Machine Learning with H2O on HAL

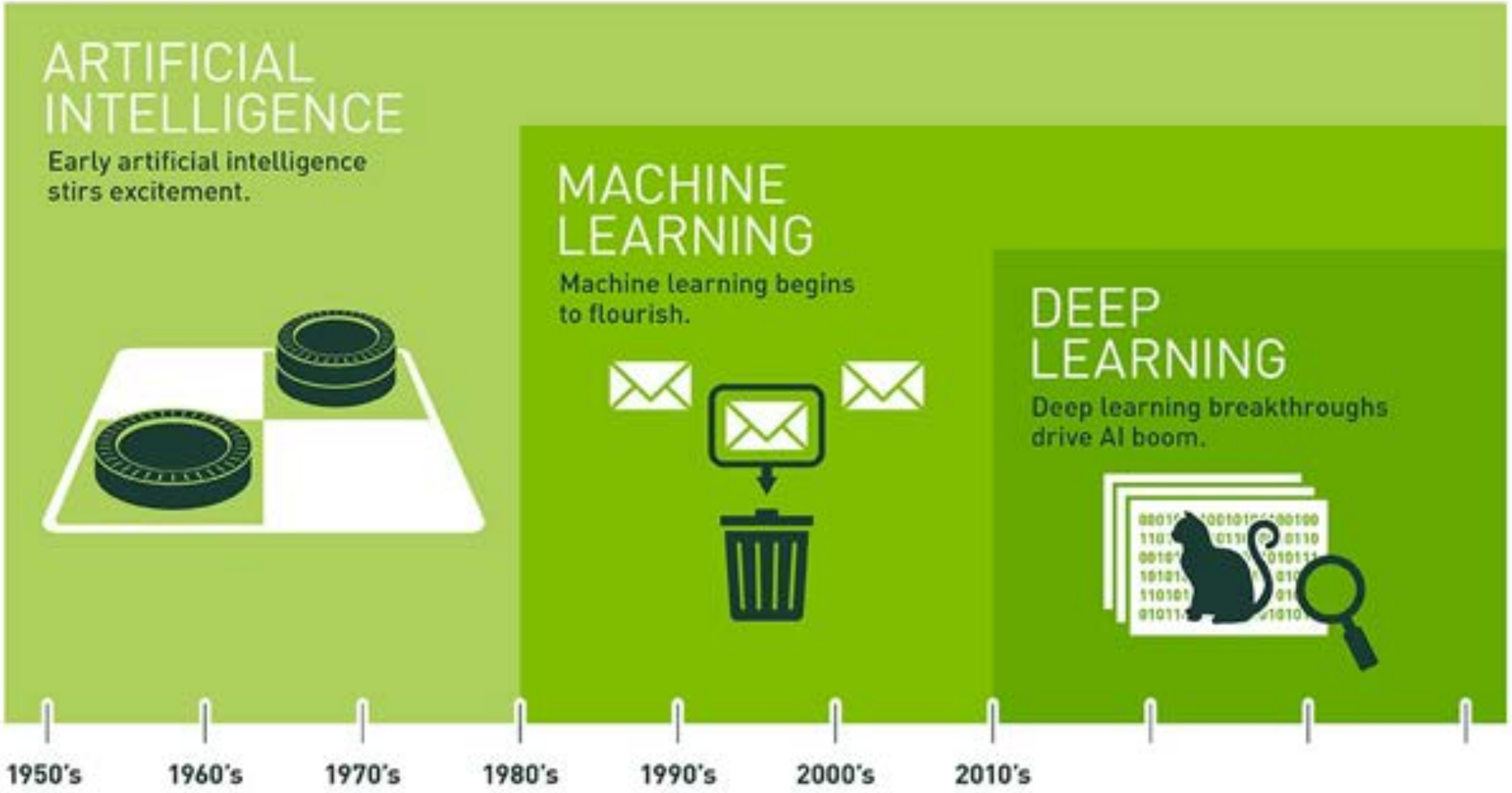
2021.10.13

Dawei Mu

ILLINOIS

NCSA | National Center for Supercomputing Applications

Introduction of AI / ML / DL



ARTIFICIAL INTELLIGENCE

Early artificial intelligence stirs excitement.

MACHINE LEARNING

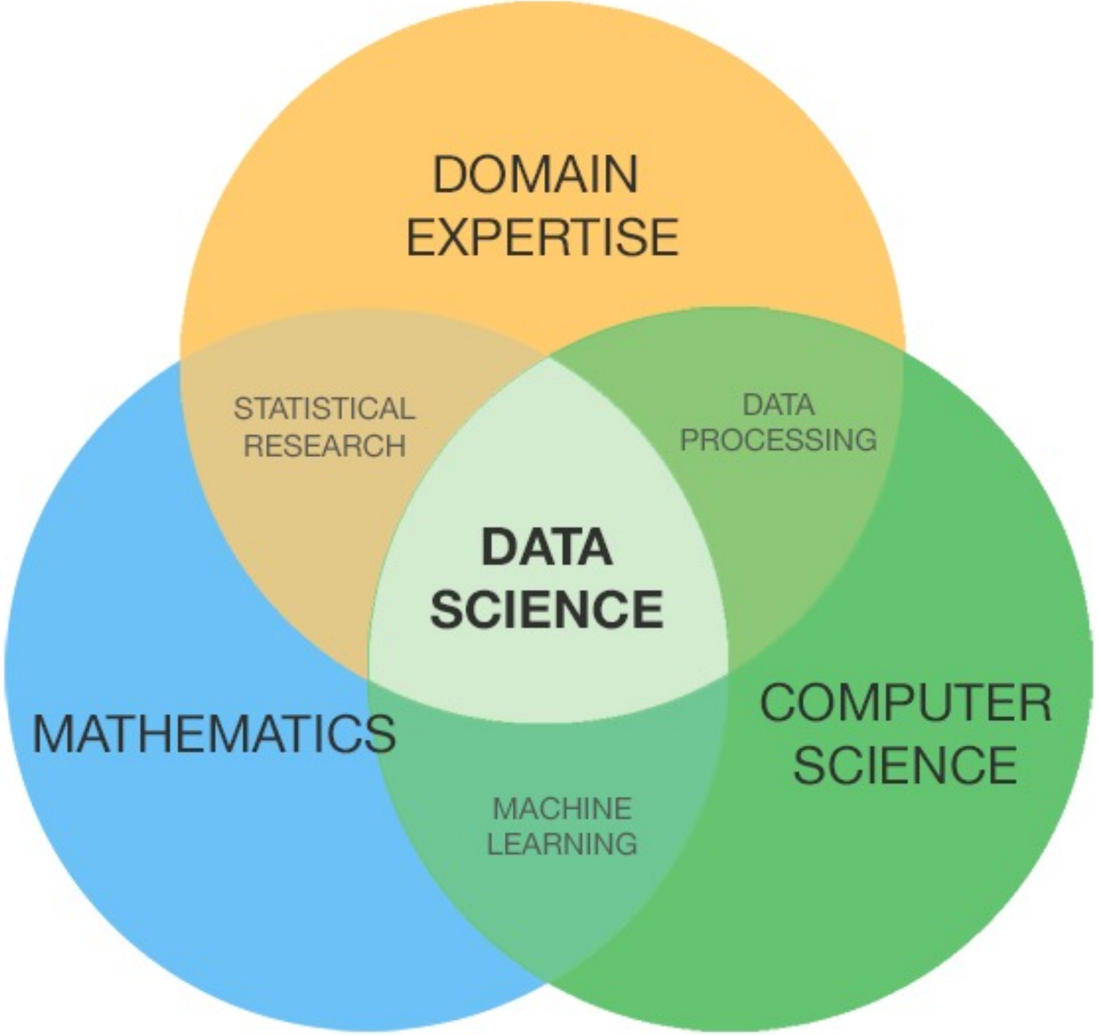
Machine learning begins to flourish.

DEEP LEARNING

Deep learning breakthroughs drive AI boom.

Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

This is an Interdisciplinary Field



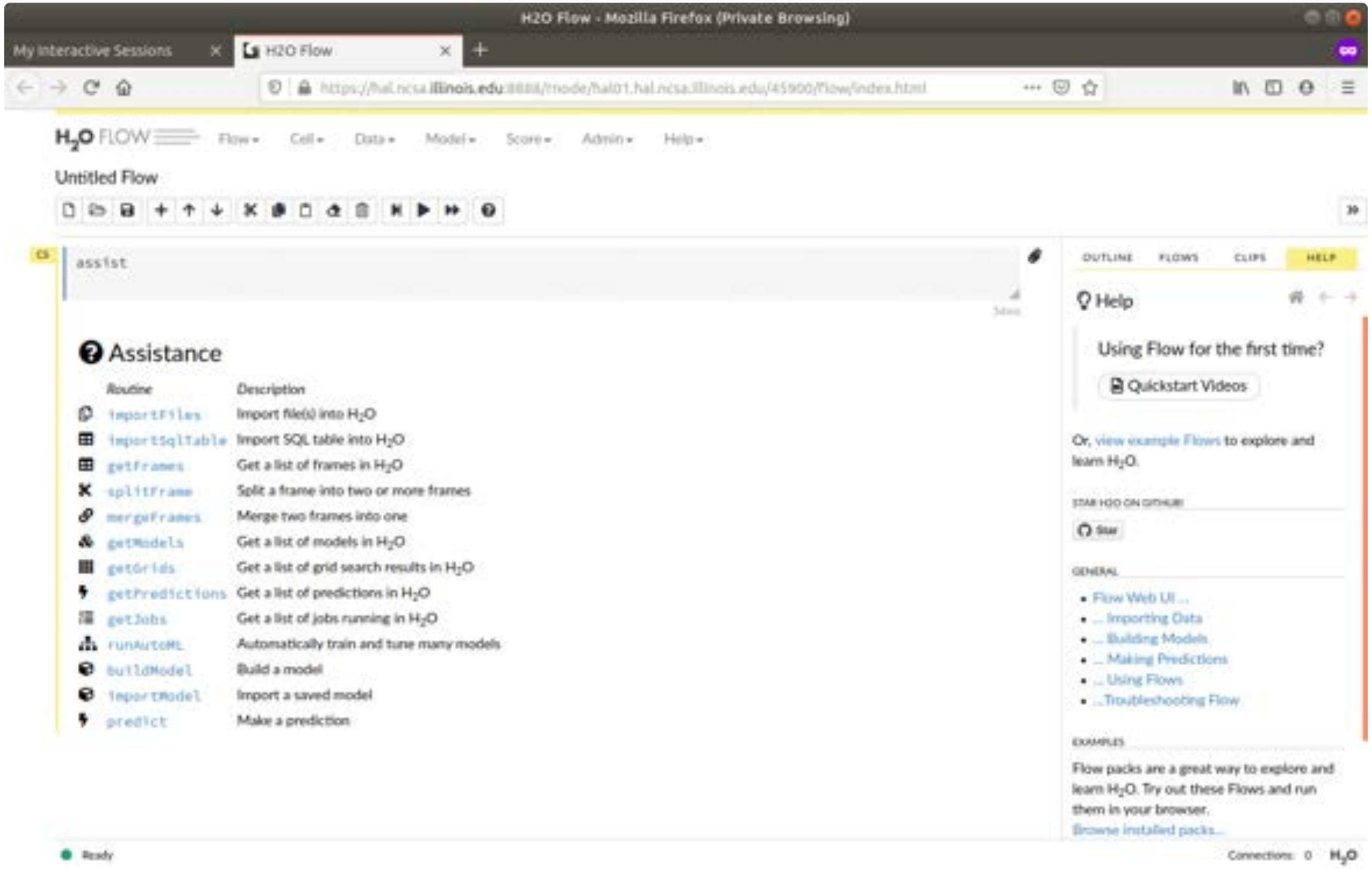
But Why Now?

- Hardware: high-performance GPUs, TPUs
- Datasets: large datasets collected from internet
- Algorithmic advances: *activation functions, optimization schemes.*

Introduction of H2O

- What is H2O.ai?
  - H2O.ai is the company behind open-source Machine Learning (ML) products like H2O, aimed to make ML easier for all.
- What is H2O?
  - An open source, Java-based, in-memory, distributed, ML and predictive analytics platform allowing you to build and productionize ML models.
  - Contains supervised and unsupervised models in R and Python, as well as a simple to use web-UI called Flow.

H2O Flow



H2O Flow - Mozilla Firefox (Private Browsing)
My Interactive Sessions
H2O Flow
https://hal.ncsa.illinois.edu:8888/rnode/hal01.hal.ncsa.illinois.edu/45900/flow/index.html
H₂O FLOW
Flow
Cell
Data
Model
Score
Admin
Help
Untitled Flow
CS
OUTLINE
FLOWS
CLIPS
HELP
Ready
Connections: 0
H₂O

```
assist
```

Assistance

Routine	Description
importFiles	Import file(s) into H_2O
importSqlTable	Import SQL table into H_2O
getFrames	Get a list of frames in H_2O
splitFrame	Split a frame into two or more frames
mergeFrames	Merge two frames into one
getModels	Get a list of models in H_2O
getGrids	Get a list of grid search results in H_2O
getPredictions	Get a list of predictions in H_2O
getJobs	Get a list of jobs running in H_2O
runAutoML	Automatically train and tune many models
buildModel	Build a model
importModel	Import a saved model
predict	Make a prediction

Help

Using Flow for the first time?

Quickstart Videos

Or, view example Flows to explore and learn H_2O.

STAR H2O ON GITHUB!

Star

GENERAL

- Flow Web UI ...
- ... Importing Data
- ... Building Models
- ... Making Predictions
- ... Using Flows
- ...Troubleshooting Flow

EXAMPLES

Flow packs are a great way to explore and learn H_2O. Try out these Flows and run them in your browser.

