve for β:

$$\beta = (X^T X) X^T Y$$

```r
## Load in data
data(mtcars)

## Define X matrix
X <- as.matrix(cbind(1,
                     mtcars$cyl,
                     mtcars$hp))


## Define y matrix
y <- as.matrix(mtcars$mpg)

## Solve for beta_hat
beta_hat <- solve(t(X)%*%X)%*%t(X)%*%y


## Fit the same data using lm
fit <- lm(mpg ~ cyl + hp,
          data = mtcars)

fit$coefficients
```

Regression Model Metrics

- Sum of Squares

$$\sum_{i=1}^{n}(y_i - \bar{y})^2 = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2 + \sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

$$SSTotal = SSReg + SSError$$

- R^2 : The Coefficient of Determination

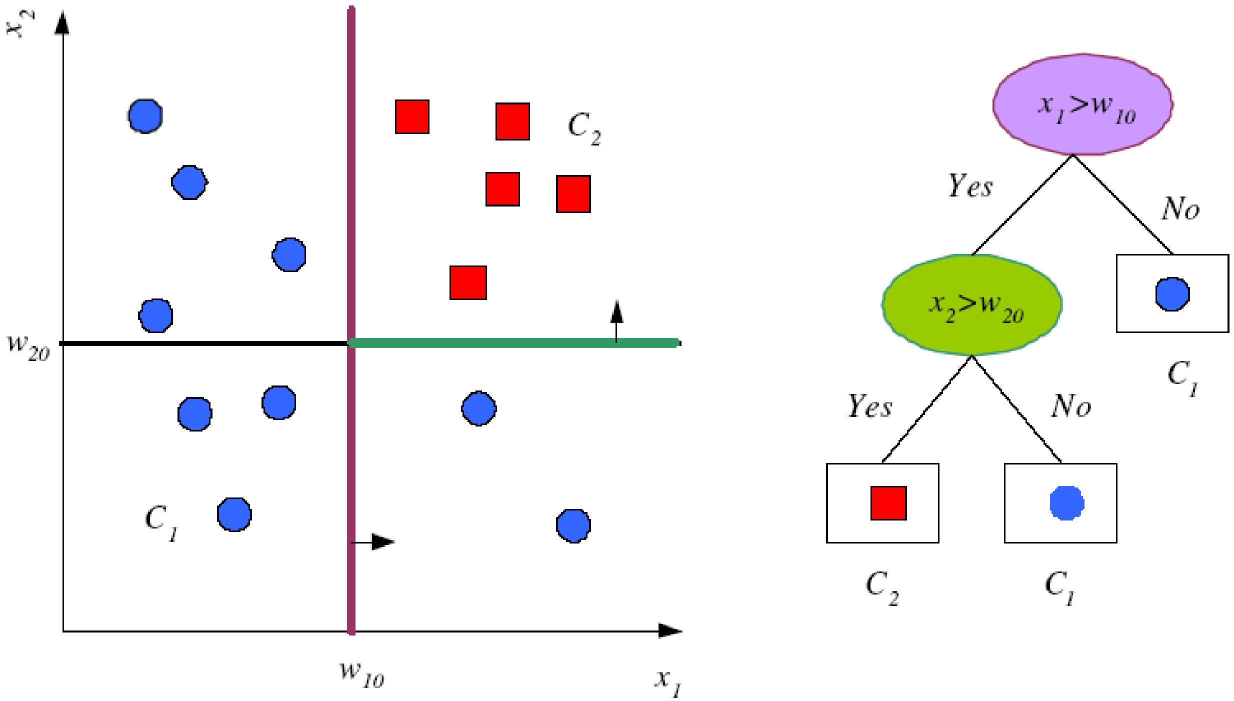
$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST}$$

- Other things to look at:
  - p-values
  - RMSE
  - MAPE

Tree-Based Classification

- Decision Trees
  - Pick the most discriminable factor first, then continue to split
- Random Forests
  - Generate a bunch of random decision trees and vote on the most popular answer

