Hands on Introduction to Machine Learning with IBM Watson Studio

Power of data. Simplicity of design. Speed of innovation.

Bernie Beekman
Michael Cronk
James Parry
Prithvi Rao

Agenda

Time	Description
09:00 AM – 09:15 AM	Welcome, Agenda
09:15 AM – 10:00 AM	Introduction to Machine Learning (Overview, Data) Lab 1-2 Overview
10:00 AM – 10:15 AM	**Lab-1 Set up Environment**
10:15 AM – 11:00 AM	**Lab-2 Data Refinery**
11:00 AM – 12:00 PM	Introduction to Machine Learning (Modeling/Evaluation) Introduction to Watson Studio
12:00 PM – 12:30 PM	**Lunch Break**
12:30 PM – 12:45 PM	Lab 3,4,5, Overview
12:45 PM – 01:30 PM	**Lab 3 – SPSS Modeler**
01:30 PM – 02:00 PM	**Lab 4 – Auto AI**
02:00 PM – 02:30 PM	**Lab 5 – Heart Disease Notebook**
02:30 PM – 02:45 PM	Introduction to Watson OpenScale Lab 6 Overview
02:45 PM – 03:30 PM	**Lab-6 – Watson OpenScale**
03:30 PM – 04:00 PM	Introduction to Neural Networks, Adversarial Robustness Toolkit Lab 7-8 Overview
04:00 PM – 04:30 PM	**Lab 7- Neural Network Lab**
04:30 PM – 05:00 PM	**Lab 8 – Adversarial Robustness Toolkit**

Introduction to Machine Learning

- Overview
- Data Science Methodology
- Data Understanding
- Data Preparation

------------Labs -------------------------------------

- Categories of Machine Learning
- Learning Challenges
- Machine Learning Algorithms
- Evaluation

------------Labs--------------------------------------

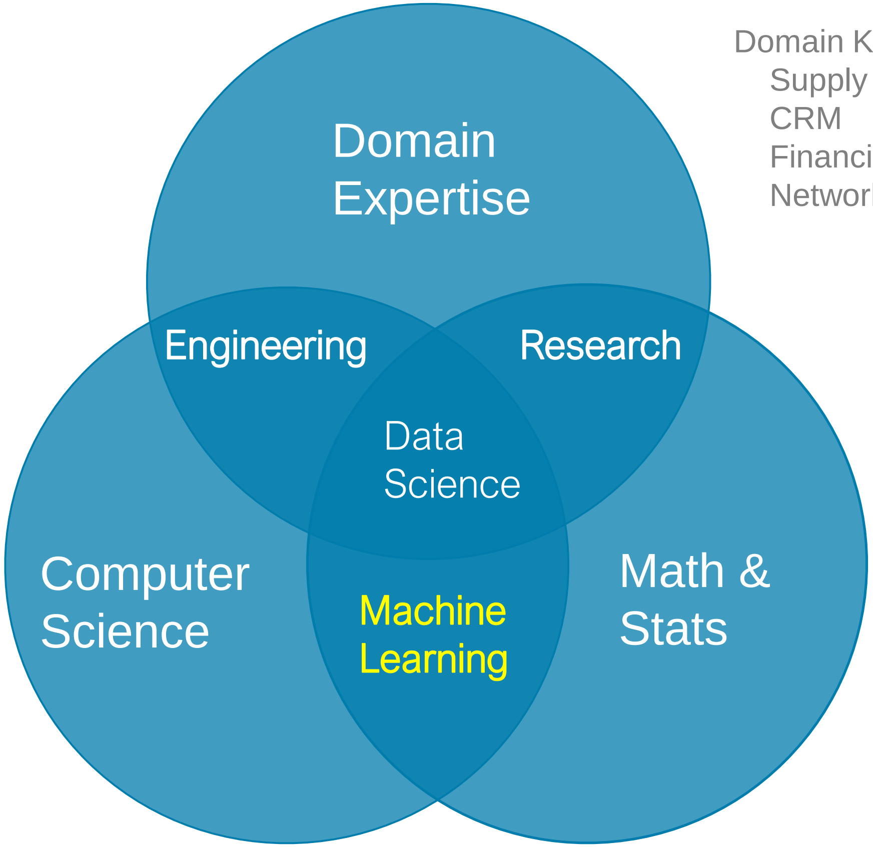
- Deep Learning

------------Labs--------------------------------------

Machine Learning and Data Science….

Domain Knowledge
Supply Chain
CRM
Financials
Networking

Scripting, SQL
Python, R Scala
Data Pipelines
Big Data/ Apache Spark

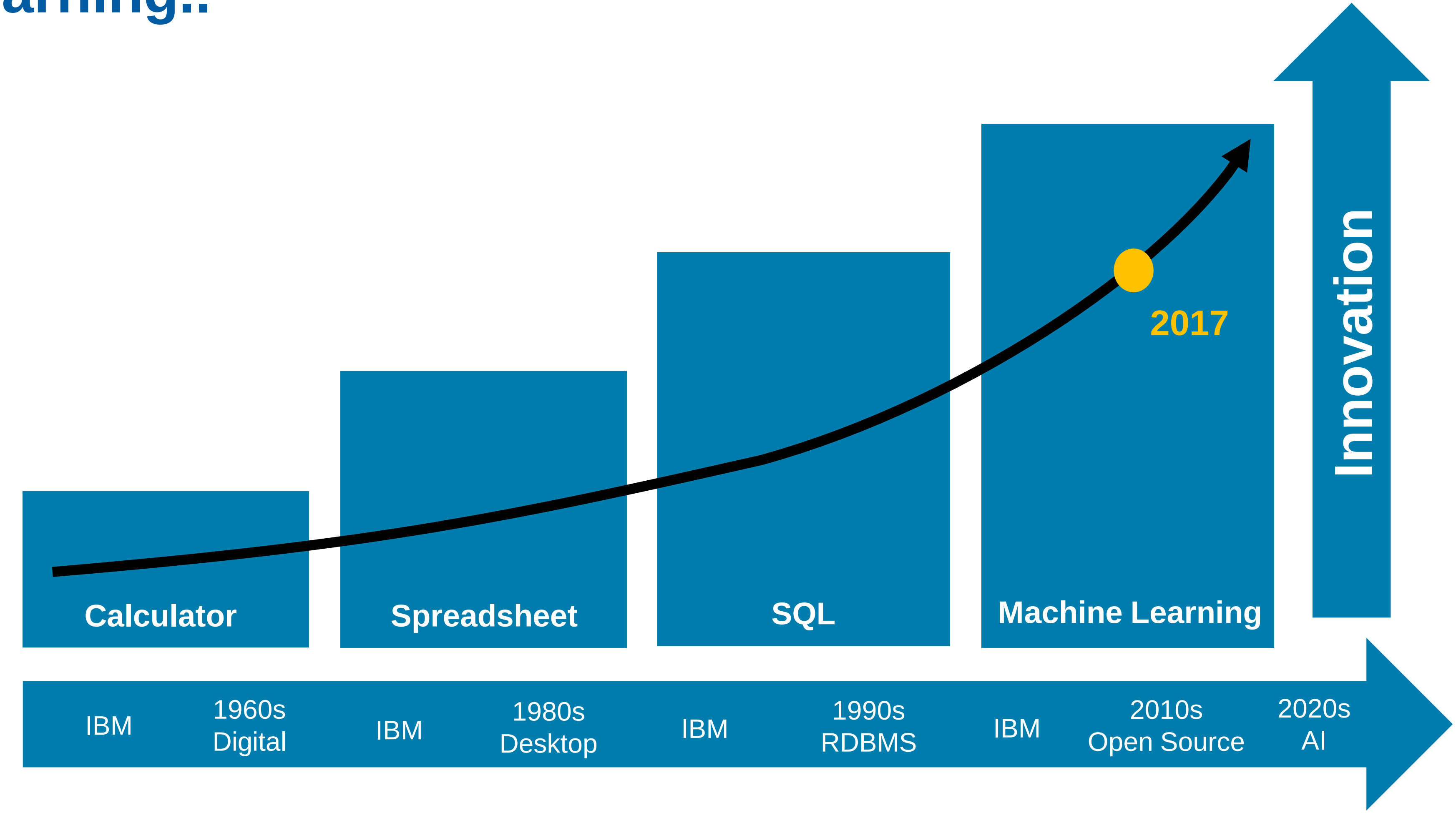


Mathematics
Computational

Data Science Projects Require Multiple Skills

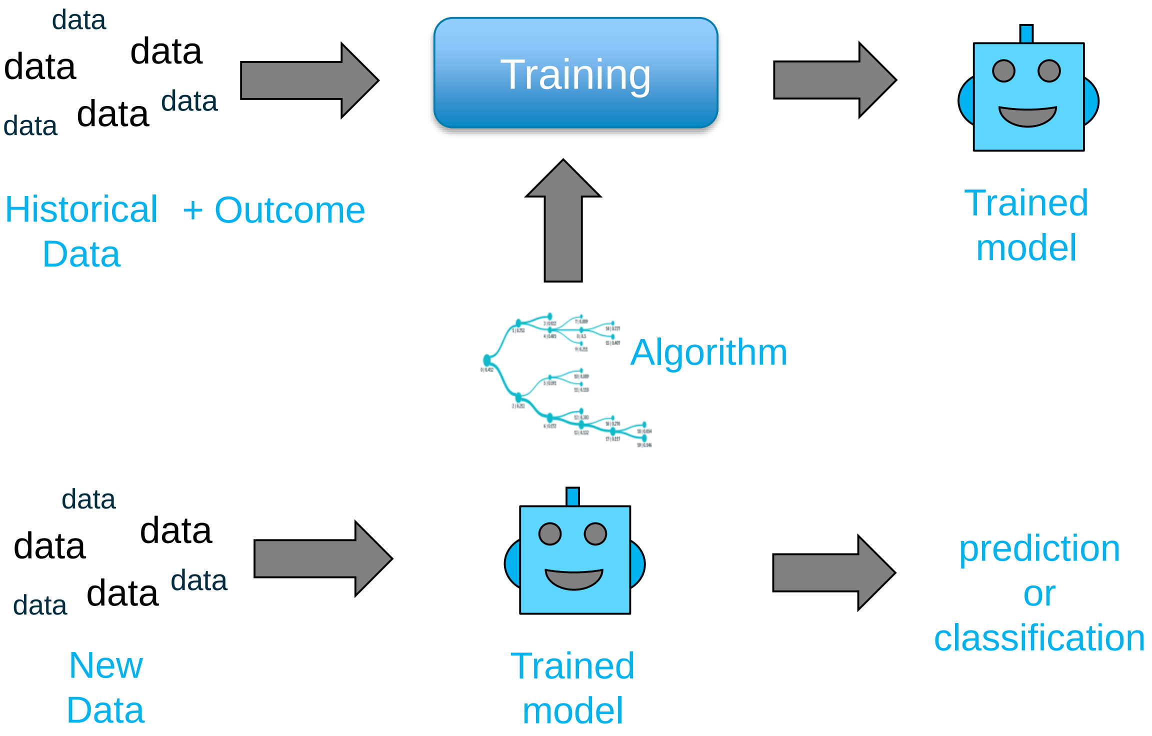
Modified from Drew Conway's Venn Diagram

Future of Data Science is Democratizing Machine Learning..



But what is Machine Learning?

“Computers that learn without being explicitly programmed”



data data data data data data

Historical Data + Outcome

Training

Trained model

Algorithm

data data data data data data

New Data

Trained model

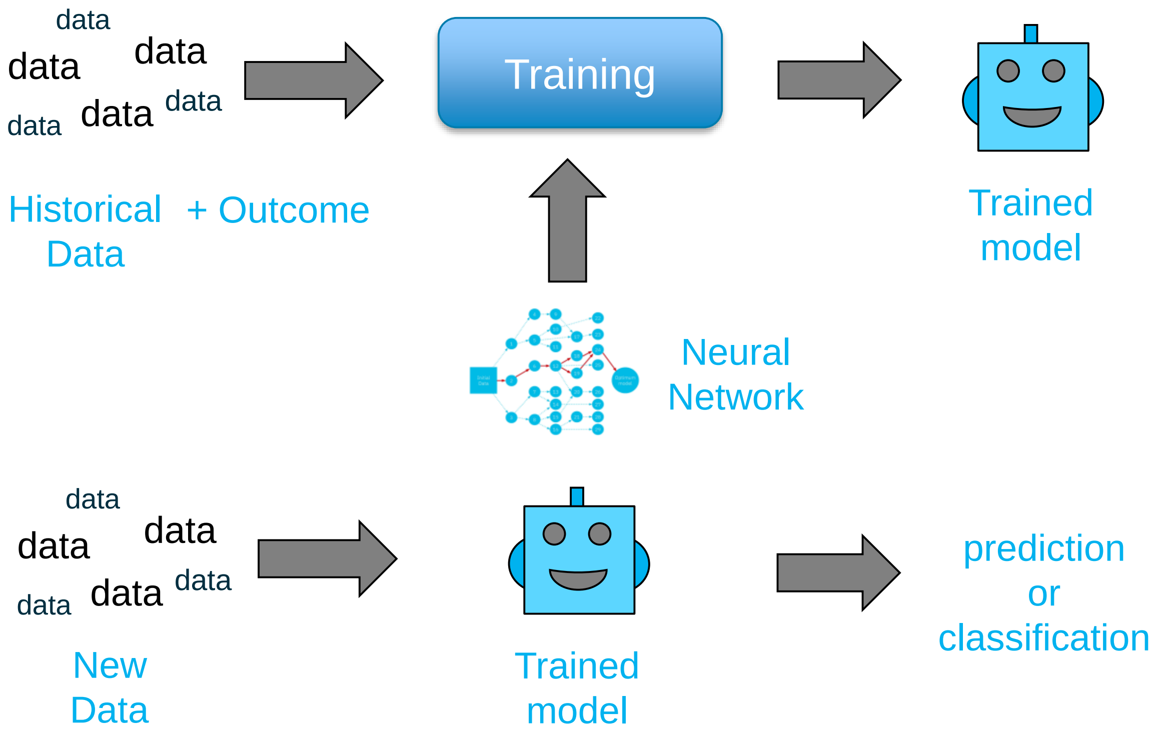
prediction or classification

1 A machine learning model is trained to recognize patterns in historical data

2 The model is then shown new data and asked to predict or classify it. If the patterns In the new data match the training data then the model makes accurate predictions.

But what is Deep Learning?

"Computers that learn without being explicitly programmed"



1 A machine learning model is trained to recognize patterns in historical data

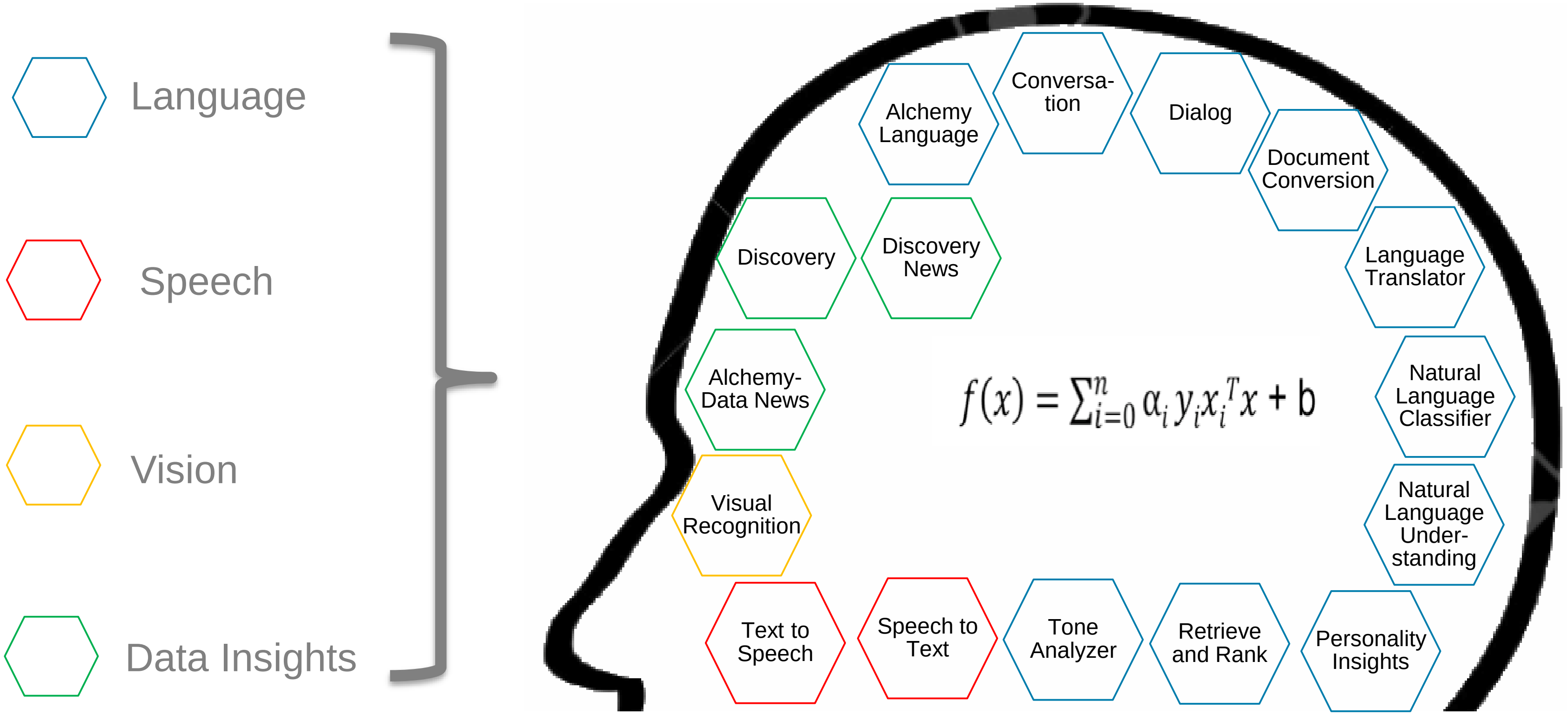
2 The model is then shown new data and asked to predict or classify it. If the patterns In the new data match the training data then the model makes accurate predictions.

But what is Artificial Intelligence?

A theory and development of computer systems able to perform tasks that normally require human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages..

Machine Learning = Artificial Intelligence???