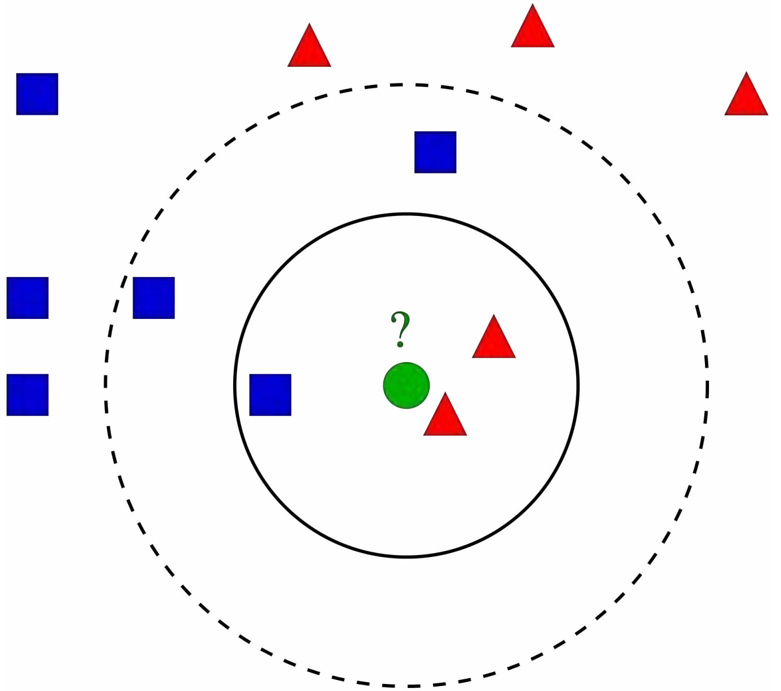
Browse installed packs...

Common Machine Learning Algorithms

- Machine Learning has 3 main functions: classification, prediction, clustering.
- Big 3 Basic Algorithms
  - K-Nearest Neighbor
  - Linear Regression
  - K-Mean Clustering
- Other Common Algorithms
  - Decision Tree / Random Forest / Naive Bayes / Support Vector Machine

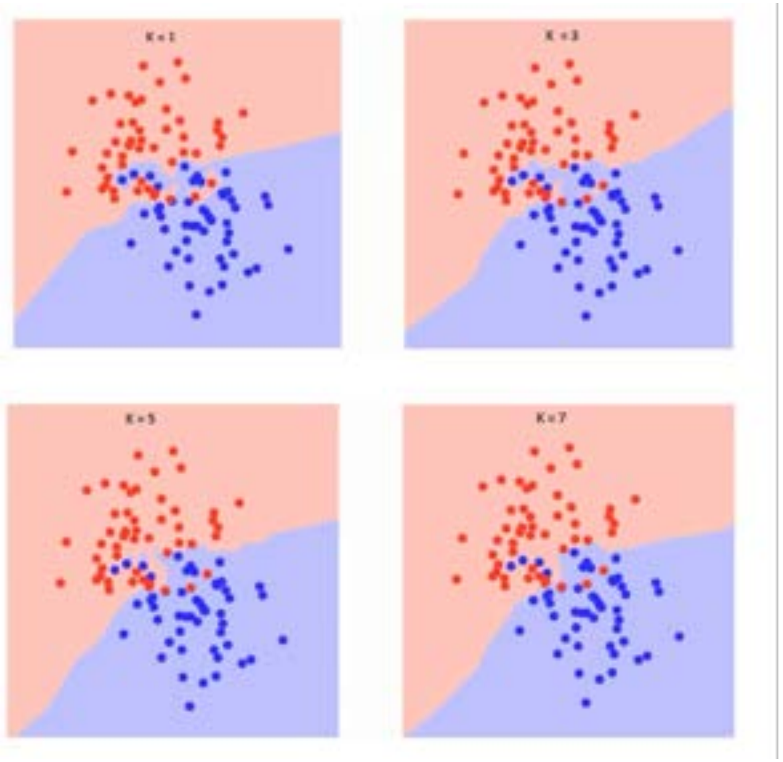
K Nearest Neighbors

- An object is classified by a majority vote of its neighbors
  - One of the simplest classification algorithm.
  - Often used in classification.
  - Computed from a simple majority vote of the nearest neighbors of each point.
  - K is constant specified by user.
  - KNN is computationally expensive.



K Nearest Neighbors

- How do we choose the factor K
  - Choosing K could be a challenge.
  - Boundary becomes smoother with increasing value of K.



K Nearest Neighbors

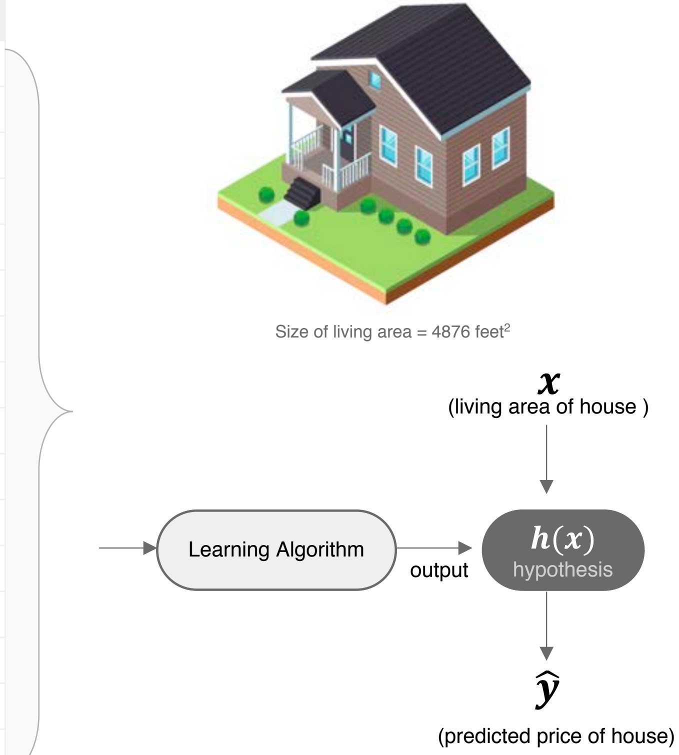
- Pros and Cons of KNN
  - Pros
    - It is beautifully simple and logical
  - Cons
    - It may be driven by the choice of K, which may be a bad choice.
    - Generally, larger values of K reduce the effect of noise on the classification, but make boundaries between classes less distinct.
    - The accuracy of the algorithm can be severely degraded by the presence of noisy or irrelevant features.
    - It is important to review the sensitivity of the solution to different values of K.

Linear Regression

Living Area (Feet²)	Price ($)
1180	221,900
2570	538,000
770	180,000
1960	604,000
1680	510,000
5420	1,225,000
1715	257,500
1060	291,850
1780	229,500
1890	323,000
3560	662,500
1160	468,000
1430	310,000
1370	400,000
1810	530,000
...	...

x y

Size of living area = 4876 feet²



x

(living area of house)

Learning Algorithm

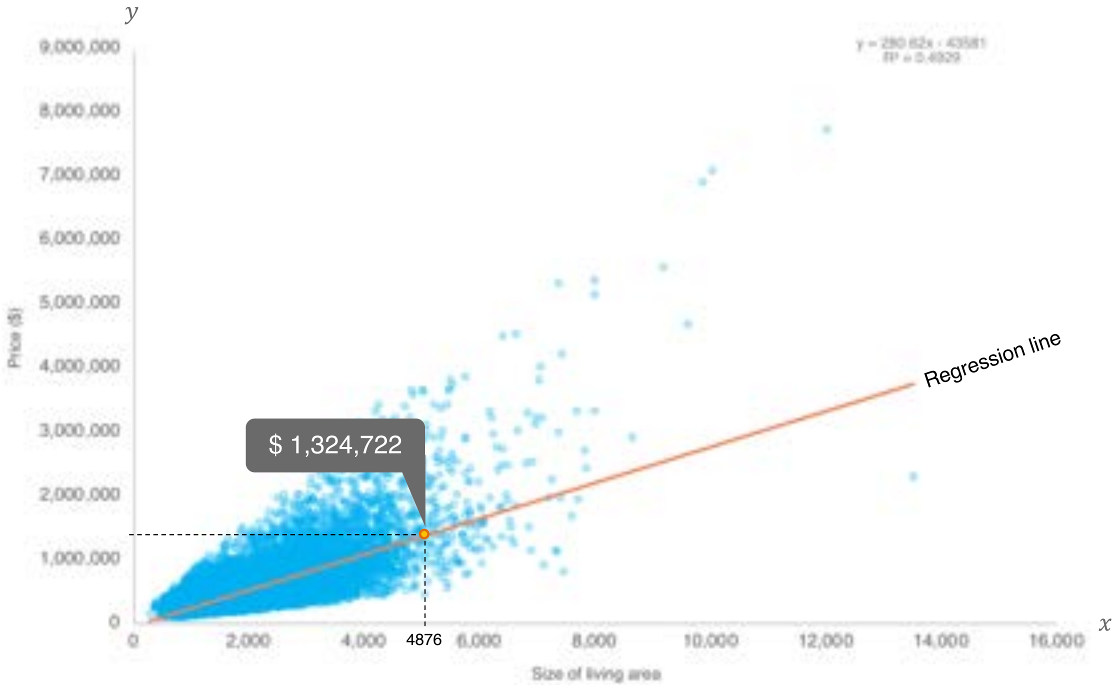
output

$h(x)$

hypothesis

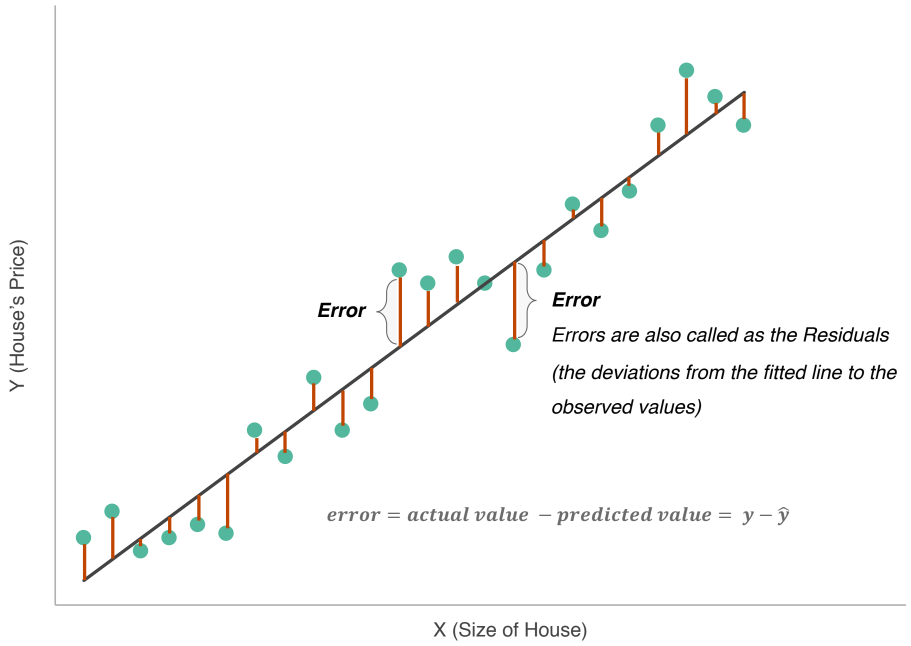
$\hat{y}$

(predicted price of house)



Linear Regression

- The goal of linear regression is to find the best fit line.
  - minimizes the sum of the “squared differences” between the points and the regression line.



$$h(x) = \theta_0 + \theta_1 x$$

How to find the appropriate parameter θ_0 and θ_1 in order to minimize the error – i.e. cost function $J(\theta_0, \theta_1)$

To **minimize** $\underbrace{\boldsymbol{J}\,(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}$

Cost function
(also known as Loss Function)

- m is number of training instances
- $\hat{y}_i$ (y hat) is the predicted value
- y_i is the actual value

- **Normal Equation (Closed Form)**

  It’s a method to solve for θ analytically.

  Using a direct “closed-form” equation that directly computes the model parameters that best fit the model to the training set (i.e., the model parameters that minimize the cost function over the training set). [1]

  It’s suitable for small feature set (e.g. < 1000 features).

- **Gradient Descent**

  Using an iterative optimization approach, called Gradient Descent (GD), that gradually tweaks the model parameters to minimize the cost function over the training set, eventually converging to the same set of parameters as the first method. [2]

  Gradient Descent is better choice than Normal Equation when there are a large number of features, or too many training instances to fit in memory.

Linear Regression

- A gradient is the slope of a function at a specific point. The gradient of loss function/cost function is equal to the derivative (slope) of the curve.

Gradient Descent algorithm

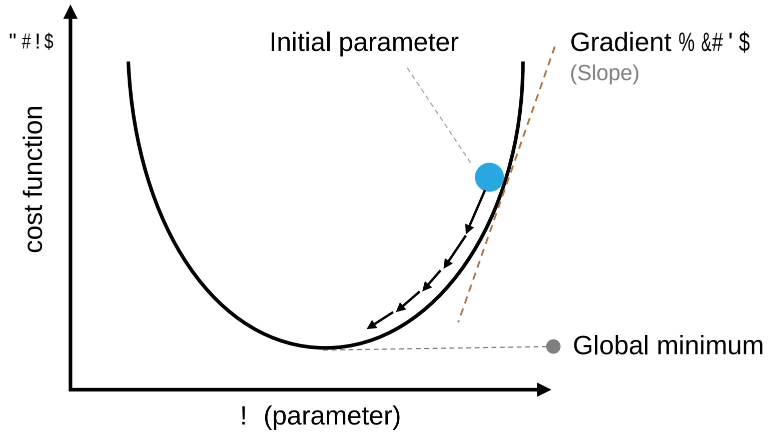
The parameter are iteratively updated in the following equation:

Learning rate (Step size)

$$'_{new} = '_{old} - (\ )\ \boxed{* \cdot \text{"\#'\$}}$$

Gradient $* \cdot \text{"(')} + \frac{,}{,\ '}\text{"(')}$

1. Pick a value for the learning rate !
2. Start with a random point '
3. Calculate the gradient * "#'$ at the point '. Follow the opposite direction of gradient to get new parameter $'_{new}$
4. Repeat until the cost function converges to the minimum

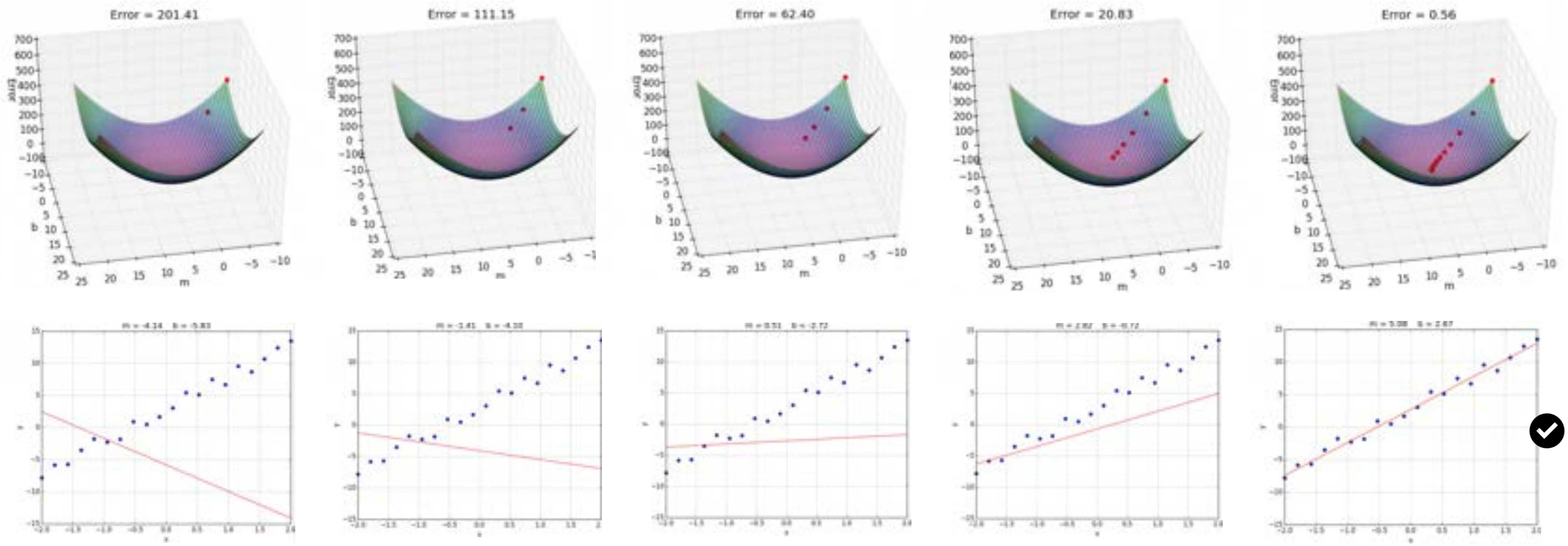


In this example, initially the slope is large and positive. So, in the update equation, ! is reduced. As ! keeps getting reduced, notice that the gradient also reduces, and hence the updates become smaller and smaller and eventually, it converges to the minimum.[1]

Linear Regression

- Find the best-fit line through Gradient Descent algorithm.

