TU 257 – Fundamentals of Data Science

Data Analytics

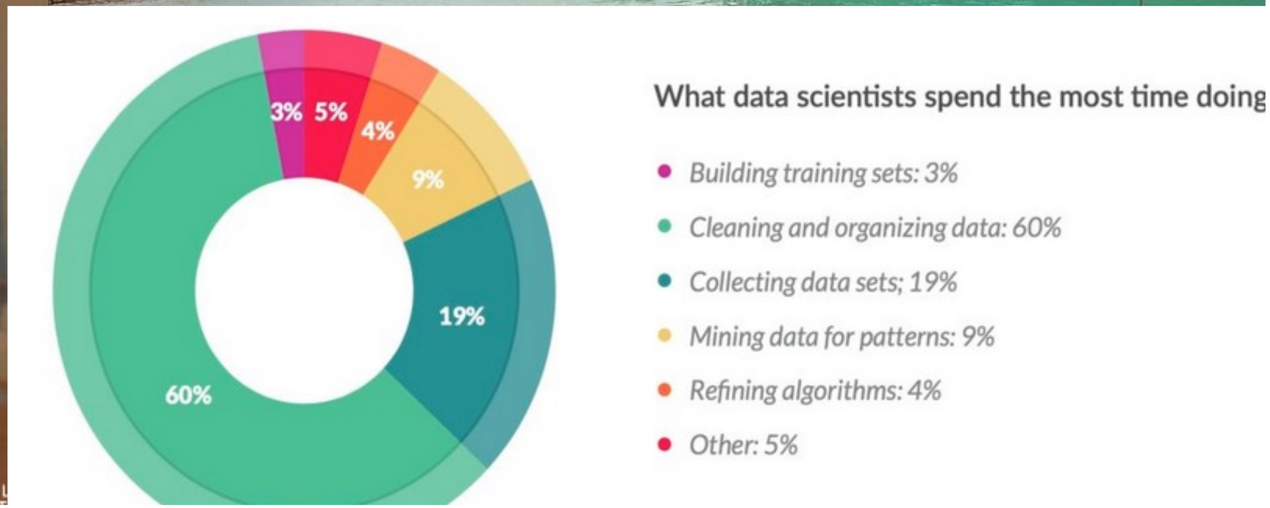
L4 – Classification – Part 1

Brendan Tierney


Who wants do do data analysis
Who wants to clean data?

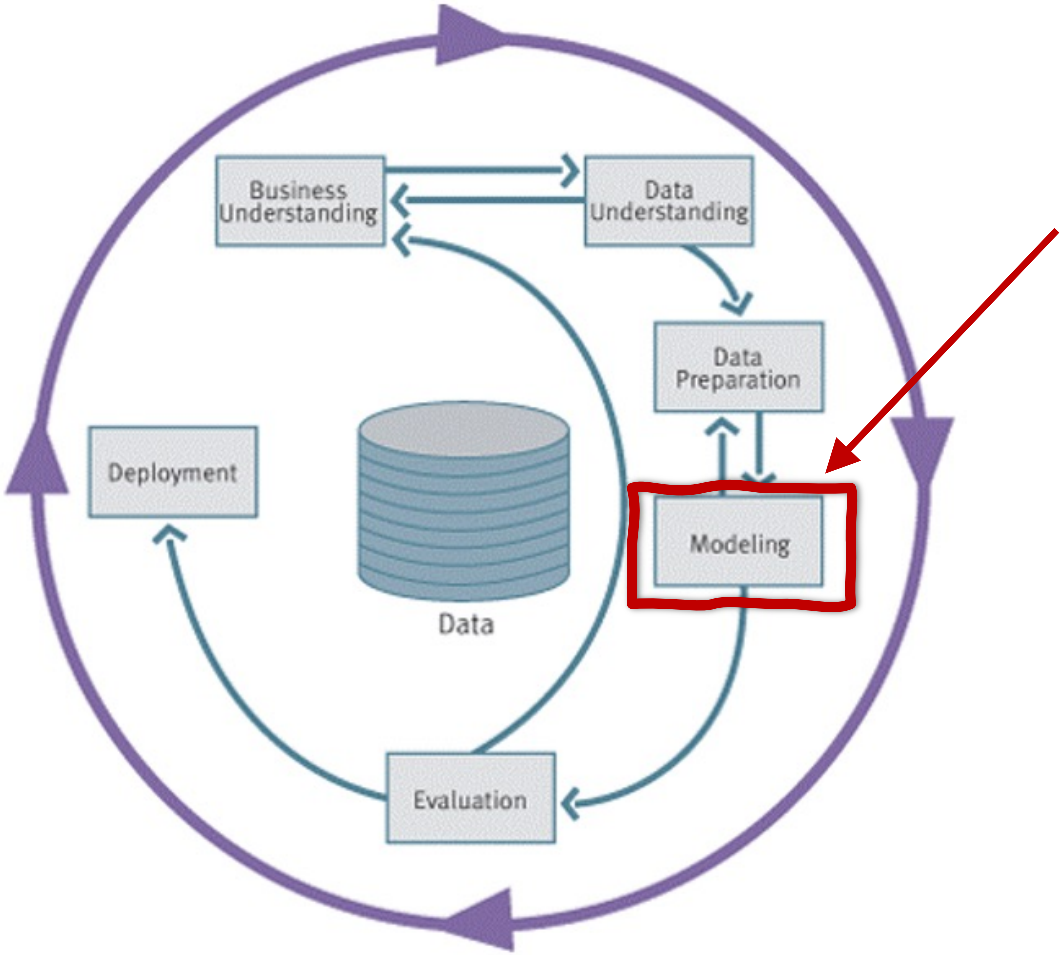

EVER
Data
Data Analysts


BAD DATA
BAD DATA EVERYWHERE


3%
5%
4%
9%
19%
60%
What data scientists spend the most time doing
Building training sets: 3%
Cleaning and organizing data: 60%
Collecting data sets; 19%
Mining data for patterns: 9%
Refining algorithms: 4%
Other: 5%

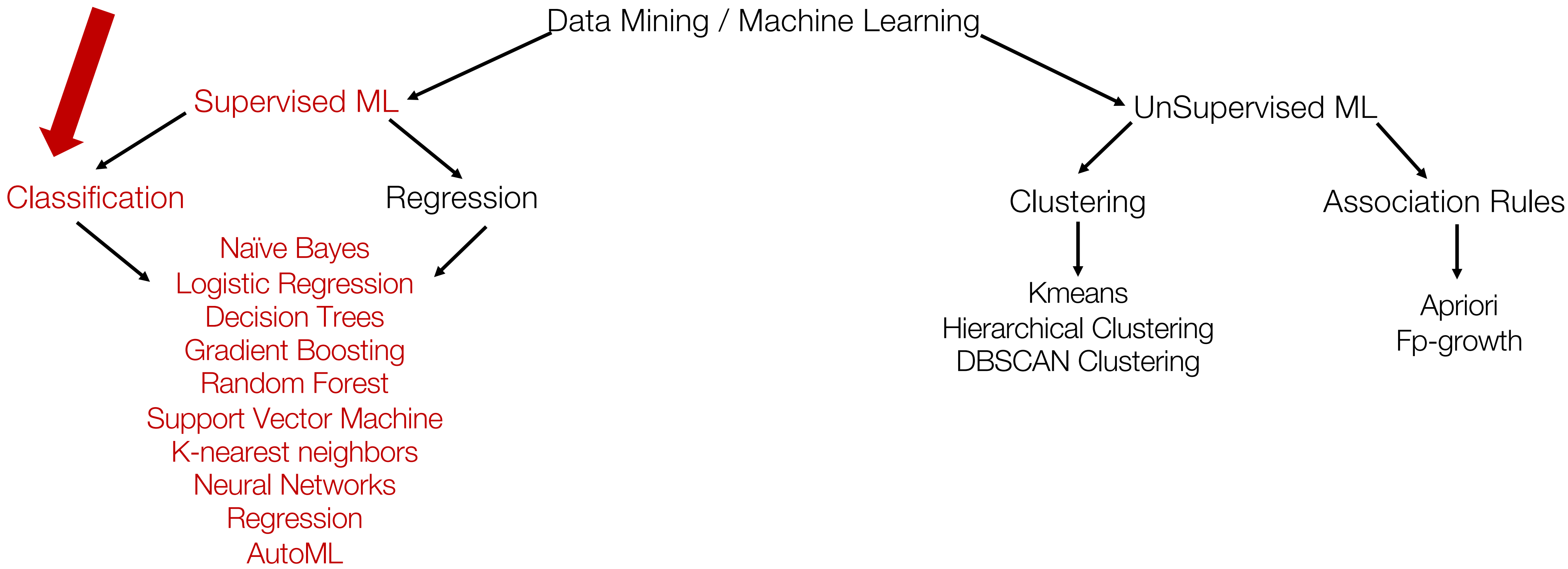
Agenda

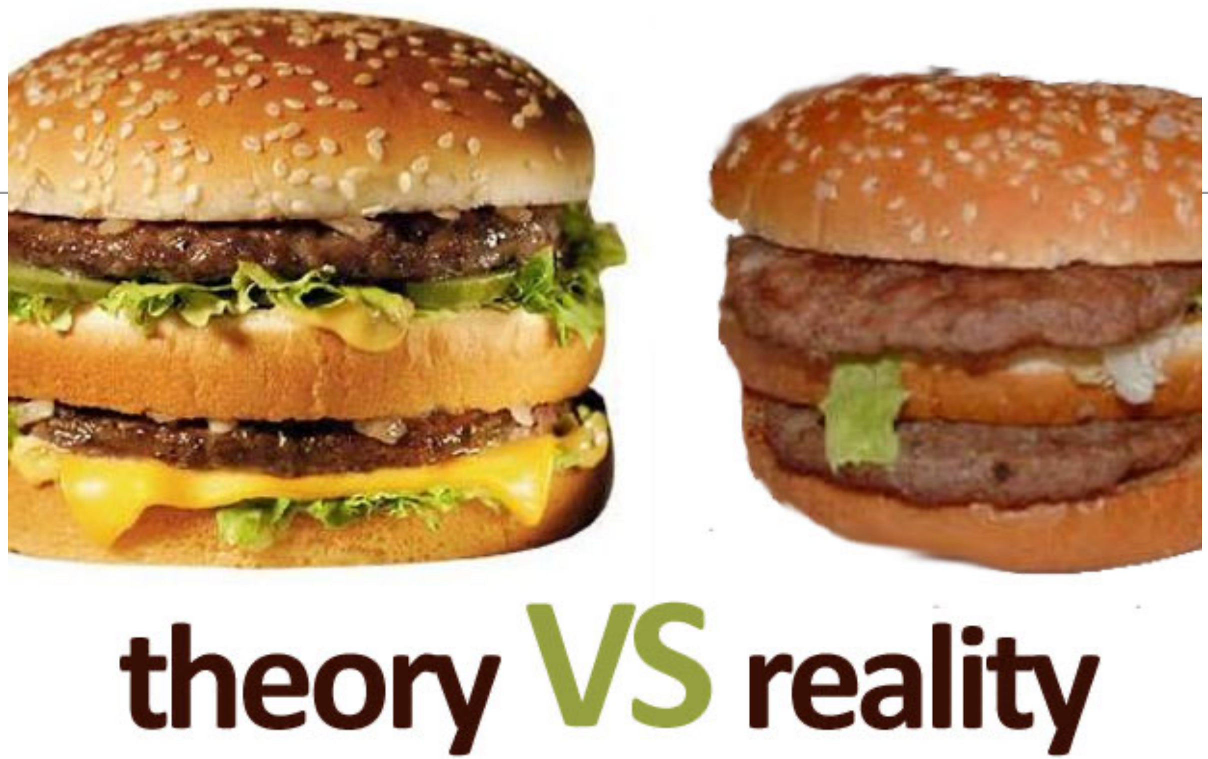
- What is Classification
- What type of problems
- The Typical Process
- Preparing Data
- Lots of Algorithms
  - Subset this week
  - More next week
  - Not all will be covered
- Some details/background/under-the-hood at the algorithms
  - Inner details are not needed. Can be explored in a Machine Learning module
- How do you measure if it’s any good