Data + Algorithms = Scored AI Models



Language

Speech

Vision

Data Insights

Alchemy Language

Conversa-tion

Dialog

Document Conversion

Discovery

Discovery News

Language Translator

Alchemy-Data News

$$f(x) = \sum_{i=0}^{n} \alpha_i y_i x_i^T x + b$$

Natural Language Classifier

Visual Recognition

Natural Language Under-standing

Text to Speech

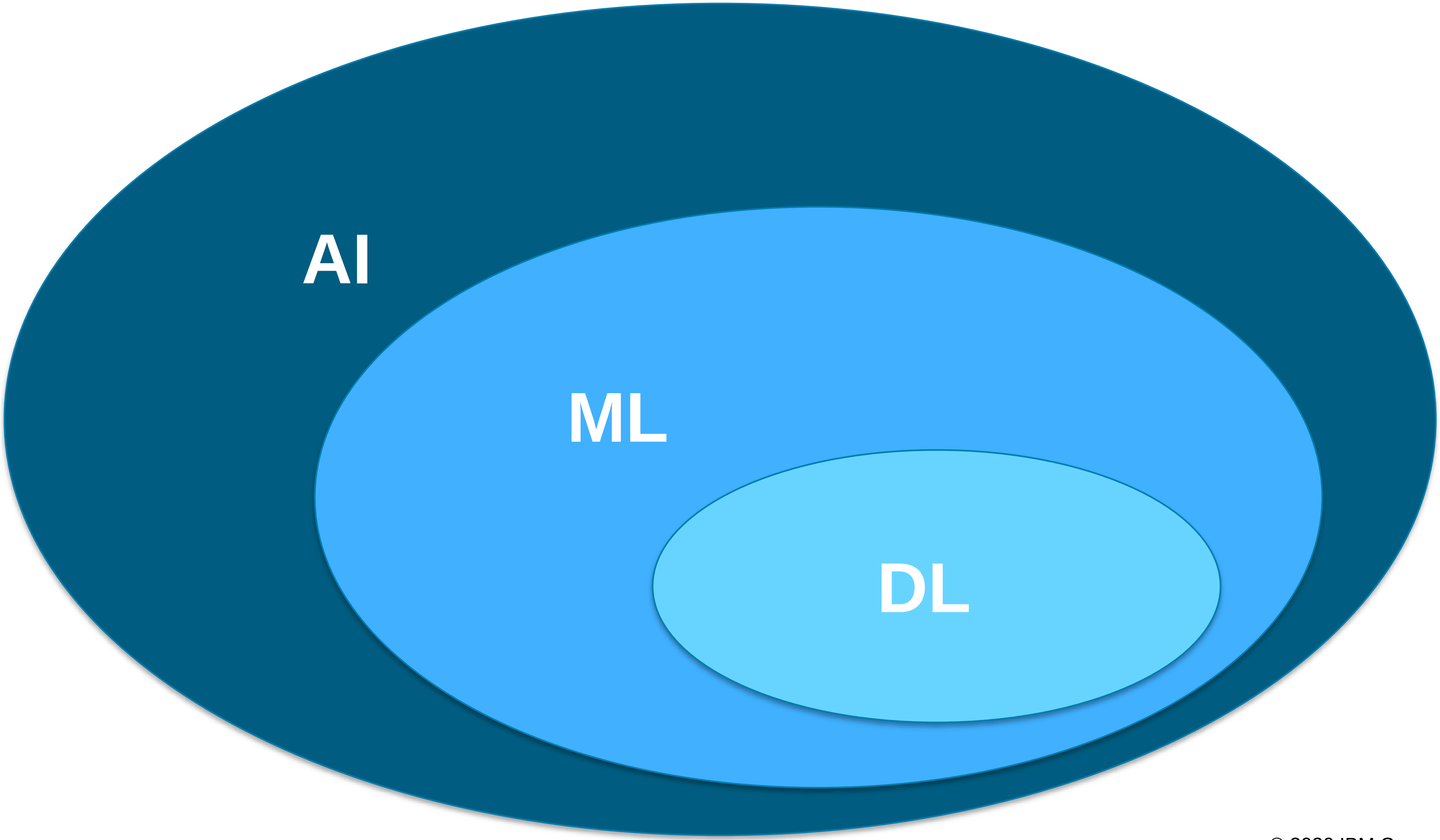
Speech to Text

Tone Analyzer

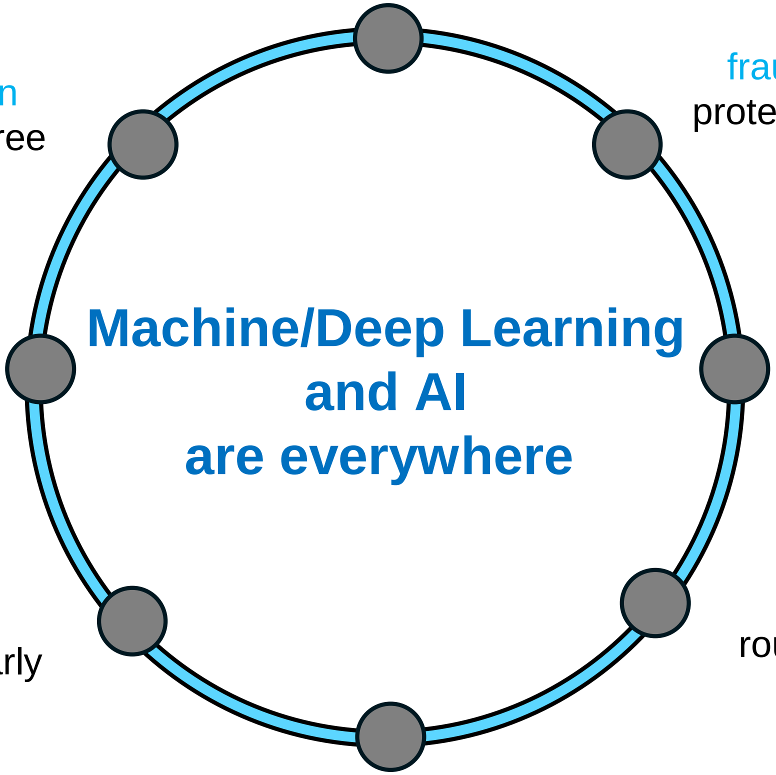
Retrieve and Rank

Personality Insights

Understanding AI, ML & DL Relationship…



The future is now



Machine/Deep Learning and AI are everywhere

facial recognition
unlocks your phone

fraud detection
protects your credit

recommendations
help you shop faster

chat bots
route calls quicker

spam detection
unclogs your Inbox

machine vision
detects cancer early

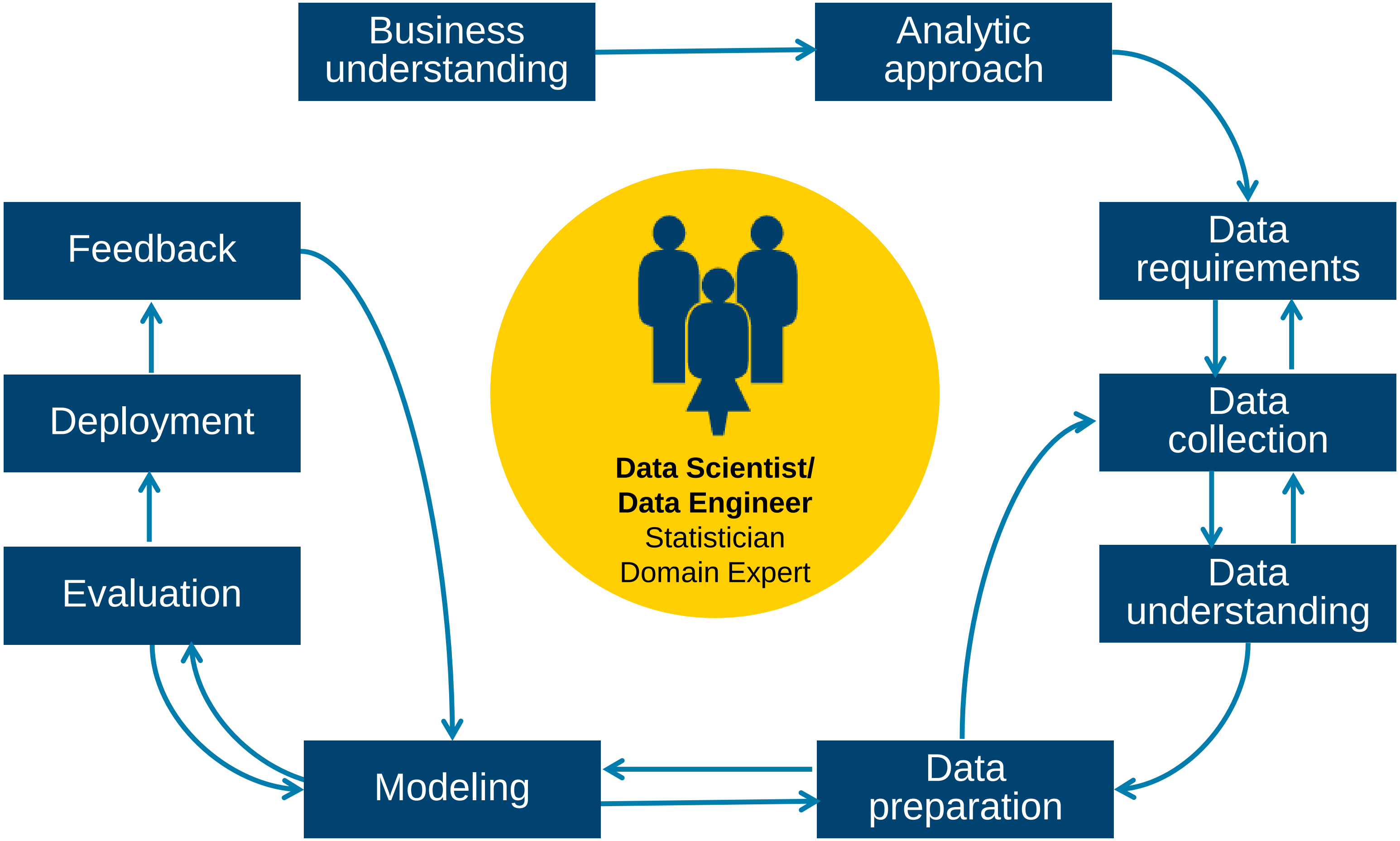
autonomous vehicles
detect pedestrians

speech recognition
lets you go hands-free

Introduction to Machine Learning

- Overview
- Data Science Methodology
- Data Understanding
- Data Preparation
- Categories of Machine Learning
- Learning Challenges
- Machine Learning Algorithms
- Model Evaluation
- Deep Learning

Data Science Methodology



Business understanding

Analytic approach

Data requirements

Data collection

Data understanding

Data preparation

Modeling

Evaluation

Deployment

Feedback

Data Scientist/
Data Engineer
Statistician
Domain Expert

Matrix for Machine Learning

<ins>Known as:</ins>

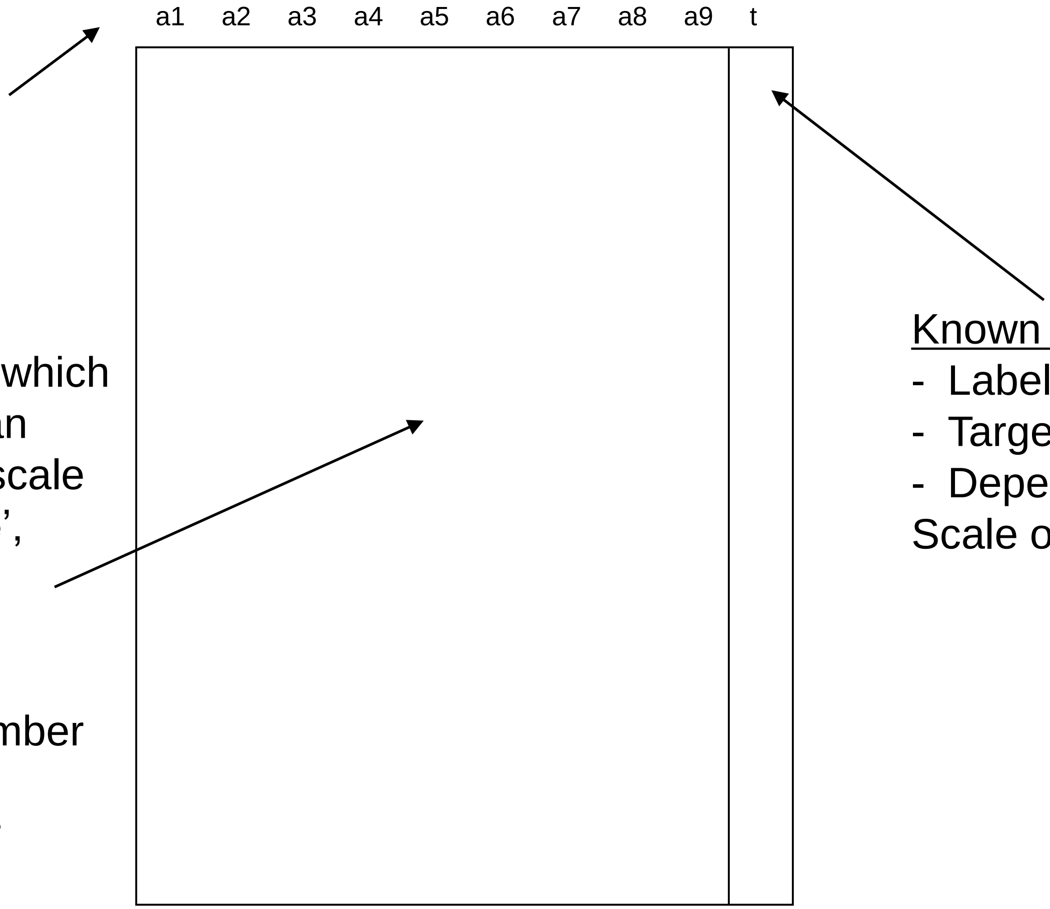
- Attributes
- Features
- Predictor variables
- Explanatory variables

<ins>Scale variables:</ins>

- Continuous variables, which can be measured on an interval scale or ratio scale
- ‘Weight’, ‘Temperature’, ‘Salary’, etc…

<ins>Categorical variables:</ins>

- Data with a limited number of distinct values or categories (nominal or ordinal)
- ‘Hair color’, ‘Gender’, ‘Grape varieties’, etc…



<ins>Known as:</ins>

- Label
- Target variable
- Dependent variable

Scale or Categorical

Introduction to Machine Learning

- Overview
- Data Science Methodology
- Data Understanding
- Data Preparation
- Categories of Machine Learning
- Learning Challenges
- Machine Learning Algorithms
- Model Evaluation
- Deep Learning

Data Understanding – Data Audit

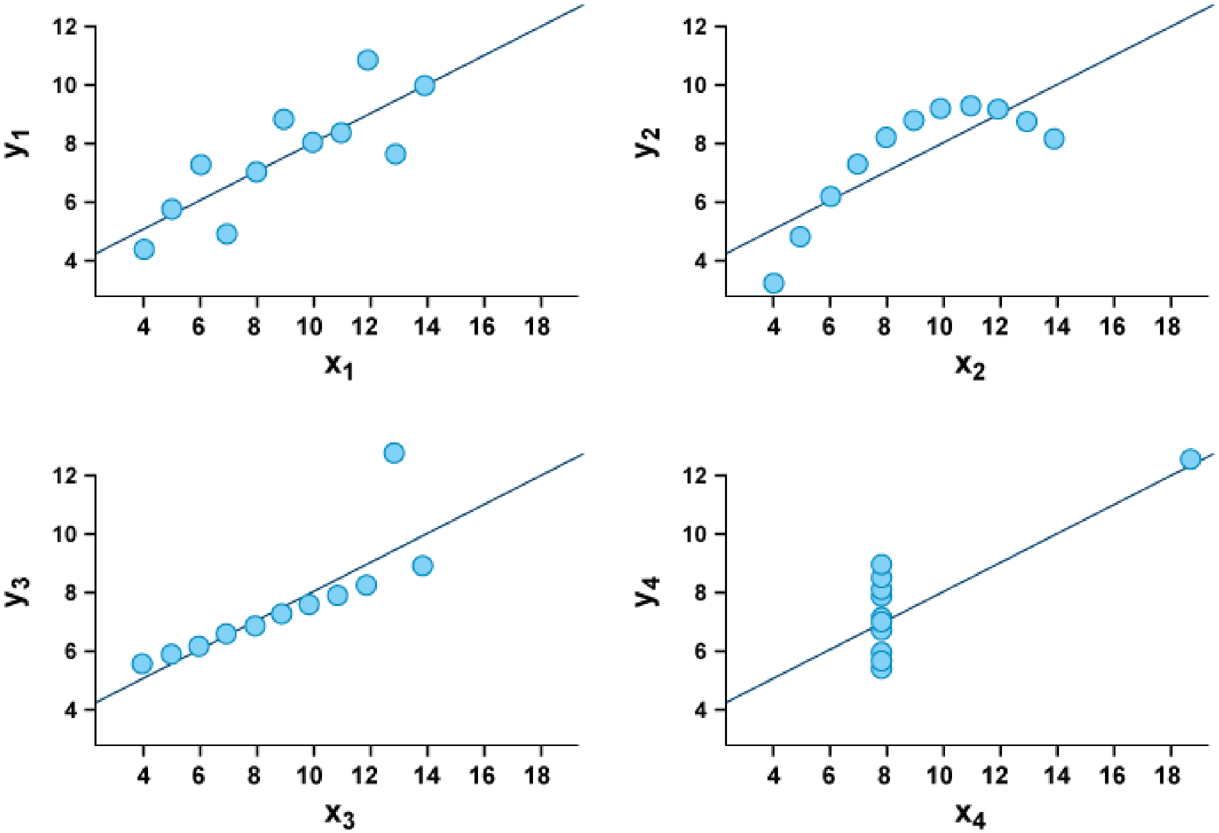
- **Data can be missing values**
  - Blank fields
  - Fields with dummy values (9999)
  - Fields with “U” or “Unknown”

- **Data can be corrupt or incoherent or anomalous:**
  - Data fields can be in the wrong format (strings where numbers are expected)
  - Spurious “End of Line” characters can chop original lines of data into several lines and cause data fields in the wrong place
  - Data entered in different formats: USA / US / United States
  - Data can be anomalous – outlier detection

- **Data can be duplicated**

- **Handling these data quality issues (as part of data preparation) is often referred to as:**
  - Data cleansing

Data Understanding: Visualizations



The four data sets have similar statistical properties:

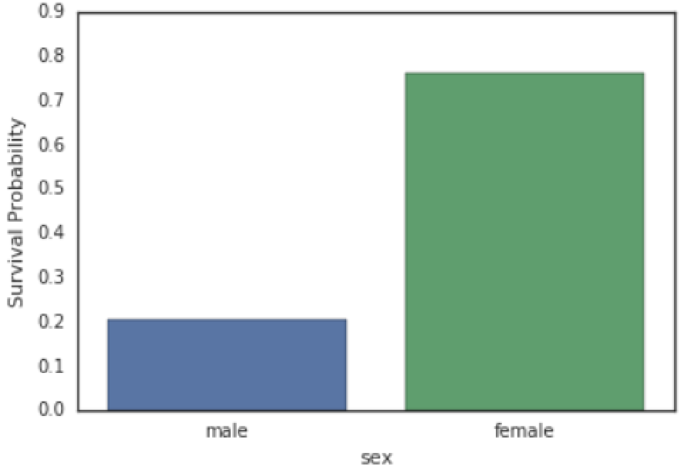
- The mean of x is 9
- The variance of x is 11
- The mean of y is approx. 7.50
- The variance of y is approx. 4.12
- The correlation is 0.816

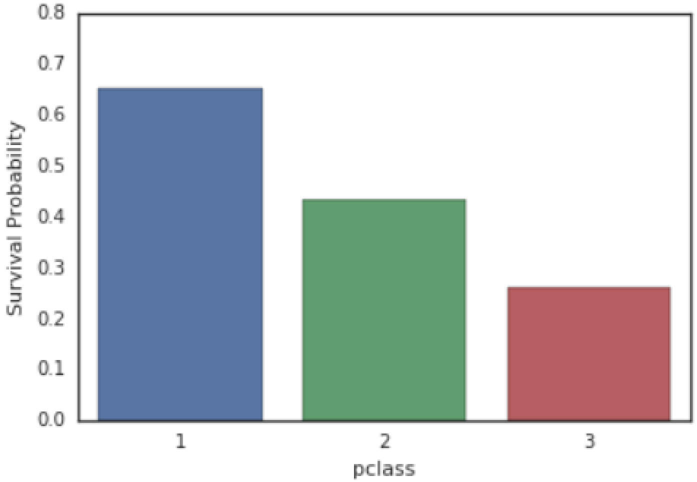
As shown the linear regression lines are approx. y=3.00+0.500x.

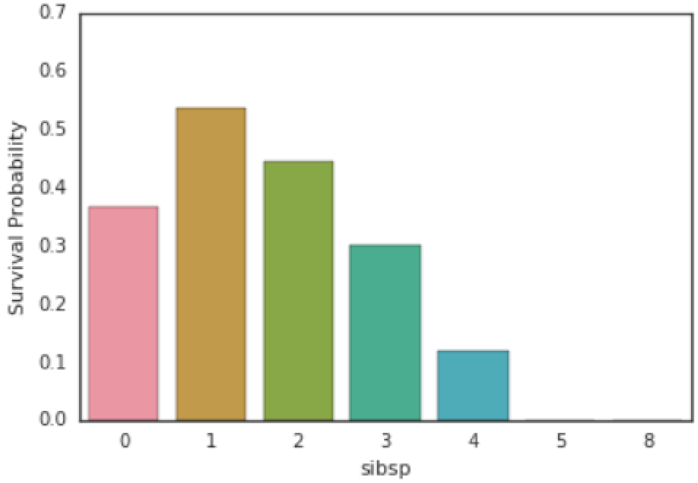
- **Anscombe's quartet**
  - The four datasets have nearly identical statistical properties (mean, variance, correlation), yet the differences are striking when looking at the simple visualization

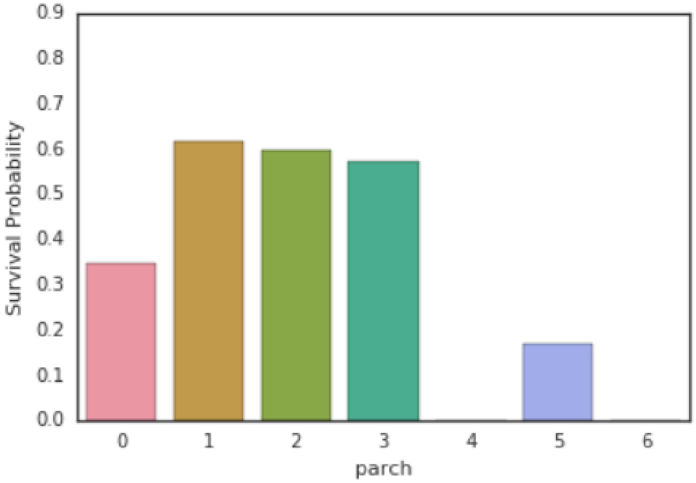
Data Understanding: Visualizations

- **Titanic Data**
- **Univariate Relationships**



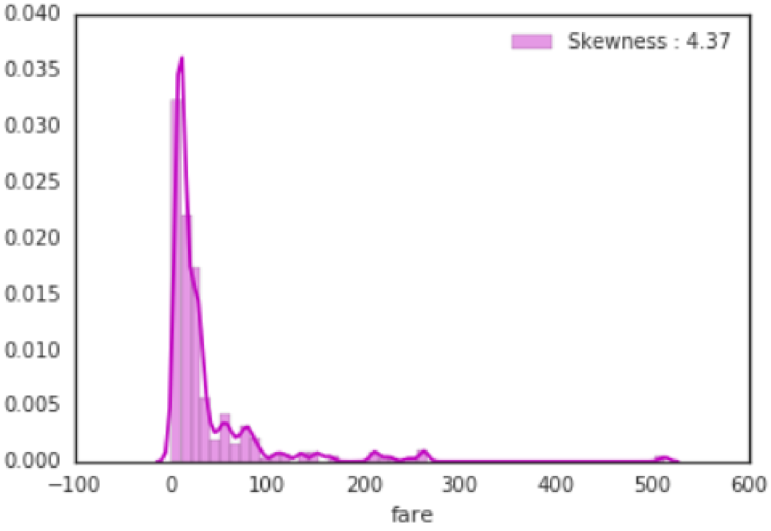




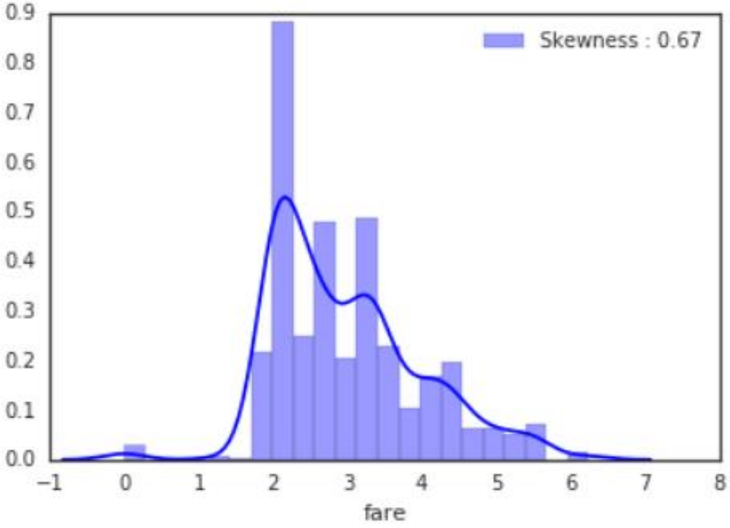


Data Understanding: Visualizations

- **Titanic Data**
- **Skewed Data**



Original Data



After Log Transform

Introduction to Machine Learning

- Overview
- Data Science Methodology
- Data Understanding
- Data Preparation ⬅
- Categories of Machine Learning
- Learning Challenges
- Machine Learning Algorithms
- Model Evaluation
- Deep Learning