Iteratively find the minimum of cost function

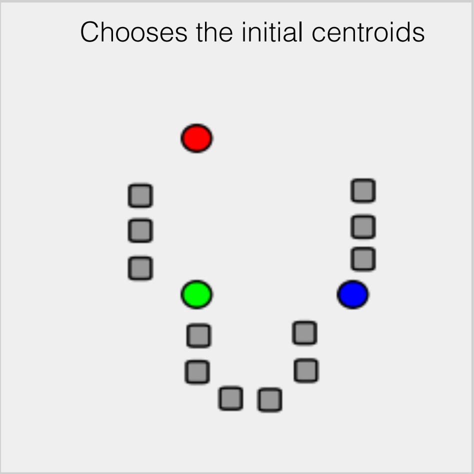


K-Mean Clustering

- Discover the structure within the un-labeled data.
- Clustering is a technique for finding similarity groups in a data, called clusters.
- It attempts to group individuals in a population together by similarity, but not driven by a specific purpose.
- Clustering is often called an unsupervised learning, as you don’t have prescribed labels in the data and no class values denoting a priori grouping of the data instances are given.

K-Mean Clustering

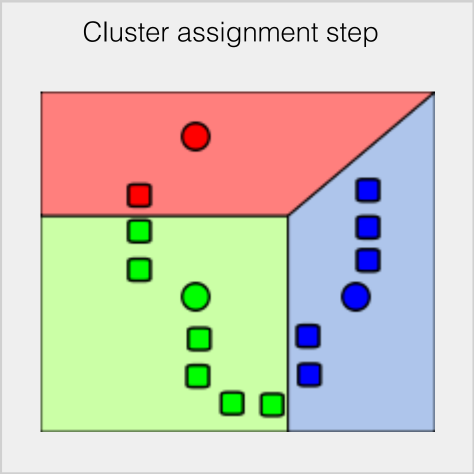
- A graphical view of K-means algorithm.



Chooses the initial centroids

1

k initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

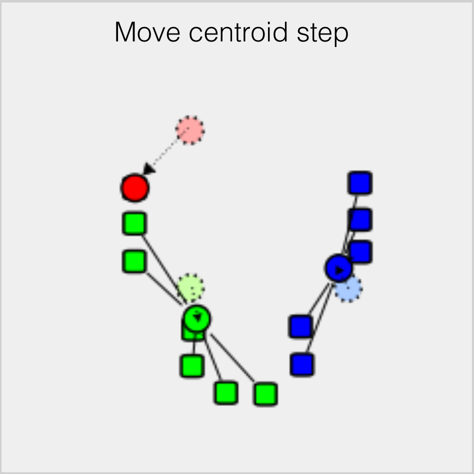


Cluster assignment step

2

k clusters are created by associating every observation with the nearest mean.

In Cluster assignment step, the algorithm goes through each of the data points and depending on which cluster is closer, whether the red cluster centroid or the blue cluster centroid or the green; It assigns the data points to one of the three cluster centroids.

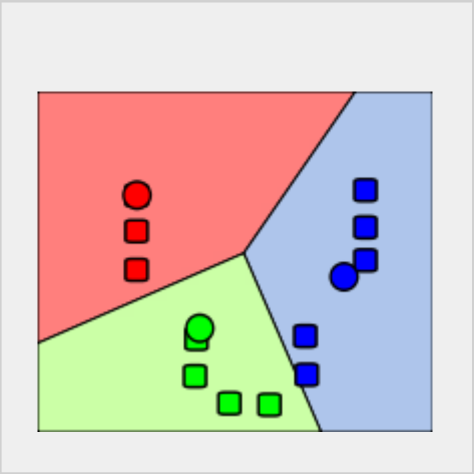


Move centroid step

3

The centroid of each of the *k* clusters becomes the new mean.

In move centroid step, K-means moves the centroids to the average of the points in a cluster. In other words, the algorithm calculates the average of all the points in a cluster and moves the centroid to that average location.



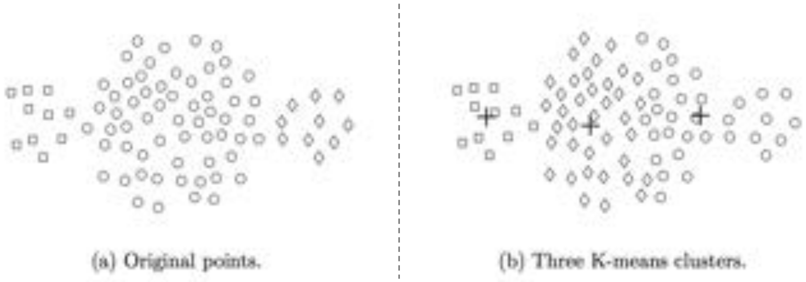
4

Steps 2 and 3 are repeated until convergence has been reached

In other words, it repeats until the centroids do not move significantly.

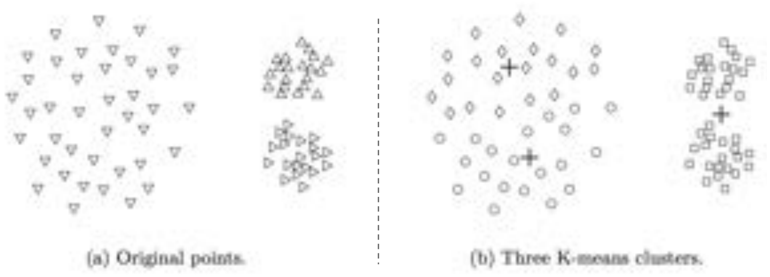
K-Mean Clustering

- Weakness of K-means
  - The number of cluster “*k*” must be specified in advance.
  - Sensitive to initial centroids selection, which leads to unwanted solution.
  - k-means can only handle numerical data.
  - The algorithm may get stuck in the local optimum.
  - Sensitive to outliers and noise, which results in an inaccurate partition.
  - K-means cannot handle non-globular clusters or clusters of different sizes and densities.


(a) Original points.

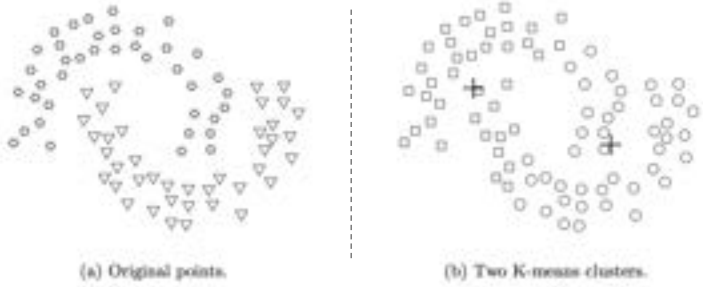
(b) Three K-means clusters.

Figure 1: K-means with clusters of <u>different size</u>


(a) Original points.

(b) Three K-means clusters.

Figure 2: K-means with clusters of <u>different density</u>


(a) Original points.

(b) Two K-means clusters.

Figure 3: K-means with <u>non-globular clusters</u>

Naïve Bayes

- Naïve Bayes is a simple but important probabilistic model
  - It based on applying Bayes’ theorem with the “naive” assumption of independence between the features.
  - It computes the conditional probability distribution of each feature given label, and then it applies Bayes’ theorem to compute the conditional probability distribution of label given an observation and use it for prediction.
  - It classifies new data based on the highest probability of its belonging to a particular class.
  - Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods.

Naïve Bayes

$$\overbrace{P(A \mid B)} = \frac{P(B \mid A)\, P(A)}{P(B)}$$

Day	Outlook	Temperature	Humidity	Wind	Play Tennis ?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Outlook	Play = Yes	Play = No	Total
Sunny	2/9	3/5	5/14
Overcast	4/9	0/5	4/14
Rain	3/9	2/5	5/14

Temperature	Play = Yes	Play = No	Total
Hot	2/9	2/5	4/14
Mild	4/9	2/5	6/14
Cool	3/9	1/5	4/14

Humidity	Play = Yes	Play = No	Total
High	3/9	4/5	7/14
Normal	6/9	1/5	7/14

Wind	Play = Yes	Play = No	Total
Strong	3/9	3/5	6/14
Weak	6/9	2/5	8/14

Naïve Bayes

If X = (Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong)， then

P (Play=Yes I X) = P (Play=Yes | Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong)

$$= \frac{\text{P(Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong I Play=Yes) * P(Play=Yes)}}{\text{P(Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong)}}$$

$$= \frac{\text{P(Outlook = Sunny I Play=Yes） * P(Temperature = Cool I Play=Yes) * P(Humidity = High I Play=Yes)* P(Wind = Strong I Play=Yes) * P(Play=Yes)}}{\text{P(Outlook=Sunny) * P(Temperature=Cool) * P(Humidity=High) * P(Wind=Strong)}}$$

$$= \frac{(2/9) * (3/9) * (3/9) * (3/9) * (9/14)}{(5/14) * (4/14) * (7/14) * (6/14)}$$

$$= \frac{0.0053}{0.02186} = \mathbf{0.2424}$$

P (Play= No I X) = P (Play= NO | Outlook = Sunny, Temperature = Cool, Humidity = High, Wind = Strong)

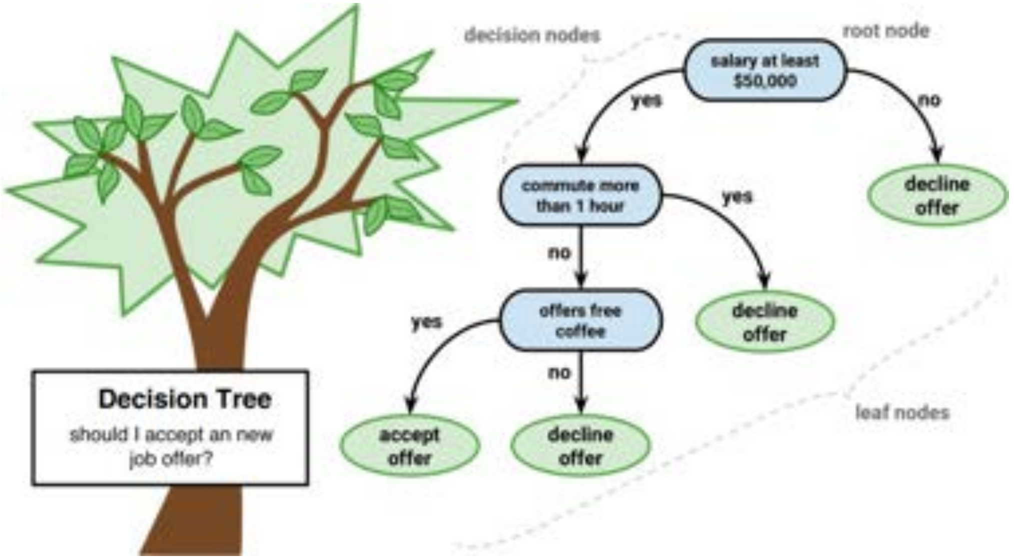
$$= \frac{(3/5) \ast (1/5) \ast (4/5) \ast (3/5) \ast (5/14)}{(5/14) * (4/14) * (7/14) * (6/14)} = \frac{0.0206}{0.02186} = \mathbf{0.9421}$$

- P(Play=Yes I X) = 0.2424
- P(Play=No I X) = 0.9421

Since 0.9421 is greater than 0.2424 then the answer is ‘no’, we cannot play a game of tennis today.