Finishes based on some entropy/purity criterion



https://www.cmpe.boun.edu.tr/~ethem/i2ml3e/3e_v1-0/i2ml3e-chap9.pptx

Divide and Conquer

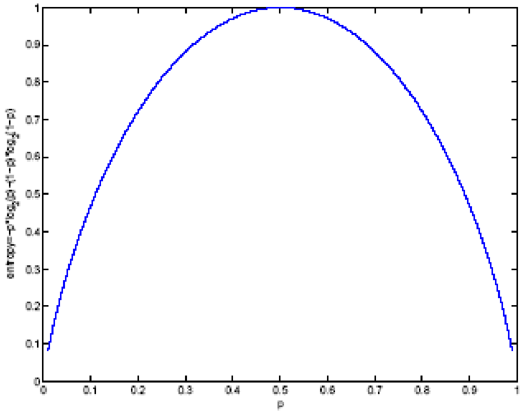
- Internal decision nodes
  - Univariate: Uses a single attribute, x_i
    - Numeric x_i : Binary split : $x_i > w_m$
    - Discrete x_i : n-way split for n possible values
  - Multivariate: Uses all attributes, $\boldsymbol{x}$
- Leaves
  - Classification: Class labels, or proportions
  - Regression: Numeric; r average, or local fit
- Learning is greedy; find the best split recursively (Breiman et al, 1984; Quinlan, 1986, 1993)

Classification Trees (ID3,CART,C4.5)

- For node m, N_m instances reach m, N^i_m belong to C_i

$$\hat{P}(C_i \mid \mathbf{x}, m) \equiv p^i_m = \frac{N^i_m}{N_m}$$

- Node m is pure if p^i_m is 0 or 1
- Measure of impurity is entropy



$$\mathcal{I}_m = -\sum_{i=1}^{K} p^i_m \log_2 p^i_m$$

Best Split

- If node m is pure, generate a leaf and stop, otherwise split and continue recursively
- Impurity after split: N_{mj} of N_m take branch j. N^i_{mj} belong to C_i

$$\hat{P}(C_i \mid \mathbf{x}, m, j) \equiv p^i_{mj} = \frac{N^i_{mj}}{N_{mj}} \qquad \mathcal{I}'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p^i_{mj} \log_2 p^i_{mj}$$

- Find the variable and split that min impurity (among all variables -- and split positions for numeric variables)

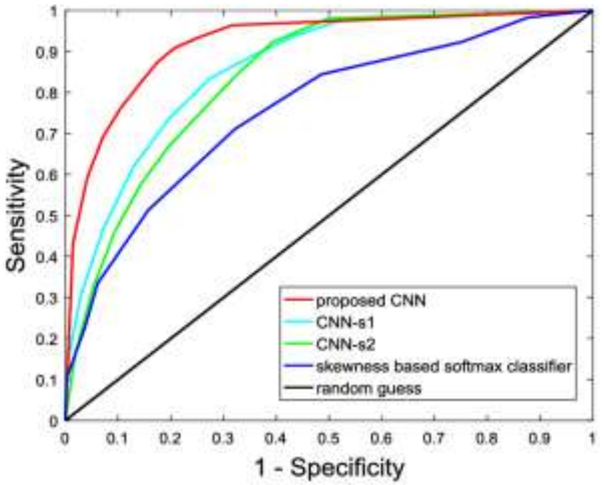
Pruning Trees

- Remove subtrees for better generalization (decrease variance)
  - Prepruning: Early stopping
  - Postpruning: Grow the whole tree then prune subtrees that overfit on the pruning set
- Prepruning is faster, postpruning is more accurate (requires a separate pruning set)

“Accuracy is the best indication of a good model.” – Wrong People

Classification Model Metrics

- ROC Curve
  - A measure of the false positive rate vs. the true positive rate



https://www.researchgate.net/figure/ROC-curves-of-different-classifiers-The-ROC-curve-of-an-ideal-classifier-should-be_fig5_319939197