Business Understanding
Data Understanding
Data Preparation
Deployment
Data
Modeling
Evaluation

Data Mining / Machine Learning

- the use and development of computer systems that are able to learn and adapt without following explicit instructions, by using algorithms and statistical models to analyse and draw inferences from patterns in data.

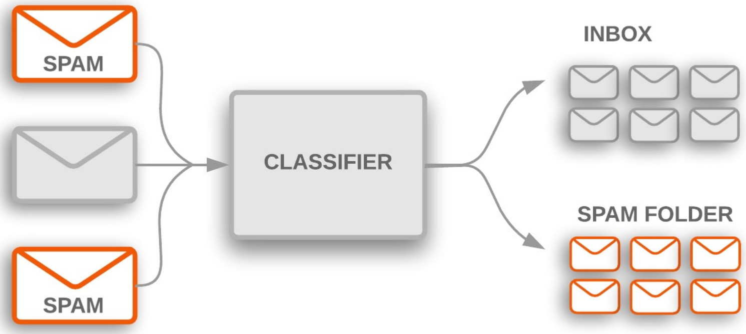




- Most Data Analytics etc can be done in a few lines of code
- Don't worry about the Theory, we might touch upon some of it, but it isn't necessary to know in-depth
- You'll never have to write an algorithm from scratch

What is Classification

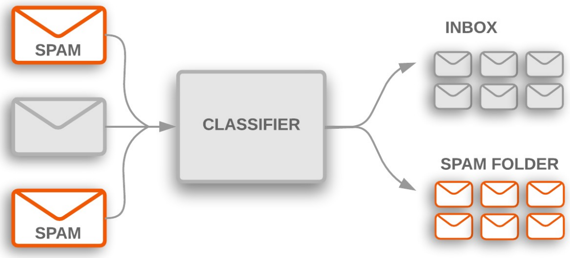
- Classification is a task that requires the use of (machine learning) algorithms that learn how to assign a class label to examples from the problem domain.
  - We learn from the past to predict the future
  - An easy to understand example is classifying emails as "*spam*" or "*not spam*."



- Classification predictive modeling involves:
  - Looking at historical data representing a particular scenario
  - Using algorithms to find patterns in the data
    - What attributes/features contribute towards determining the scenario being investigated
    - Assign a class label.

Classification – Different types

- Binary classification refers to predicting one of two classes
  - Yes / No
  - 0 / 1
  - Buy / Not-Buy
  - Spam / Not-Spam



- Multi-class Classification is when we have more than two class values
  - Different Fruits
  - Credit Ratings
  - Different Products

Classification: Definition

- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class*.

- Find a *model* for the class attribute as a function of the values of other attributes.

- Goal: previously unseen records should be assigned a class as accurately as possible.
  - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

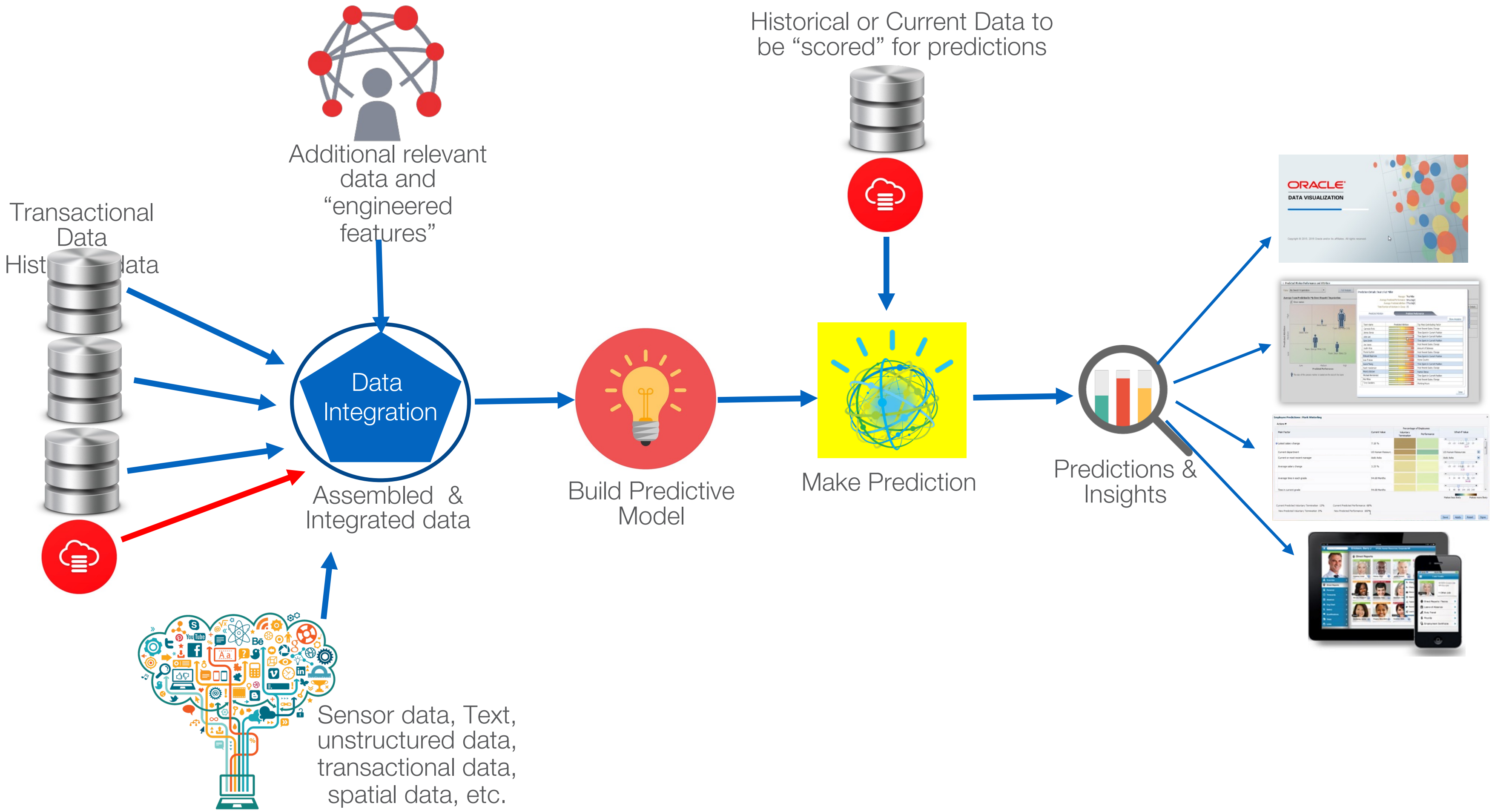
Classification—A Two-Step Process

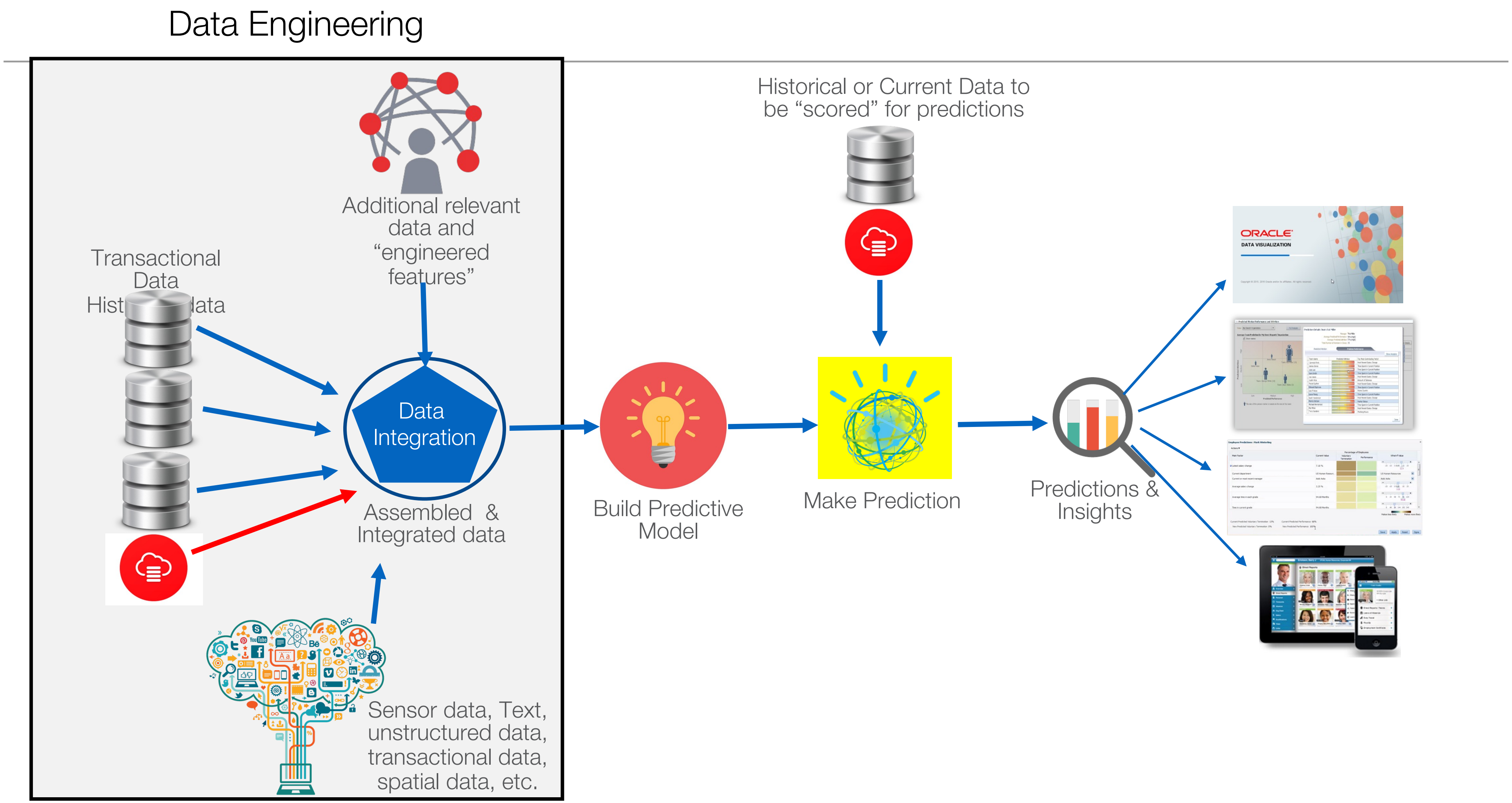
- Step 1 - Model Construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - Create a sub-set of tuples to be used for model construction: training dataset
  - Create a models using different algorithms
    - Each algorithms takes the training dataset as input