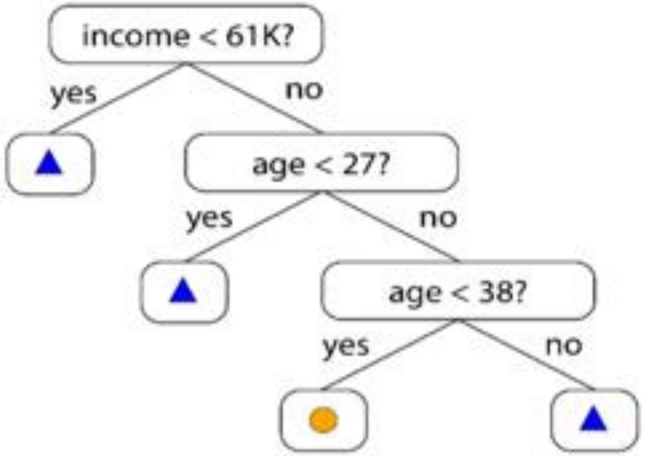
Decision Tree

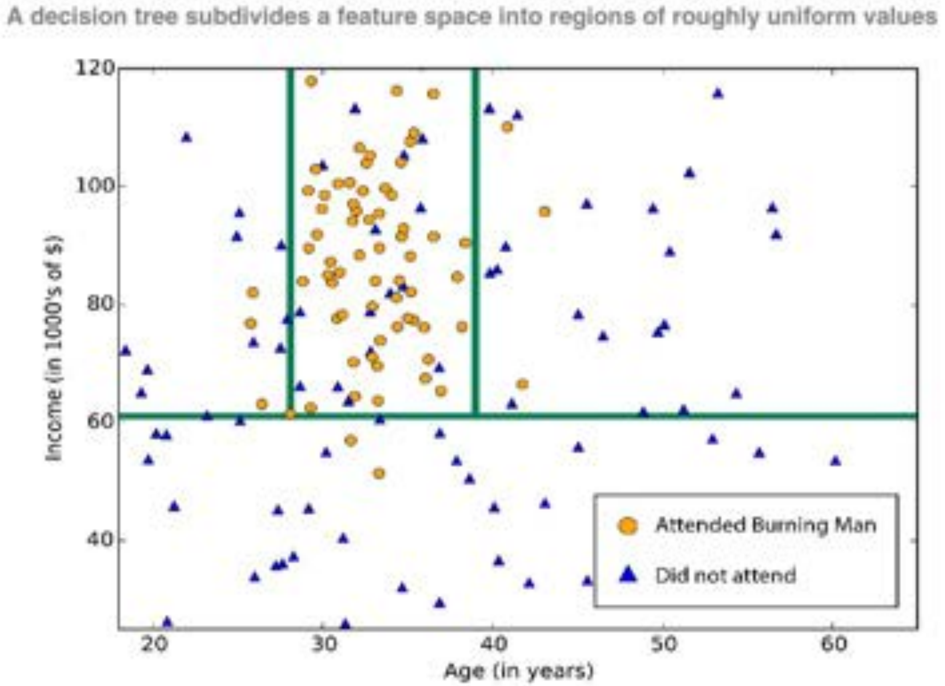
- Decision tree builds classification or regression models in the form of a tree structure.
  - predict the value of a target variable by following the decisions in the tree from the root (beginning) down to a leaf node.
  - A tree consists of branching conditions where the value of a predictor is compared to a trained weight.
  - Decision trees are prone to overfitting, additional modification, or pruning, may be used to simplify the model.



Decision Tree

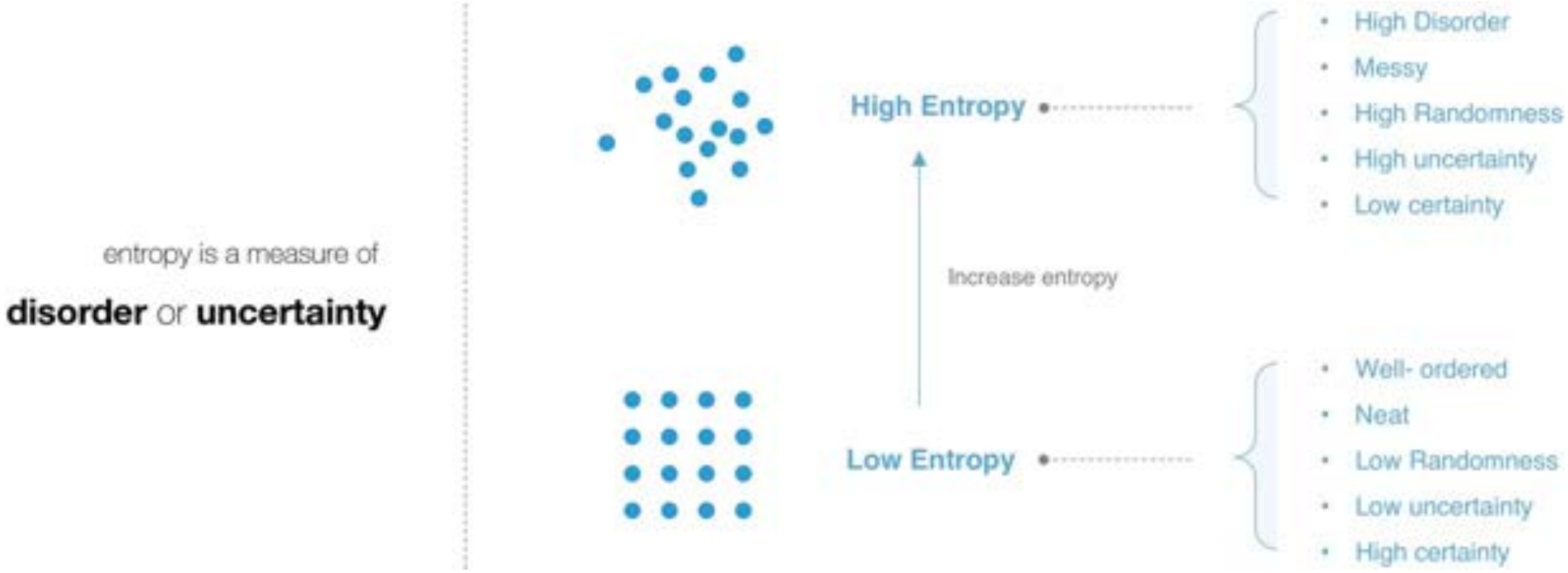
- Growing a tree involves deciding on which features to choose and what conditions to use for splitting, along with knowing when to stop.





Decision Tree

- Generally entropy is a measure of disorder or uncertainty
  - Entropy is a concept used in physics, mathematics, computer science (information theory) and other fields of science. The concept of entropy originated in thermodynamics as a measure of molecular disorder: entropy approaches zero when molecules are still and well ordered.



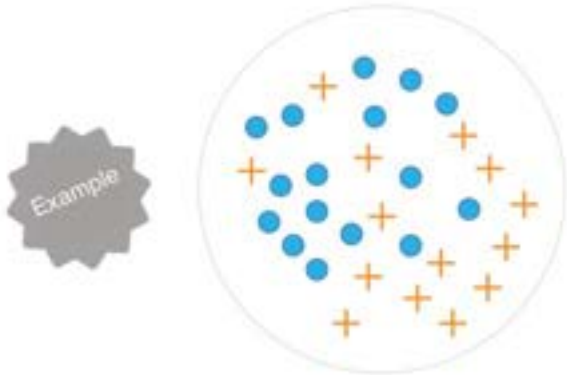
Decision Tree

- Mathematical Definition of Entropy

$$Entropy = -\sum_{i=1}^{n} p_i \log_2 p_i$$

Where p_i is the **probability** of getting the i^{th} value when randomly selecting one from the set.

In other words, Where there are n classes, and p_i is the probability an object from the i^{th} class appearing.



16/30 are blue circles:

$$\log_2\left(\frac{16}{30}\right) = -0.9\,;$$

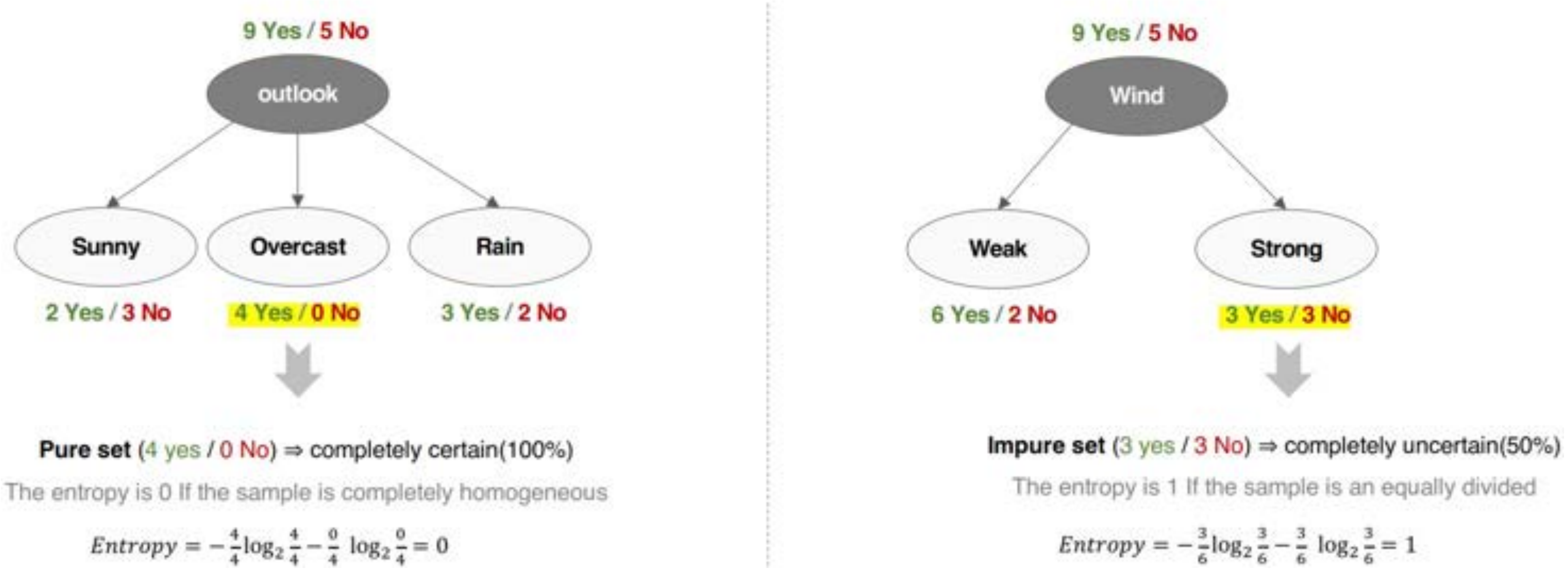
14/30 are orange crosses:

$$\log_2\left(\frac{14}{30}\right) = -1.1\,;$$

$$Entropy = -\sum_{i=1}^{n} p_i \log_2 p_i = -\left(\frac{16}{30}\right)(-0.9) - \left(\frac{14}{30}\right)(-1.1) = 0.99$$

Decision Tree

- Use entropy to measure the “purity” of the split



Pure set (4 yes / 0 No) ⇒ completely certain(100%)

The entropy is 0 If the sample is completely homogeneous

$$Entropy = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\ \log_2\frac{0}{4} = 0$$

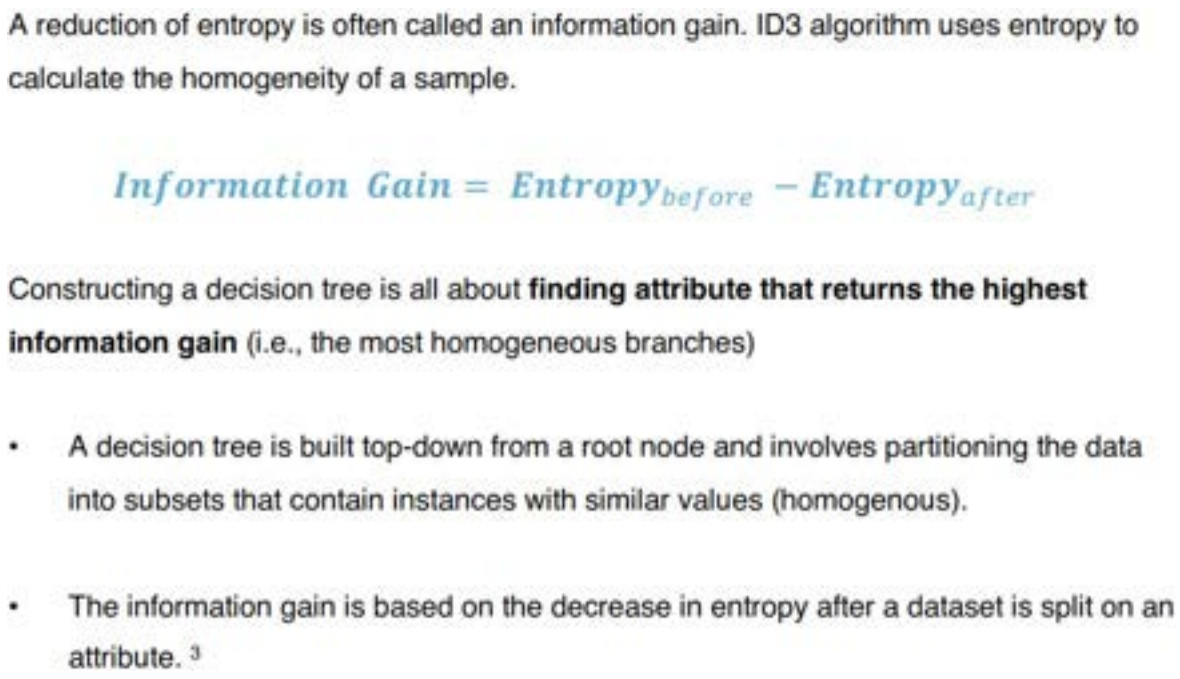
Impure set (3 yes / 3 No) ⇒ completely uncertain(50%)

The entropy is 1 If the sample is an equally divided

$$Entropy = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\ \log_2\frac{3}{6} = 1$$

Decision Tree with ID3 Algorithm

- Measures that can be used to capture the purity of split.
    - Information Gain