Data Preparation

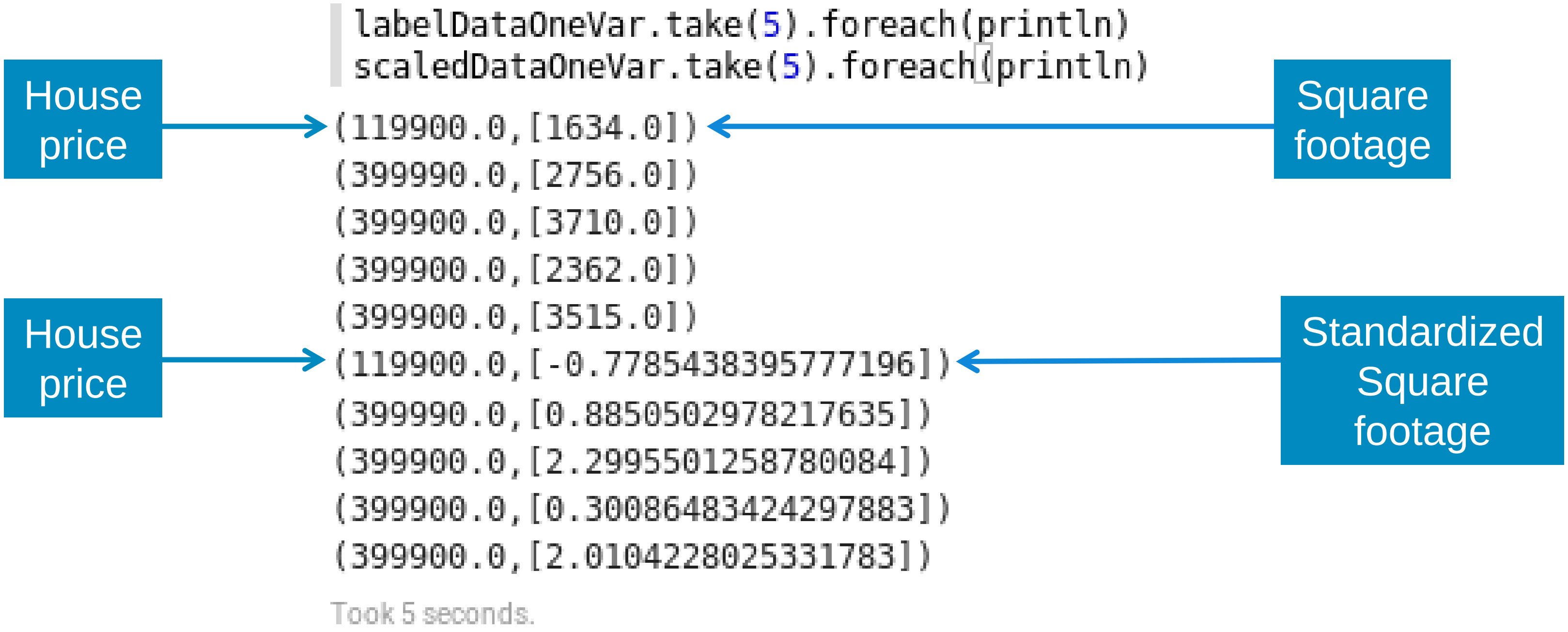
- **Data preparation can be very time consuming depending on:**
  - The state of the original data
    - Data is typically collected in a “human” friendly format
  - The desired final state of the data (as required by the machine learning models and algorithms)
    - The desired final state is typically some “algorithm” friendly format
  - There may be a need for a (long) pipeline of transformations before the data is ready to be consumed by a model:
    - These transformations can be done manually (write code)
    - These transformations can be done through tools

Data Preparation – Transformation

- **Data may need to be transformed to match algorithms requirements:**
  - Tokenizing (typical in text processing)
  - Vectorizing (several algorithms in Spark MLlib require this)
    - Transform data into Vector arrays
    - Can be done manually (write Python or Scala code)
    - Can be done using tools (VectorAssembler in the new ML package
  - Bucketizing
    - Transform a range of continuous values into a set of buckets

Data Preparation – Transformation

- **Data may need to be transformed to match algorithms requirements:**
  - Standardization
    - Transform numerical data to values with zero mean and unit standard deviation
    - Linear Regression with SGD in Spark MLlib requires this



```
labelDataOneVar.take(5).foreach(println)
scaledDataOneVar.take(5).foreach(println)
```

```
(119900.0,[1634.0])
(399990.0,[2756.0])
(399900.0,[3710.0])
(399900.0,[2362.0])
(399900.0,[3515.0])
(119900.0,[-0.7785438395777196])
(399990.0,[0.8850502978217635])
(399900.0,[2.2995501258780084])
(399900.0,[0.30086483424297883])
(399900.0,[2.0104228025331783])
```

Took 5 seconds.

Data Preparation – Transformation

- **Data may need to be transformed to match algorithms requirements:**
  - Normalization
    - Transform data so that each Vector has a Unit norm.

$$x' = \frac{x}{||x||}$$

    - Transform data so that each feature has a value between 0 and 1

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

  - Categorical values need to be converted to numbers
    - This is required by Spark MLlib classification trees
    - Marital Status: {“Widowed”, “Married”, “Divorced”, “Single”}
    - Marital Status: {0, 1, 2, 3}
    - You cannot do this if the algorithm could infer: Single = 3 X Married ☺

Data Preparation – Transformation

- **Data may need to be transformed to match algorithms requirements:**
  - Dummy encoding
    - When categorical values cannot be converted to consecutive numbers
    - Marital Status: {“Single”, “Married”, “Divorced”, “Widowed”}
    - Marital Status: {“0001”, “0010”, “0100”, “1000”}
    - This is necessary if the algorithm could make some wrong inference from the numerical based categorical encoding:
      - ❑ Single = 3
      - ❑ Married = 2
      - ❑ Divorced = 1
      - ❑ Widowed = 0
        - > Single = Married + Divorced
        - > Single = Divorced x 3
        - > (this is a contrived example, but you get the idea ☺, replace marital status with colors…)

Data Preparation – Dimensionality Reduction

- **Data dimensionality may need to be reduced:**

- **The idea behind reducing data dimensionality is that raw data tends to have two subcomponents:**
  - "Useful features" (aka structure)
  - Noise (random and irrelevant)
  - Extracting the structure makes for better models

  - Examples of applications of dimensionality reduction
    - Extracting the important features in face/pattern recognition
    - Removing stop words when working on text classification
    - Stemming: fishing, fished, fisher ➔ fish

  - Examples methods of dimensionality reduction
    - Principal Component Analysis
    - Singular Value Decomposition
    - Autoencoders

Introduction to Machine Learning

- Overview
- Data Science Methodology
- Data Understanding
- Data Preparation
- Categories of Machine Learning
- Learning Challenges
- Machine Learning Algorithms
- Model Evaluation
- Deep Learning

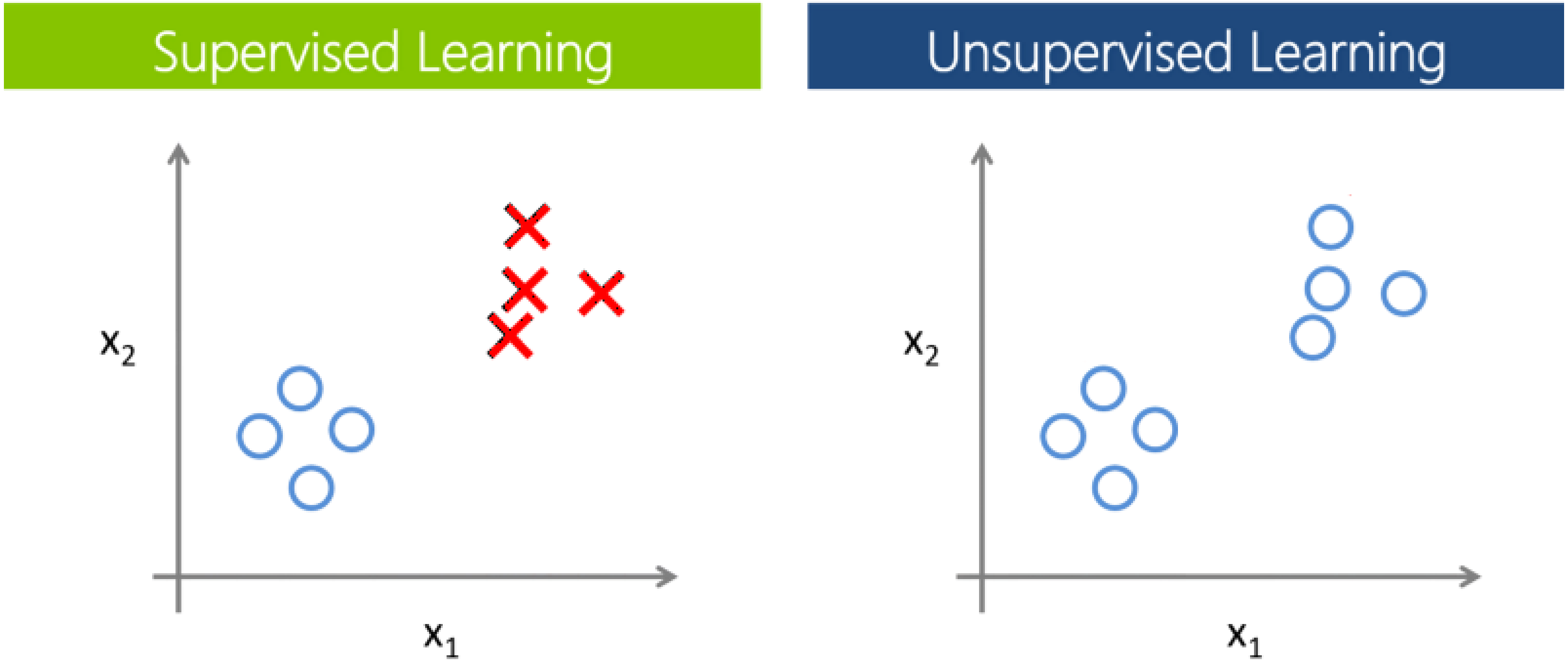
Categories of Machine Learning

- **Supervised learning**
  - The program is “trained” on a pre-defined set of “training examples”, which then facilitate its ability to reach an accurate conclusion when given new data
  - The algorithm is presented with example inputs and their desired outputs (correct results)
  - The goal is to learn a general rule that maps inputs to outputs

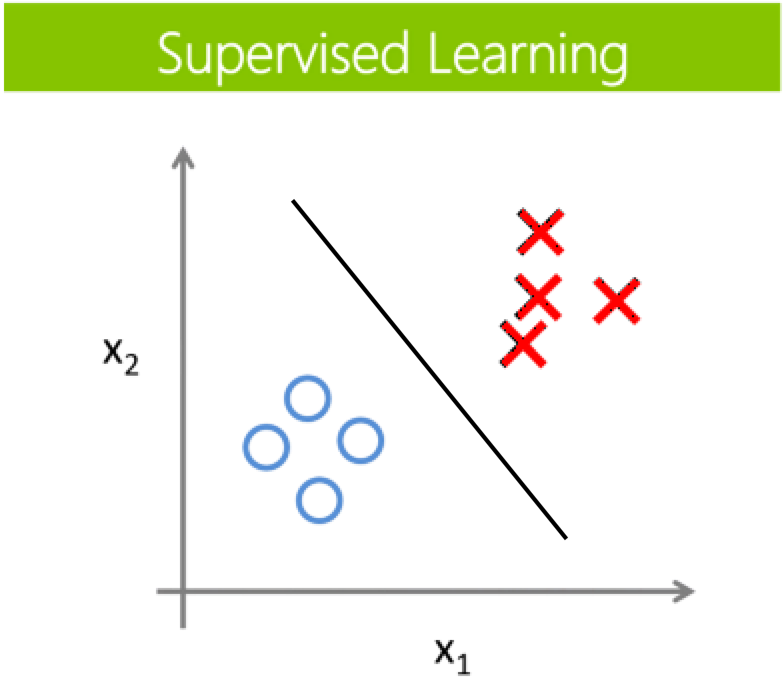
- **Unsupervised learning**
  - No labels are given to the learning algorithm, leaving it on its own to find structure (patterns and relationships) in its input
  - Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning)

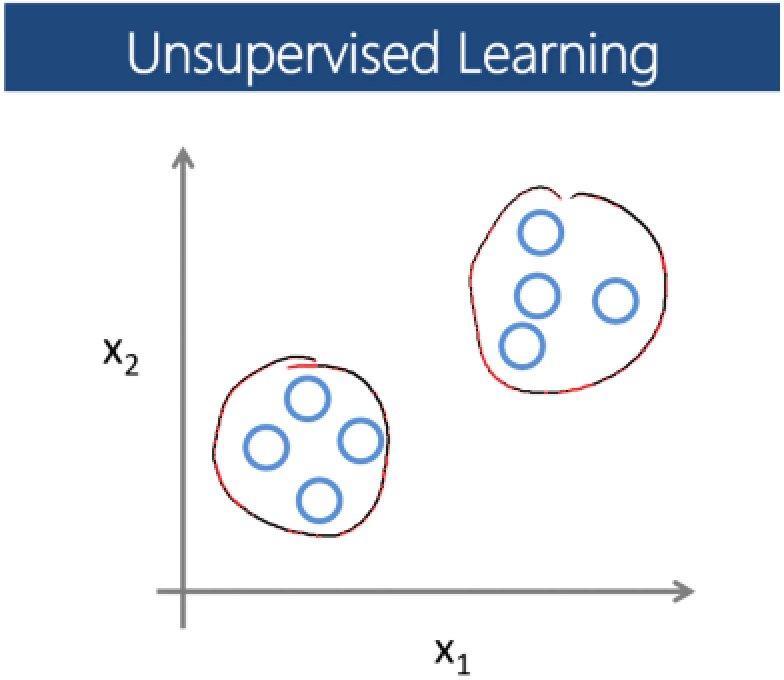
- **Reinforcement learning**
  - Models an agent interacting with an environment. The agent observes the environment, takes an action, and may receive an award (+ or -). The goal is to learn the set of actions to maximize the reward.

Supervised vs. Unsupervised Learning



Supervised vs. Unsupervised Learning





Categories of Machine Learning

Technique	Usage	Algorithms
Classification (or prediction)	• Used to predict group membership (e.g., will this employee leave?) or a number (e.g., how many widgets will I sell?)	• Decision Trees • Logistic Regression • Random Forests • **Naïve Bayes** • Linear Regression • Lasso Regression etc
Segmentation	• Used to classify data points into groups that are internally homogenous and externally heterogeneous. • Identify cases that are unusual	• K-means • Gaussian Mixture • Latent Dirichlet allocation etc
Association	• Used to find events that occur together or in a sequence (e.g., market basket)	• FP Growth etc

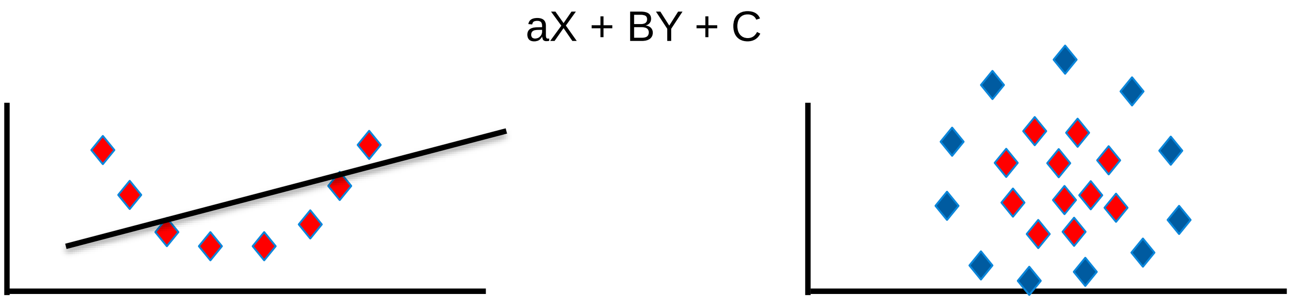
Introduction to Machine Learning

- Overview
- Data Science Methodology
- Data Understanding
- Data Preparation
- Categories of Machine Learning
- Learning Challenges
- Machine Learning Algorithms
- Model Evaluation
- Deep Learning

Learning challenges

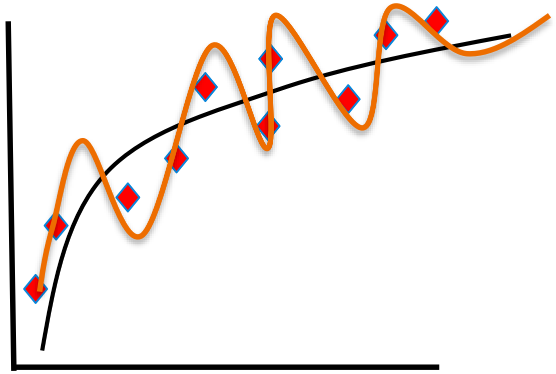
- **Under fitting:**
  - Not knowing enough “basic” concepts, i.e. not being well-equipped enough to tackle learning at hand:
    - You can’t study calculus without knowing some algebra.
    - You can’t learn playing hockey without knowing how to skate.
    - You can’t learn polo without knowing how to ride.

  - This can lead to under fitting in Machine Learning: The chosen model is just not “sophisticated”, “rich”, enough to capture the concept.



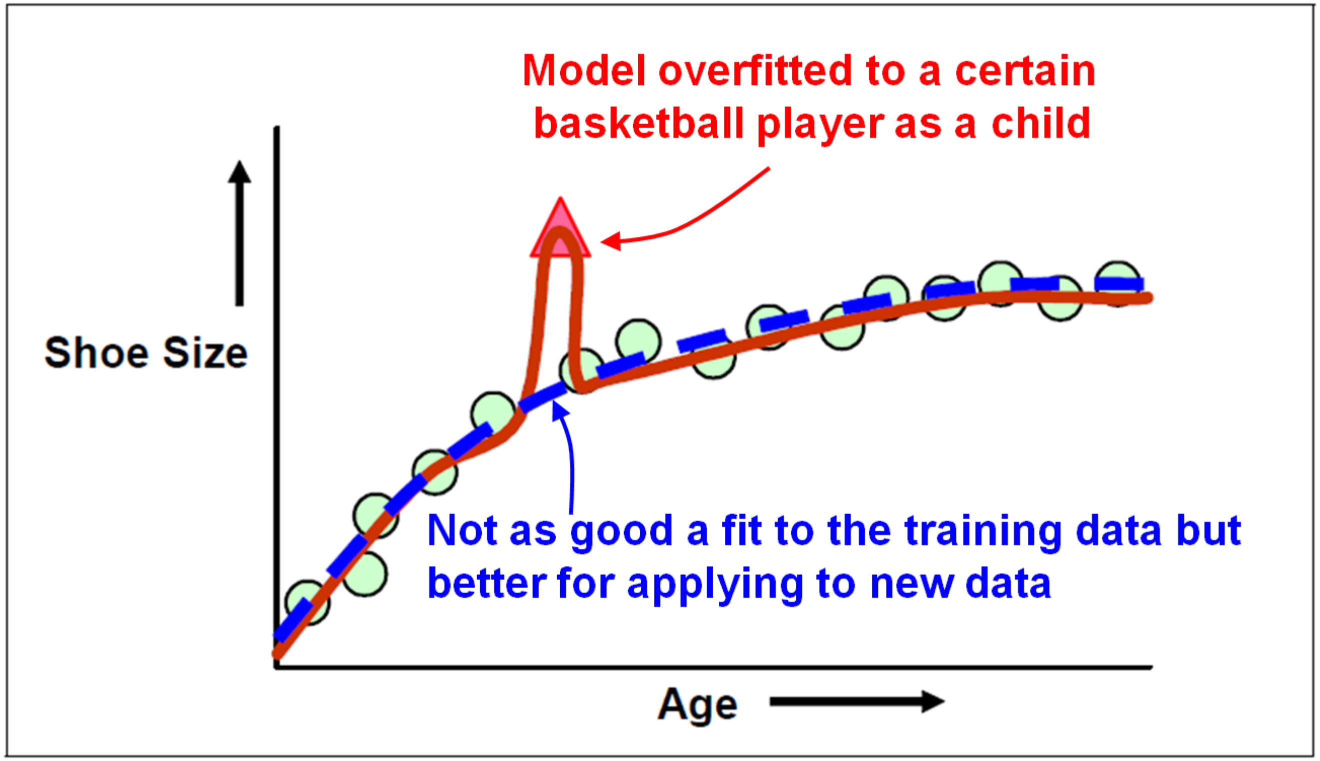
Learning challenges

- **Over fitting:**
  - Hyper-sensitivity to minor fluctuations, ending up in modeling a lot of the unwanted noise in the data:
  - This can lead to over fitting in Machine Learning.



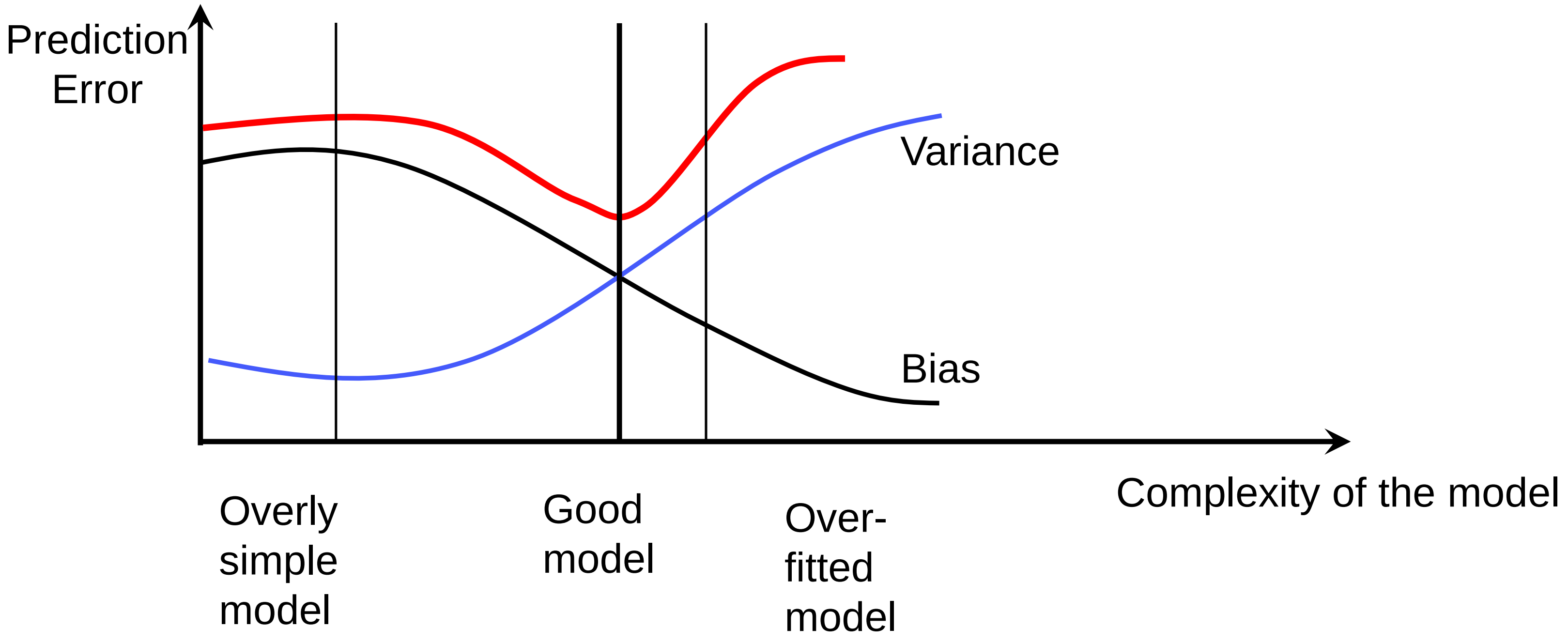
Model overfitting

- When building a predictive model, there is a risk of overfitting the model to the training data.
- The model fits the training data very well, but it does not perform well when applied to new data.