- $\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}}$
- $\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}$
- $\text{Recall} = \text{Sensitivity} = \frac{\text{TP}}{\text{TP}+\text{FN}}$
- $\text{Specificity} = \frac{\text{TN}}{\text{TN}+\text{FP}}$
- $\text{F1} = 2 \cdot \frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$

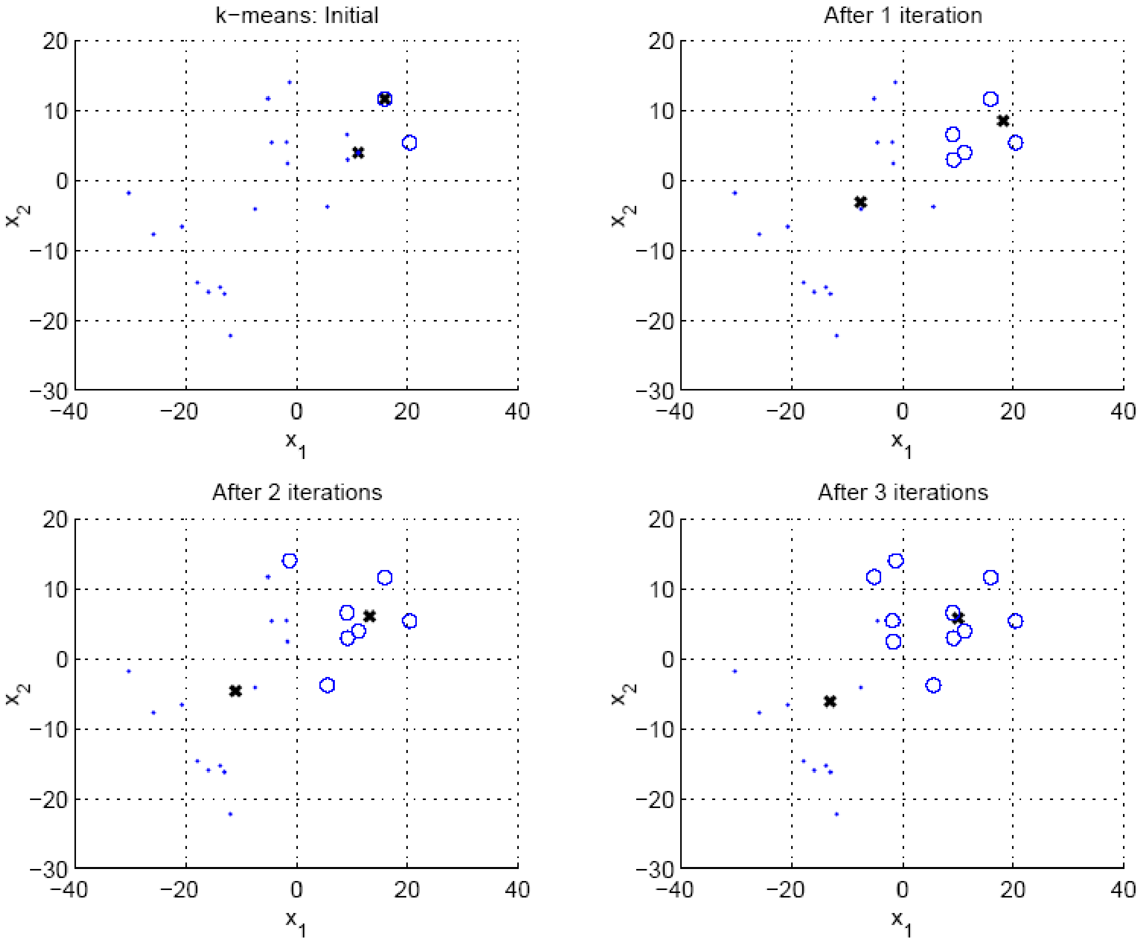
Clustering

Multiple methods:

- Agglomerative – “Bottom-Up”
  - UPGMA
- Divisive – “Top-Down”
  - k-Means

Based on some distance metric and some linkage criteria.