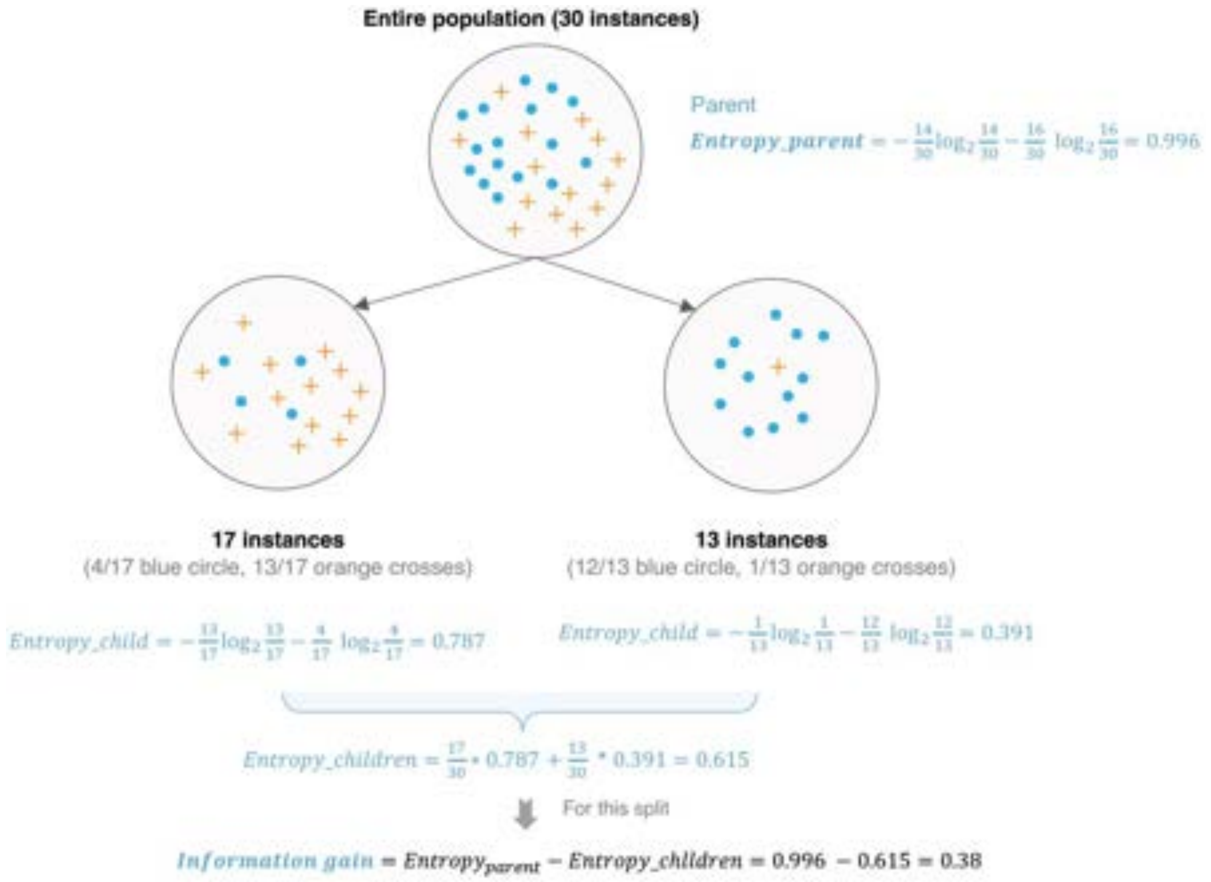
A reduction of entropy is often called an information gain. ID3 algorithm uses entropy to calculate the homogeneity of a sample.

$$Information\ Gain = Entropy_{before} - Entropy_{after}$$

Constructing a decision tree is all about **finding attribute that returns the highest information gain** (i.e., the most homogeneous branches)

- A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous).

- The information gain is based on the decrease in entropy after a dataset is split on an attribute. [3]



Entire population (30 instances)

Parent

$$Entropy\_parent = -\frac{14}{30}\log_2\frac{14}{30} - \frac{16}{30}\log_2\frac{16}{30} = 0.996$$

17 instances
(4/17 blue circle, 13/17 orange crosses)

13 instances
(12/13 blue circle, 1/13 orange crosses)

$$Entropy\_child = -\frac{13}{17}\log_2\frac{13}{17} - \frac{4}{17}\log_2\frac{4}{17} = 0.787$$

$$Entropy\_child = -\frac{1}{13}\log_2\frac{1}{13} - \frac{12}{13}\log_2\frac{12}{13} = 0.391$$

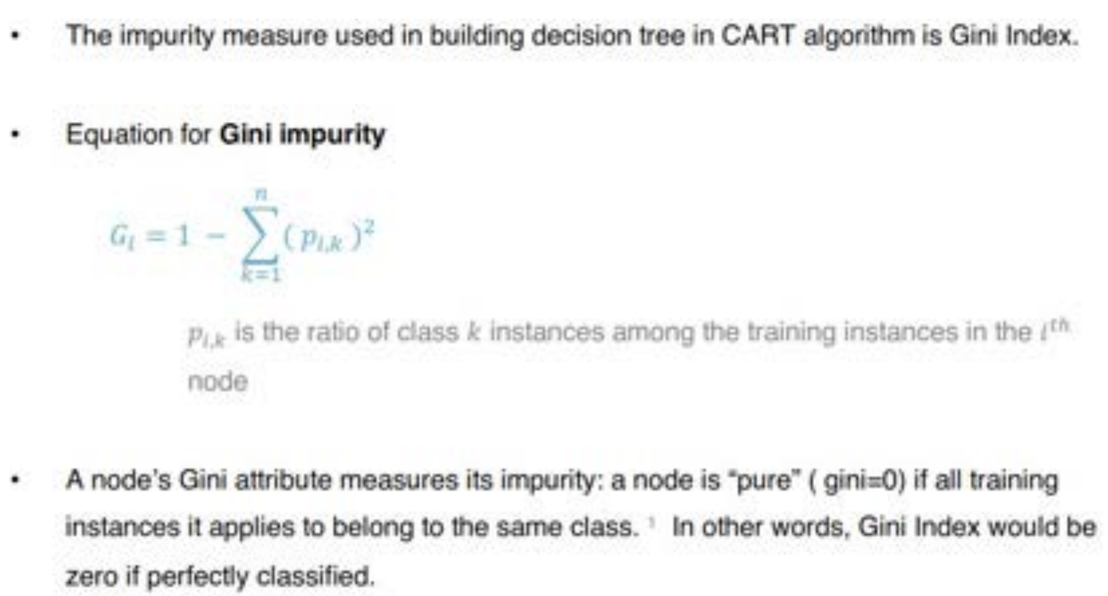
$$Entropy\_children = \frac{17}{30} * 0.787 + \frac{13}{30} * 0.391 = 0.615$$

For this split

$$Information\ gain = Entropy_{parent} - Entropy\_children = 0.996 - 0.615 = 0.38$$

Decision Tree with CART Algorithm

- Measures that can be used to capture the purity of split.
  - Gini impurity Index

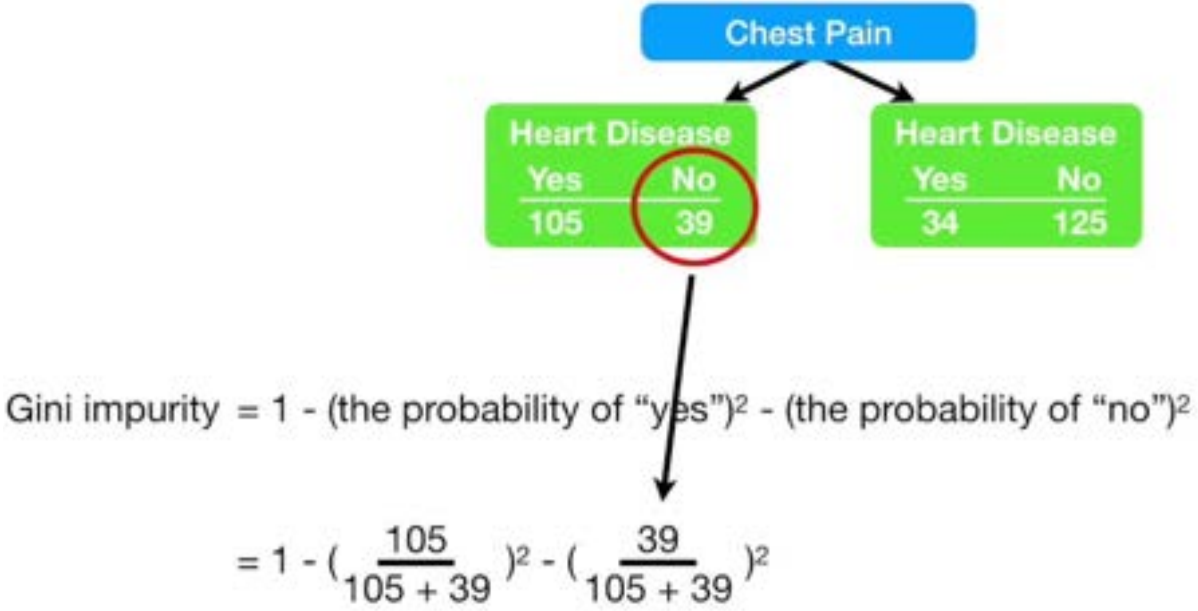


- The impurity measure used in building decision tree in CART algorithm is Gini Index.
- Equation for **Gini impurity**

$$G_i = 1 - \sum_{k=1}^{n} (p_{i,k})^2$$

$p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node

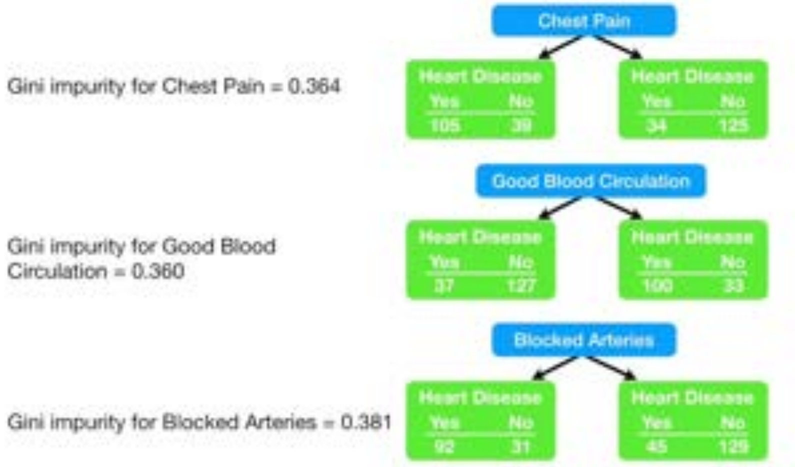
- A node's Gini attribute measures its impurity: a node is "pure" (gini=0) if all training instances it applies to belong to the same class. [1] In other words, Gini Index would be zero if perfectly classified.



Gini impurity $= 1 -$ (the probability of "yes")^2 $-$ (the probability of "no")^2

$$= 1 - \left(\frac{105}{105 + 39}\right)^2 - \left(\frac{39}{105 + 39}\right)^2$$

Decision Tree with CART Algorithm

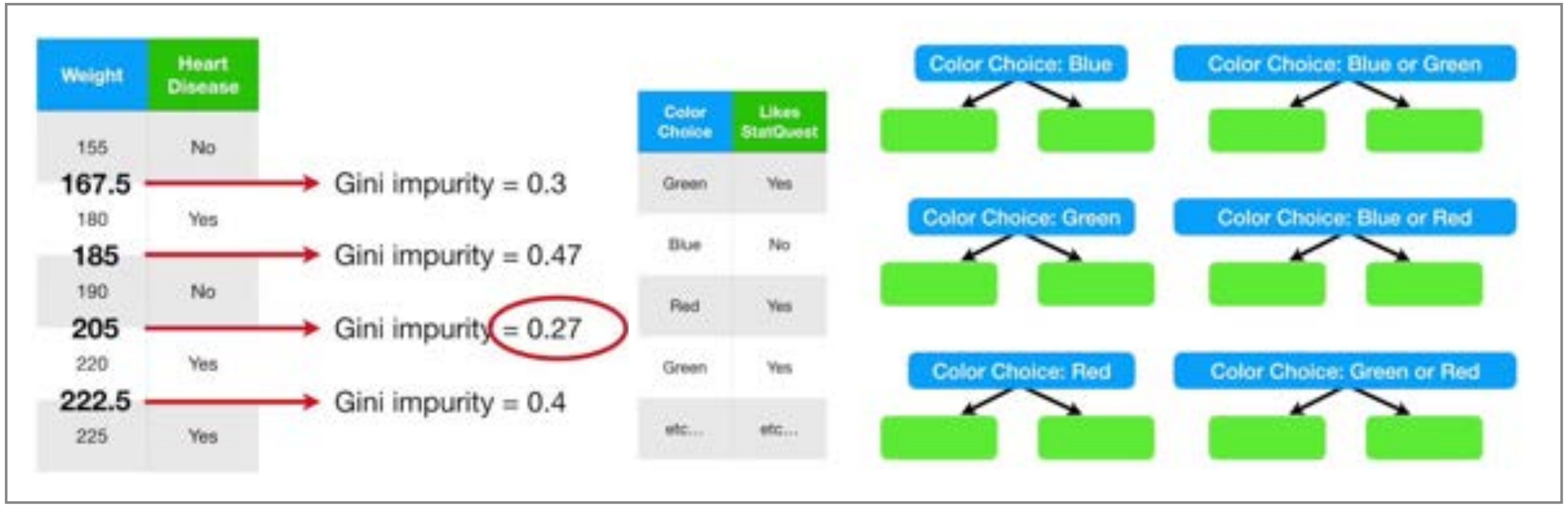


Gini impurity for Chest Pain = 0.364

Gini impurity for Good Blood Circulation = 0.360

Gini impurity for Blocked Arteries = 0.381

Chest Pain
Heart Disease Yes No 105 39
Heart Disease Yes No 34 125
Good Blood Circulation
Heart Disease Yes No 37 127
Heart Disease Yes No 100 33
Blocked Arteries
Heart Disease Yes No 92 31
Heart Disease Yes No 45 129



Weight	Heart Disease
155	No
180	Yes
190	No
220	Yes
225	Yes

167.5 → Gini impurity = 0.3

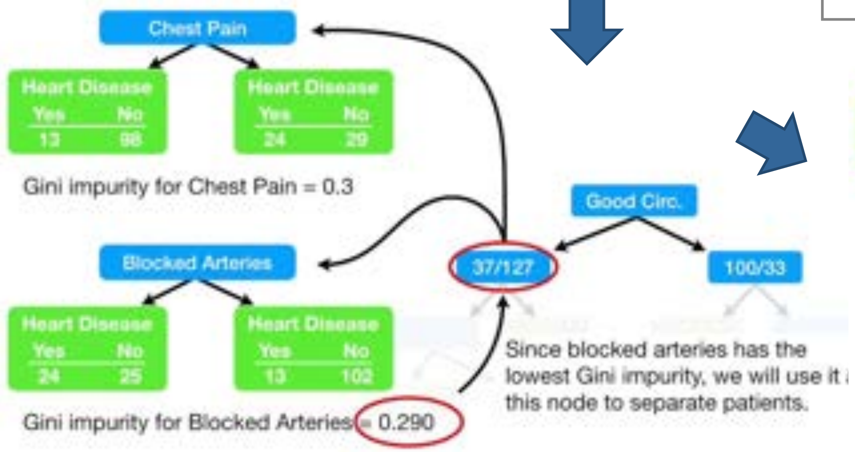
185 → Gini impurity = 0.47

205 → Gini impurity = 0.27

222.5 → Gini impurity = 0.4

Color Choice	Likes StatQuest
Green	Yes
Blue	No
Red	Yes
Green	Yes
etc...	etc...