Learning challenges

- Compromise between bias and variance:



Graphical illustration of bias vs variance

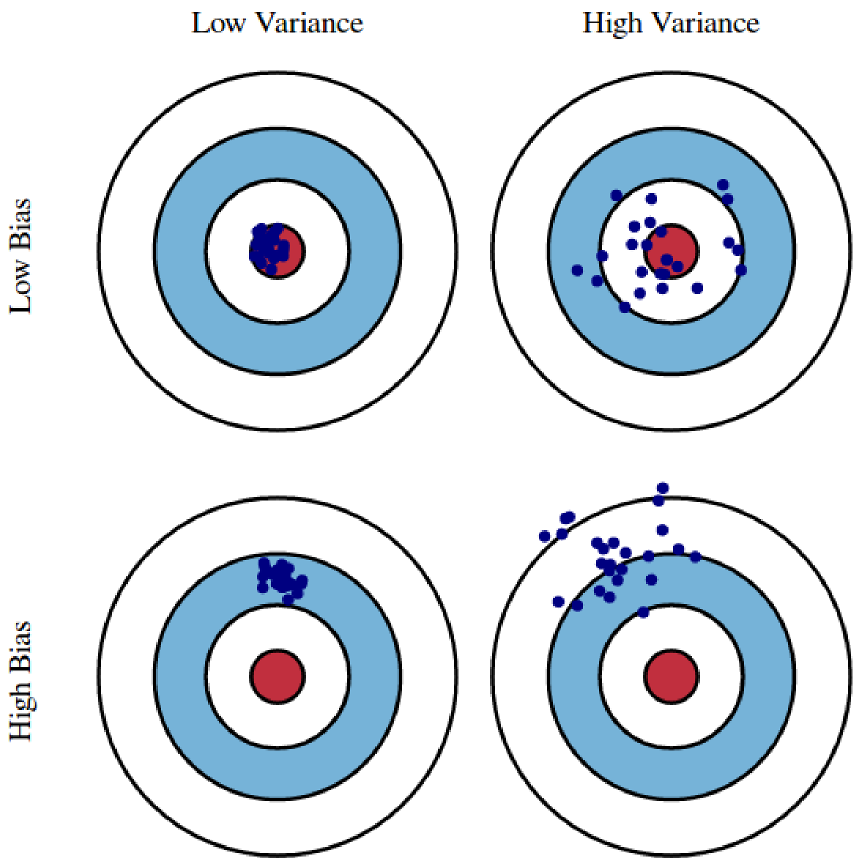
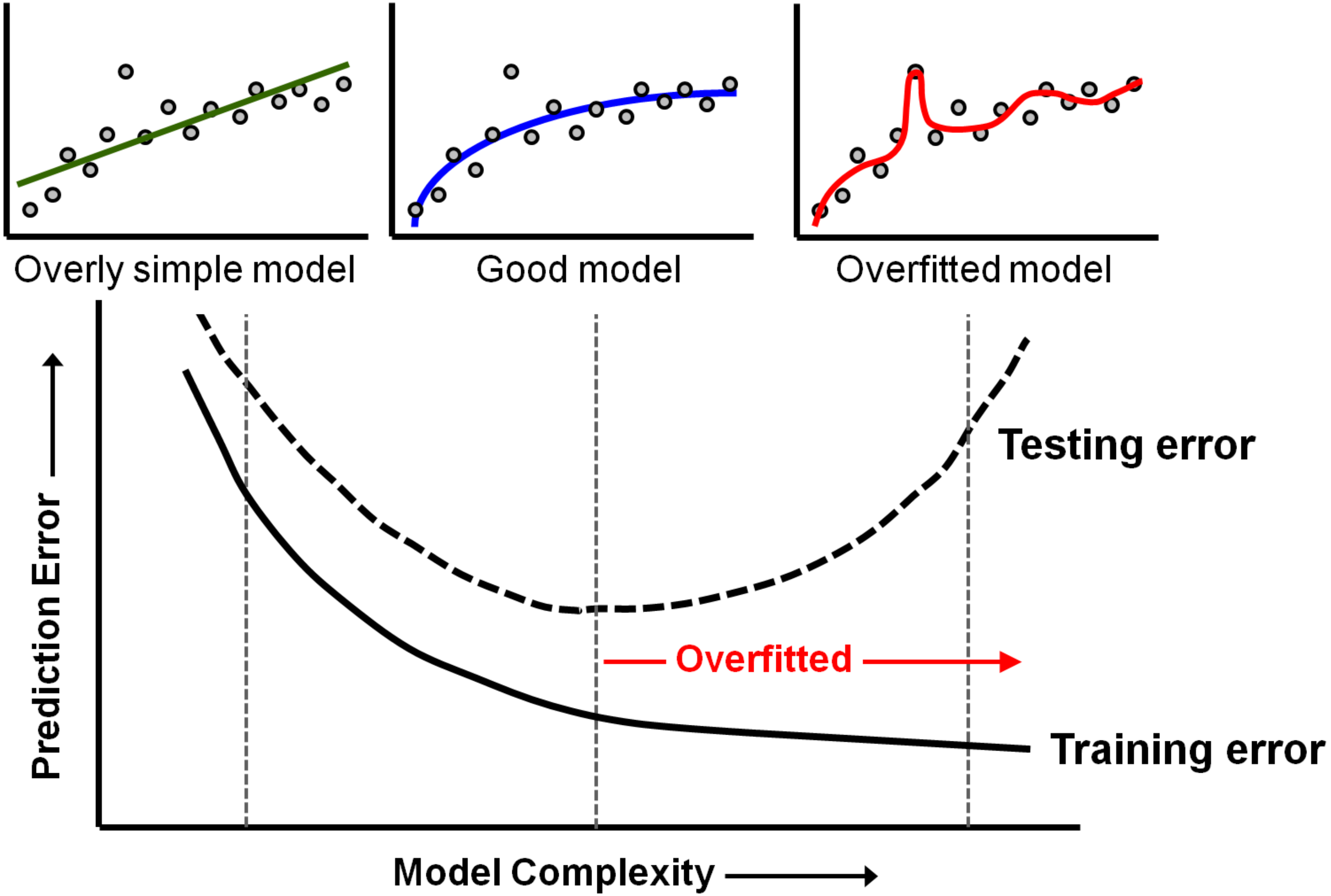


Fig. 1 Graphical illustration of bias and variance.

Source: http://scott.fortmann-roe.com/docs/BiasVariance.html

Indication of Overfitting



Introduction to Machine Learning

- Overview
- Data Science Methodology
- Data Understanding
- Data Preparation
- Categories of Machine Learning
- Learning Challenges
- Machine Learning Algorithms
- Model Evaluation
- Deep Learning

Classification – Naïve Bayes (supervised)

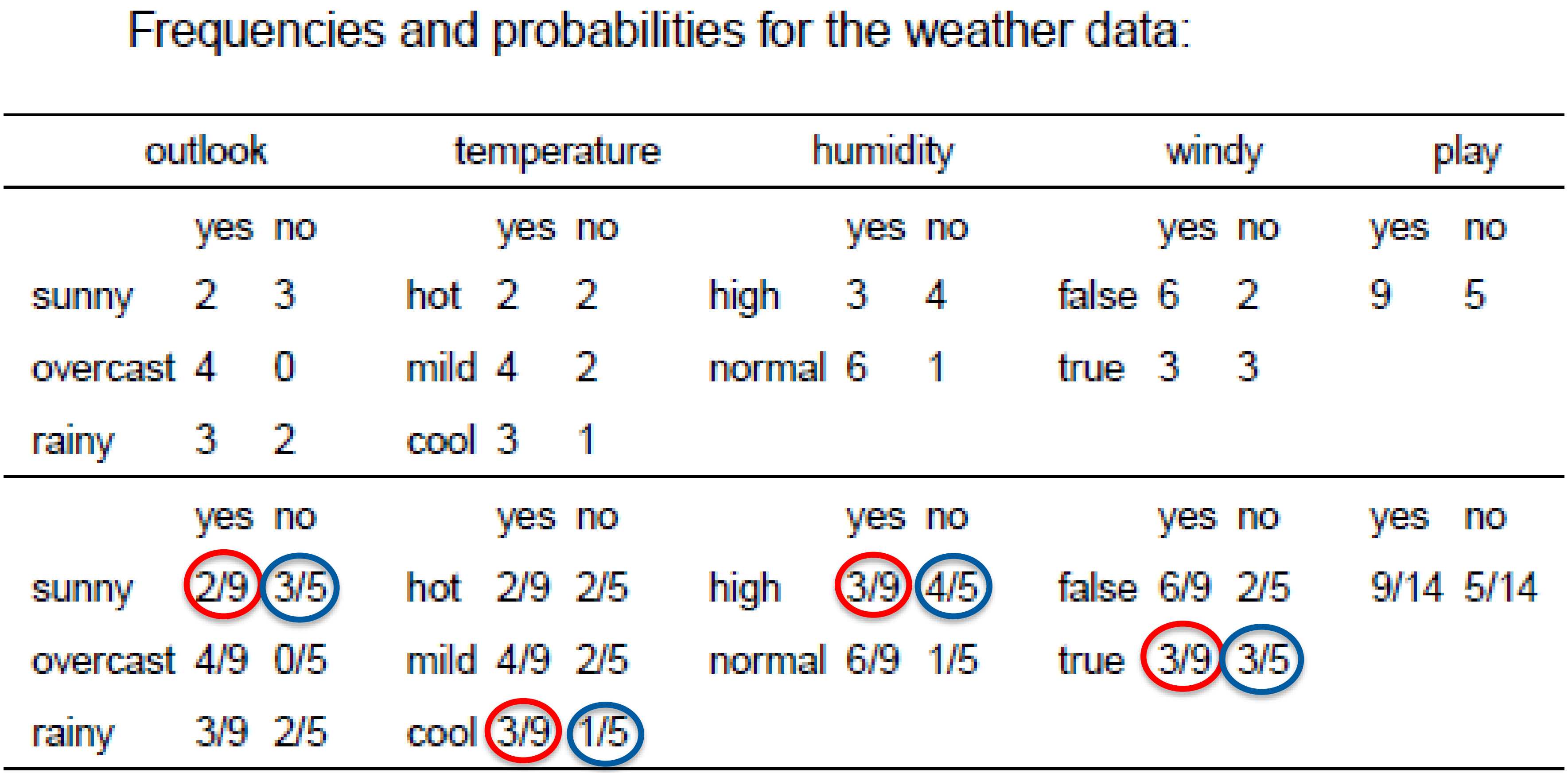
- **Two or more outcomes.**
- **Assumes independence among explanatory variables, which is rarely true (thus “naïve”).**
- **Despite its simplicity, often performs very well… widely used.**
- **Significant use cases:**
  - Text categorization (spam vs. legitimate, sports or politics, etc.) using word frequencies as the features
  - Medical diagnosis (*e.g.*, automatic screening)
  - Check a piece of text expressing positive emotions, or negative emotions?
  - Used for face recognition software.
- **Maximum conditional probability**
  - $Prob(Target|Input) = Prob(Input|Target) * \frac{Prob(Target)}{Prob(Input)}$

Classification – Naïve Bayes

Input → ← Target

Outlook	Temp	Humidity	Windy	Play golf
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

Classification – Naïve Bayes



Frequencies and probabilities for the weather data:

outlook			temperature			humidity			windy			play	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2	3	hot	2	2	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

Today's weather prediction sunny, cool, high humidity, windy → play golf ??

$$\boldsymbol{Prob(sunny|yes) \ast Prob(cool|yes) \ast Prob(high\ humidity|yes) Prob(windy|yes)}$$

$$\boldsymbol{Prob(sunny|no) \ast Prob(cool|no) \ast Prob(high\ humidity|no) Prob(windy|no)}$$

Classification – Naïve Bayes

Prob(Input | yes) = 2/9 * 3/9 * 3/9 * 3/9 = 0.0082

Prob(Input | no) = 3/5 * 1/5 * 4/5 * 3/5 = 0.0577

P(yes) = 9/14

P(no) = 5/14

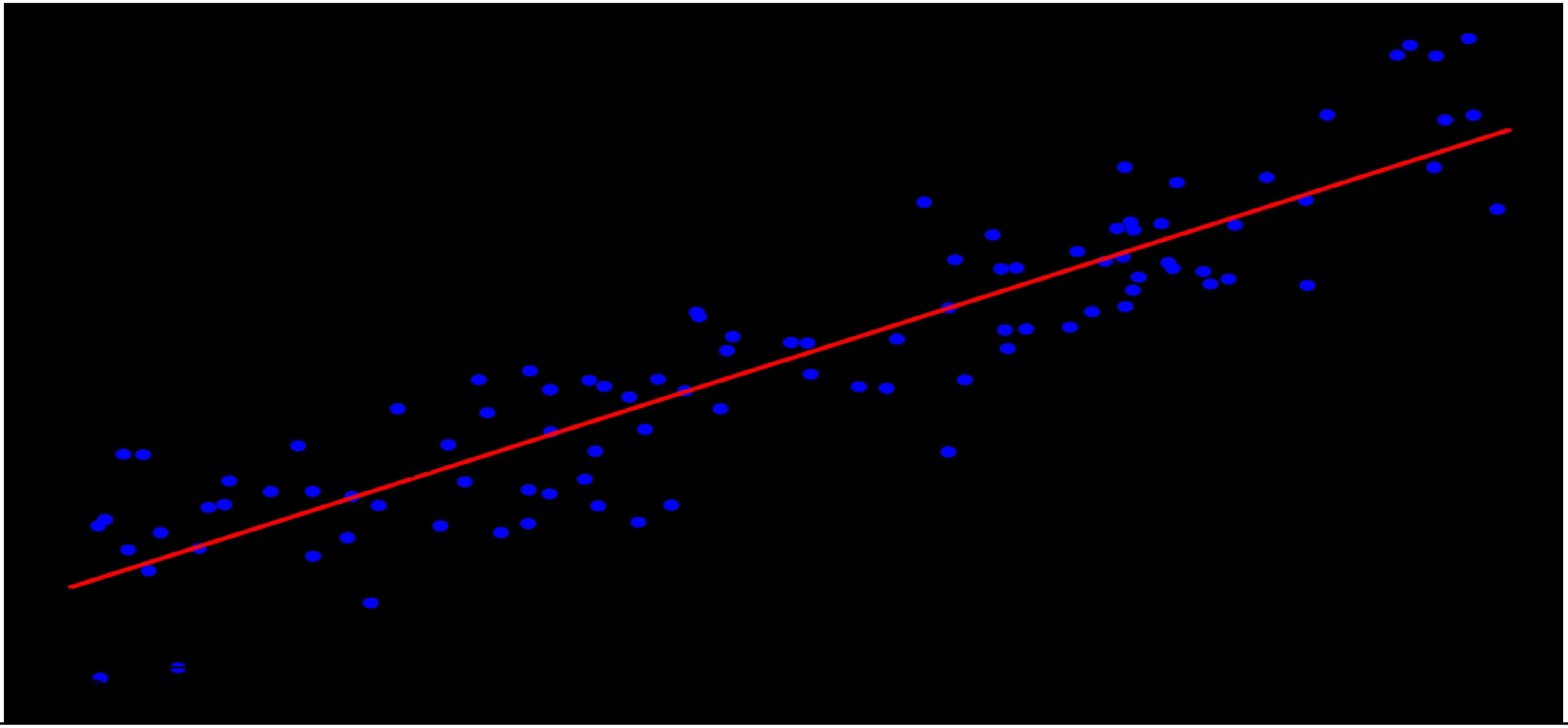
Prob(Input|yes)* Prob(yes) = 0.0082*(9/14) = 0.0053

Prob(Input|no) * Prob(no) = 0.0577*(5/14) = 0.0206

The prediction would be: NO.

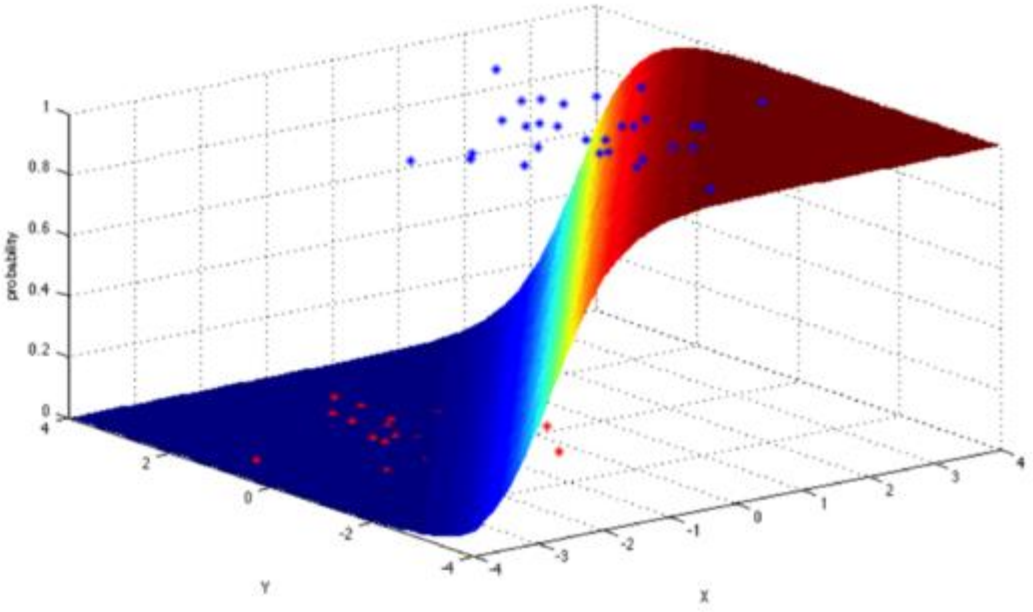
Linear Regression (supervised)

- **Draw a line, and then for each of the data points, measure the vertical distance between the point and the line, and add these up; the fitted line would be the one where this sum of distances is as small as possible.**
- **Use case:**
  - Housing prices



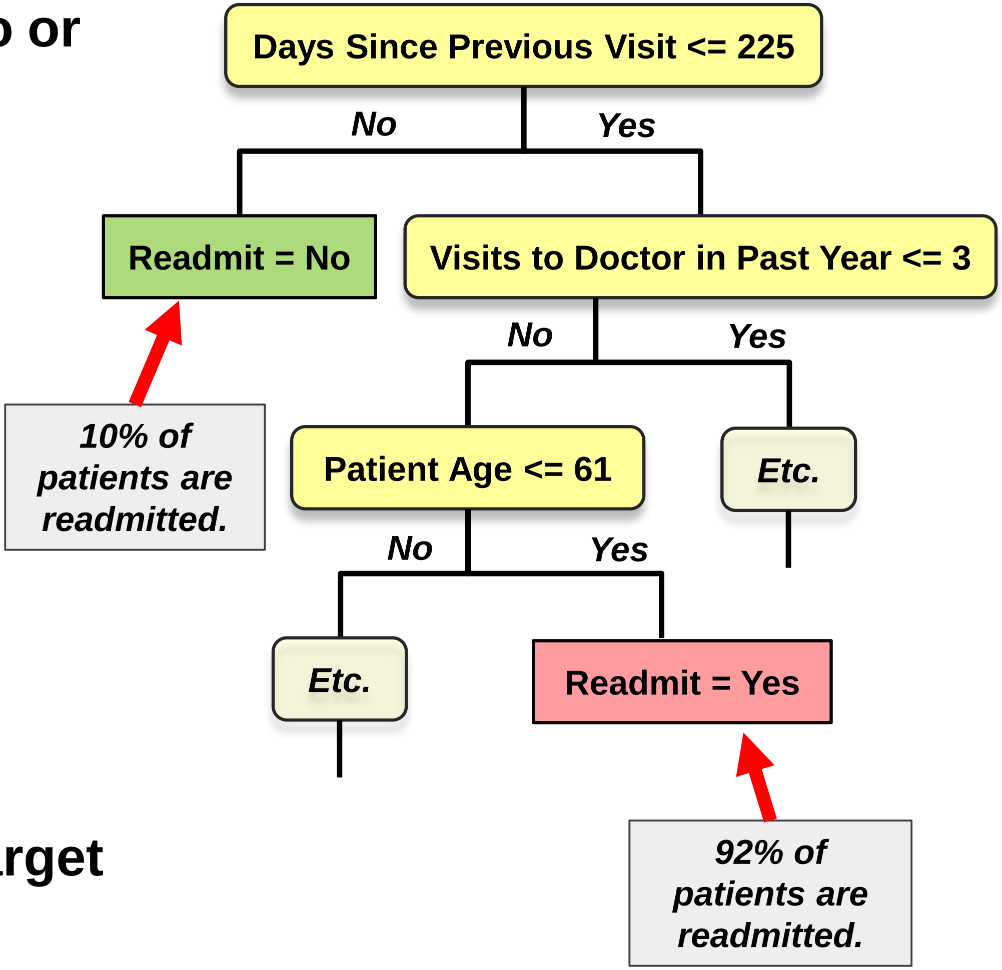
Logistic Regression (supervised)

- **Logistic regression is a powerful statistical way of modeling a binomial outcome with one or more explanatory variables. It measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution.**



Classification – Decision tree (supervised)