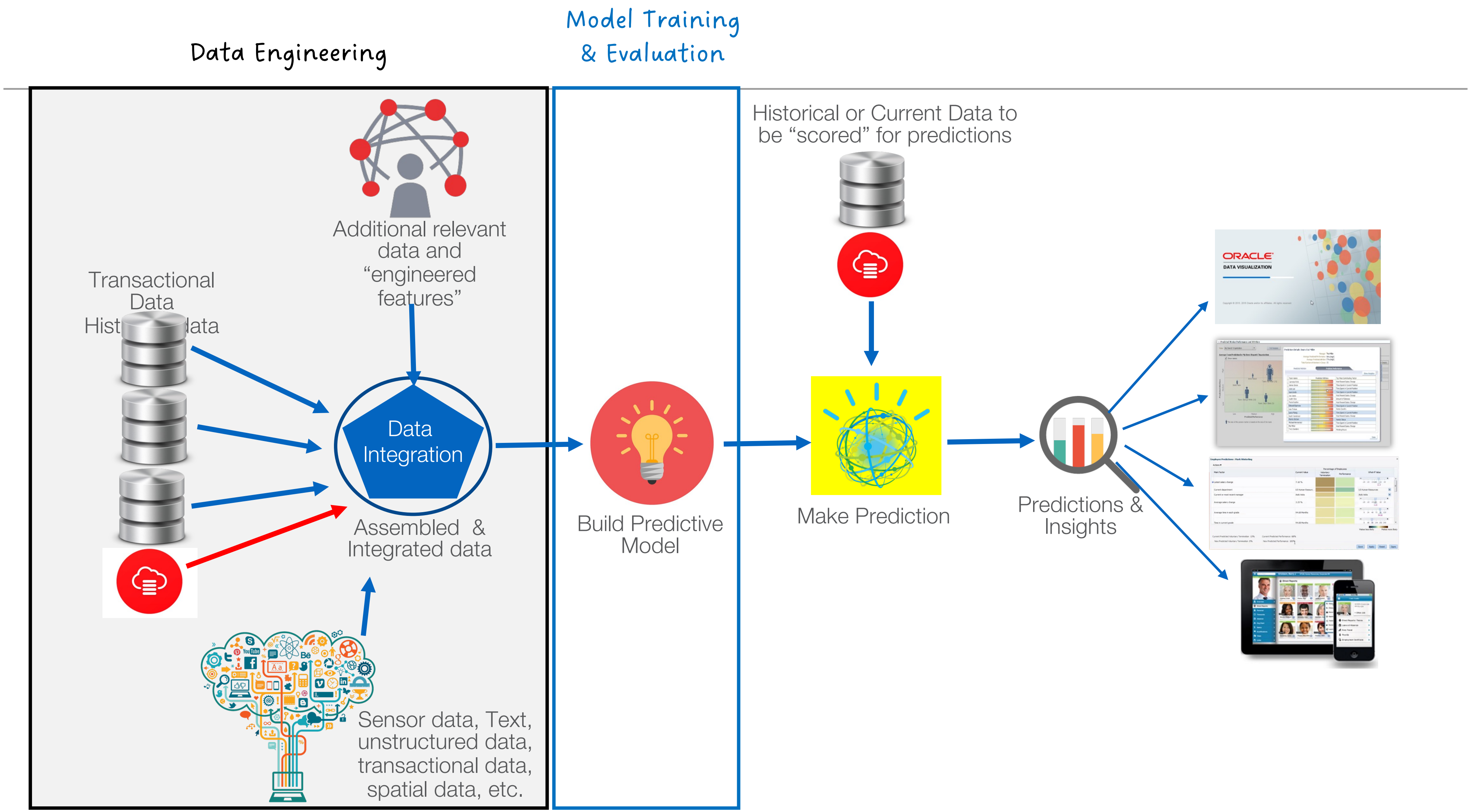
- Step 2 – Model Test & Evaluation
  - The model is represented as classification rules, decision trees, or mathematical formulae
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur

Classification—A Two-Step Process or is it a Three Steps

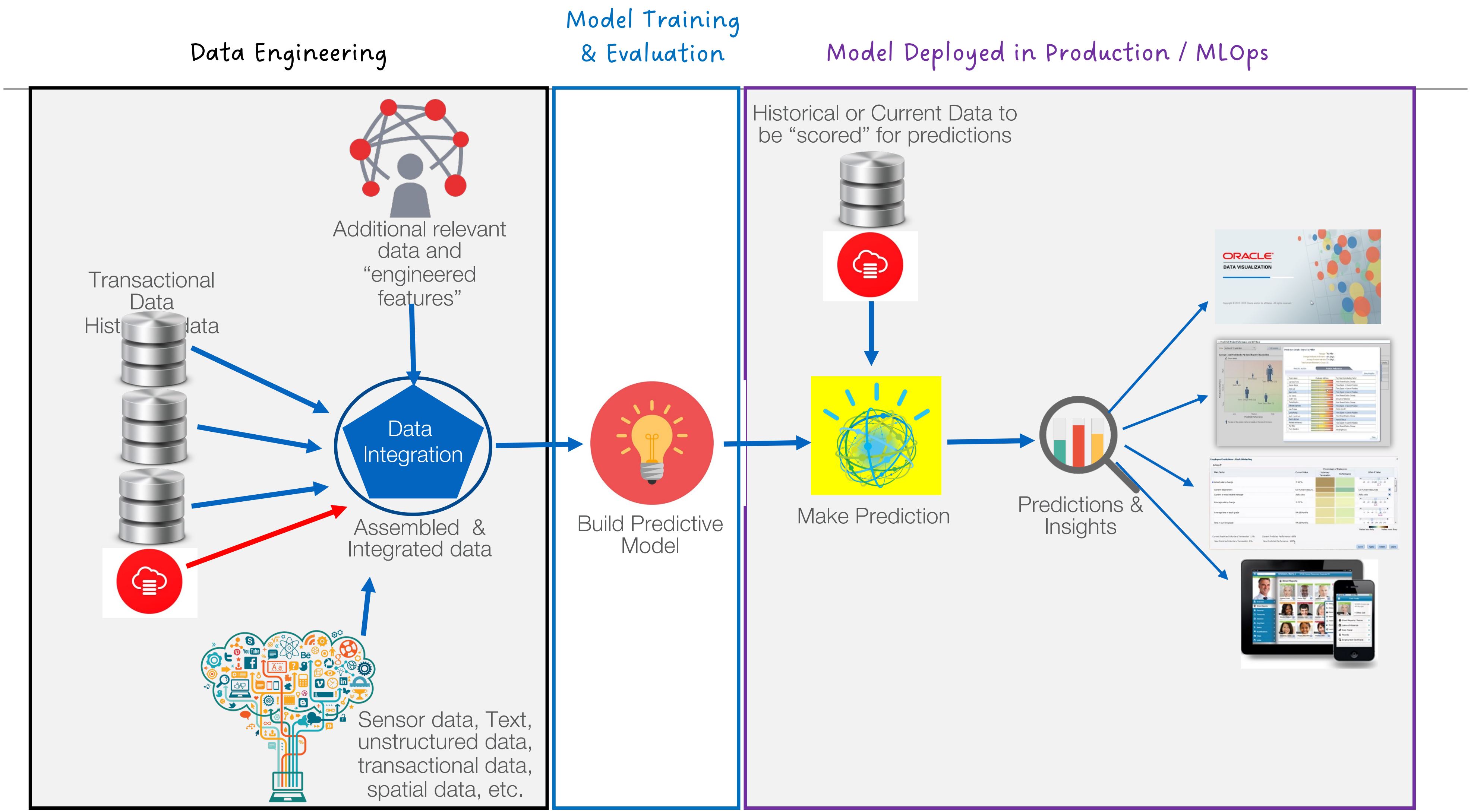
- Step 1 - Model Construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - Create a sub-set of tuples to be used for model construction: training dataset
  - Create a models using different algorithms
    - Each algorithms takes the training dataset as input
- Step 2 – Model Test & Evaluation
  - The model is represented as classification rules, decision trees, or mathematical formulae
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur
- Step 3 – Using the Model on new data
  - Put into production
  - Use on newly generated data
  - Need to constantly Review and assess if the model needs to be update
  - Iterative


Additional relevant data and “engineered features”
Historical or Current Data to be “scored” for predictions
Transactional Data
Data Integration
Assembled & Integrated data
Build Predictive Model
Make Prediction
Predictions & Insights
ORACLE
DATA VISUALIZATION
Sensor data, Text, unstructured data, transactional data, spatial data, etc.