Initialize $\boldsymbol{m}_i, i = 1, \ldots, k$, for example, to k random $\boldsymbol{x}^t$
Repeat
  For all $\boldsymbol{x}^t \in \mathcal{X}$
$$b_i^t \leftarrow \begin{cases} 1 & \text{if } \|\boldsymbol{x}^t - \boldsymbol{m}_i\| = \min_j \|\boldsymbol{x}^t - \boldsymbol{m}_j\| \\ 0 & \text{otherwise} \end{cases}$$
  For all $\boldsymbol{m}_i, i = 1, \ldots, k$
$$\boldsymbol{m}_i \leftarrow \sum_t b_i^t \boldsymbol{x}^t / \sum_t b_i^t$$
Until $\boldsymbol{m}_i$ converge



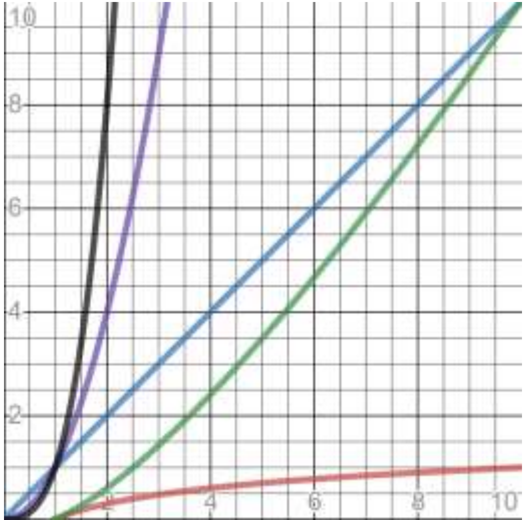
https://www.cmpe.boun.edu.tr/~ethem/i2ml3e/3e_v1-0/i2ml3e-chap7.pptx

—

Computational Complexity

- A measure of inherent difficulty in completing a computational task
- In other words, if I increase the number of input points (either variables, observations, or parameters), by what factor does the computational time increase?

Algorithm	Training Complexity	Prediction Complexity
Linear Regression	$O(p^2 n + p^3)$	$O(p)$
Decision Trees	$O(n^2 p)$	$O(p)$
Random Forest	$O(n^2 pa)$	$O(pa)$
k-Means Clustering	N/A	$O(nk)$
UPGMA	N/A	$O(n^3)$ or $O(n^2 \log n^2)$ or $O(n^2)$



Where:

$n = number\ of\ observations$

$p = number\ of\ features$

$a = number\ of\ trees$

https://www.thekerneltrip.com/machine/learning/computational-complexity-learning-algorithms/

Just Keep Modelin'

Proving the model is valid in more scenarios than just on the training data.

- Parameter Tuning
  - Grid Search
  - Regularization
  - Gradient Decent
- Cross-Validation
  - k-Fold CV
  - LpOCV/LOOCV
- Intro to Training Parallelization

—

Grid Search

- Most traditional way of parameter sweeping is to build a grid of multiple parameter options
- Parameters can be discrete or continuous
- Must pick a finite set of reasonable options
- The locally optimal set is the combination of parameters that minimize or maximize a criterion. *(Such as RMSE, R^2, AUC, etc.)*

Parameter Options

$$\alpha \in \{1, 10, 100\}$$
$$\beta \in \{0.01, 0.05, 0.1\}$$
$$\gamma \in \{10, 11, 12\}$$

```
## in R
alpha <- c(1, 10, 100)
beta <- c(0.01, 0.05, 0.1)
gamma <- c(10, 11, 12)

grid <- expand.grid(alpha, beta, gamma)
```