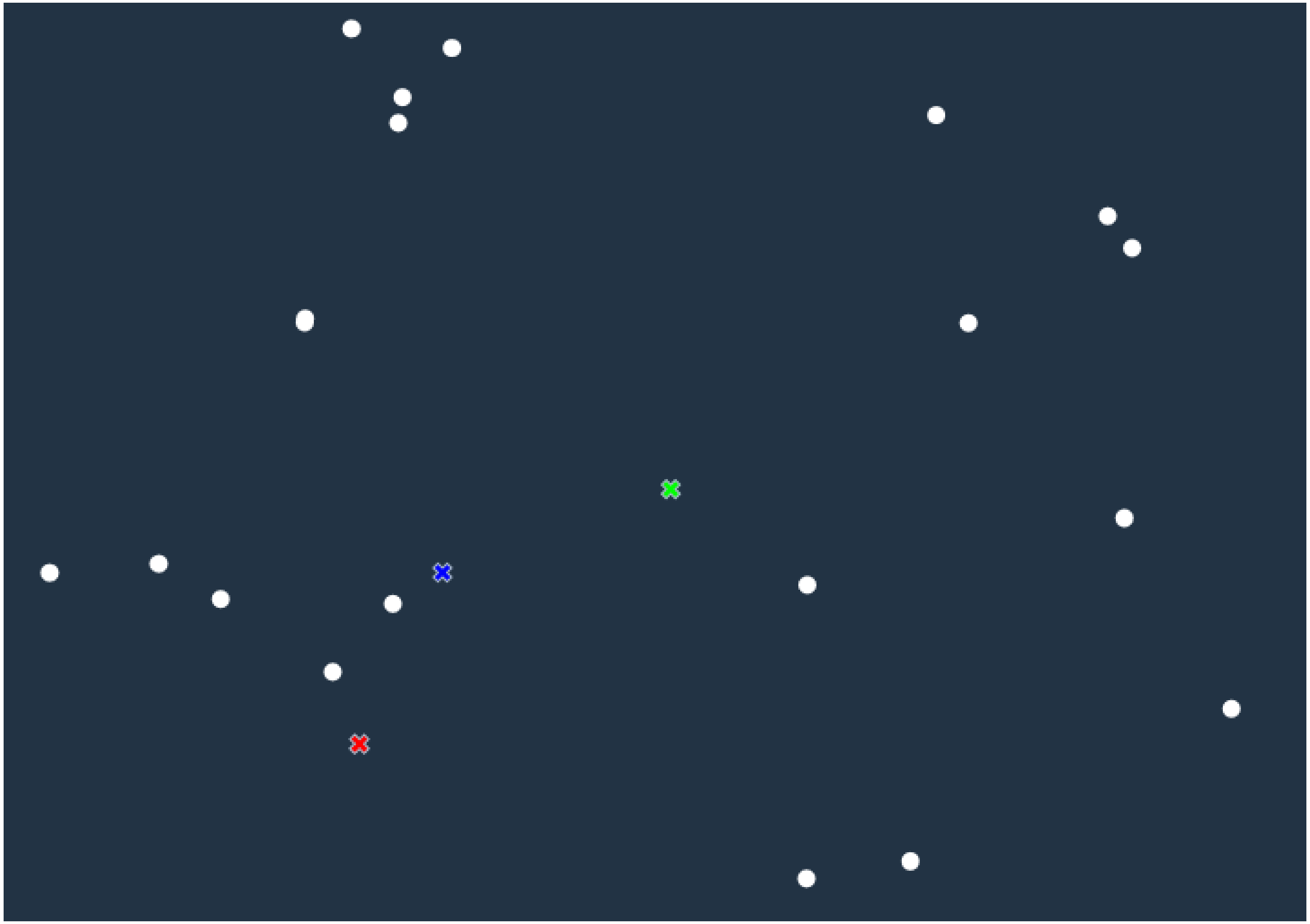
- **Class variable (target) with two or more outcomes.**
- **Splits records in a tree-like series of nodes along mutually-exclusive paths.**
  - Algorithm decides which variable and threshold value to use at each split
  - New records are predicted (classified) based on the leaf assignment
  - Accurate
  - Explicit decision paths
- **Can also handle continuous target (“regression tree”).**


Days Since Previous Visit <= 225
No
Yes
Readmit = No
Visits to Doctor in Past Year <= 3
No
Yes
Patient Age <= 61
Etc.
No
Yes
Etc.
Readmit = Yes

10% of patients are readmitted.

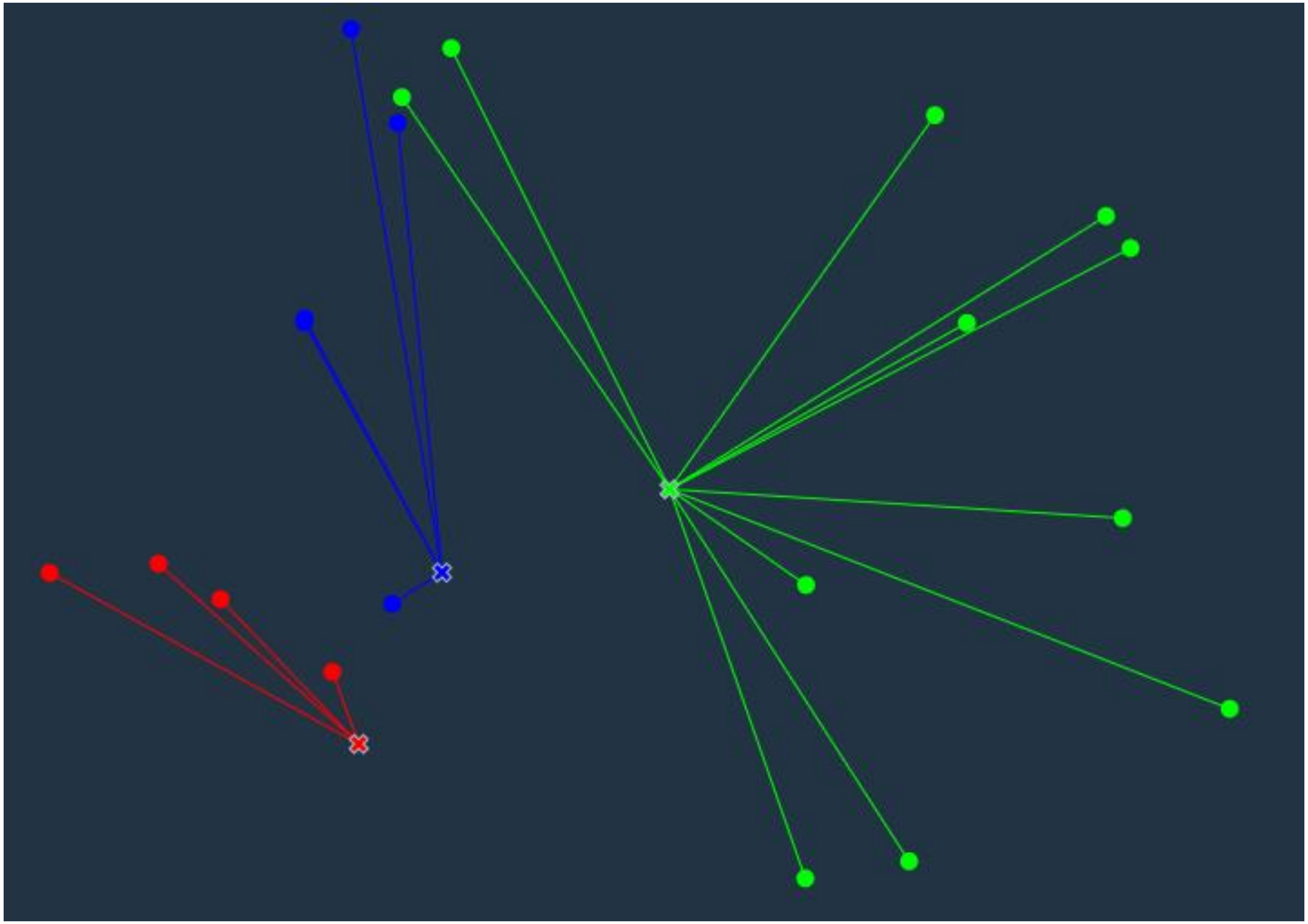
92% of patients are readmitted.

Clustering – K-means method (unsupervised)



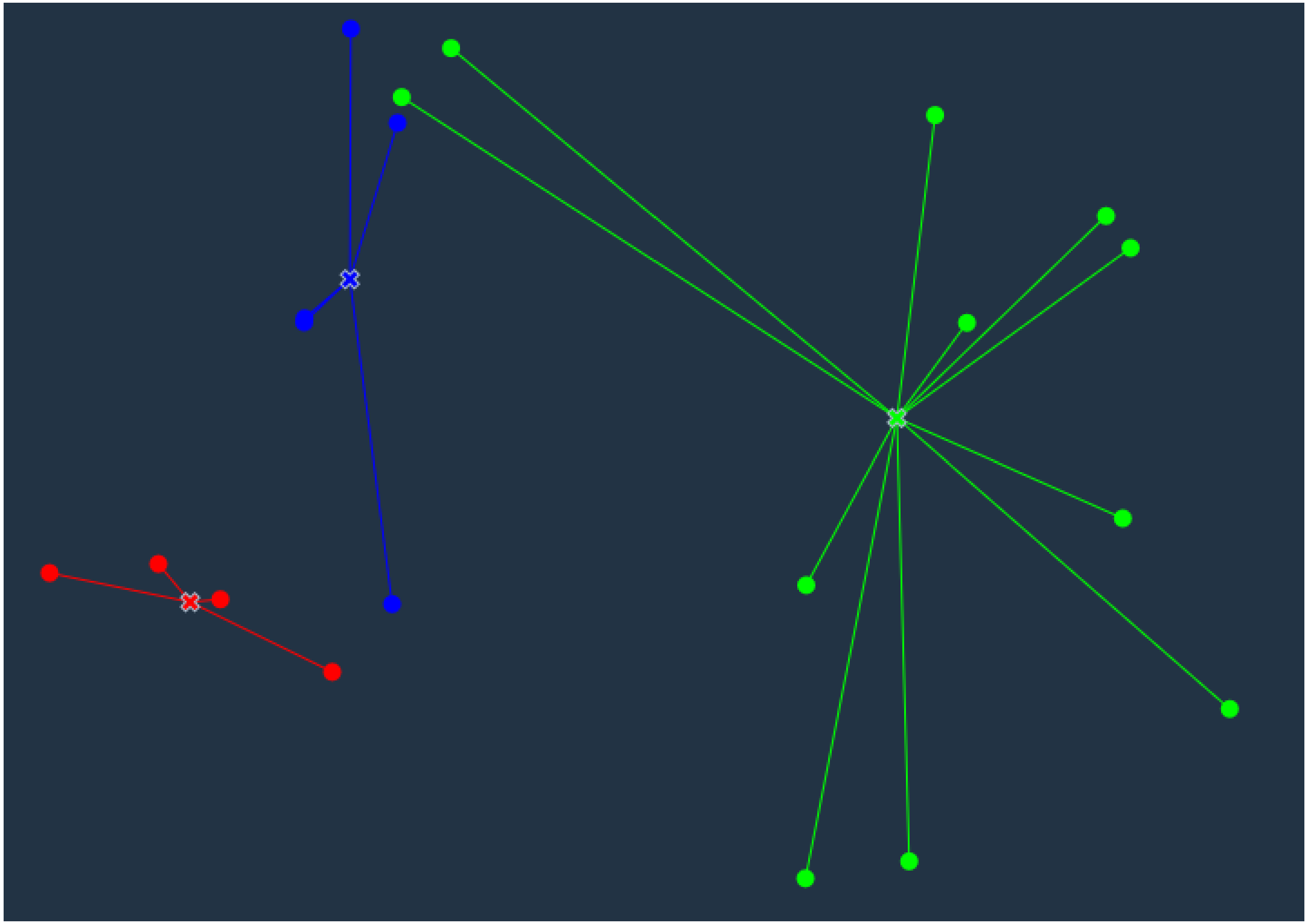
Start with 20 data points and 3 clusters

Clustering – K-means method



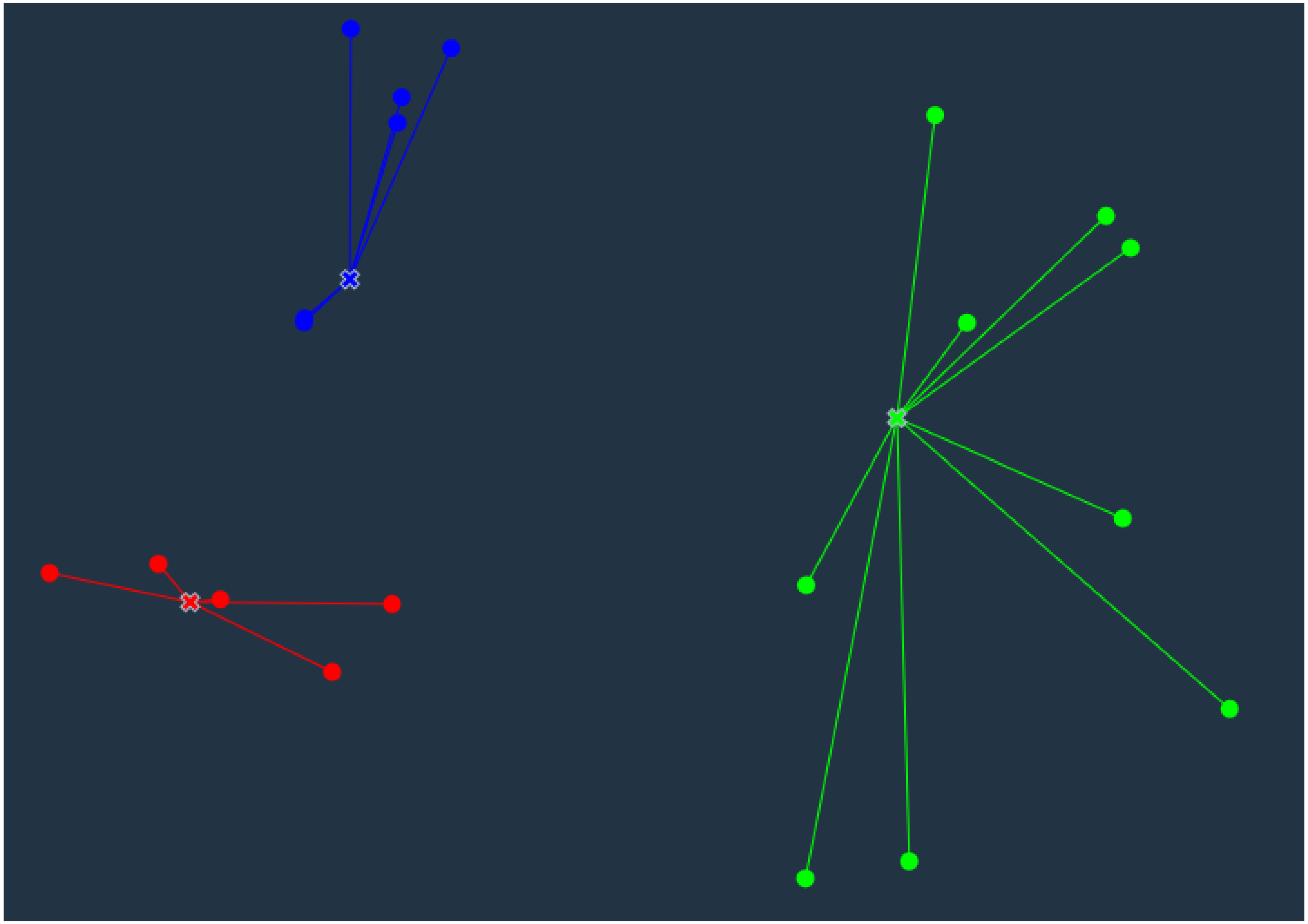
Assign each data point to the nearest cluster

Clustering – K-means method



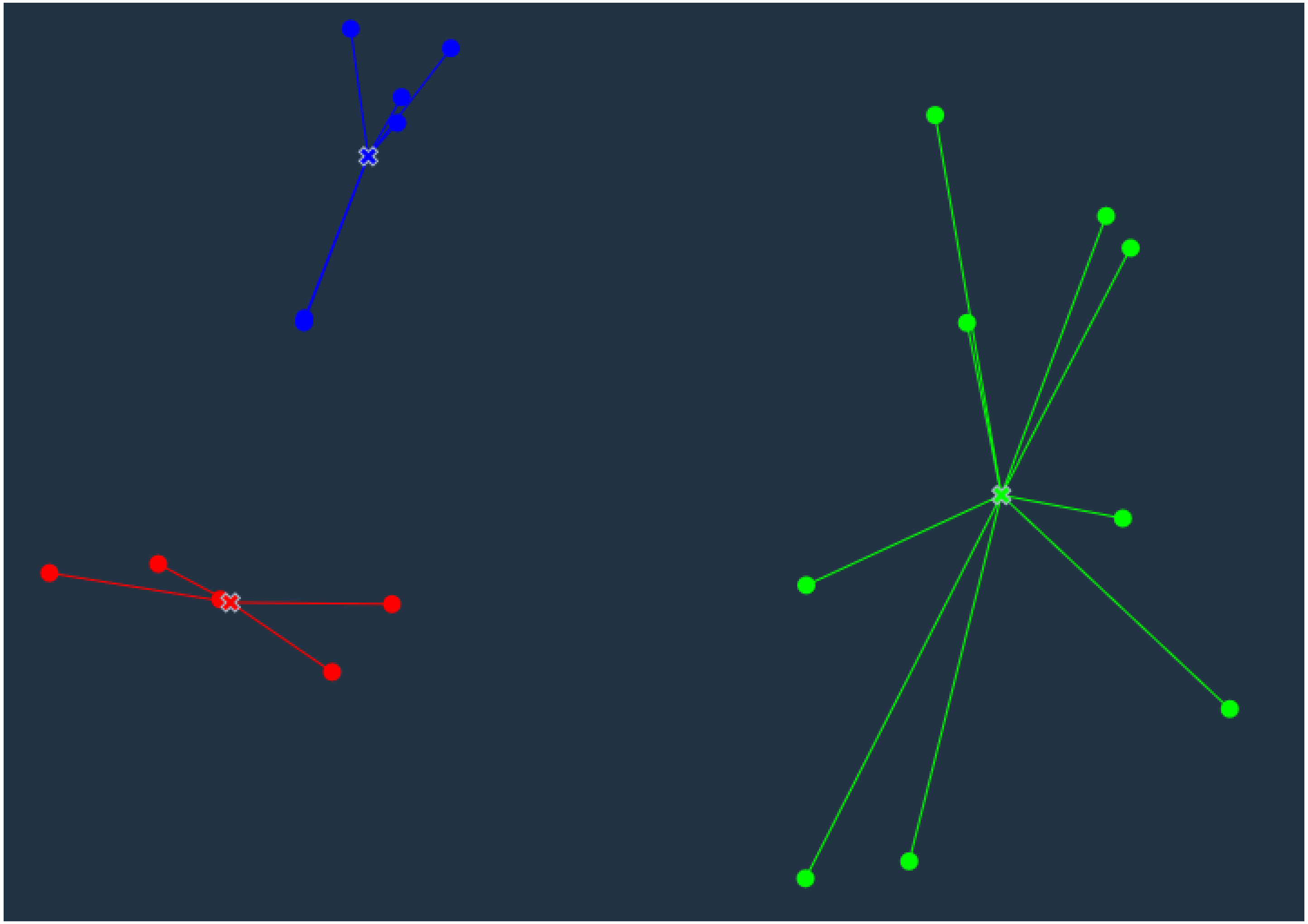
Calculate centroids of new clusters

Clustering – K-means method



Assign each data point to the nearest cluster

Clustering – K-means method



Calculate centroids of new clusters…until convergence

Ensemble Modeling

- **Use a collection or ensemble of models instead of a single model to create more reliable and accurate predictive models**

- **Bagging**
  - New training datasets are generated based on random sampling with replacement of the original data set
  - Models are constructed for each sample and the results are combined
  - Random Forest is bagging applied to Decision Trees
- **Boosting**
  - Successive models are built to predict observations misclassified from earlier models.
  - Gradient boosting - train each subsequent model on the residuals (error between predicted value and actual value).

Introduction to Machine Learning

- Overview
- Data Science Methodology
- Data Understanding
- Data Preparation
- Categories of Machine Learning
- Learning Challenges
- Machine Learning Algorithms
- Model Evaluation
- Deep Learning

Training, testing, & validation sets

- **During the model development process, supervised learning techniques employ training and testing sets and sometimes a validation set.**
  - Historical data with known outcome (*target*, *class*, *response*, or *dependent variable*)
  - Source data randomly split or sampled… mutually exclusive records
- **Why?**
  - Training set ➔ build the model (**iterative**)
  - Validation set ➔ tune the parameters & variables during model building (**iterative**)
    - Assess model quality during training process
    - Avoid overfitting the model to the training set
  - Testing set ➔ estimate accuracy or error rate of model (**once**)
    - Assess model's expected performance when applied to new data

K-Fold Cross Validation

- **Instead of using a separate validation set**

- **Shuffle Training Samples and sub-divide into "K" folds (groups)**

- **Train "K" models using K-1 folds as training data and 1 Fold as Test Data**

- **For example, K=4**
  - Model 1 Train on 1,2,3 Test on 4 – calculate and store E1 (Error)
  - Model 2 Train on 2,3,4 Test on 1 – E2
  - Model 3 Train on 3,4,1 Test on 2 - E3
  - Model 4 Train on 4,1,2 Test on 3 - E4
  - E = (E1+E2+E3+E4)/4

- **A common value for K is 10**

Model Evaluation: Confusion Matrix

Confusion matrix is more useful measure than simply using prediction accuracy

- Provides a better visualization of the performance of the algorithm
- Examine the count of each of these boxes

		Predicted	
		Has Disease	No Disease
Actual	Has Disease	true positive (tp) ✓	false negative (fn) No Treatment
	No Disease	false positive (fp) Unnecessary Treatment	true negative (tn) ✓

Precision = tp/(tp + fp) Recall = sensitivity= True Positive Rate tp/(tp + fn)

FPR = fp/(fp + tn) 1 – specificity ROC = plot of TPR/FPR at different thresholds

Model Evaluation: Confusion Matrix

Confusion matrix is more useful measure than simply using prediction accuracy

- Provides a better visualization of the performance of the algorithm
- Examine the count of each of these boxes

		Predicted	
		Has Disease	No Disease
Actual	Has Disease	true positive (tp) ✓	false negative (fn) No Treatment
	No Disease	false positive (fp) Unnecessary Treatment	true negative (tn) ✓

Precision = tp/(tp + fp)   Recall = sensitivity= True Positive Rate tp/(tp + fn)

FPR = fp/(fp + tn)   1 – specificity   ROC = plot of TPR/FPR at different thresholds

Model Evaluation: Confusion Matrix

Confusion matrix is more useful measure than simply using prediction accuracy

- Provides a better visualization of the performance of the algorithm
- Examine the count of each of these boxes

		Predicted	
		Has Disease	No Disease
Actual	Has Disease	true positive (tp) ✓	false negative (fn) No Treatment
	No Disease	false positive (fp) Unnecessary Treatment	true negative (tn) ✓

Precision = tp/(tp + fp)   Recall = sensitivity= True Positive Rate tp/(tp + fn)

FPR = fp/(fp + tn)   1 – specificity   ROC = plot of TPR/FPR at different thresholds

Model Evaluation: Confusion Matrix

Confusion matrix is more useful measure than simply using prediction accuracy

- Provides a better visualization of the performance of the algorithm
- Examine the count of each of these boxes

		Predicted	
		Has Disease	No Disease
Actual	Has Disease	true positive (tp) ✓	false negative (fn) No Treatment
	No Disease	false positive (fp) Unnecessary Treatment	true negative (tn) ✓

Precision = tp/(tp + fp) Recall = sensitivity= True Positive Rate tp/(tp + fn)

FPR = fp/(fp + tn) 1 – specificity ROC = plot of TPR/FPR at different thresholds

Model Evaluation

- **When you are building a classifier, it is important to understand the PREVALANCE of the condition that you are building a model for,**
  i.e. how common or uncommon this condition effectively is…

- **Imagine you are working towards building a classifier for some medical condition and your training and testing data sets yield the following model**

	Test positive	**Test negative**
Disease (100)	95 (True Positive)	5 (False Negative)
Normal (100)	5 (False Positive)	95 (True Negative)

Accuracy = 95% **Recall = 95%** **Precision=95%**

Model Evaluation

- **What truly matters to the users of your new model / test (doctors, bankers, practitioners) is the PREDICTIVE VALUE of the test:**
  - If the test is positive, then what is the actual chance of being sick?
  - Is it 95% ?