Data Engineering
Additional relevant data and "engineered features"
Transactional Data
Data Integration
Assembled & Integrated data
Sensor data, Text, unstructured data, transactional data, spatial data, etc.
Historical or Current Data to be "scored" for predictions
Build Predictive Model
Make Prediction
Predictions & Insights
ORACLE
DATA VISUALIZATION

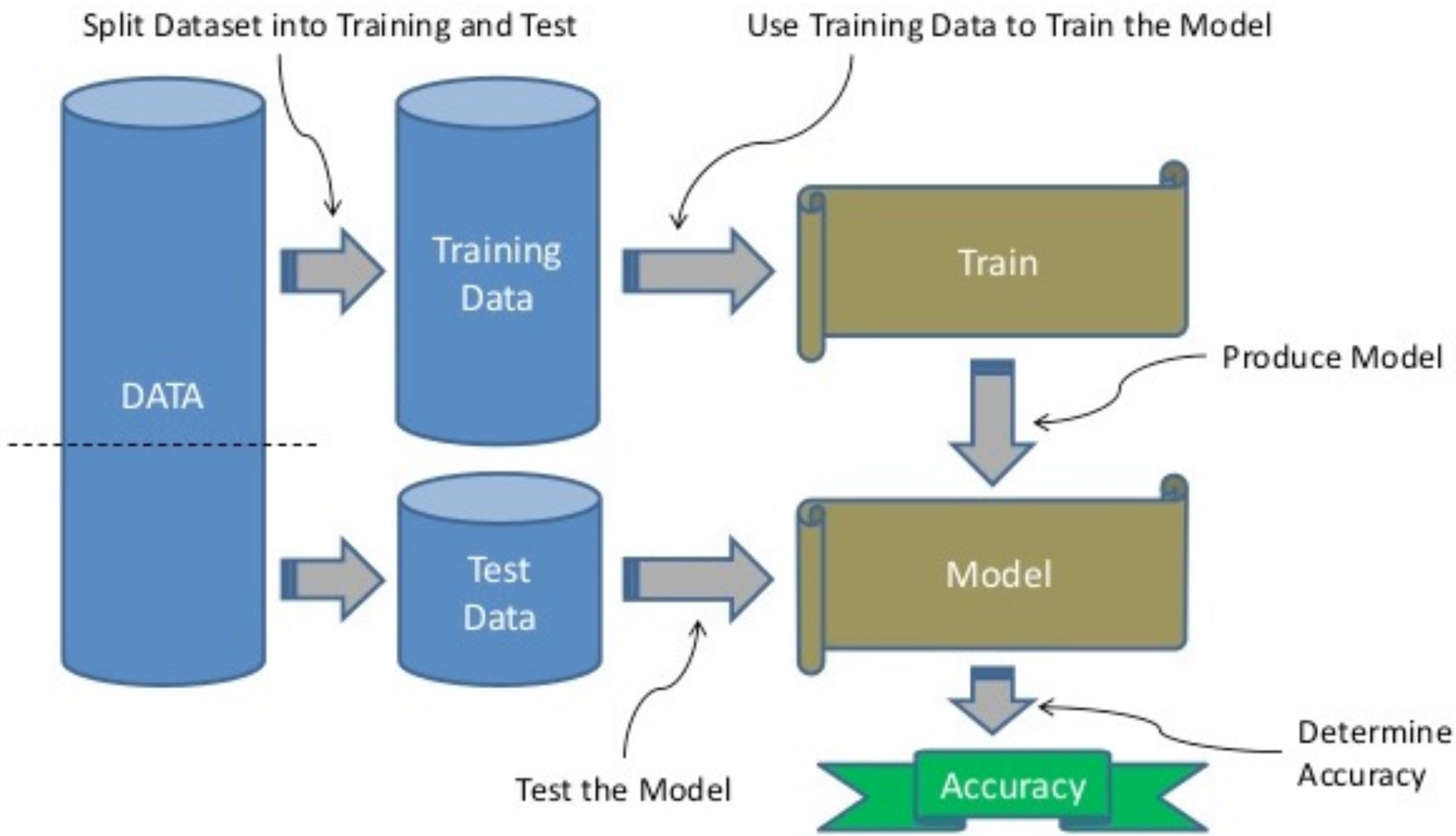


Data Engineering
Model Training & Evaluation
Historical or Current Data to be "scored" for predictions
Additional relevant data and "engineered features"
Transactional Data
Data Integration
Assembled & Integrated data
Build Predictive Model
Make Prediction
Predictions & Insights
ORACLE
DATA VISUALIZATION
Sensor data, Text, unstructured data, transactional data, spatial data, etc.

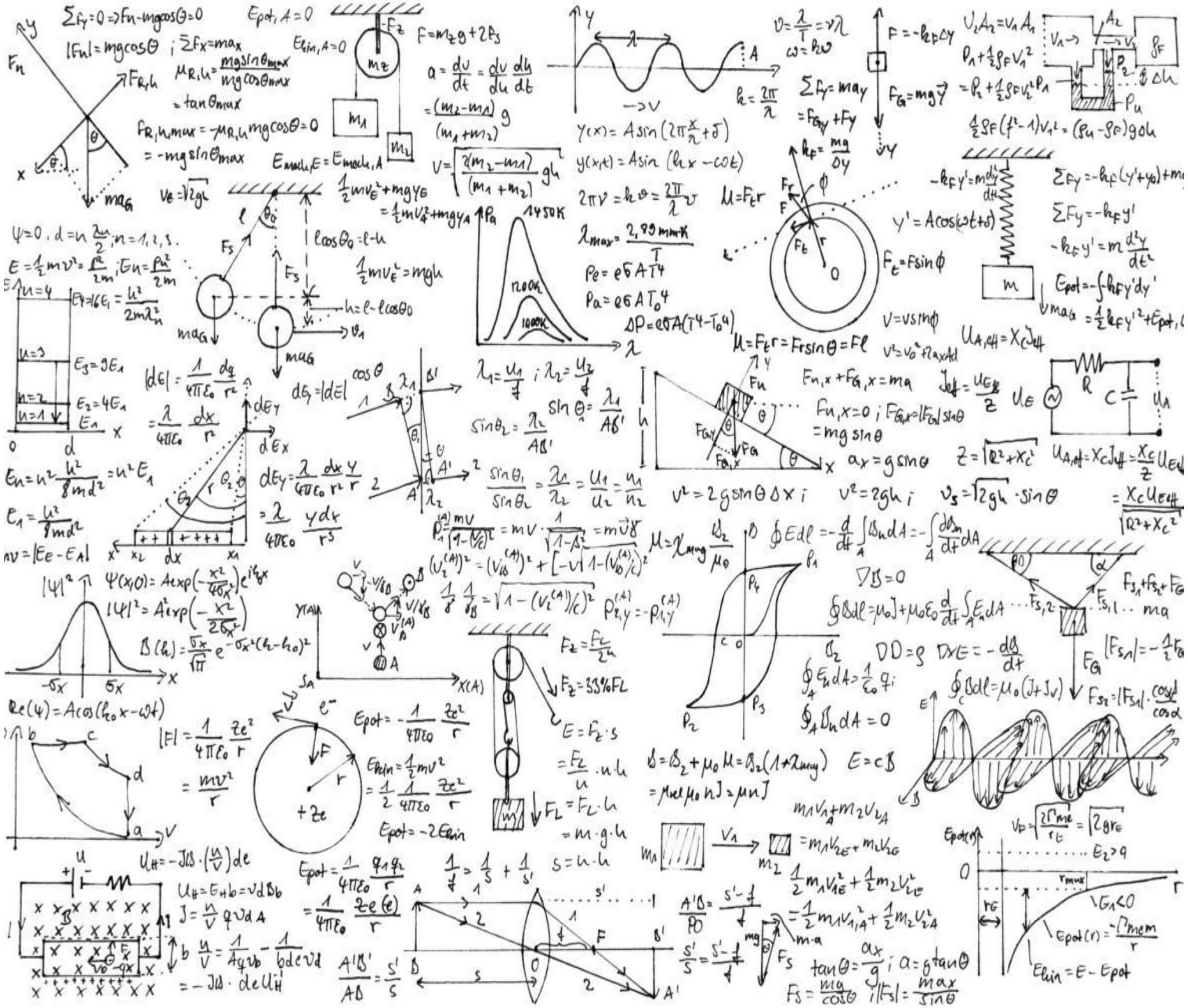


Data Engineering
Model Training
& Evaluation
Model Deployed in Production / MLOps
Additional relevant data and "engineered features"
Transactional Data
Hist data
Data Integration
Assembled & Integrated data
Sensor data, Text, unstructured data, transactional data, spatial data, etc.
Build Predictive Model
Historical or Current Data to be "scored" for predictions
Make Prediction
Predictions & Insights
ORACLE
DATA VISUALIZATION