Parameter Grid

#	α	β	γ
1	1	0.01	10
2	10	0.01	10
3	100	0.01	10
4	1	0.05	10
5	10	0.05	10
6	100	0.05	10
7	1	0.10	10
8	10	0.10	10
9	100	0.10	10
10	1	0.01	11
11	10	0.01	11
12	100	0.01	11
13	1	0.05	11
14	10	0.05	11
15	100	0.05	11
16	1	0.10	11
17	10	0.10	11
18	100	0.10	11
19	1	0.01	12
20	10	0.01	12
21	100	0.01	12
22	1	0.05	12
23	10	0.05	12
24	100	0.05	12
25	1	0.10	12
26	10	0.10	12
27	100	0.10	12

Regularization

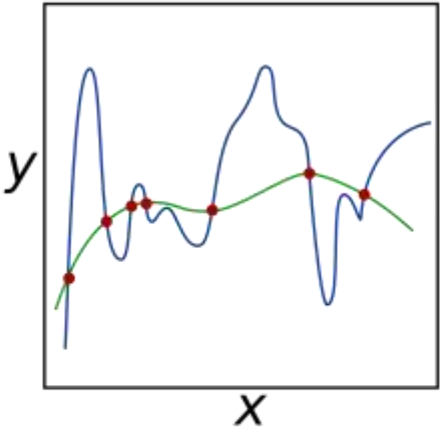
- Adding in information to prevent overfitting of the model to training data.
- Usually denoted by α or λ_k
- In combination, can be referred to as an Elastic Net

LASSO: Least Absolute Shrinkage and Selection Operator

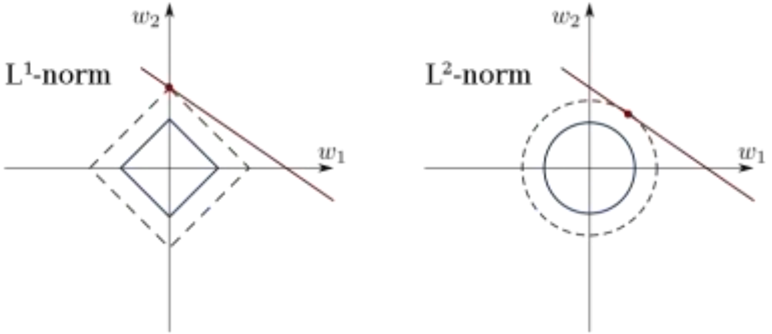
- Based on the L_1 distance (a.k.a. the taxicab metric)
- $\text{SSError} + (\lambda \times |\text{slope}|)$

Ridge Regression (a.k.a. Weight Decay)

- Most commonly used.
- Based on L_2 distance (Euclidean distance)
- $\min_{\beta,\beta_0} \left\{\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \beta_0 - x_i^T\beta\right)^2\right\}$ subject to $\left(\sum_{i=1}^{N}|\beta|^p\right)^{1/p}$
- $\text{SSError} + (\lambda \times \text{slope}^2)$



https://commons.wikimedia.org/wiki/File:Regularization.svg



https://en.wikipedia.org/wiki/File:L1_and_L2_balls.svg

Other Papers:

- Elastic Nets
- LASSO
- Ridge Regression
- L_1 vs L_2 Regularization

Common Parameters to Sweep

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

https://spark.apache.org/docs/2.2.0/mllib-decision-tree.html

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn.linear_model.LinearRegression

https://spark.apache.org/docs/2.2.0/ml-classification-regression.html#regression

https://spark.apache.org/docs/2.2.0/mllib-naive-bayes.html#naive-bayes-sparkmllib

- Trees:
  - Max Depth
  - Max Bins
  - Number of Trees (in a Random Forest)
  - Max Splits
  - Max Features
  - Min Samples per Leaf
  - Max Leaf Nodes
  - Min Impurity Decrease
  - Min Impurity to Split
  - … and more
- Regression and Other Algorithms:
  - Regularization
  - Elastic Net
  - Maximum Iterations
  - Smoothing (in Naïve Bayes)
  - … and more