- **Let's run the test on a population of 1,000,000 where 1% individuals (10,000) are actually suffering from this condition:**

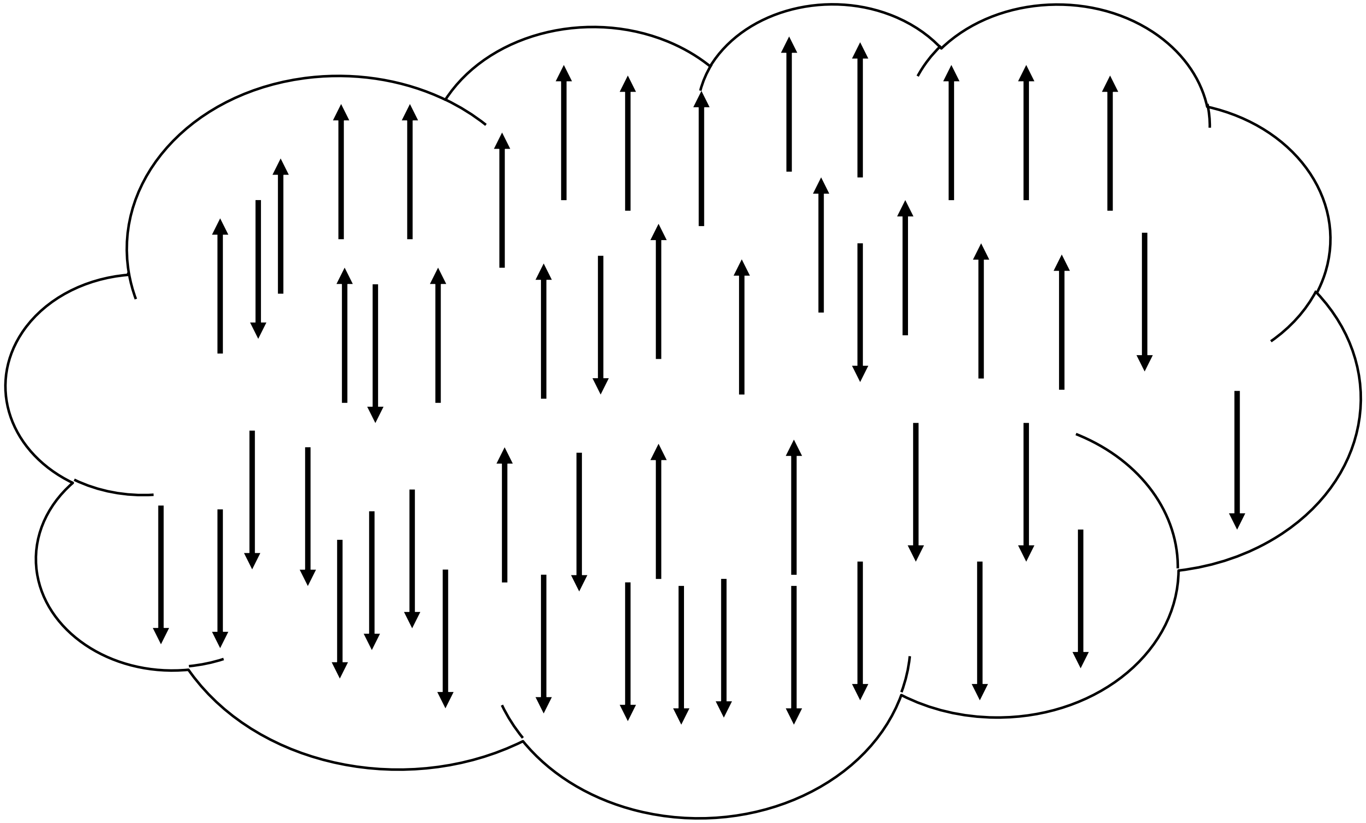
	Test positive	**Test negative**
Disease (10000)	9500 (95% True Positive)	500 (5% False Negative)
Normal (990000)	49500 (5% False Positive)	940500 (95% True Negative)

Accuracy = 95%   Precision=16.1%   Recall = 95%

What is happening here:

The condition is RARE and the 5% FALSE POSITIVES are still way higher in numbers than the true positives. Need 99% or higher specificity.

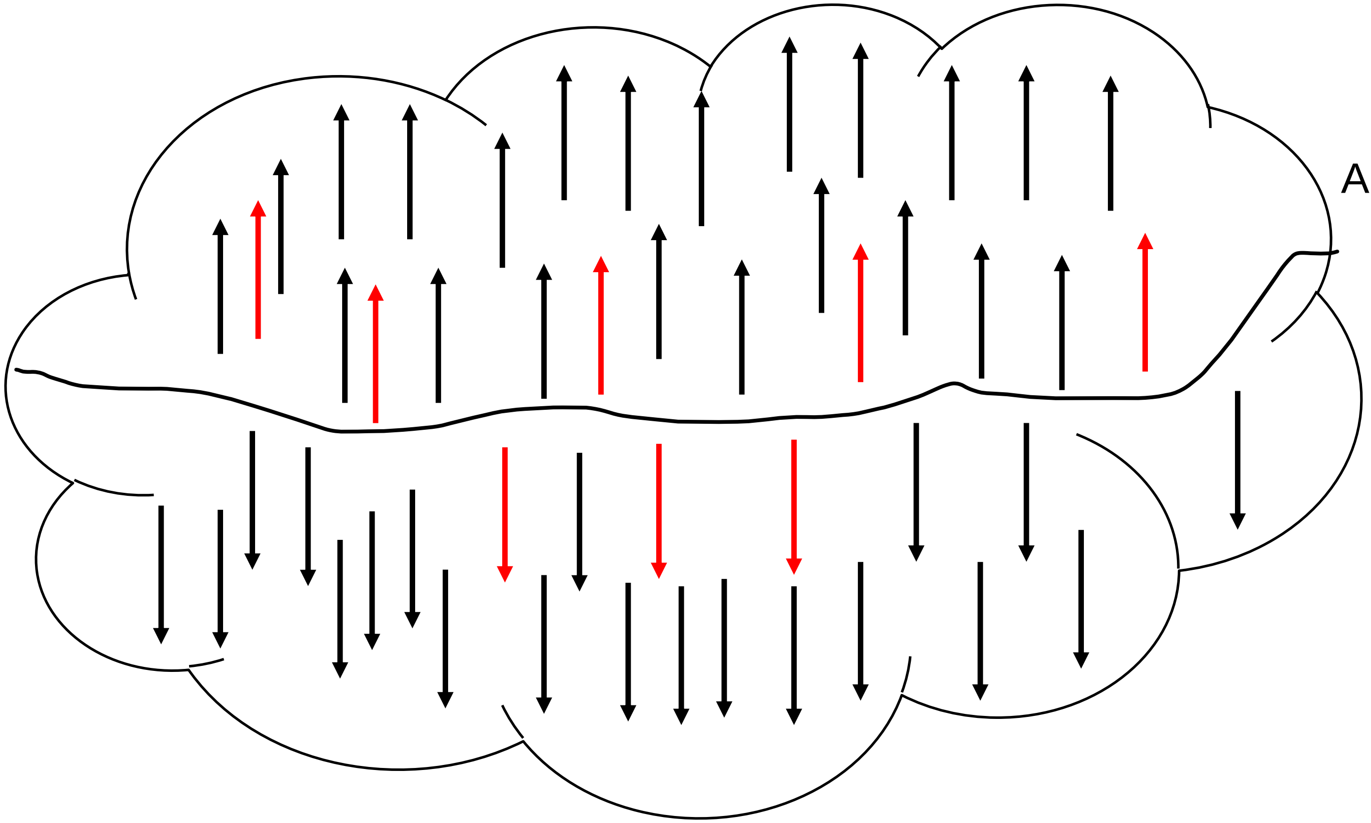
Model Evaluation – Visual Example



↑ Has Disease (HD) 25

↓ No Disease (ND) 25

Evaluation Metrics – Confusion Matrix

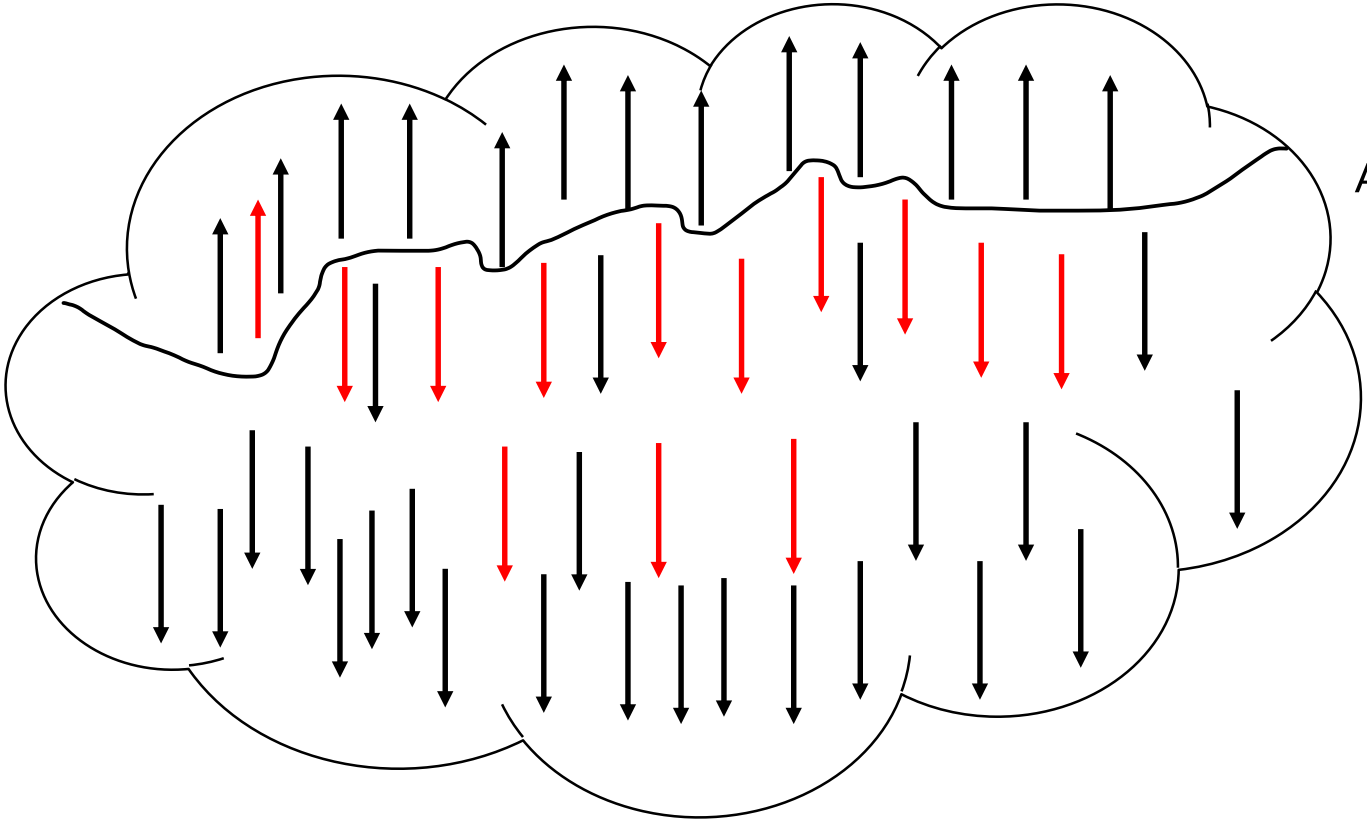


		Predicted	
		↑ HD	↓ ND
Actual	↑ HD	↑ tp 22	↓ fn 3
	↓ ND	↑ fp 5	↓ tn 20

Precision = tp/(tp + fp) = 81%   Recall = sensitivity= True Positive Rate tp/(tp + fn) = 88%

FPR = fp/(fp + tn) = 20%   ROC = plot of TPR/FPR at different thresholds

Evaluation Metrics – Confusion Matrix



		Predicted	
		↑ HD	↓ ND
Actual	↑ HD	↑ tp 13	↓ fn 12
	↓ ND	↑ fp 1	↓ tn 24

Precision = tp/(tp + fp) = 92.3% Recall = sensitivity= True Positive Rate tp/(tp + fn) = 52%

FPR = fp/(fp + tn) = 4% ROC = plot of TPR/FPR at different thresholds

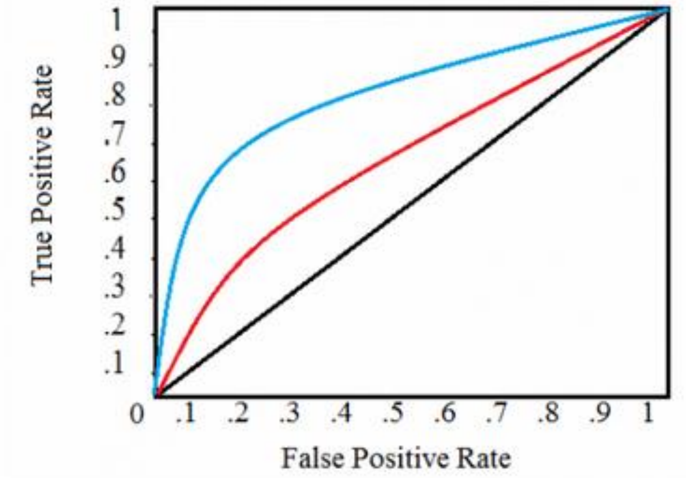
Model Evaluation - Metrics

- $Accuracy = \frac{Correct\ Predictions}{Total\ Predictions} = \frac{Tp+Tn}{Tp+Tn+Fp+Fn}$

- $F1 = 2 * \frac{Recall*Precision}{Recall+Precision}$

- **Area under Receiver Operating Characteristic (ROC)**

Source

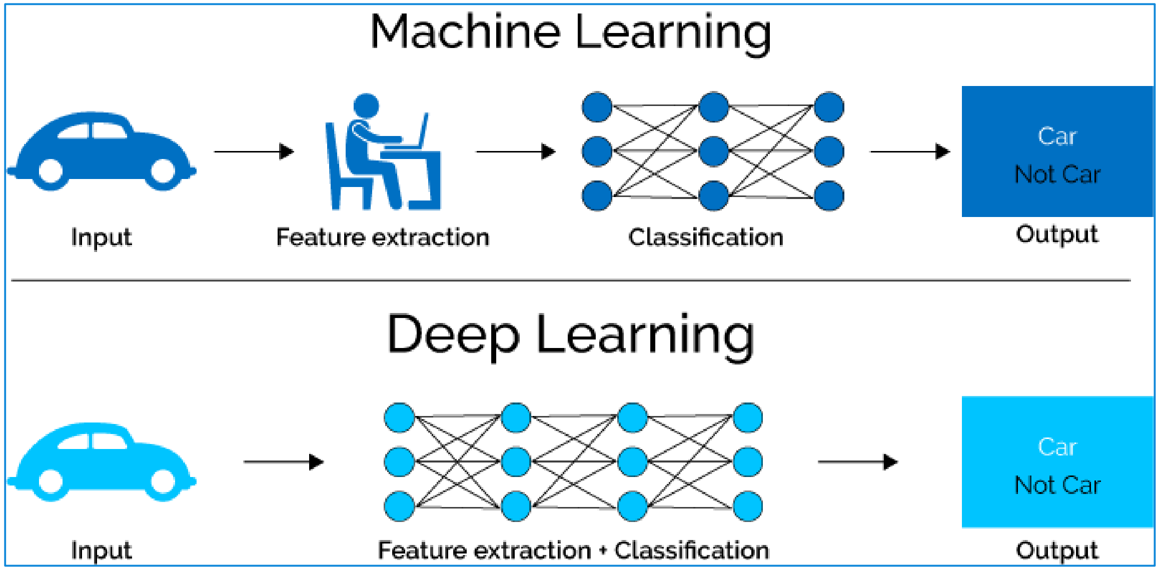


Introduction to Machine Learning

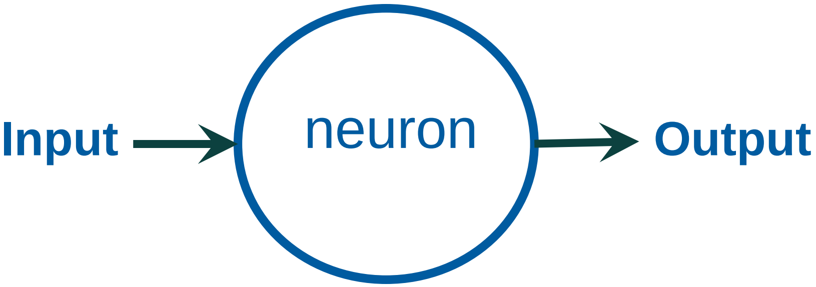
- Overview
- Data Science Methodology
- Data Understanding
- Data Preparation
- Categories of Machine Learning
- Learning Challenges
- Machine Learning Algorithms
- Model Evaluation
- Deep Learning

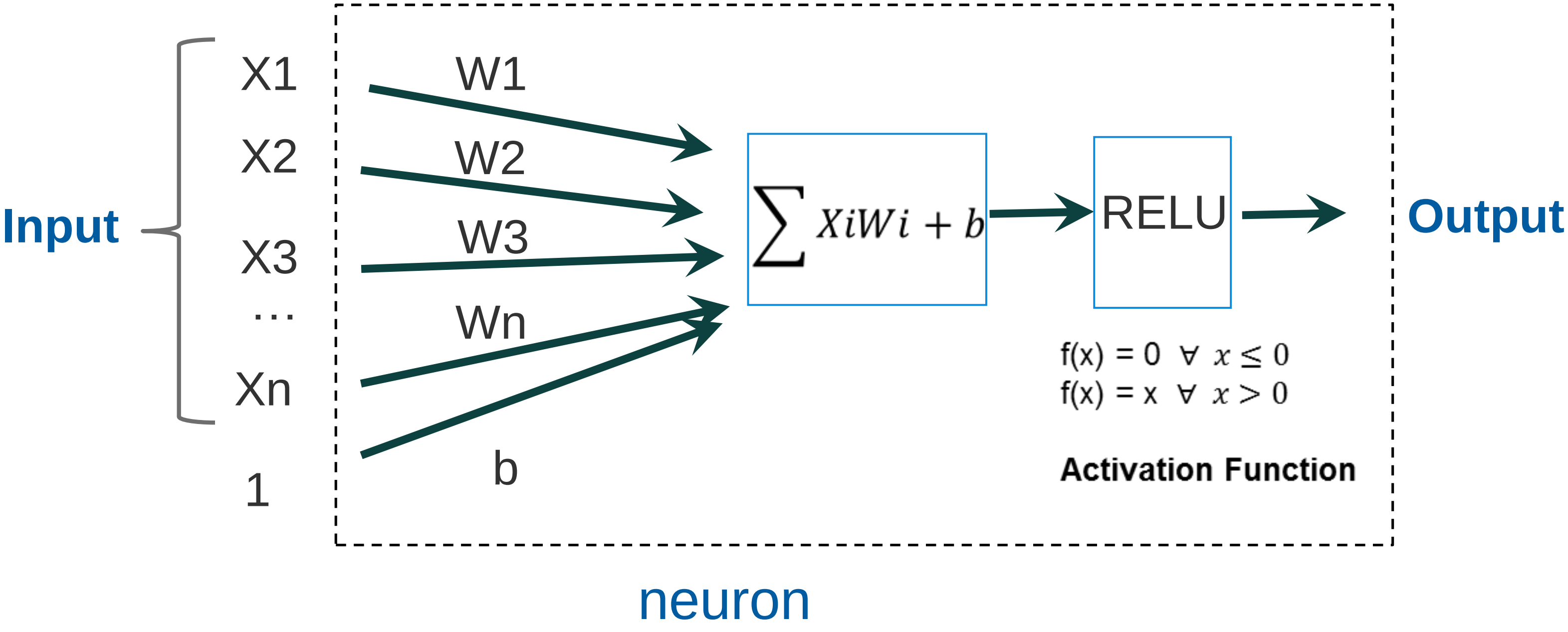
Deep Learning

- Deep Learning is a machine learning method.
- Could be supervised or unsupervised
- Originated in 1940s
- Became very popular this decade
  - Hardware Improvements/Cost – GPUs, Storage
  - Availability of Large Datasets for Training
  - Better performing algorithms.
- Especially good for human perception type task



What is an Artificial Neuron



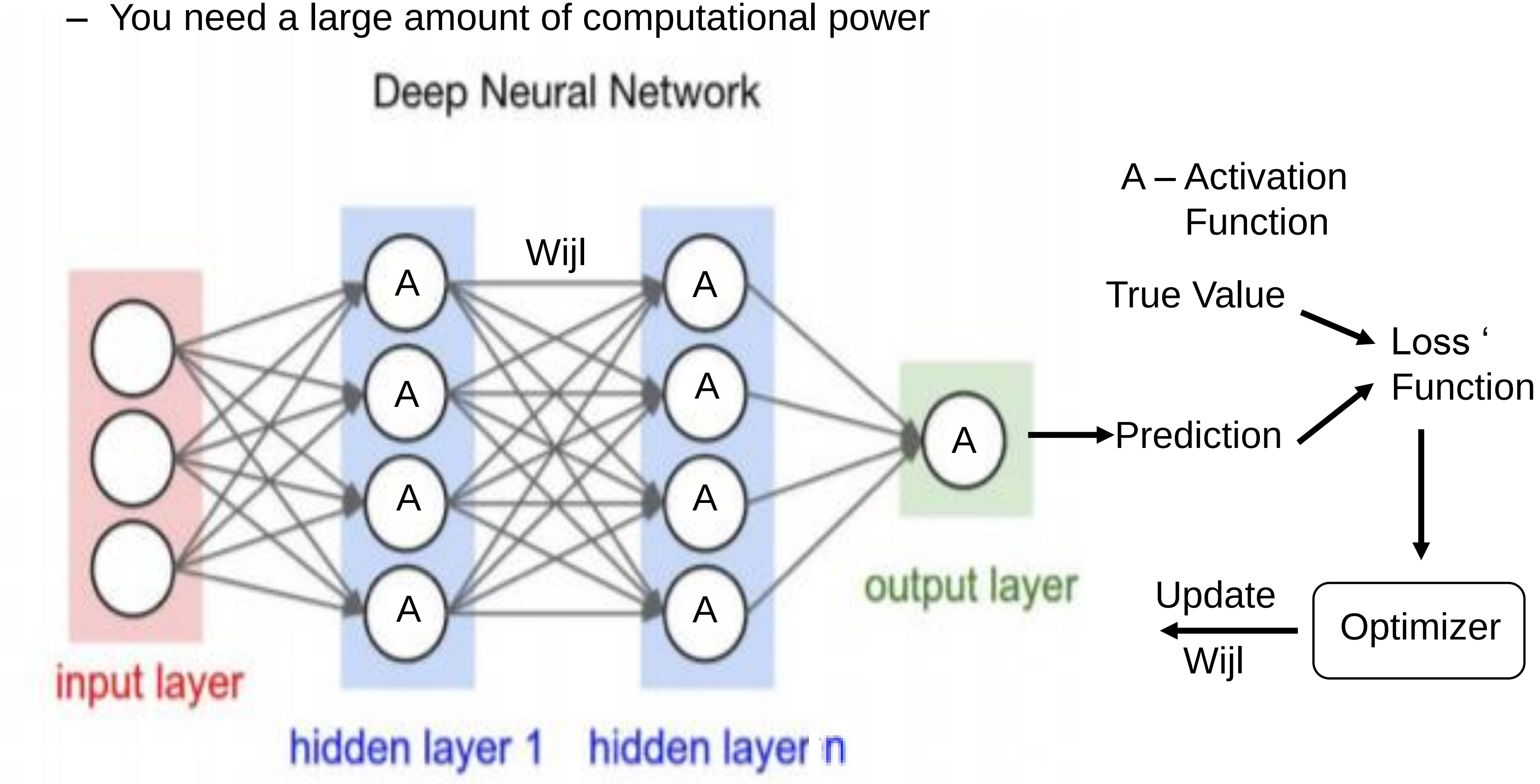


Modeling

Neural Network

Training the AI is the hardest part of Deep Learning.