Color Choice: Blue
Color Choice: Blue or Green
Color Choice: Green
Color Choice: Blue or Red
Color Choice: Red
Color Choice: Green or Red

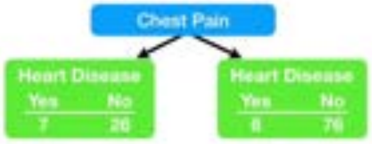


Gini impurity for Chest Pain = 0.3

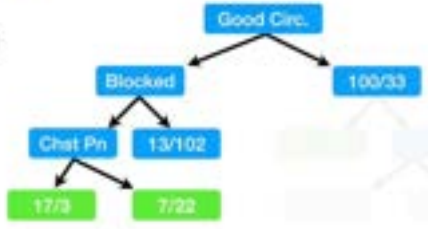
Gini impurity for Blocked Arteries = 0.290

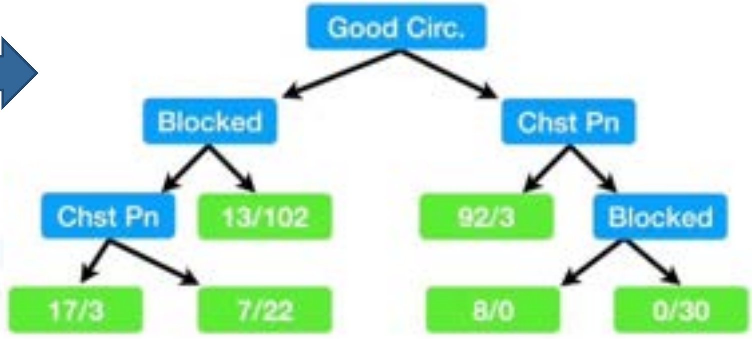
Since blocked arteries has the lowest Gini impurity, we will use it at this node to separate patients.

Chest Pain
Heart Disease Yes No 13 98
Heart Disease Yes No 24 29
Blocked Arteries
Heart Disease Yes No 24 25
Heart Disease Yes No 13 102
Good Circ.
37/127
100/33



Do these new leaves separate patients better than what we had before?



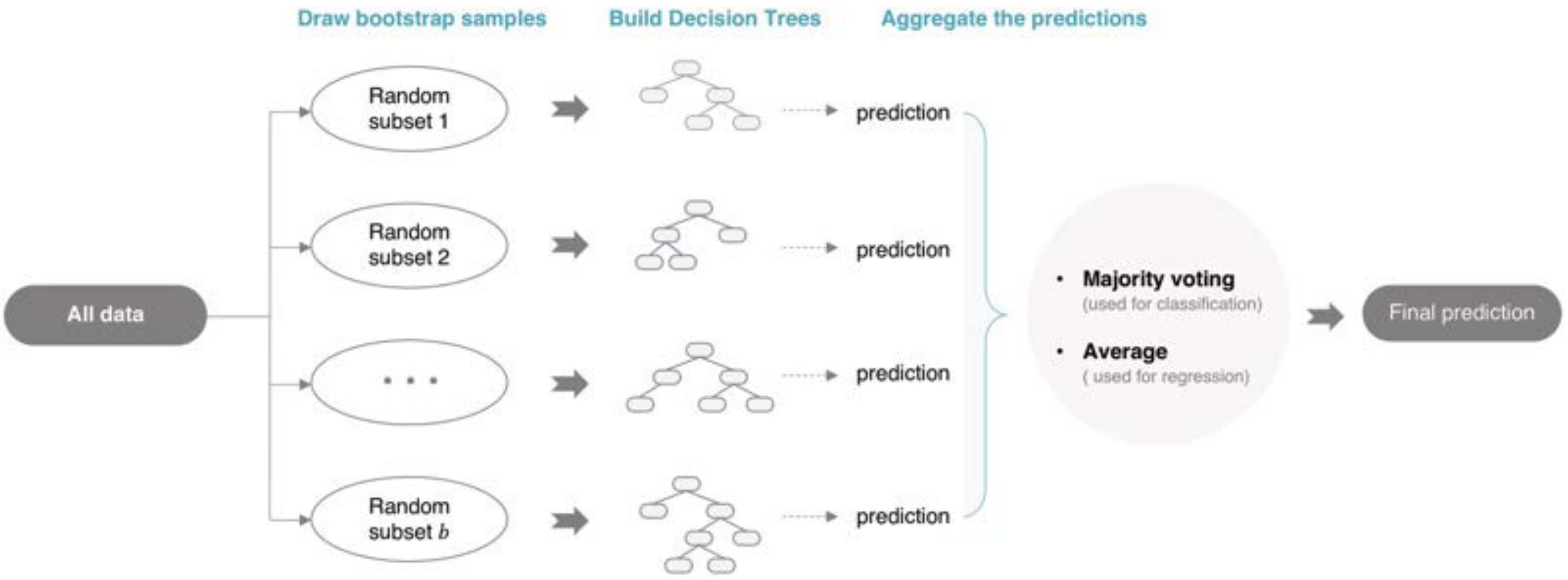


Decision Tree

- Pros and Cons of Decision Tree
  - Pros
    - Simple to understand and to interpret.
    - To build decision tree requires little data preparation.
    - Handle both continuous and categorical variables.
    - Implicitly perform feature selection.
  - Cons
    - They are prone to over-fitting.
    - create biased trees in case of unbalanced data.
    - Instability.
    - Greedy approach used by Decision tree doesn’t guarantee best solution.
    - Standard decision trees are restricted by hard, axis-aligned splits of the input space.

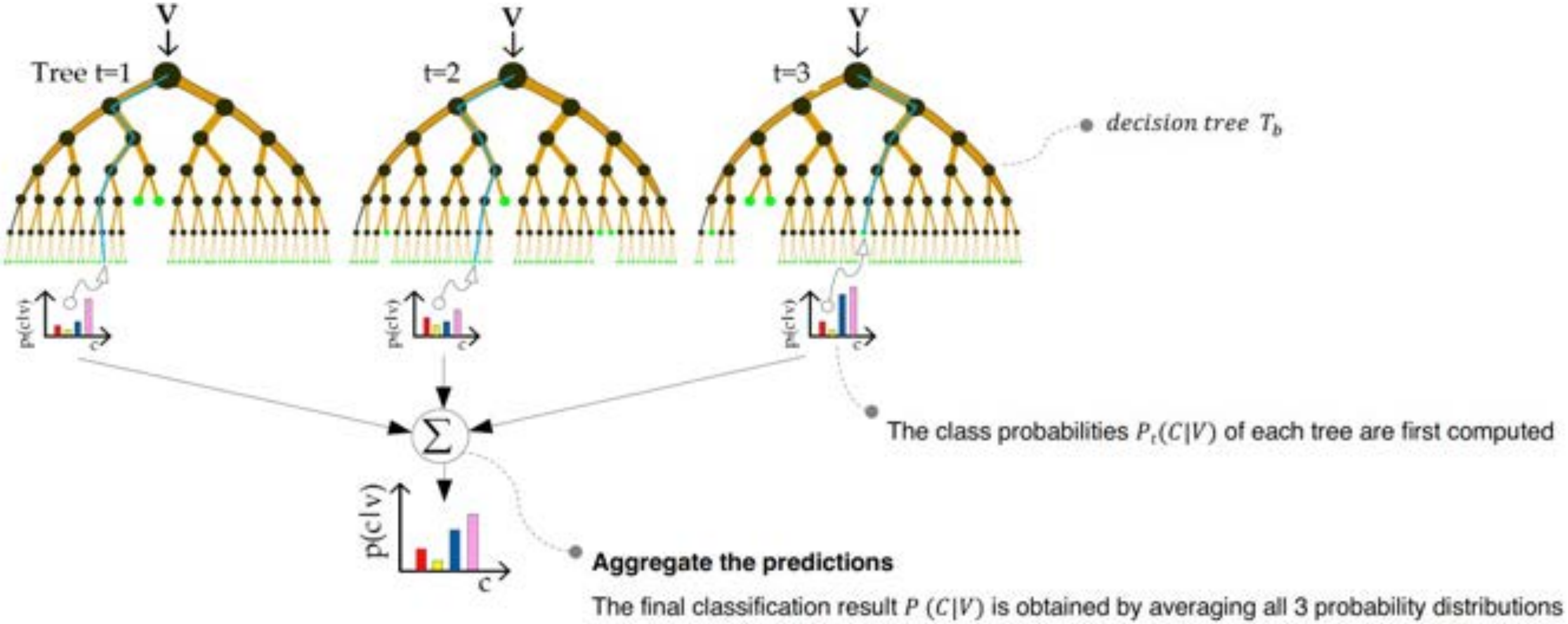
Random Forest

- Random Forest is one of the most used algorithms.
- Random Forest = Bagging + Full-grown CART decision Tree



Random Forest

- Classification example: use Random Forest to classify data
  - After training, a tree set $\{T\}$ can be obtained to predict the classes of the unseen samples by taking the majority vote from all individual classification trees.



decision tree T_b

The class probabilities $P_t(C|V)$ of each tree are first computed

Aggregate the predictions

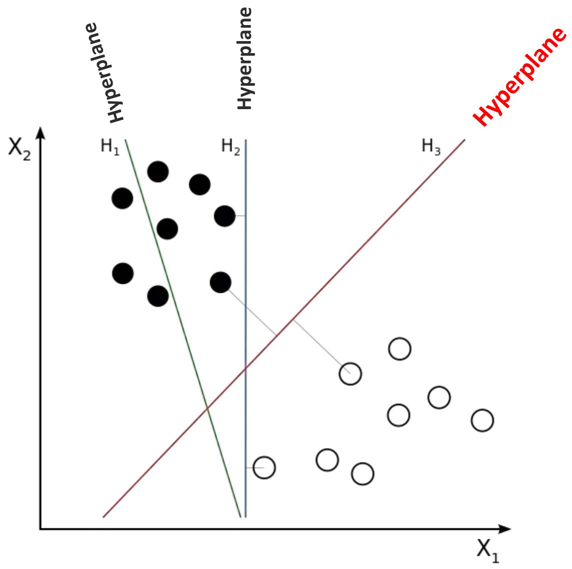
The final classification result $P\ (C|V)$ is obtained by averaging all 3 probability distributions

Random Forest

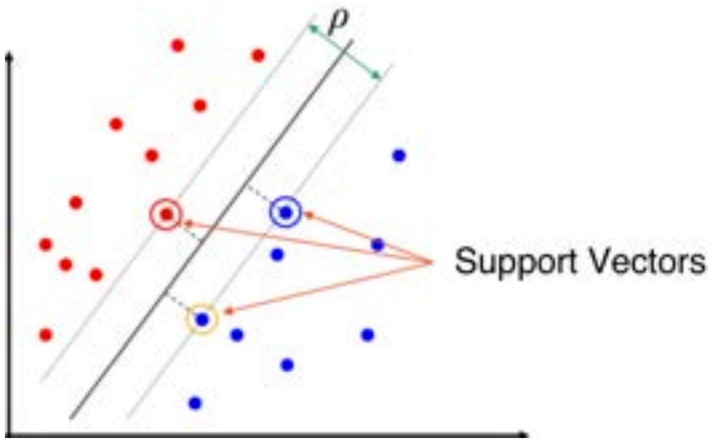
- Pros and Cons of Random Forest
  - Pros
    - Random Forest algorithms can be grown in parallel.
    - Random Forest has higher classification accuracy.
    - Able to deal with the missing value and maintain accuracy in case of missing data.
    - Help data scientists save data preparation time.
  - Cons
    - Large number of decision trees in the random forest can slow down the algorithm.
    - Good job at classification but not as good as for regression.
    - like a black box approach, random forest is not easily interpretable.

Support Vector Machine

- A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane.

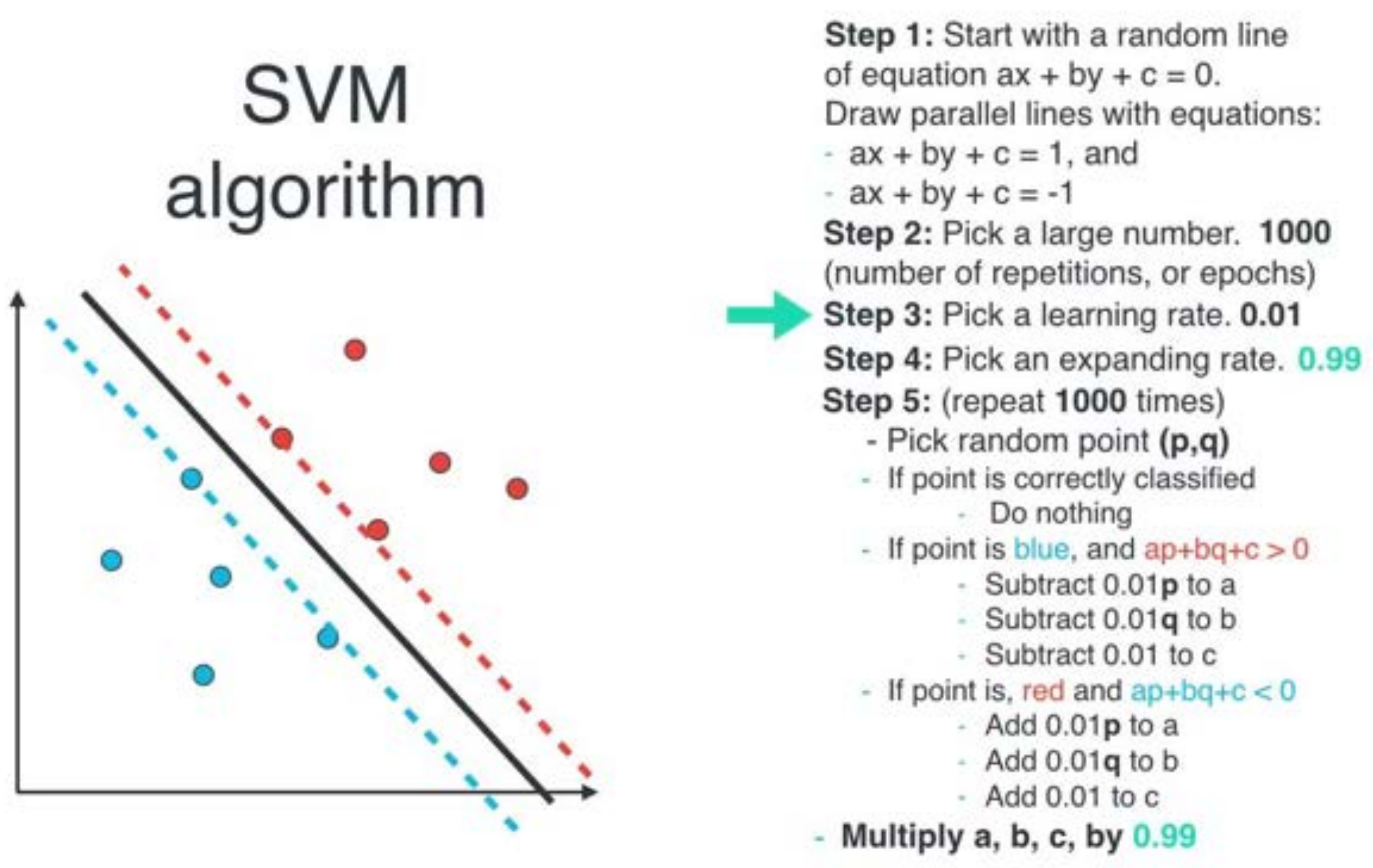


- H_1 does not separate the classes.
- H_2 does, but only with a small margin.
- **H_3 separates them with the maximum margin.**



- Examples closet to the hyper-plane are ***support vectors***
- **Margin** ρ of the separator is the distance between support vectors.