Data Set = Training + Test sata sets



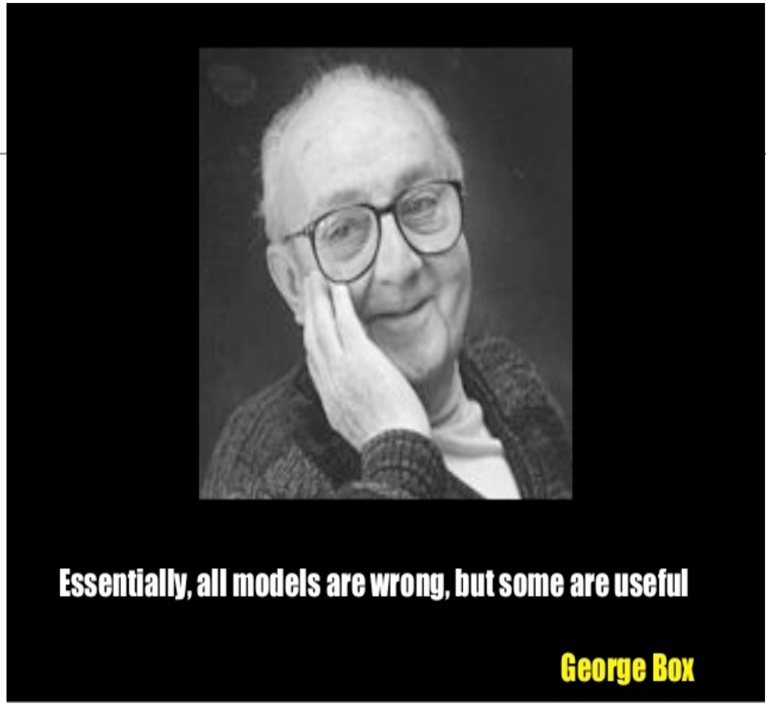
The Algorithms



The Algorithms

- This week
  - Naive Bayes
  - Decision Trees
  - Random Forests
  - XGBoost

- Other algorithms next week

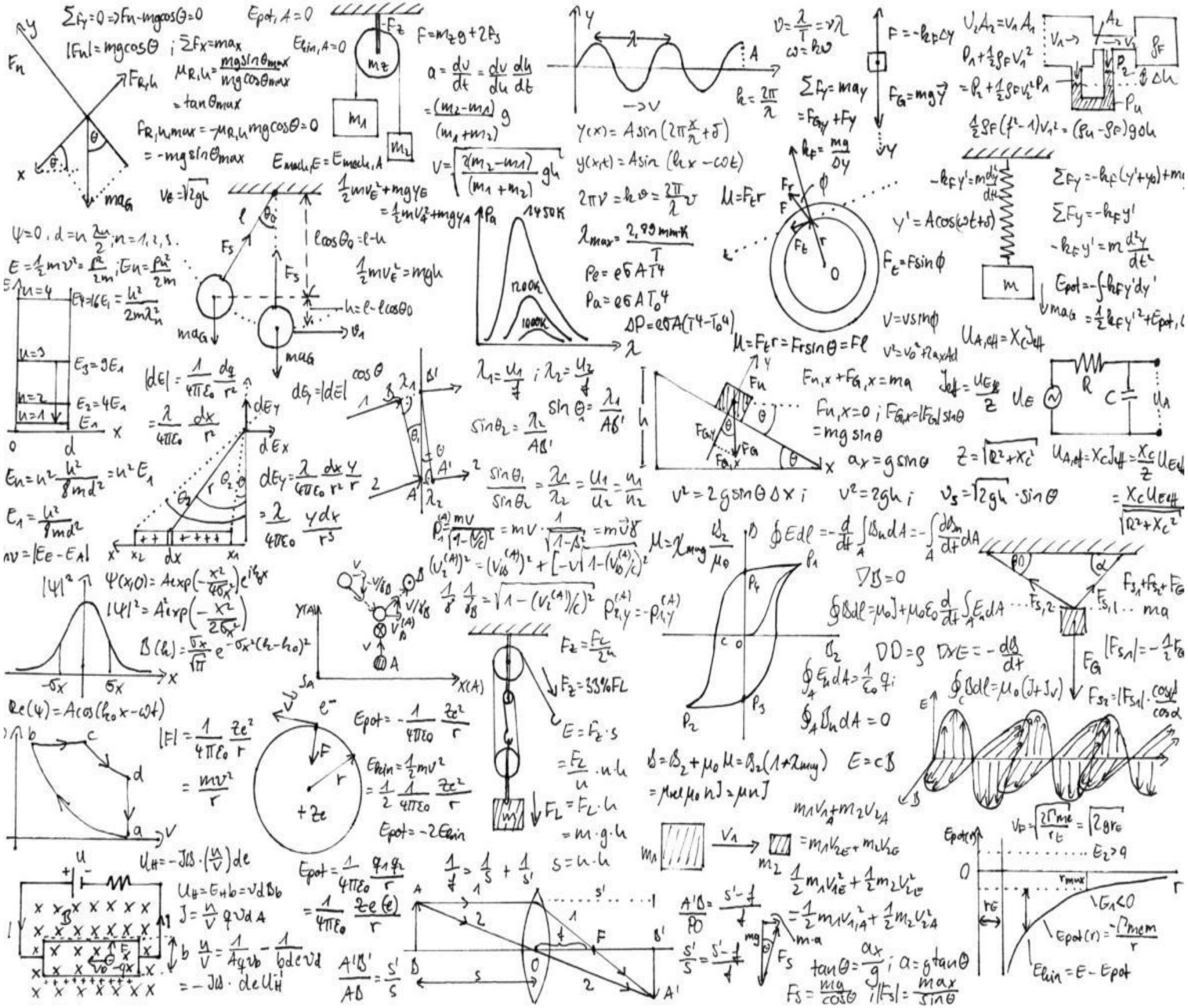


A model is a simplification or approximation of reality
and hence will not reflect all of reality.

His paper was published in the Journal of the American Statistical Association, 1976
Book *Empirical Model-Building and Response Surfaces*, 1987

The Algorithms

- Naive Bayes
- Decision Trees
- Random Forests
- XGBoost



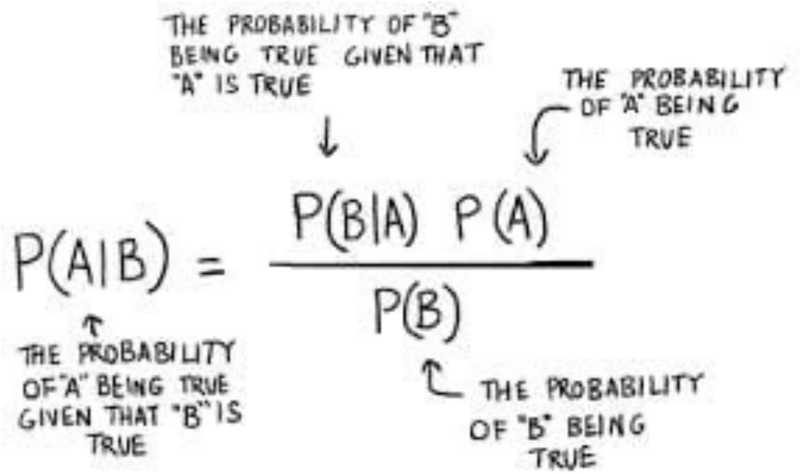
Naïve Baye

- Naive Bayes is a probabilistic classifier in Machine Learning which is built on the principle of Bayes theorem.

- Naive Bayes classifier assumes that one particular feature in a class is unrelated to any other feature and that is why it is known as naïve
  - It is based on probability models that incorporate strong independence assumptions.
  - The independence assumptions often have little impact on reality. Therefore they are considered as naive.

- Sounds Complicated!

- 1786!



Naïve Baye

- Naive Bayes classifier calculates the probability of an event in the following steps:
  - Step 1: Calculate the prior probability for given class labels
  - Step 2: Find Likelihood probability with each attribute for each class
  - Step 3: Put these values in Bayes Formula and calculate posterior probability.
  - Step 4: See which class has a higher probability, given the input belongs to the higher probability class.

- Let’s look at an example

Play-tennis example: estimating $P(x_i|C)$

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

P(p) = 9/14

P(n) = 5/14

Play-tennis example: estimating $P(x_i|C)$

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

P(p) = 9/14
P(n) = 5/14

outlook	
P(sunny|p) = 2/9	P(sunny|n) = 3/5
P(overcast|p) = 4/9	P(overcast|n) = 0
P(rain|p) = 3/9	P(rain|n) = 2/5
temperature	
P(hot|p) = 2/9	P(hot|n) = 2/5
P(mild|p) = 4/9	P(mild|n) = 2/5
P(cool|p) = 3/9	P(cool|n) = 1/5
humidity	
P(high|p) = 3/9	P(high|n) = 4/5
P(normal|p) = 6/9	P(normal|n) = 2/5
windy	
P(true|p) = 3/9	P(true|n) = 3/5
P(false|p) = 6/9	P(false|n) = 2/5

Play-tennis example: classifying X

outlook	
P(sunny|p) = 2/9	P(sunny|n) = 3/5
P(overcast|p) = 4/9	P(overcast|n) = 0
P(rain|p) = 3/9	P(rain|n) = 2/5
temperature	
P(hot|p) = 2/9	P(hot|n) = 2/5
P(mild|p) = 4/9	P(mild|n) = 2/5
P(cool|p) = 3/9	P(cool|n) = 1/5
humidity	
P(high|p) = 3/9	P(high|n) = 4/5
P(normal|p) = 6/9	P(normal|n) = 2/5
windy	
P(true|p) = 3/9	P(true|n) = 3/5
P(false|p) = 6/9	P(false|n) = 2/5

- An unseen sample X = <rain, hot, high, false>

- P(X|**p**)·P(**p**) =
  P(rain|**p**)·P(hot|**p**)·P(high|**p**)·P(false|p)·P(**p**) = 3/9·2/9·3/9·6/9·9/14 = 0.010582

- P(X|**n**)·P(**n**) =
  P(rain|**n**)·P(hot|**n**)·P(high|**n**)·P(false|**n**)·P(**n**) = 2/5·2/5·4/5·2/5·5/14 = 0.018286

- Sample X is classified in class n (don't play)

Naïve Baye

- There's a lot of Maths/Calculations

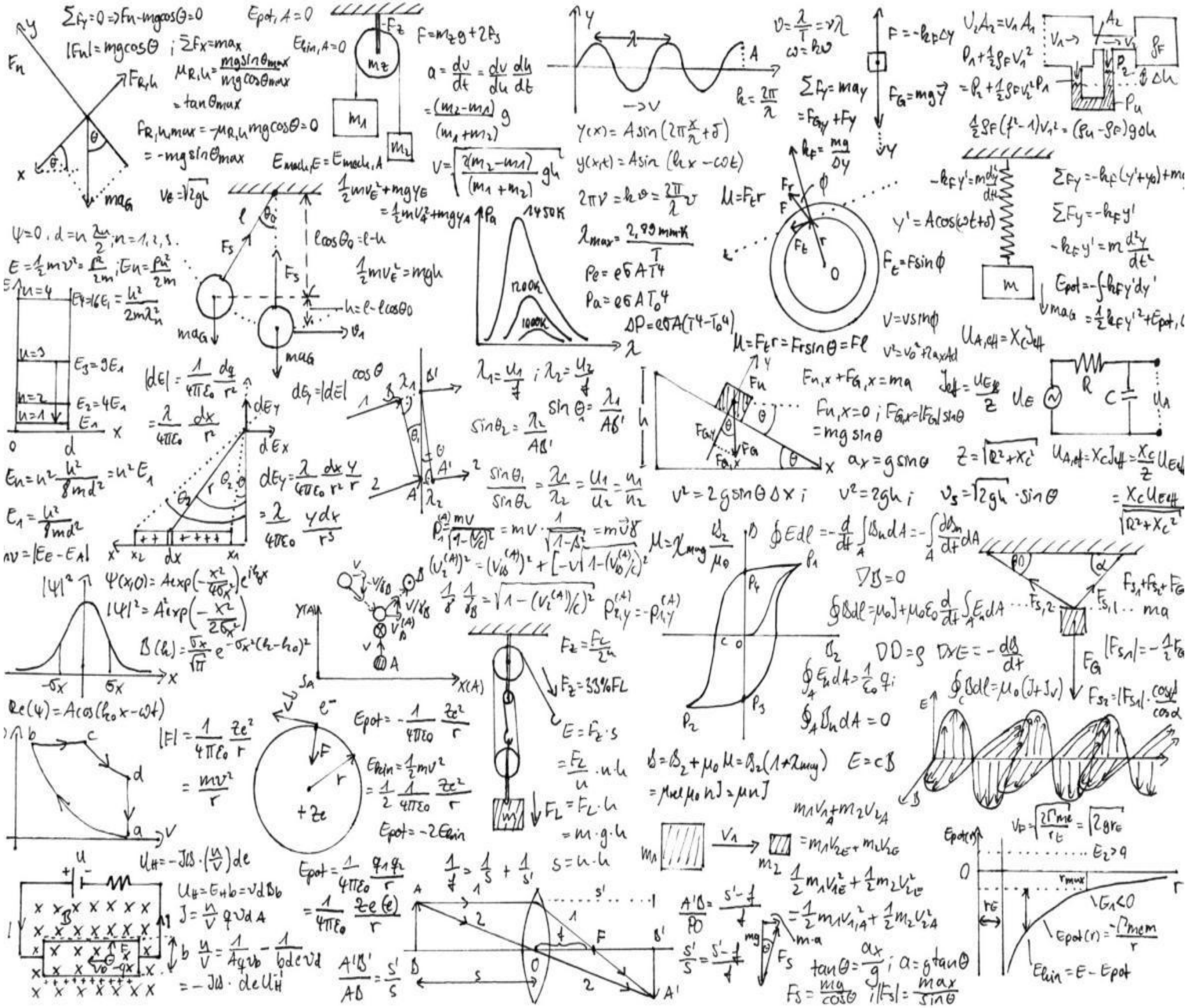
- They are all very simple calculations
  - Counting
  - Multiplication