- You need a large data set.
- You need a large amount of computational power



A – Activation Function

True Value

Prediction

Loss ‘ Function

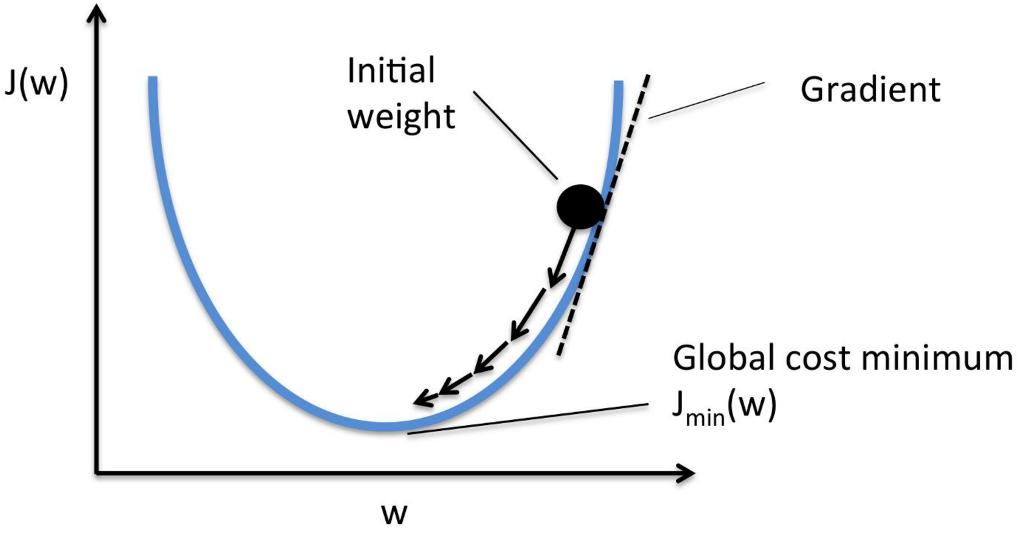
Optimizer

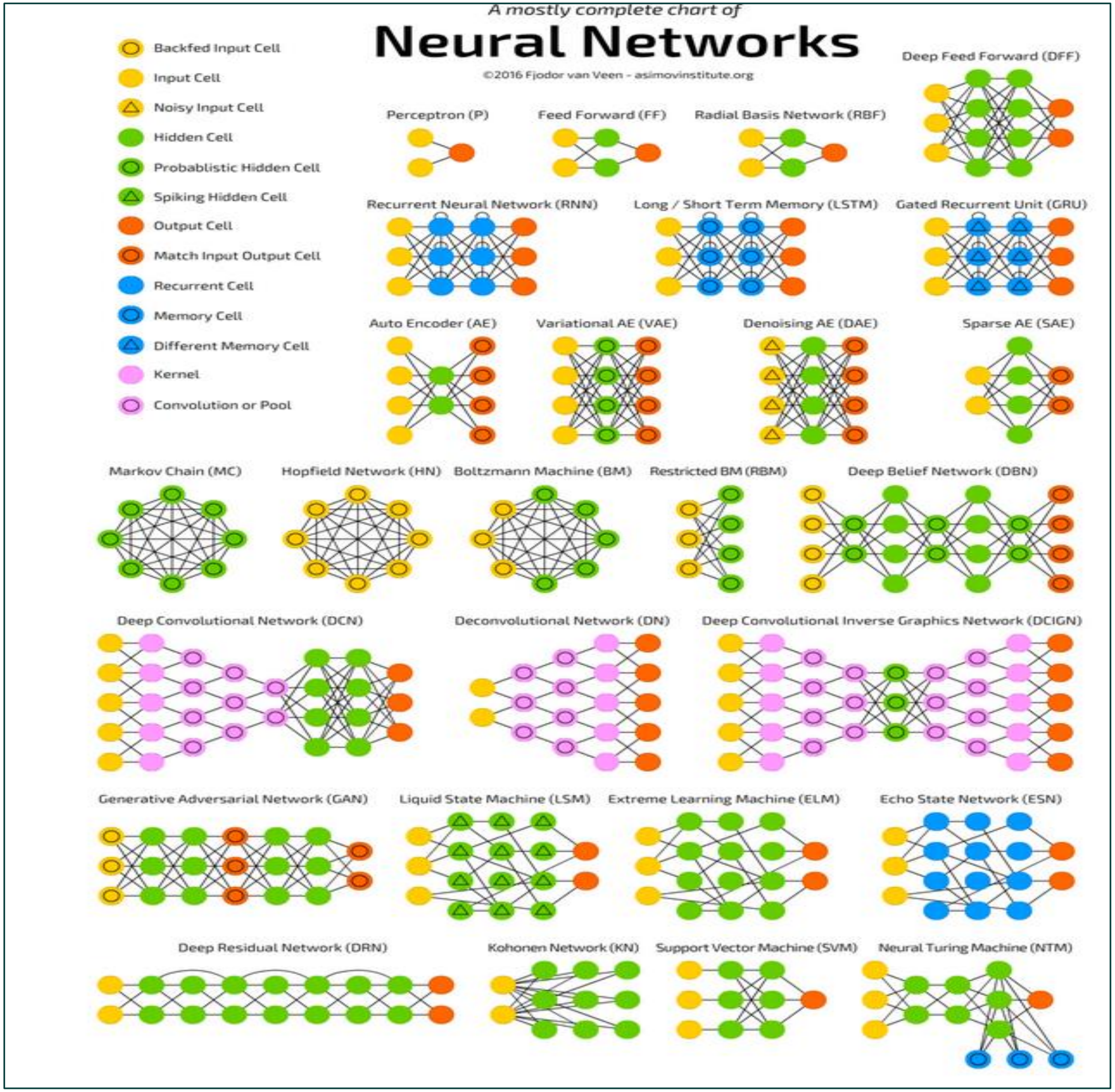
Update Wijl

Wijl – weight from neuron (i) in level (l-1) to neuron (j) in level (l)

Cost Function

- During training we need to know how our DNN is doing!
  - Compare it’s predictions to dataset’s output
  - Based on how far it is from actual value → update weights
- This function that does this is called ”Loss Function”
- Ideally, we want our loss function to be zero.
  - Does not happen in real world
  - use techniques like “Gradient Descent” → allows us to find the minimum of a function by iterating through dataset and updating the weights




A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

Common Types of Deep Neural Networks

Convolutional Neural Networks

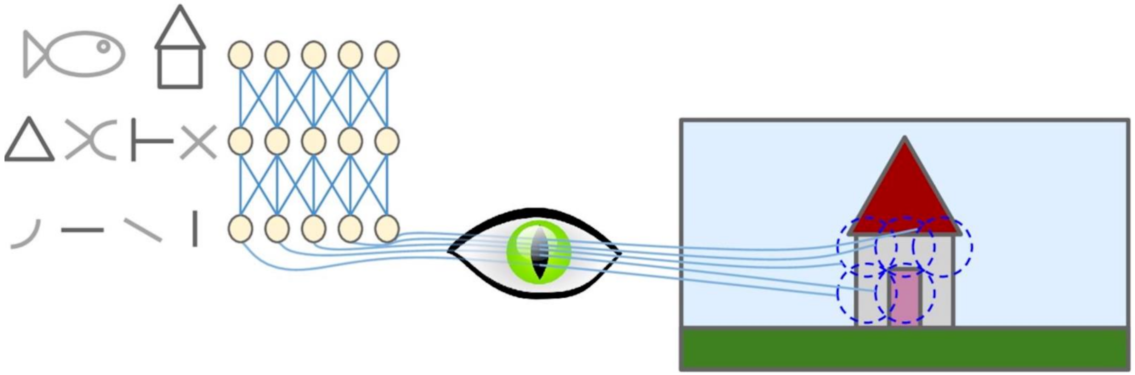
- Image classifications
- Object detection
- Recognizing faces
- Natural language processing
- ..

Recurrent Neural Networks

- Speech Recognition
- Handwriting Recognition
- Machine Translation
- Sequence prediction
- Natural Language Processing
- …

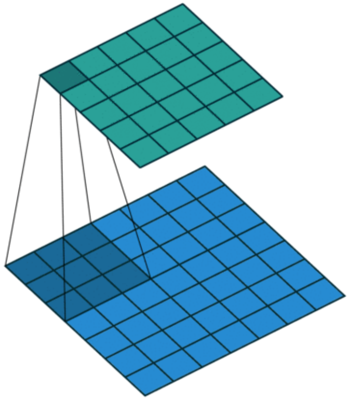
Convolutional Neural Networks (CNN)

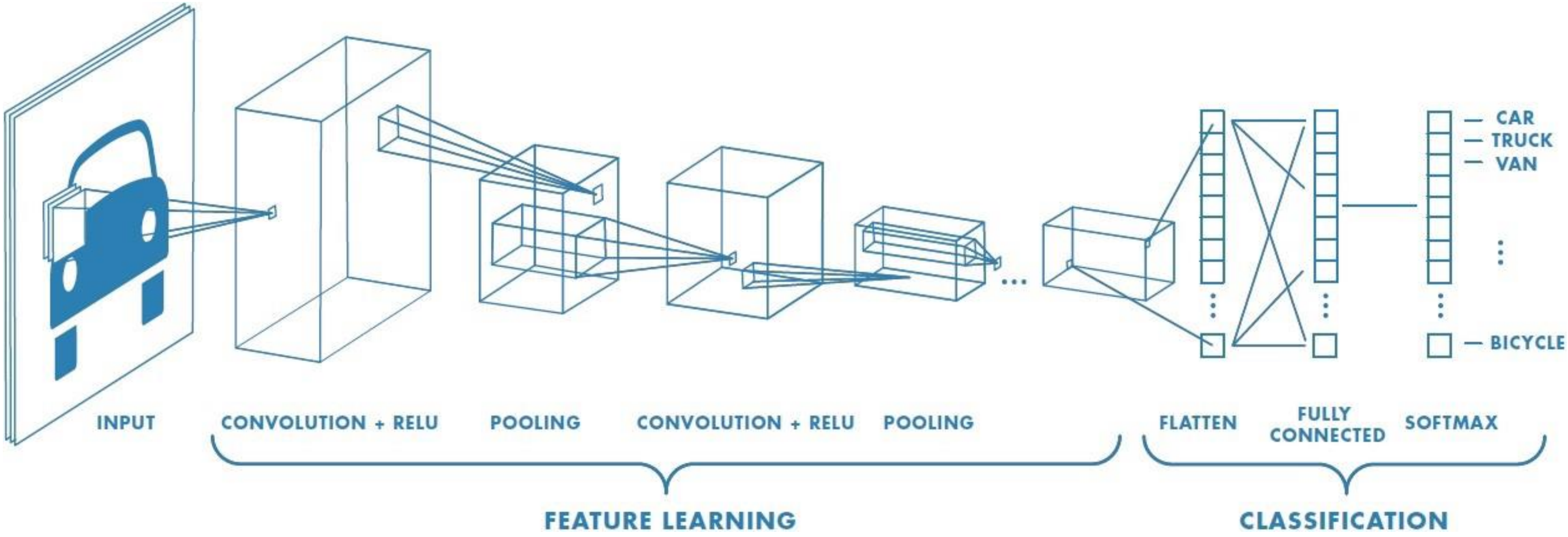
- Inspired by the architecture of the visual cortex
- Some neurons only react to horizontal lines, while others reacts to lines with different orientations
- Some neurons have larger receptive fields, so they react to more complex patterns based on output of lower-level neuron
- Two building blocks: **Convolutional** layers and **Pooling** layers



Convolutional Layer

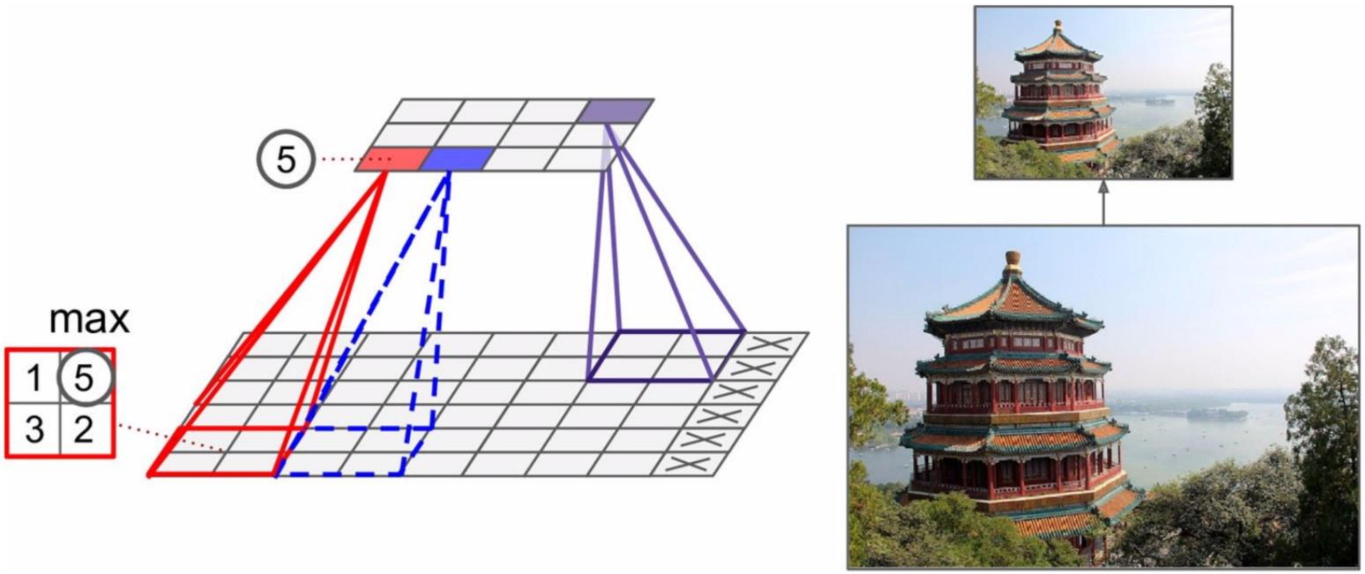
- Convolution is the first layer to extract features from an input image.
- Neurons in the first convolutional layer are not connected to every pixel in the input image, but **only to pixels in their receptive fields**
- Each neuron in the second convolutional layer is connected only to **neurons located within a small rectangle** in the first layer.
- Convolution preserves the relationship between pixels by learning image features using small squares of input data.



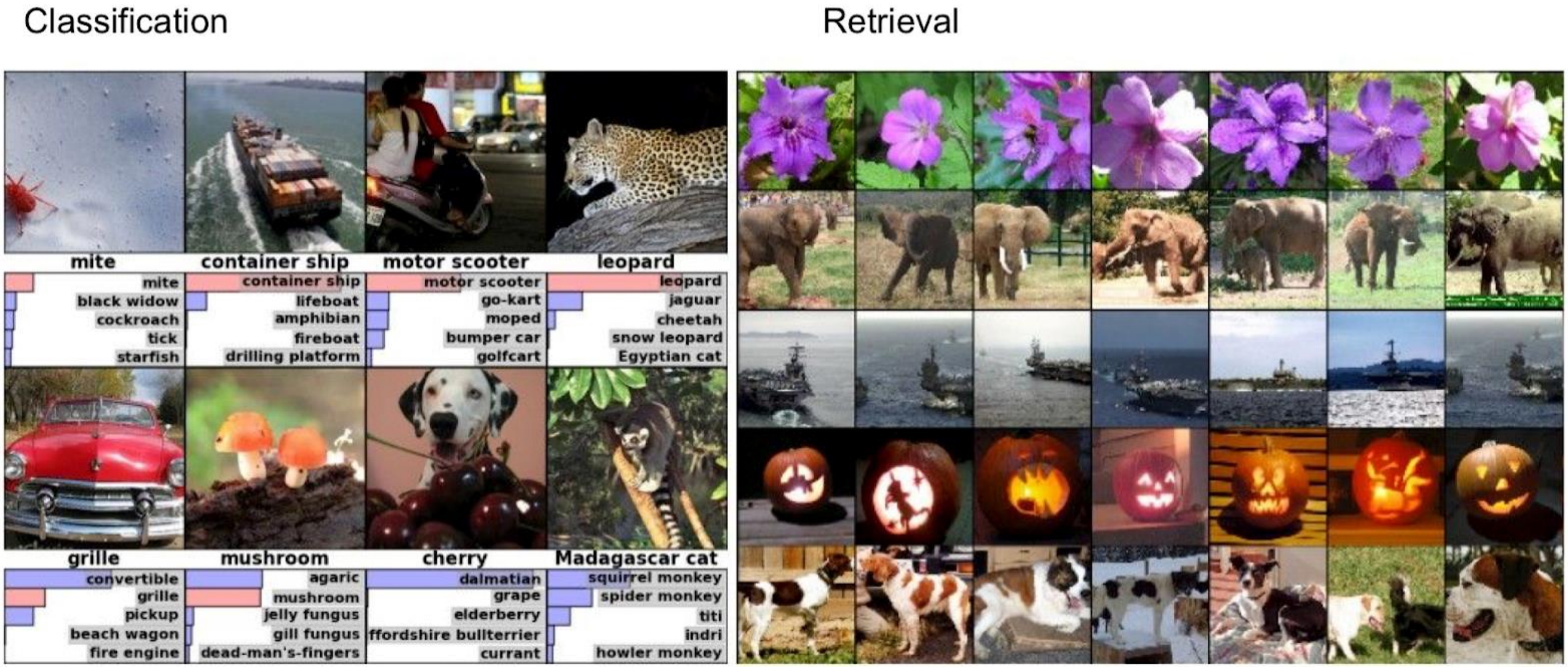


Pooling Layer

- Pooling Layer reduce the number of parameters when the images are too large.
  - Max Pooling
  - Average Pooling
  - Sum Pooling



CNN Applications



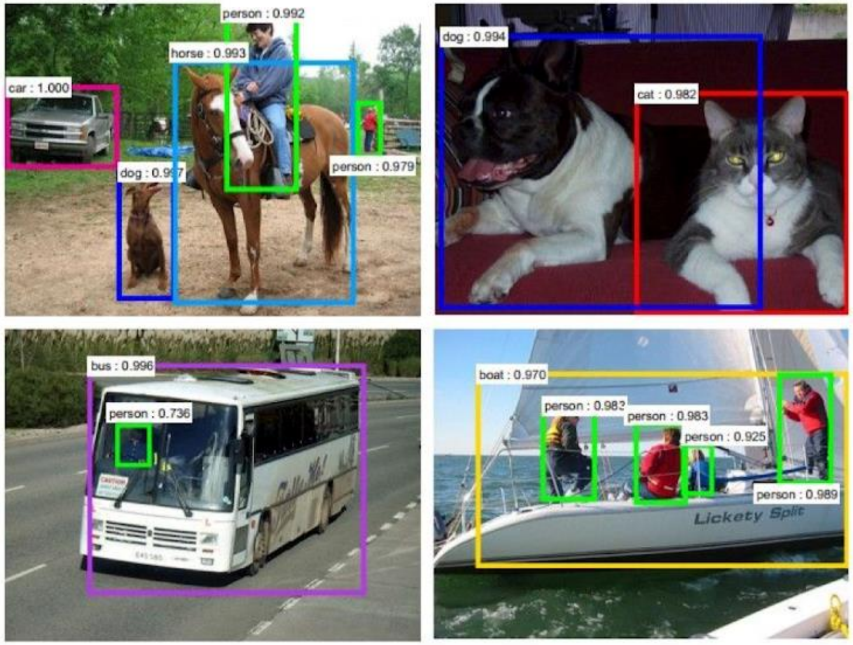
Classification

Retrieval

Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

CNN Applications

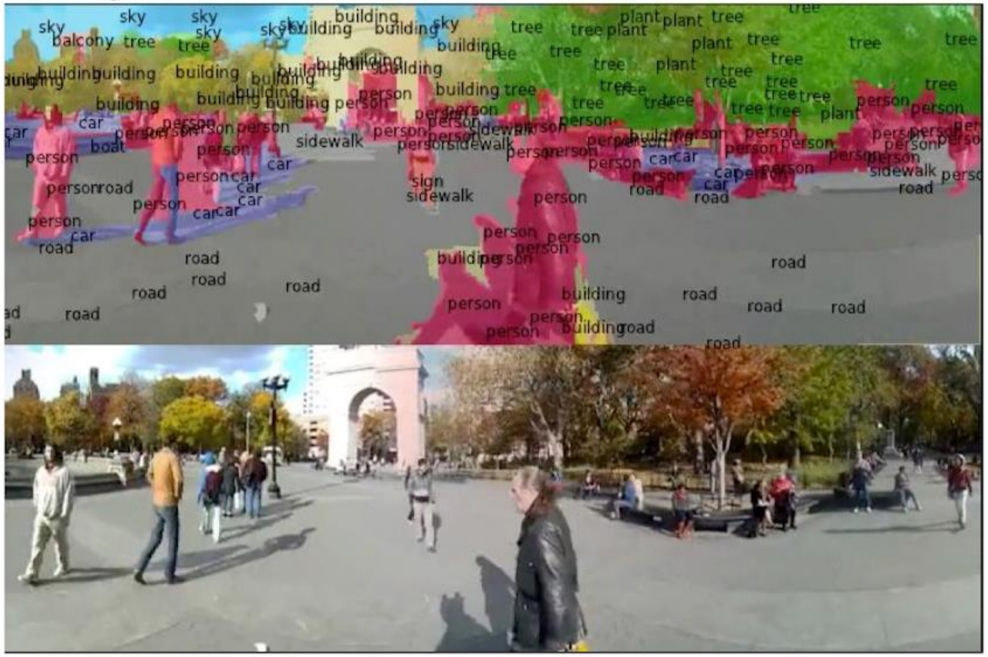
Detection



Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation

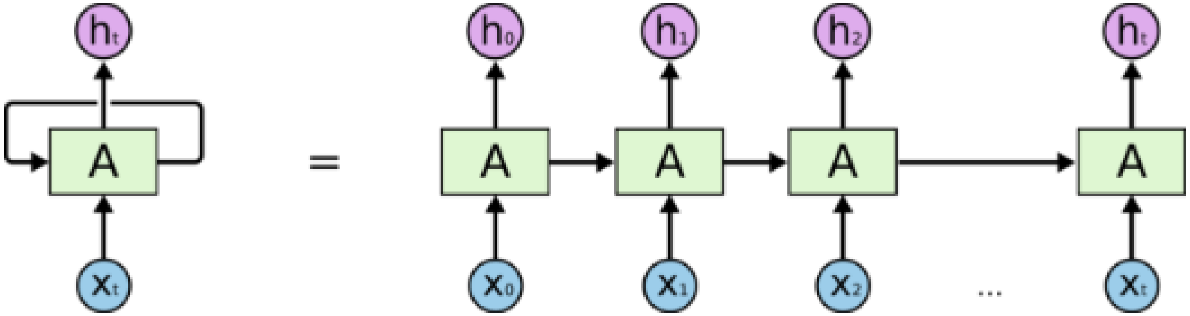


Figures copyright Clement Farabet, 2012. Reproduced with permission.