Support Vector Machine



SVM algorithm

Step 1: Start with a random line of equation ax + by + c = 0.
Draw parallel lines with equations:
- ax + by + c = 1, and
- ax + by + c = -1

Step 2: Pick a large number. **1000** (number of repetitions, or epochs)

Step 3: Pick a learning rate. **0.01**

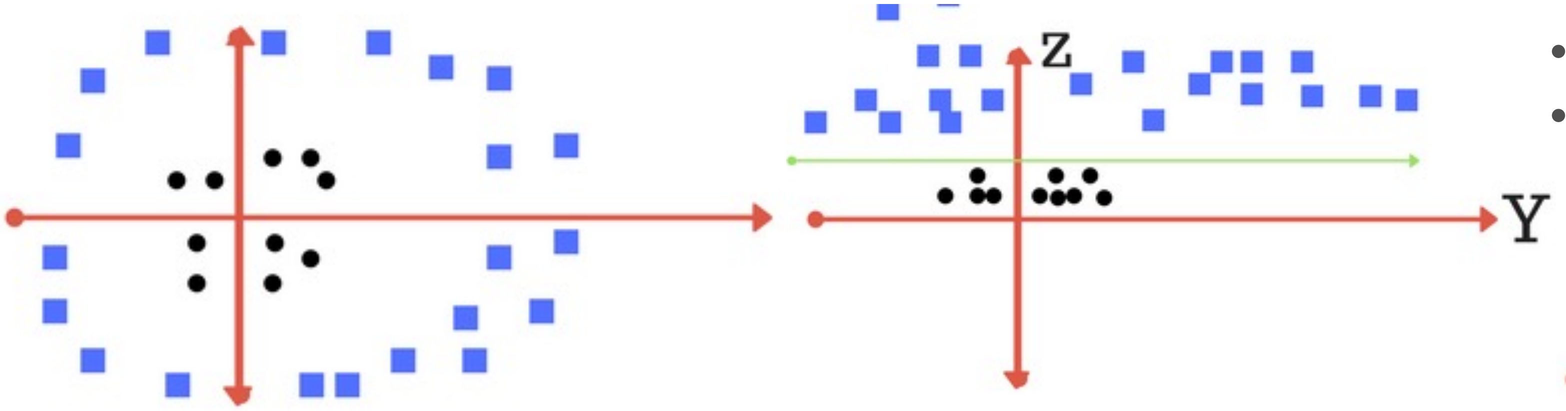
Step 4: Pick an expanding rate. 0.99

Step 5: (repeat **1000** times)

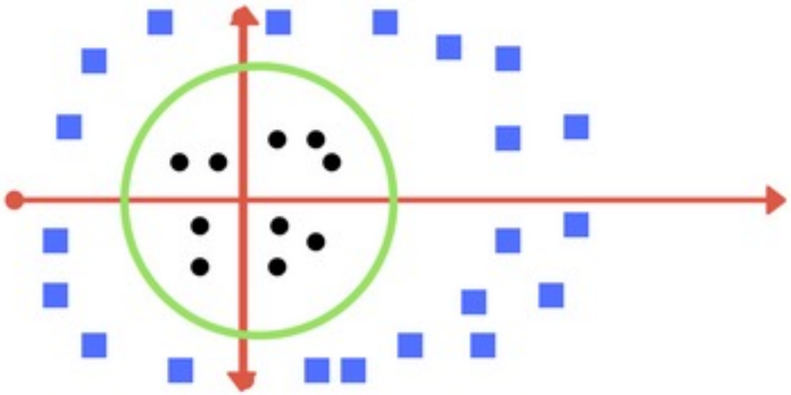
- Pick random point **(p,q)**
- If point is correctly classified
    - Do nothing
- If point is blue, and ap+bq+c > 0
    - Subtract 0.01**p** to a
    - Subtract 0.01**q** to b
    - Subtract 0.01 to c
- If point is, red and ap+bq+c < 0
    - Add 0.01**p** to a
    - Add 0.01**q** to b
    - Add 0.01 to c
- **Multiply a, b, c, by** 0.99

Support Vector Machine

- SVMs sometimes use a kernel transform to transform nonlinearly separable data into higher dimensions where a linear decision boundary can be found, the kernel trick.

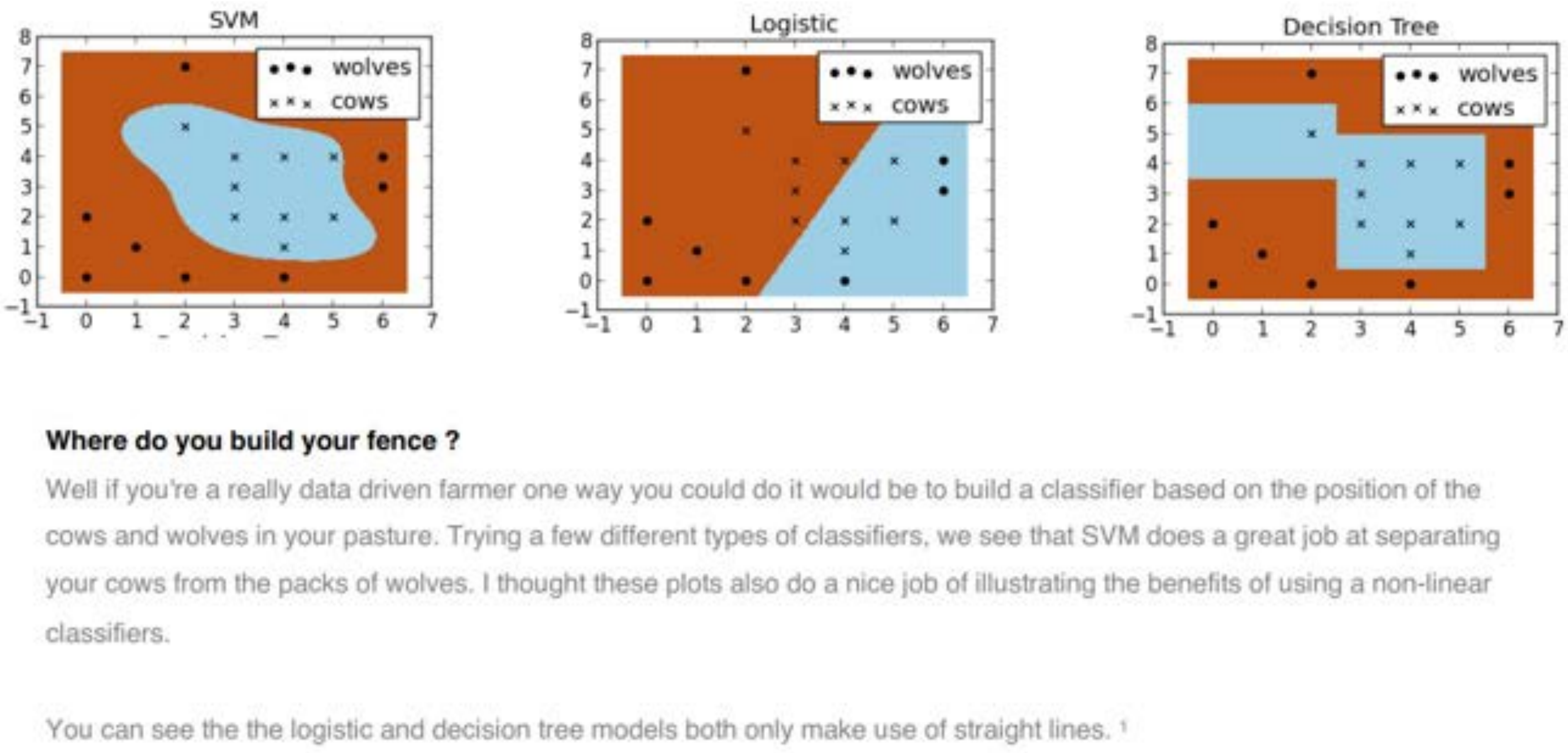


- apply transformation
- $z = x^2 + y^2$
- clear separation is visible
- transform back to original plane



Support Vector Machine

- SVM using a Non-Linear Kernel



Where do you build your fence ?

Well if you're a really data driven farmer one way you could do it would be to build a classifier based on the position of the cows and wolves in your pasture. Trying a few different types of classifiers, we see that SVM does a great job at separating your cows from the packs of wolves. I thought these plots also do a nice job of illustrating the benefits of using a non-linear classifiers.

You can see the the logistic and decision tree models both only make use of straight lines. [1]

Support Vector Machine

- Pros and Cons of Support Vector Machine
  - Pros
    - The training is relatively easy.
    - No local optimal, unlike in neural networks.
    - SVMs have a regularization parameter, which can help avoid over-fitting.
    - Effective in high dimensional spaces.
  - Cons
    - For classification, the SVM is only directly applicable for two-class tasks.
    - SVMs do not directly provide probability estimates.
    - Parameters of a solved model are difficult to interpret.
    - Long training time on large data sets.
    - Choosing a “good” kernel function can be tricky.

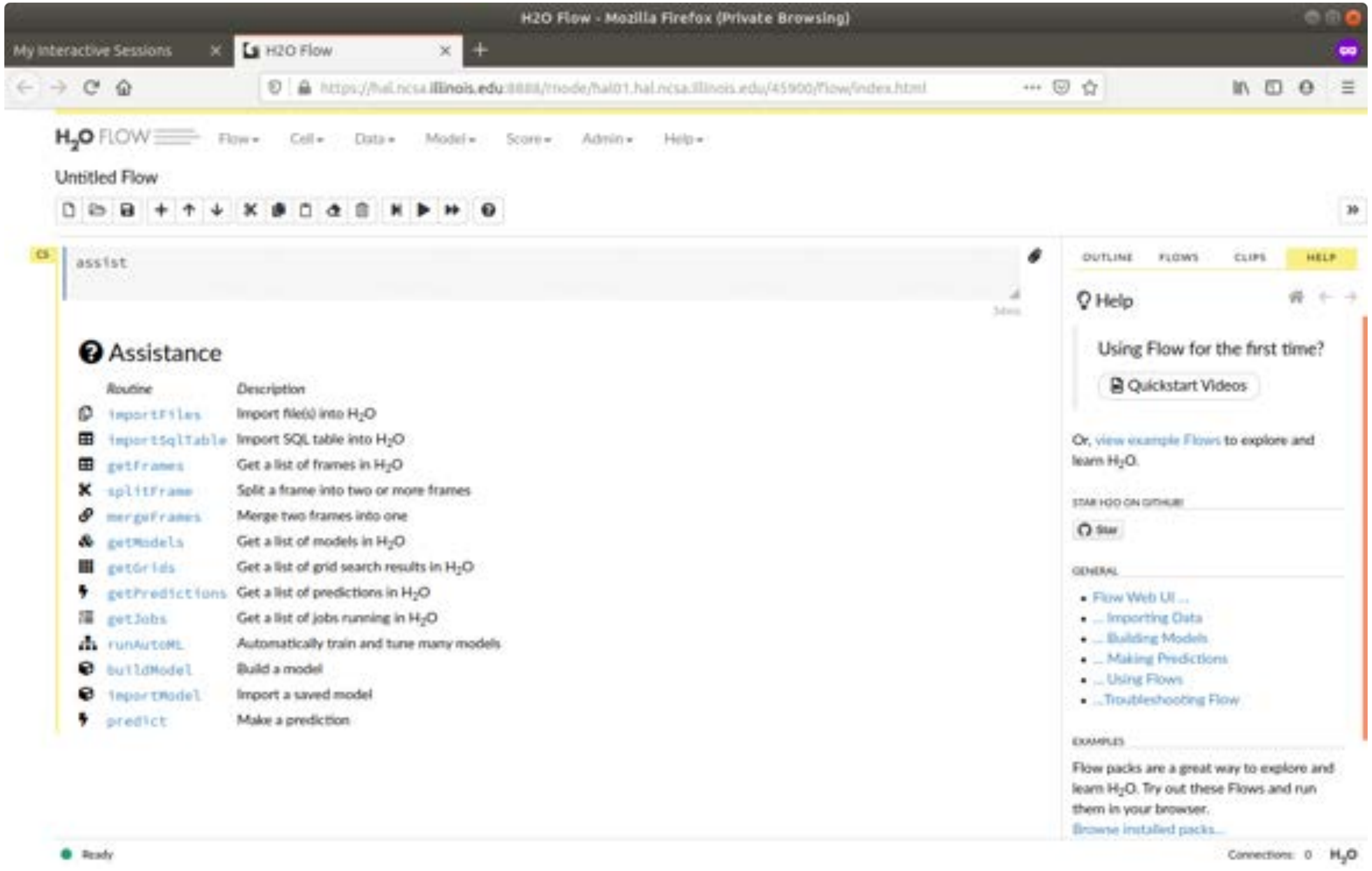
Now it is

Hands-on time, but first let’s get familiar with H2O Flow.

(Recommended Web Browser : Firefox)

(ReservationName=uiuc_21)

H2O Flow



```
assist
```

Assistance

Routine	Description
importFiles	Import file(s) into H_2O
importSqlTable	Import SQL table into H_2O
getFrames	Get a list of frames in H_2O
splitFrame	Split a frame into two or more frames
mergeFrames	Merge two frames into one
getModels	Get a list of models in H_2O
getGrids	Get a list of grid search results in H_2O
getPredictions	Get a list of predictions in H_2O
getJobs	Get a list of jobs running in H_2O
runAutoML	Automatically train and tune many models
buildModel	Build a model
importModel	Import a saved model
predict	Make a prediction

Help

Using Flow for the first time?