- Computers are very good at doing simple Maths/Calculations -> Very fast
  - You will never have to do these calculations
  - It's a tool for you to use

- Quick results
  - Although may not be the most accurate
  - Can be a good starting point -> benchmark other algorithms performance

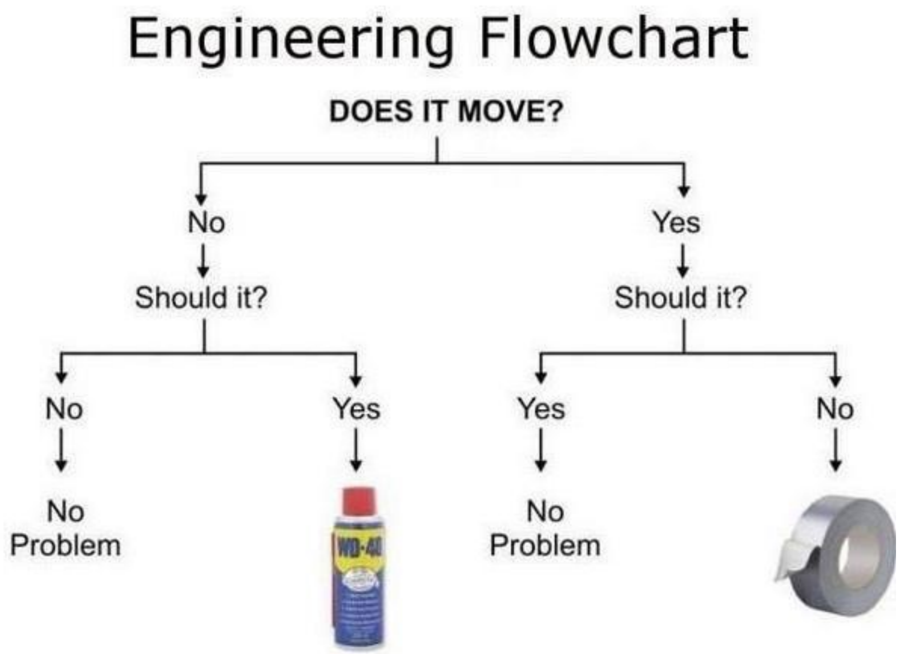
The Algorithms

- Naive Bayes
- Decision Trees
- Random Forests
- XGBoost



Decision Tree

- Does it work for just this one time or can be be used over time with different data?
  - Does it work in different situations ?
  - is a simplification or approximation of reality ? (George Box)

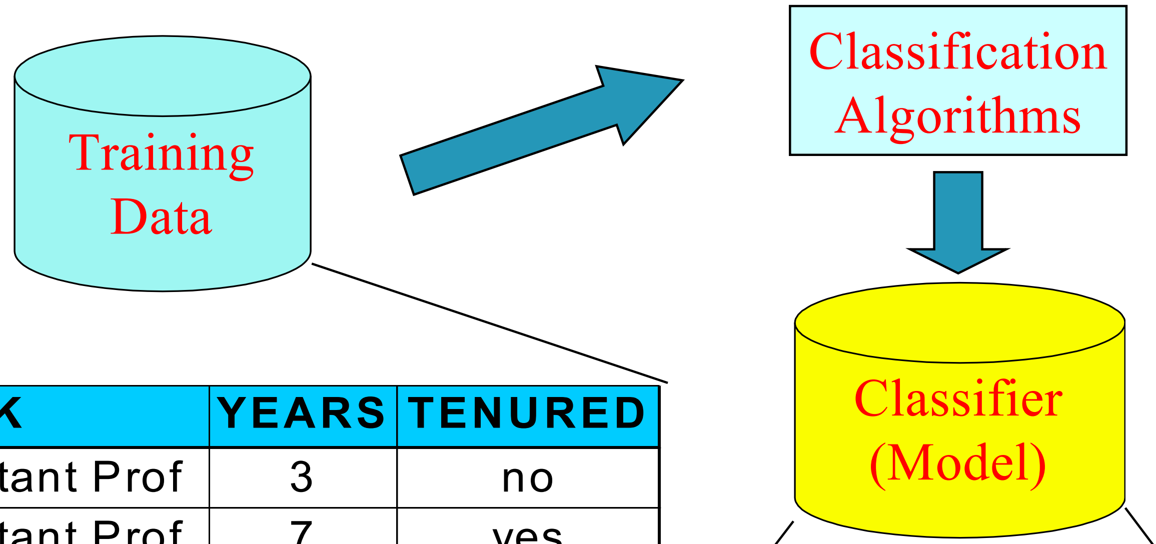


Decision Trees are :

- Simple
- Easy to understand
- Easy to explain
- Everyone can follow a DT

- IF-THEN statements
- Easy to code
- Easy to integrate into …

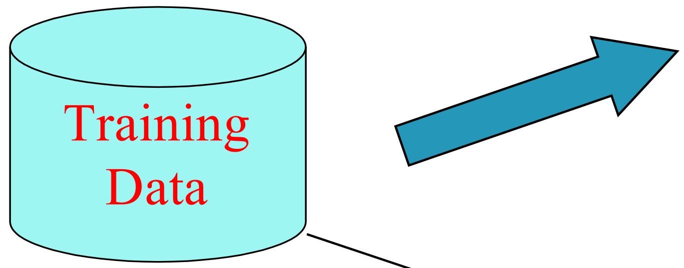
Classification Process (1): Model Construction



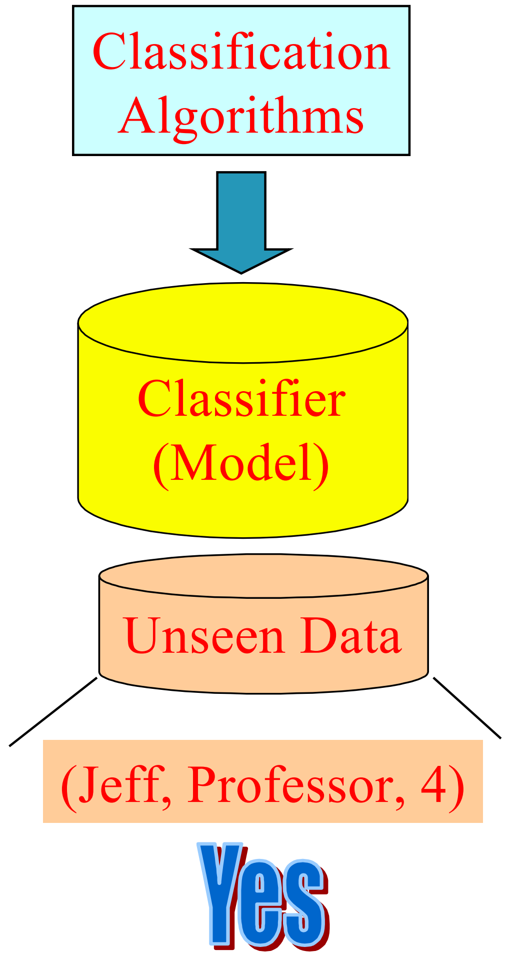
NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Classification Process (1): Model Apply



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

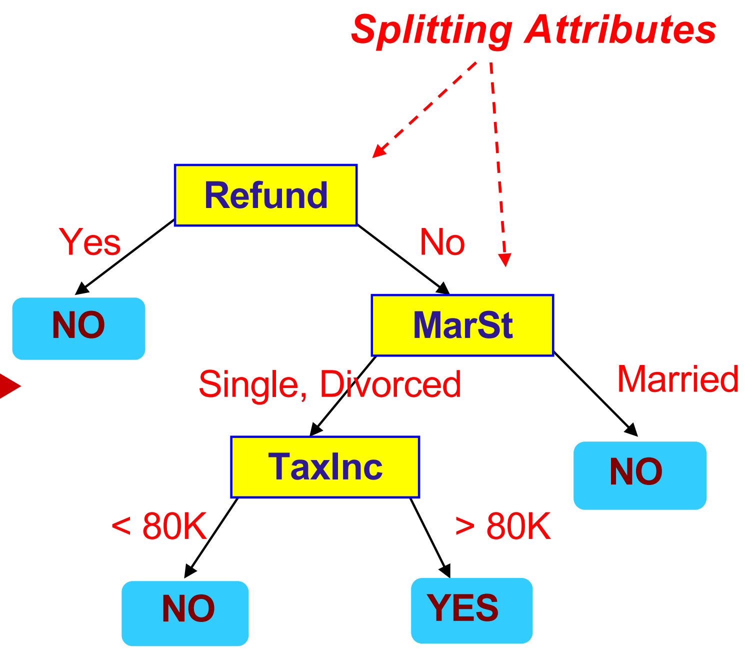


Example of a Decision Tree

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data



Model: Decision Tree

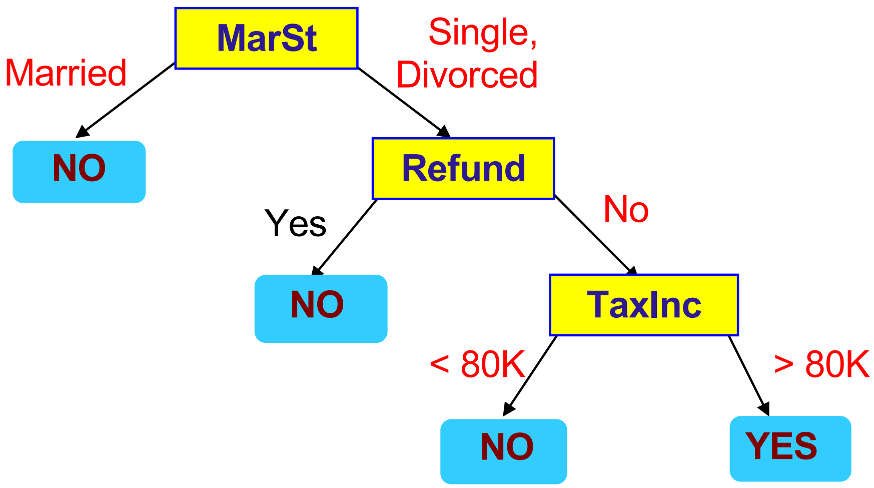
What attribute do we start with ?

This is called Information Gain

Another Example of Decision Tree

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**



There could be more than one tree that fits the same data!