[Farabet et al., 2012]

Recurrent Neural Networks (RNN)

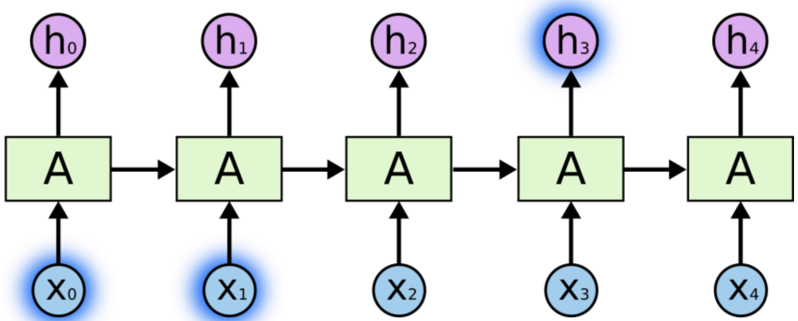
- Humans don’t start their thinking from scratch every second. We rely on our memory!
- Traditional Neural Networks **CAN NOT** help → data flows forward only
- Recurrent Neural Networks address this issue.
  - They are networks with loops in them, allowing information to persist.
- RNN Applications are: Speech recognition, Language modeling, Translation



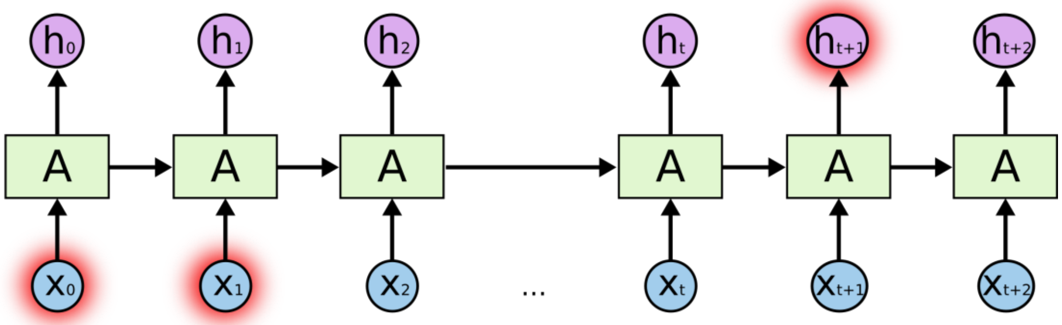
An unrolled recurrent neural network.

Long-Term Dependencies Problem

- RNNs can look at old information. But how old?
- Successfully uses recent information → Clouds are in the … [Sky]



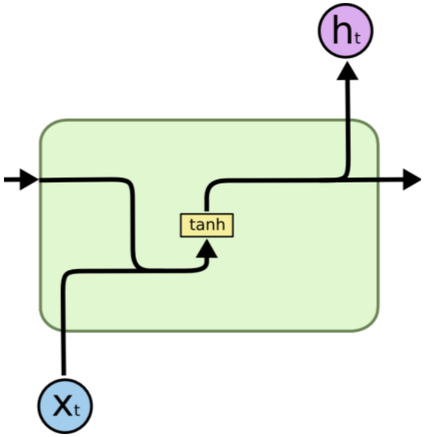
- Can NOT use older information → I grew up in France, in a small city near Paris, so I speak fluent … [??] **But why?**



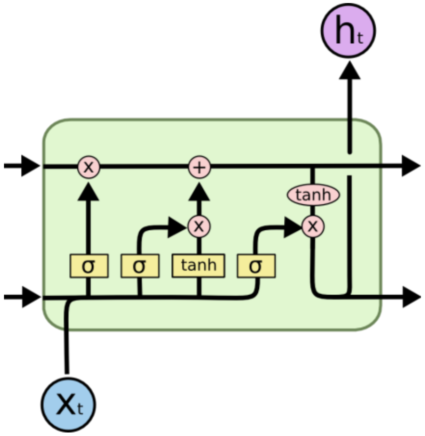
Long-Term Dependencies Problem

- **Vanishing gradients**
  - 0 < Gradient values < 1 → they leave the connection weights **unchanged**
- **Exploding gradients**
  - Calculated gradients are large values → many layers got **insanely large** weight updates, and the network becomes unstable and diverged

- **Long Short Term Memory (LSTM) fixed the gradient problem**
  - by introducing a few more gates that control access to the cell state



The repeating module in RNN contains a single layer.

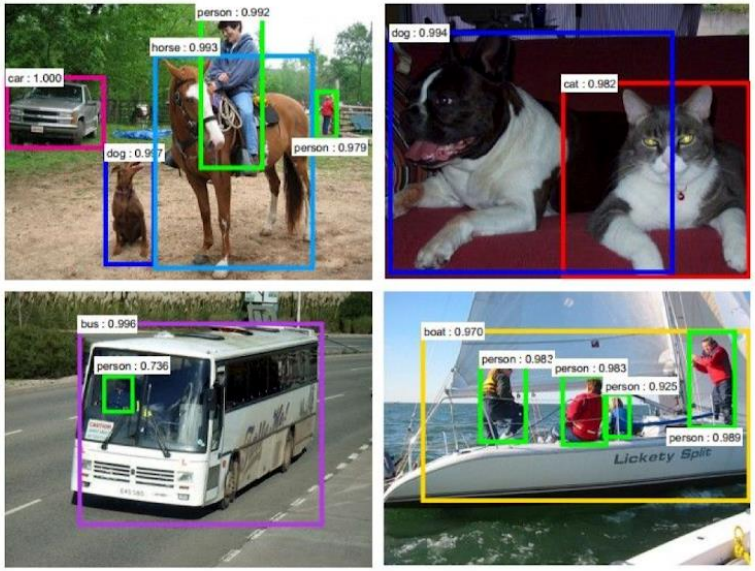


The repeating module in LSTM contains four interacting layers

DNN in Visual Recognition

- **We use DNNs for lots of things:**
  - Facial recognition → iPhone FaceID
  - Text recognition → Mobile Check Deposit
  - Self driving cars → help detect signs, pedestrians, traffic lights, etc.

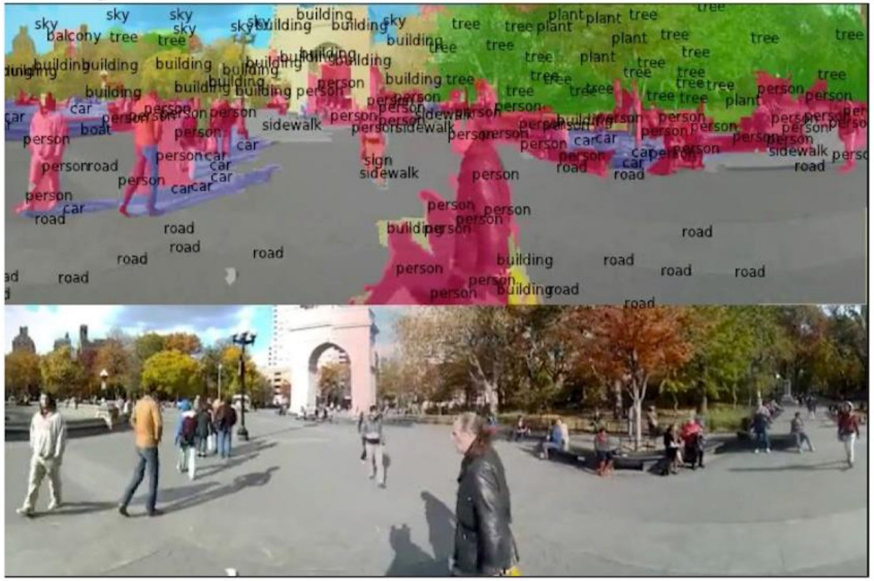
Detection



Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



Figures copyright Clement Farabet, 2012. Reproduced with permission.

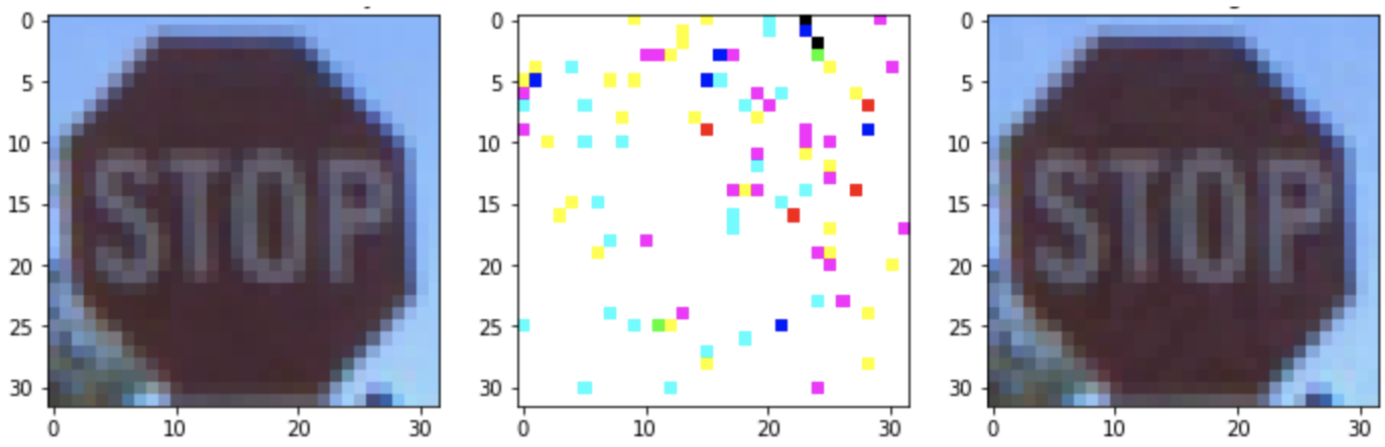
[Farabet et al., 2012]

What are Adversarial Images?

- Adversarial examples are inputs (say, images) which have deliberately been modified to produce a desired response by a DNN.

- Often, the target of adversarial examples is **misclassification** or a **specific incorrect prediction** which would benefit an attacker.

“Stop Sign“ + Noise = “Ahead Only” Sign



Adversarial Attacks

- Carefully crafted noise added to the input
- This noise gets amplified when ran through the layers of DNN
- The goal is to generate adversaries visually similar to original image

Stronger
BUT
More
Expensive

- FGSM
- Random + FGSM
- Projected Gradient Descent *
- DeepFool
- JSMA
- C&W

Threat Model

Black Box

- Attackers can only observe the outputs of a model. E.g. Attacking a model via an API
  - The adversary has no knowledge of the training algorithm or hyperparameters.

➢ Examples:
  - ➢ Boundary Attack
  - ➢ Substitute Blackbox Attack
  - ➢ Etc.

White Box

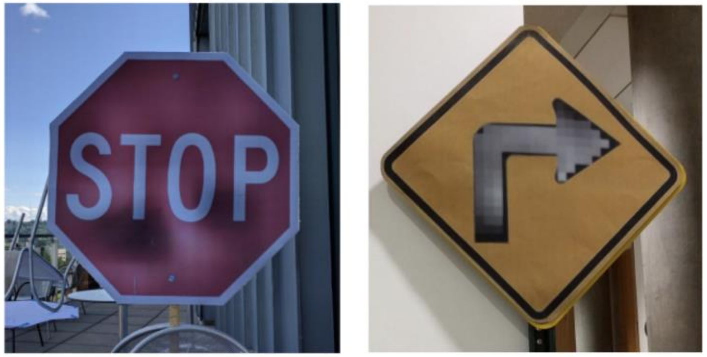
- attackers have complete access to the model that they want to attack.
- These are most effective attacks

➢ Examples:
  - ➢ Fast Gradient Sign Method (FGSM)
  - ➢ Random + FGSM
  - ➢ Projected Gradient Descent
  - ➢ Etc.

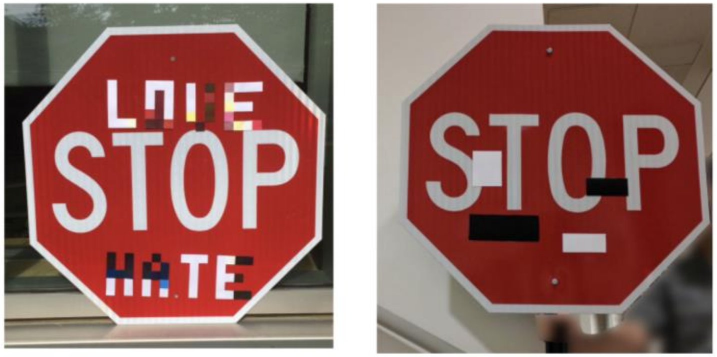
Why are they dangerous?

- **Can be crafted even if the attacker doesn't have exact knowledge of the architecture of the DNN**
- **Adversarial attacks can be launched in the physical world**
  - adversaries could evade face recognition systems by wearing specially designed glasses
  - defeat visual recognition systems in autonomous vehicles by sticking patches to traffic signs

Subtle Poster



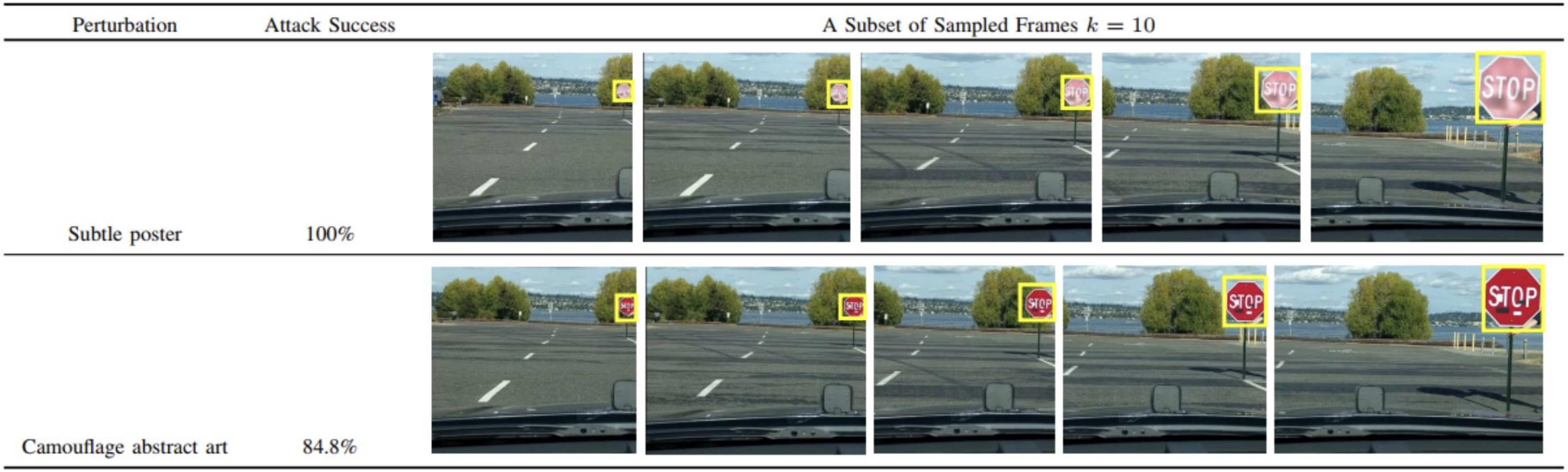
Camouflage Sticker



* Pictures from paper: Kevin Eykholt, et al. "Robust Physical-World Attacks on Deep Learning Visual Classification"

Are they effective?

- **Researchers proved that these attacks are successful! [1]**



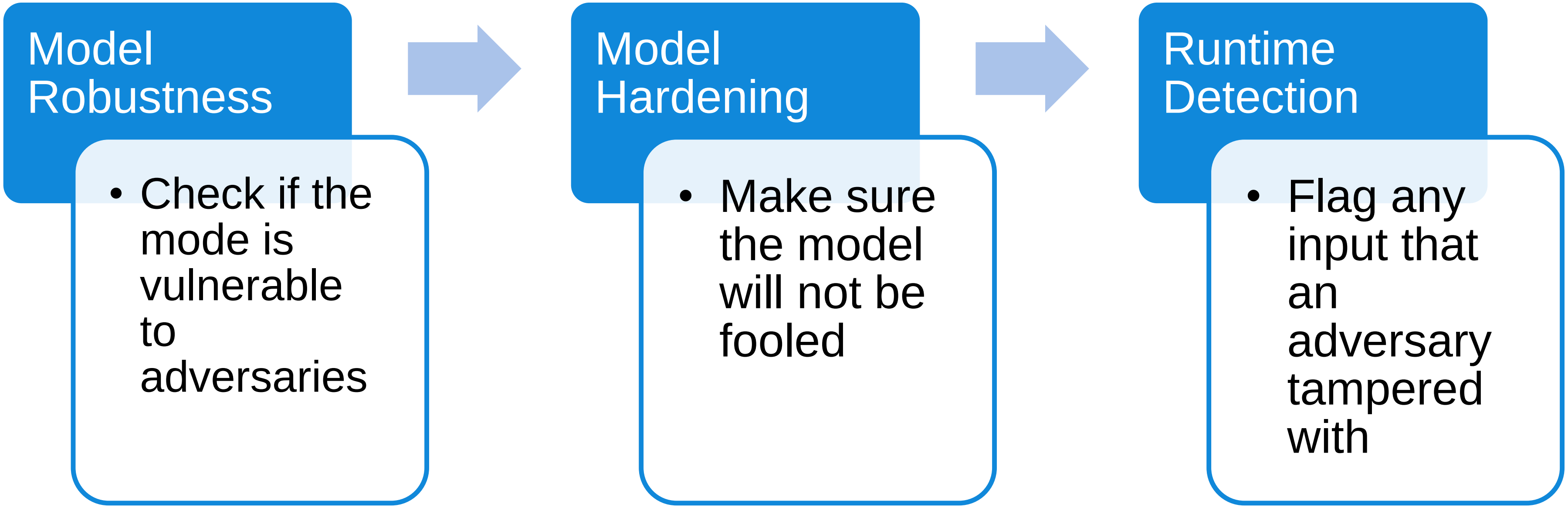
Perturbation	Attack Success	A Subset of Sampled Frames $k = 10$
Subtle poster	100%	
Camouflage abstract art	84.8%	

[1] Kevin Eykholt, et al. “Robust Physical-World Attacks on Deep Learning Visual Classification”

Adversarial Robustness Toolbox (ART)

- IBM Research team in Ireland developed the toolkit to help defend DNNs against adversarial attacks
- Open-source software library
- Written in python
- Supports most deep learning frameworks : TensorFlow, Keras, PyTorch, etc.
- It creates adversarial examples AND provides methods for defending DNNs against those.

How can ART help?



Model Robustness

- Check if the mode is vulnerable to adversaries

Model Hardening

- Make sure the model will not be fooled

Runtime Detection

- Flag any input that an adversary tampered with

Conclusions

- **Adversarial attacks are real threats**
  - Self-driving cars
  - Healthcare
  - Financial institutions
  - Insurance companies
  - ...
- **It's important to**
  - Realize there are vulnerabilities
  - Have means to protect ourselves

Some items to think about ….

- **Business**
  - What are your goals?
  - What are the criteria for success?
- **Data**
  - Do you need labeled ($$) data?
  - What is the quality of your data?
  - What features are pertinent?
  - Do you have enough data?
  - How are you going to obtain the data?
- **Models**
  - What algorithms to use?
  - What metrics to evaluate the algorithms?
  - Would ensembles help?
- **Implementation**
  - How quickly does a new instance need to be classified (online/batch)?
  - Do you need to scale?
  - What resources do you have? Memory?, GPUs?, Compute?
  - How are you going to get feedback?