Quickstart Videos

Or, view example Flows to explore and learn H_2O.

STAR H2O ON GITHUB!

Star

GENERAL

- Flow Web UI ...
- ... Importing Data
- ... Building Models
- ... Making Predictions
- ... Using Flows
- ...Troubleshooting Flow

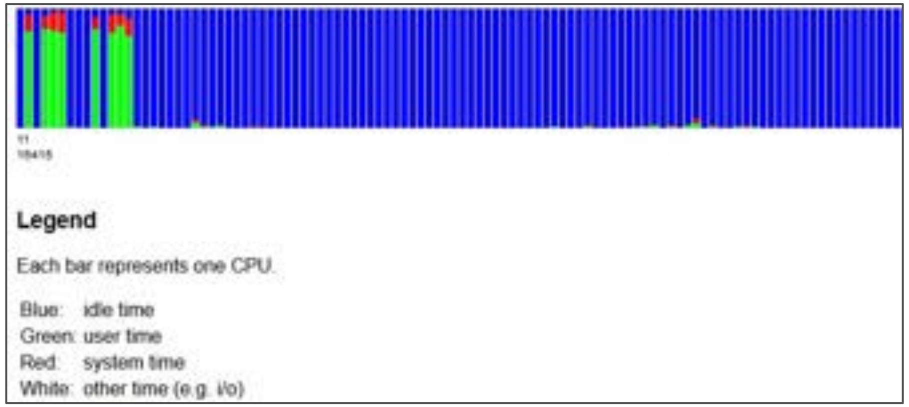
EXAMPLES

Flow packs are a great way to explore and learn H_2O. Try out these Flows and run them in your browser.

Browse installed packs...

H2O Flow

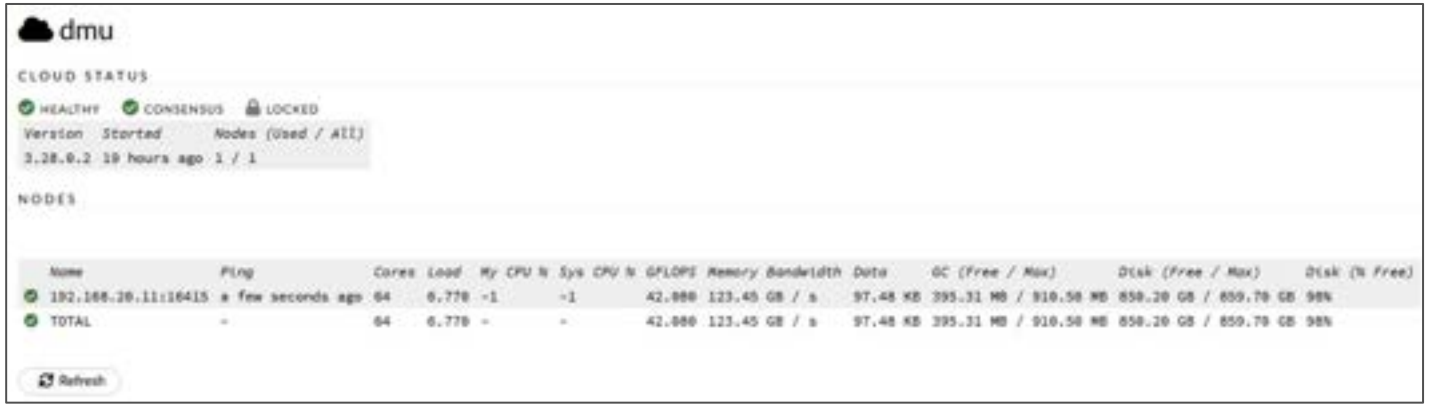
- Admin
  - Jobs / Cluster Status / Water Meter



Legend

Each bar represents one CPU.

Blue: idle time
Green: user time
Red: system time
White: other time (e.g. i/o)


dmu

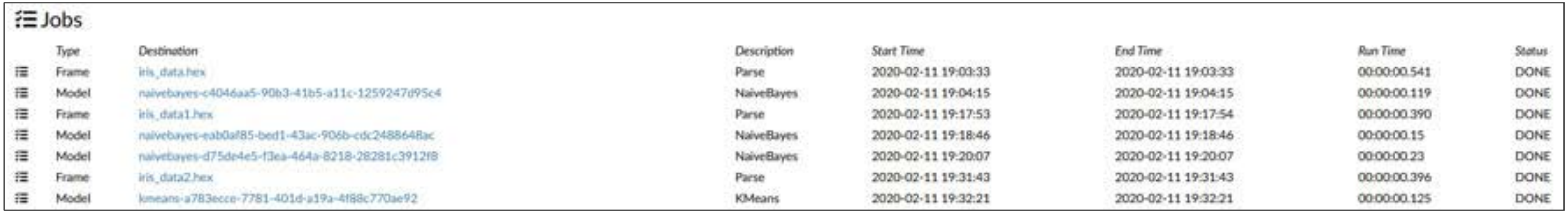
CLOUD STATUS

✓ HEALTHY ✓ CONSENSUS 🔒 LOCKED

```
Version   Started       Nodes (Used / All)
3.28.0.2  10 hours ago  1 / 1
```

NODES

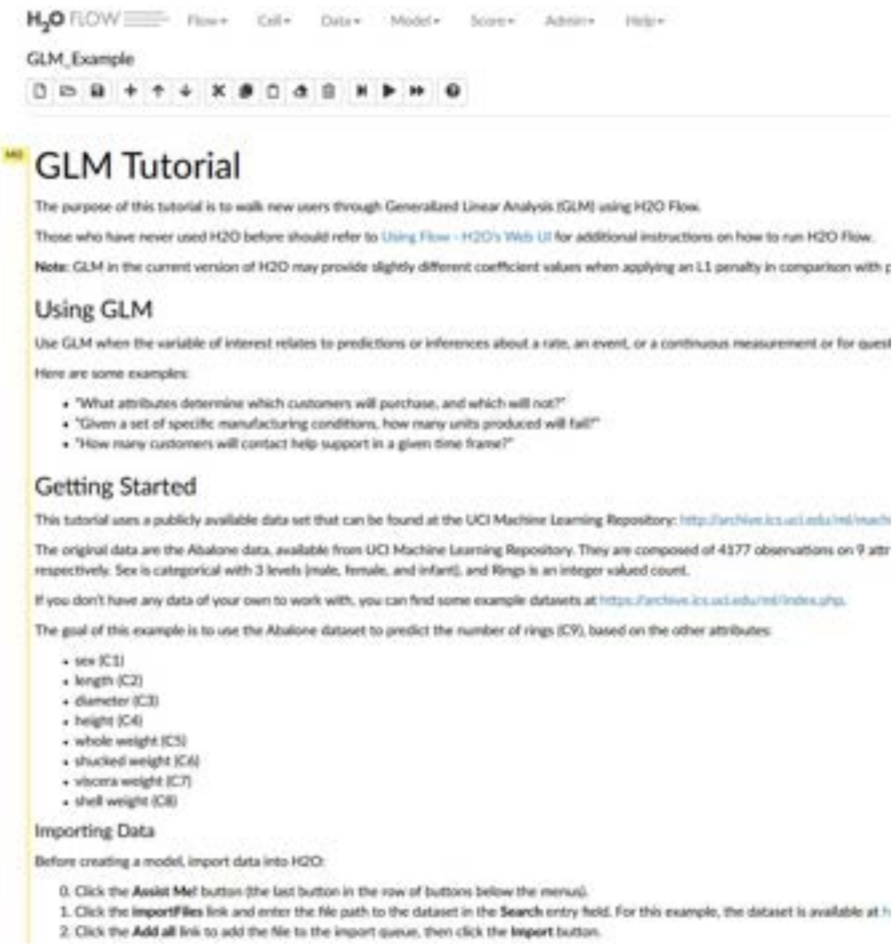
```
  Name                 Ping                Cores Load   My CPU % Sys CPU % GFLOPS Memory Bandwidth Data     GC (Free / Max)         Disk (Free / Max)         Disk (% Free)
✓ 192.168.20.11:16415  a few seconds ago   64    6.770  -1       -1        42.000 123.45 GB / s    97.48 KB 395.31 MB / 910.50 MB   850.20 GB / 859.70 GB   98%
✓ TOTAL                -                   64    6.770  -        -         42.000 123.45 GB / s    97.48 KB 395.31 MB / 910.50 MB   850.20 GB / 859.70 GB   98%
```


Jobs

	Type	Destination	Description	Start Time	End Time	Run Time	Status
☰	Frame	iris_data.hex	Parse	2020-02-11 19:03:33	2020-02-11 19:03:33	00:00:00.541	DONE
☰	Model	naivebayes-c4046aa5-90b3-41b5-a11c-1259247d95c4	NaiveBayes	2020-02-11 19:04:15	2020-02-11 19:04:15	00:00:00.119	DONE
☰	Frame	iris_data1.hex	Parse	2020-02-11 19:17:53	2020-02-11 19:17:54	00:00:00.390	DONE
☰	Model	naivebayes-eab0af85-bed1-43ac-906b-cdc2488648ac	NaiveBayes	2020-02-11 19:18:46	2020-02-11 19:18:46	00:00:00.15	DONE
☰	Model	naivebayes-d75de4e5-f3ea-464a-8218-28281c3912f8	NaiveBayes	2020-02-11 19:20:07	2020-02-11 19:20:07	00:00:00.23	DONE
☰	Frame	iris_data2.hex	Parse	2020-02-11 19:31:43	2020-02-11 19:31:43	00:00:00.396	DONE
☰	Model	kmeans-a783ecce-7781-401d-a19a-4f88c770ae92	KMeans	2020-02-11 19:32:21	2020-02-11 19:32:21	00:00:00.125	DONE

H2O Flow

- Help
  - View example Flows
    - GBM_Example.flow
    - DeepLearning_MNIST.flow
    - GLM_Example.flow
    - DRF_Example.flow
    - K-Means_Example.flow
    - Million_Songs.flow
    - KDDCup2009_Churn.flow
    - QuickStartVideos.flow
    - Airlines_Delay.flow
    - GBM_Airlines_Classification.flow
    - GBM_GridSearch.flow
    - RandomData_Benchmark_Small.flow
    - GBM_TuningGuide.flow
    - XGBoost_Example.flow



GLM Tutorial

The purpose of this tutorial is to walk new users through Generalized Linear Analysis (GLM) using H2O Flow.

Those who have never used H2O before should refer to Using Flow - H2O's Web UI for additional instructions on how to run H2O Flow.

Note: GLM in the current version of H2O may provide slightly different coefficient values when applying an L1 penalty in comparison with p

Using GLM

Use GLM when the variable of interest relates to predictions or inferences about a rate, an event, or a continuous measurement or for quest

Here are some examples:

- "What attributes determine which customers will purchase, and which will not?"
- "Given a set of specific manufacturing conditions, how many units produced will fail?"
- "How many customers will contact help support in a given time frame?"

Getting Started

This tutorial uses a publicly available data set that can be found at the UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/machi

The original data are the Abalone data, available from UCI Machine Learning Repository. They are composed of 4177 observations on 9 attr respectively. Sex is categorical with 3 levels (male, female, and infant), and Rings is an integer valued count.

If you don't have any data of your own to work with, you can find some example datasets at https://archive.ics.uci.edu/ml/index.php.

The goal of this example is to use the Abalone dataset to predict the number of rings (C9), based on the other attributes:

- sex (C1)
- length (C2)
- diameter (C3)
- height (C4)
- whole weight (C5)
- shucked weight (C6)
- viscera weight (C7)
- shell weight (C8)

Importing Data

Before creating a model, import data into H2O:

0. Click the **Assist Me!** button (the last button in the row of buttons below the menus).
1. Click the **importFiles** link and enter the file path to the dataset in the **Search** entry field. For this example, the dataset is available at h
2. Click the **Add all** link to add the file to the import queue, then click the **Import** button.

H2O Flow

- Import Data
- Parse Data
- Split Data
- Build Model
- Predict
- Save Model

K-Mean Clustering with H2O

- Seeds Data Set
- Measurements of geometrical properties of kernels belonging to three different varieties of wheat.
    - area A
    - perimeter P
    - compactness C = 4*pi*A/P^2
    - length of kernel
    - width of kernel
    - asymmetry coefficient
    - length of kernel groove

K-Mean Clustering with H2O

- Import Data:
  - importFiles ["http://s3.amazonaws.com/h2o-public-test-data/smalldata/flow_examples/seeds_dataset.txt"]
- Parse Data:
  - [“separator:9”, “number_columns:8”]
- Build Model:
  - [“K-Means”, “K:3”, “Max_iterations:100”]

Distributed Random Forest on H2O

- Internet Advertisement Data Set
  - This dataset represents a set of possible advertisements on Internet pages.
  - The features encode the geometry of the image (if available) as well as phrases occurring in the URL, the image's URL and alt text, the anchor text, and words occurring near the anchor text.
  - The task is to predict whether an image is an advertisement ("ad") or not ("nonad").

Distributed Random Forest on H2O

- Import Data:
  - importFiles ["https://s3.amazonaws.com/h2o-public-test-data/smalldata/flow_examples/ad.data.gz"]
- Parse Data:
  - [destination_frame: "ad.hex“, parse_type: "CSV“, separator: 44, number_columns: 1559, single_quotes: false]
- Build Model:
  - buildModel 'drf', {"training_frame":"ad.hex", "response_column":"C1559", "ntrees":"10", "max_depth":20, "min_rows":10, "nbins":20, "mtries":"1000", "sample_rate":0.6666667, "build_tree_one_node":false, "balance_classes":false, "class_sampling_factors":[], "max_after_balance_size":5, "seed":0}

AutoML

- The term “AutoML” (Automatic Machine Learning) refers to automated methods for model selection and/or hyperparameter optimization.
    - To enable non-experts to train high quality machine learning models.
    - To improve the efficiency of finding optimal solutions to machine learning problems.
    - explore a variety of algorithms such as Gradient Boosting Machines (GBMs), Random Forests, GLMs, and Deep Neural Networks.
- No Free Lunch Here, AutoML is slow due to heavy workload.

AutoML

- importFiles ["https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"]
- parseFiles
  - [parse_type: "CSV“, separator: 44, number_columns: 5]
- splitFrame
  - ["iris_data.hex", [0.75], ["frame_0.750","frame_0.250"], 174460]
- runAutoML
  - max_runtime_secs: 300

Thank You for Your Time !

ILLINOIS

NCSA | National Center for
Supercomputing Applications