Gradient Descent

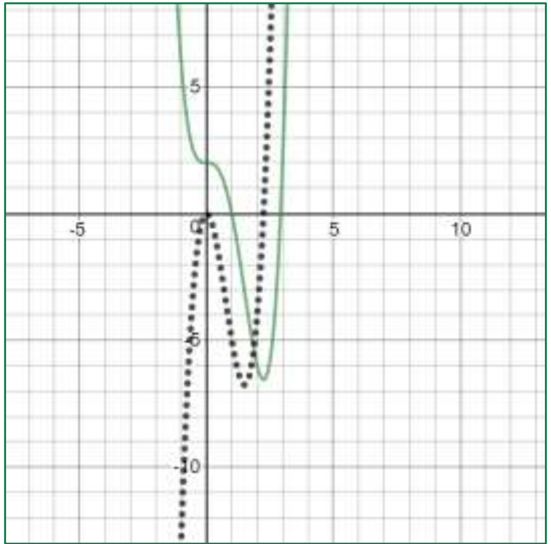
- In general, used to find the local minimum of a function
- Assume there is a multivariable function $F(x)$ that is differentiable. The function decreases fastest as you go in the direction of the negative gradient, starting at a point.

  $$x, -\nabla F(x)$$

- Assumes a smooth topology and convex topology. (Not always true for parameters)

$$f(x) = x^4 - 3x^3 + 2$$
$$f'(x) = 4x^3 - 9x^2$$

```
## Parameters
alpha = 0.001 #Stepsize
iter = 500 #Iterations

# Define the objective function
f(x) = x^4 - 3*x^3 + 2
objFun = function(x){
  return(x^4 - 3*x^3 + 2)
}

# Define the gradient of f(x) =
x^4 - 3*x^3 + 2
gradient = function(x){
  return((4*x^3) - (9*x^2))
}
```




```
# Randomly initialize a value to x
set.seed(1337)
x = floor(runif(1, 0, 10))

# Create a vector to contain all x's
for all steps
values = numeric(iter)

# Loop for Gradient Descent method to
find the minimum
for(i in seq_len(iter)){
  x = x - alpha*gradient(x)
  values[i] = x
  print(x)
}

# Output
print(paste("The minimum of f(x) is ",
            objFun(x),
            " at position x = ",
            x,
            sep = ""))
plot(values,
     type = "l")
```

Cross Validation

- k-Fold CV
  - "Non-exhaustive" – Not all ways of splitting the training data are tested. So, this is an approximation.
  - Randomly split the data into k equally-sized subsets. One subset is used as validation while the rest are used for training.
- LpOCV: Leave-p-out Cross-Validation
  - "Exhaustive" – All possible ways to dividing the training data are tested.
  - Use p observations as the validation set and the rest as the training set.
  - $C_p^n = \frac{n!}{k!(n-k)!}$ (Computationally Infeasible)