Algorithm for Decision Tree Induction

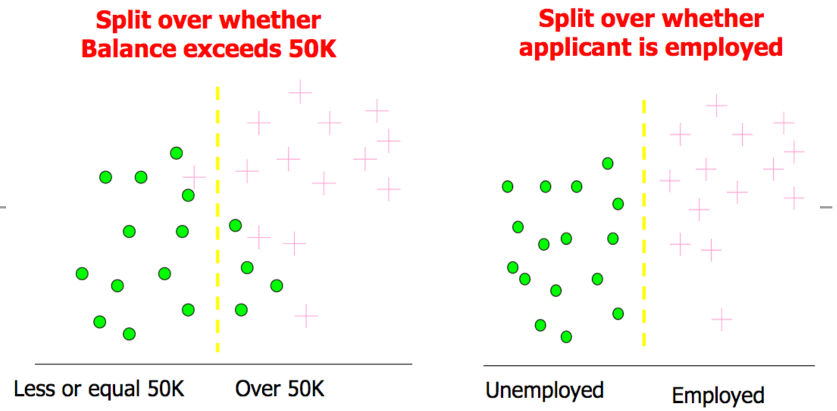
- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)

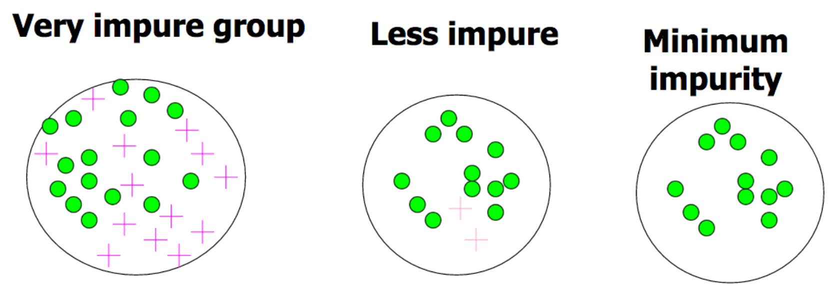
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

How to split nodes/attributes

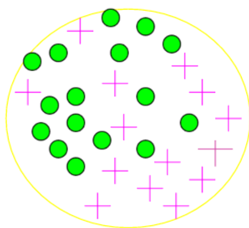
- Information Gain
  - Measures the level of impurity in a group of examples
  - We want to determine which attribute in a given set of training feature vectors is most useful for discriminating between the classes to be learned.
  - Information gain tells us how important a given attribute of the feature vectors is.

Which test is more informative?





- Entropy = $\sum_i - p_i \log_2 p_i$



p_i is the probability of class i
Compute it as the proportion of class i in the set.

16/30 are green circles; 14/30 are pink crosses
$\log_2(16/30)$ = -.9; $\log_2(14/30)$ = -1.1
Entropy = -(16/30)(-.9) –(14/30)(-1.1) = .99

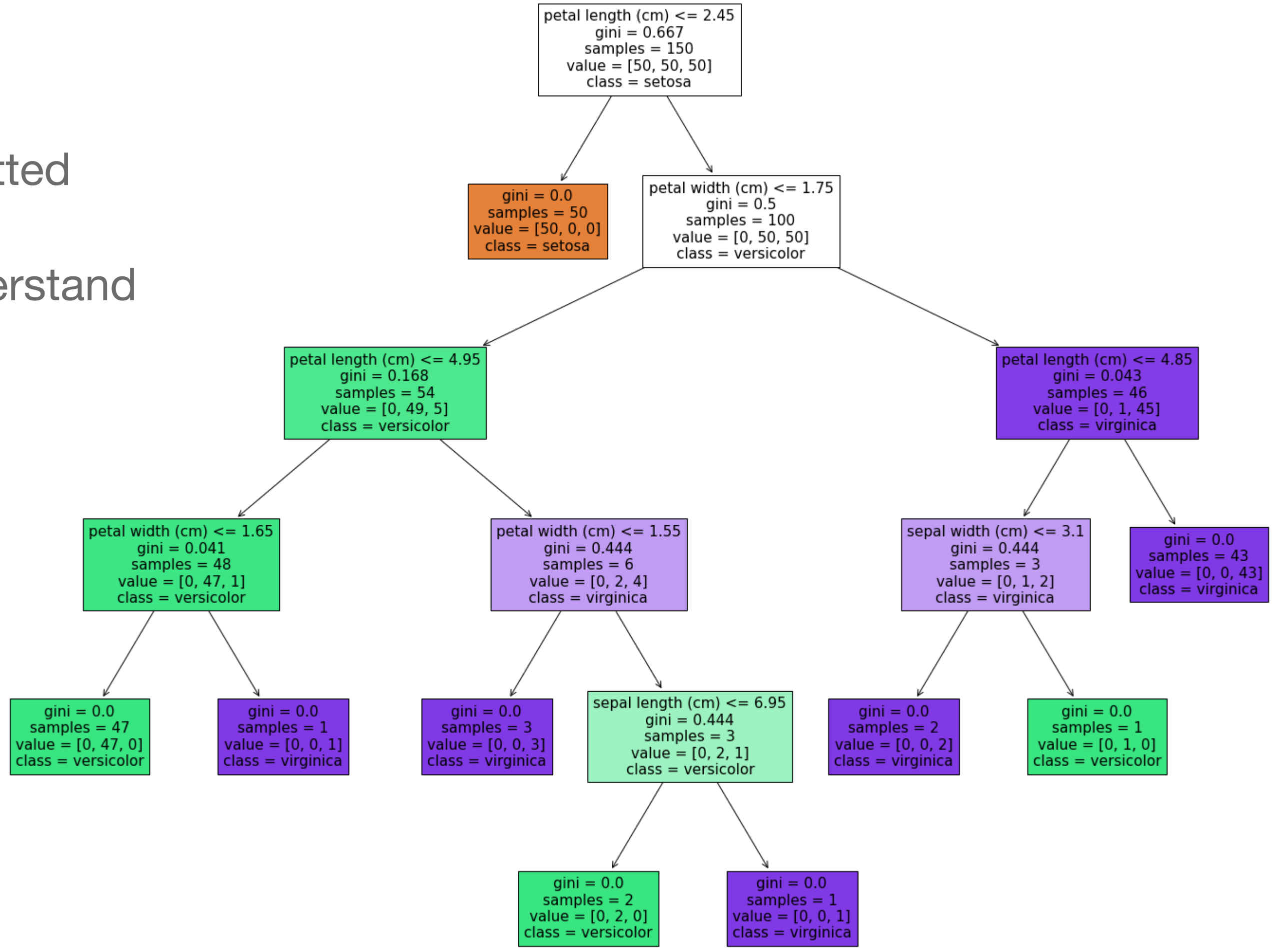
Extracting Classification Rules from Trees

- Represent the knowledge in the form of IF-THEN rules
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction
- The leaf node holds the class prediction
- Rules are easier for humans to understand
- Example

  IF *age* = “<=30” AND *student* = “*no*” THEN *buys_computer* = “*no*”
  IF *age* = “<=30” AND *student* = “*yes*” THEN *buys_computer* = “*yes*”
  IF *age* = “31…40” THEN *buys_computer* = “*yes*”
  IF *age* = “>40” AND *credit_rating* = “*excellent*” THEN *buys_computer* = “*yes*”
  IF *age* = “>40” AND *credit_rating* = “*fair*” THEN *buys_computer* = “*no*”

Decision Tree Plot

- Depends on size of Decision Tree
- Small Decision Trees are can be plotted
- Large Tree becomes difficult to understand

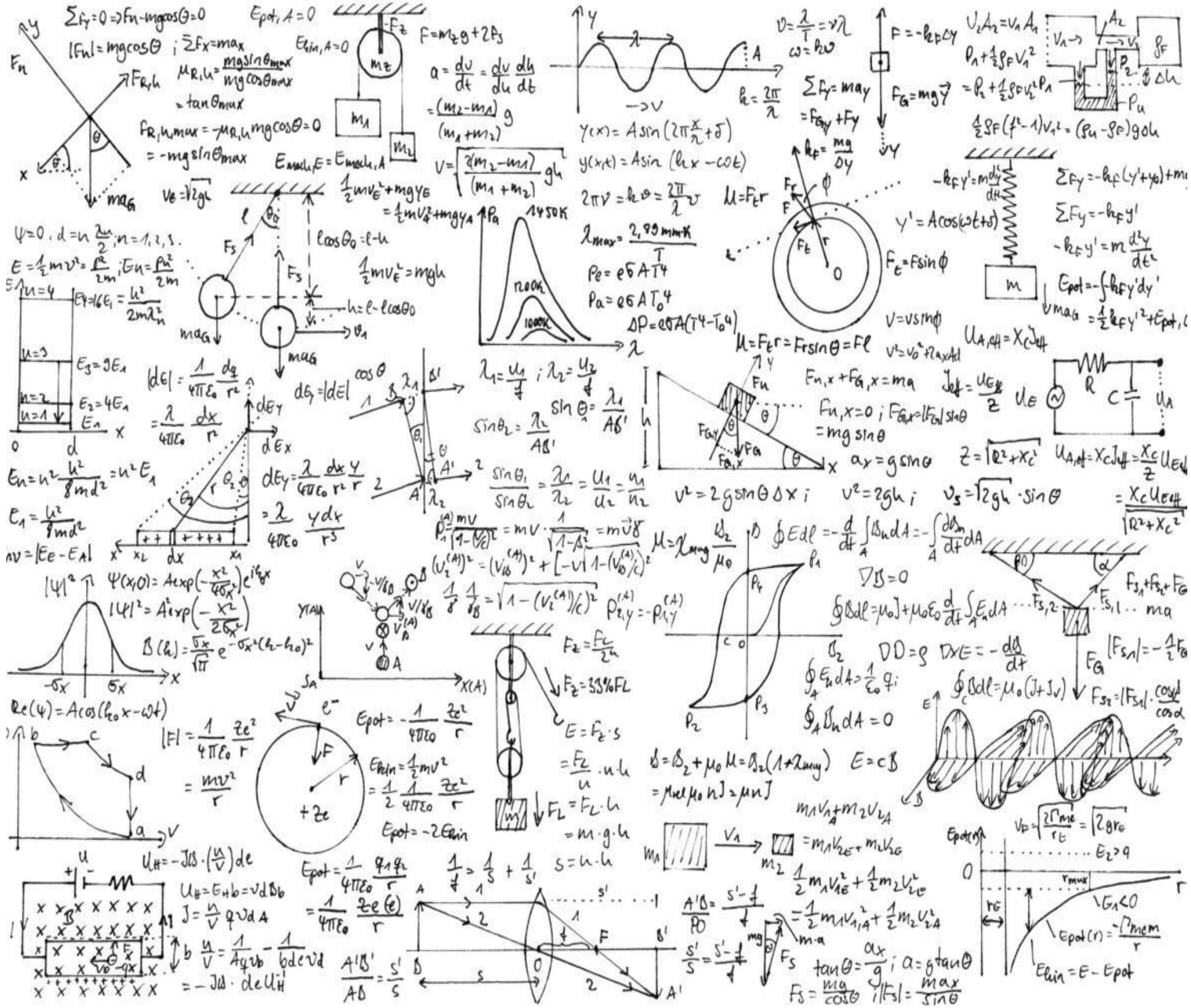


Decision Trees

- Form the basis for other algorithms
  - RandomForest
  - XGBoost
- Let’s have a look at these