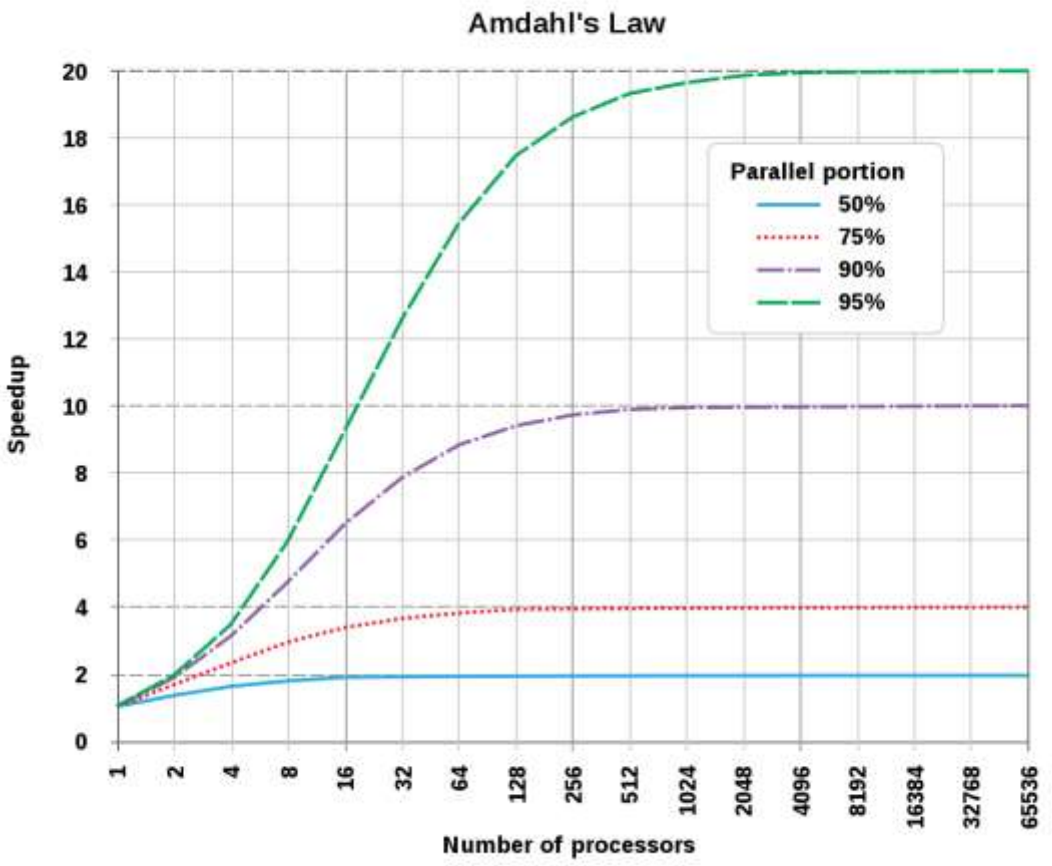
When to parallelize?

- Adding processors may seem to help with processing times, but there is always overhead to running the parallelism that will reduce the speed gain.

Amdahl's Law:

$$\text{Speedup} = \frac{1}{r_s + \frac{r_p}{n}}$$

r_s = Ratio of program that is sequential
r_p = Ratio of program that is parallel



http://www-inst.eecs.berkeley.edu/~n252/paper/Amdahl.pdf

Training Parallelization - Packages

- In R
  - parallel, foreach, doParallel, doMC, doSNOW, doMPI
  - https://cran.r-project.org/web/views/HighPerformanceComputing.html
  - Example: https://nceas.github.io/oss-lessons/parallel-computing-in-r/parallel-computing-in-r.html
- In Python
  - multiprocessing, Dask, Numba, etc.
  - https://wiki.python.org/moin/ParallelProcessing
  - Example: https://nbviewer.jupyter.org/gist/ogrisel/5115540/Model%20Selection%20for%20the%20Nystroem%20Method.ipynb

caret Example

```
fitControl <- trainControl(## 10-fold CV
                           method = "repeatedcv",
                           number = 10,
                           ## repeated ten times
                           repeats = 10)

gbmGrid <-  expand.grid(interaction.depth = c(1, 5, 9),
                        n.trees = (1:30)*50,
                        shrinkage = 0.1,
                        n.minobsinnode = 20)

gbmFit  <- train(Class ~ ., data = training,
                 method = "gbm",
                 trControl = fitControl,
                 verbose = FALSE,
                 tuneGrid = gbmGrid)
```

```
## Stochastic Gradient Boosting
##
## 157 samples
##  60 predictor
##   2 classes: 'M', 'R'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 141, 142, 141, 142, 141, 142, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy  Kappa
##   1                    50     0.78      0.56
##   1                   100     0.81      0.61
##   1                   150     0.82      0.63
##   1                   200     0.83      0.65
##   1                   250     0.82      0.65
##   1                   300     0.83      0.65
##   :                   :        :         :
##   9                  1350     0.85      0.69
##   9                  1400     0.85      0.69
##   9                  1450     0.85      0.69
##   9                  1500     0.85      0.69
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 20
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 1200,
##  interaction.depth = 9, shrinkage = 0.1 and n.minobsinnode = 20.
```

https://topepo.github.io/caret/model-training-and-tuning.html