The Algorithms

- Naive Bayes
- Decision Trees
- Random Forests
- XGBoost



Random Forests

- **Random forests**
  - are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.
  - Wisdom of Crowds

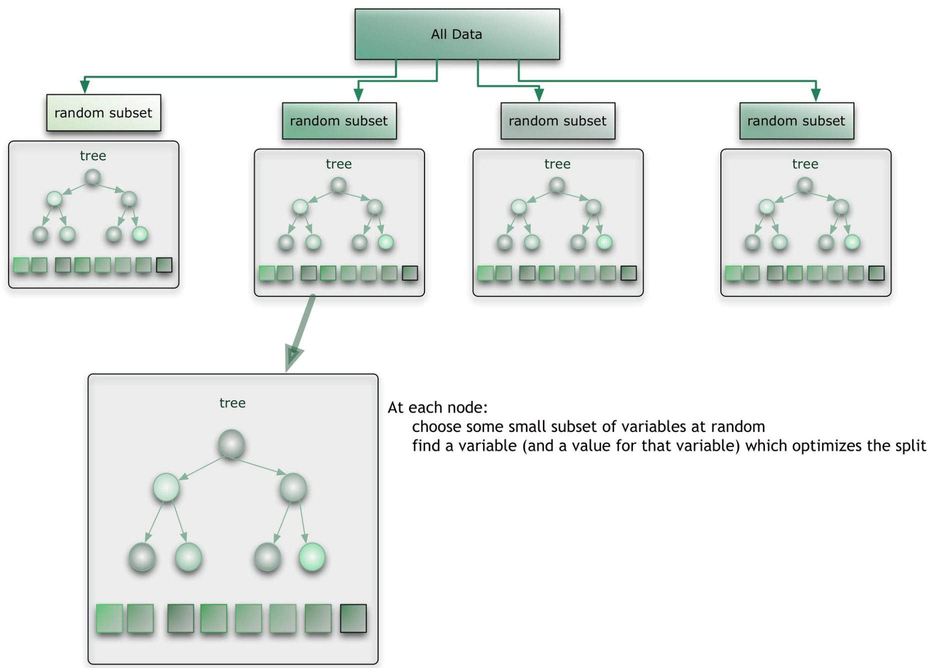
Random Forests

- The Random Forest algorithm has three main features:

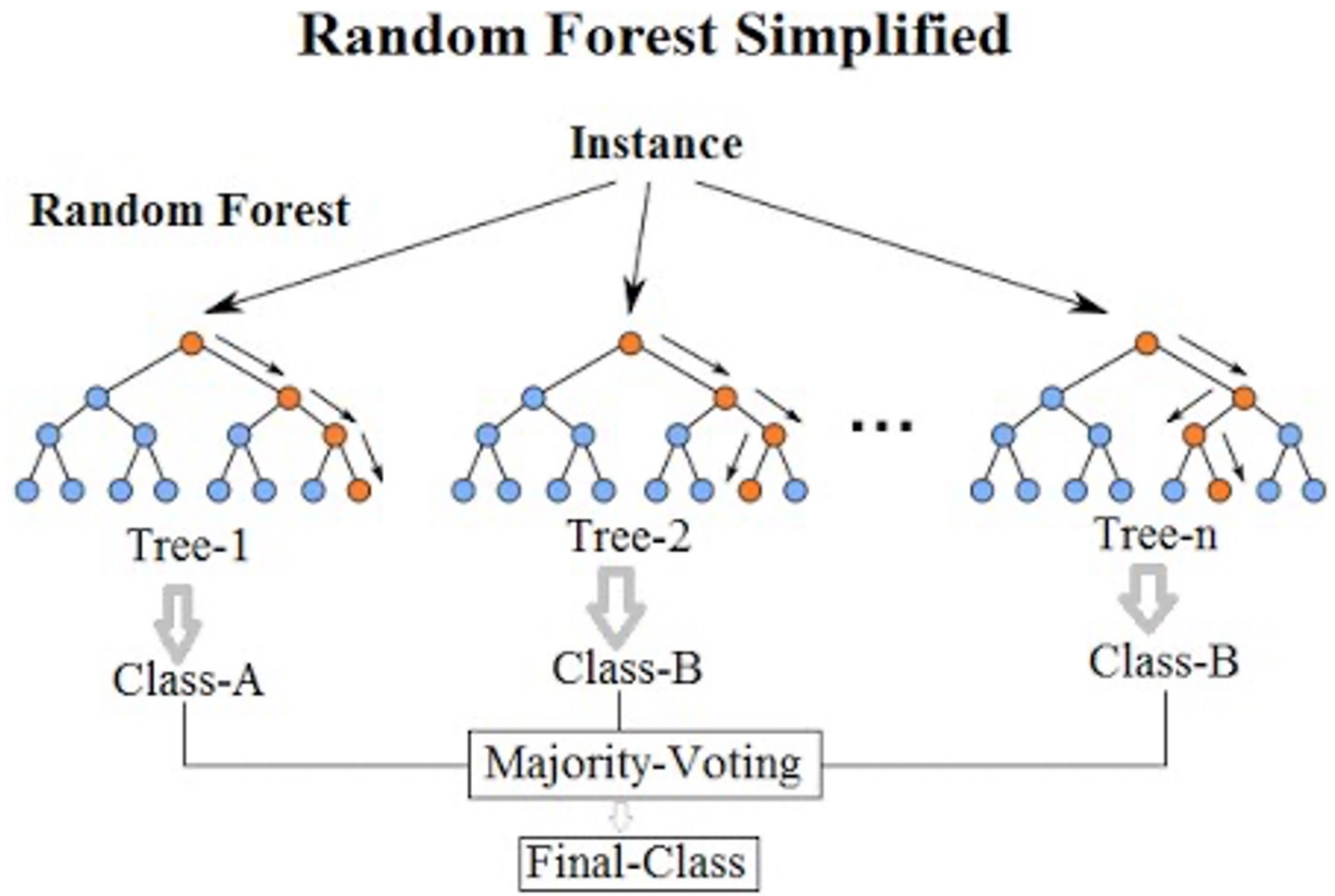
- It uses a method called bagging, to create different sub-sets of the original training data.

- It will randomly section different subsets of the features/attributes and build the decision tree based on this subset

- By creating many different decision trees, based on different subsets of the training data and different subsets of the features, will increase the probability of capturing all possible ways of modeling the data.

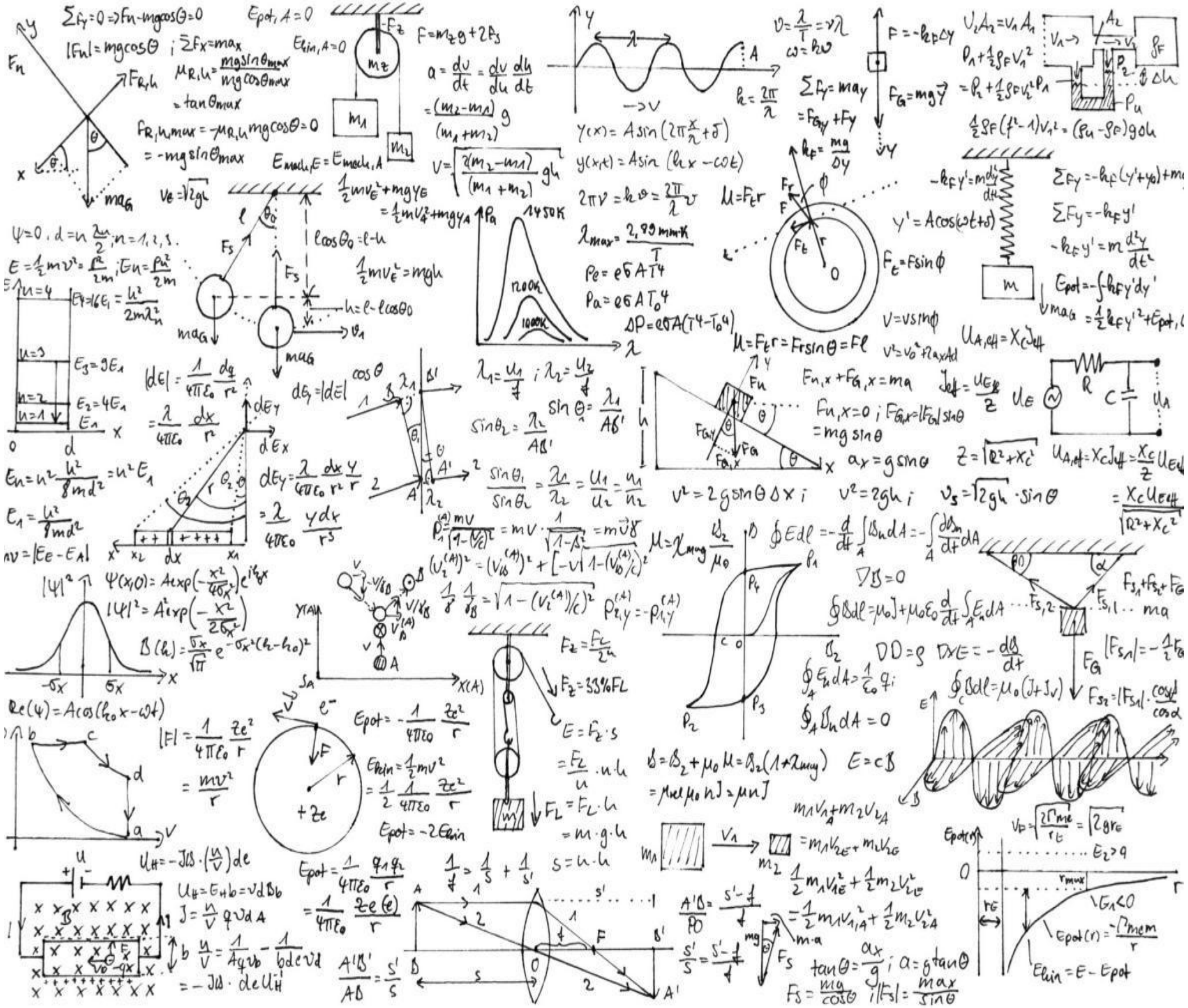


Random Forests



The Algorithms

- Naive Bayes
- Decision Trees
- Random Forests
- XGBoost



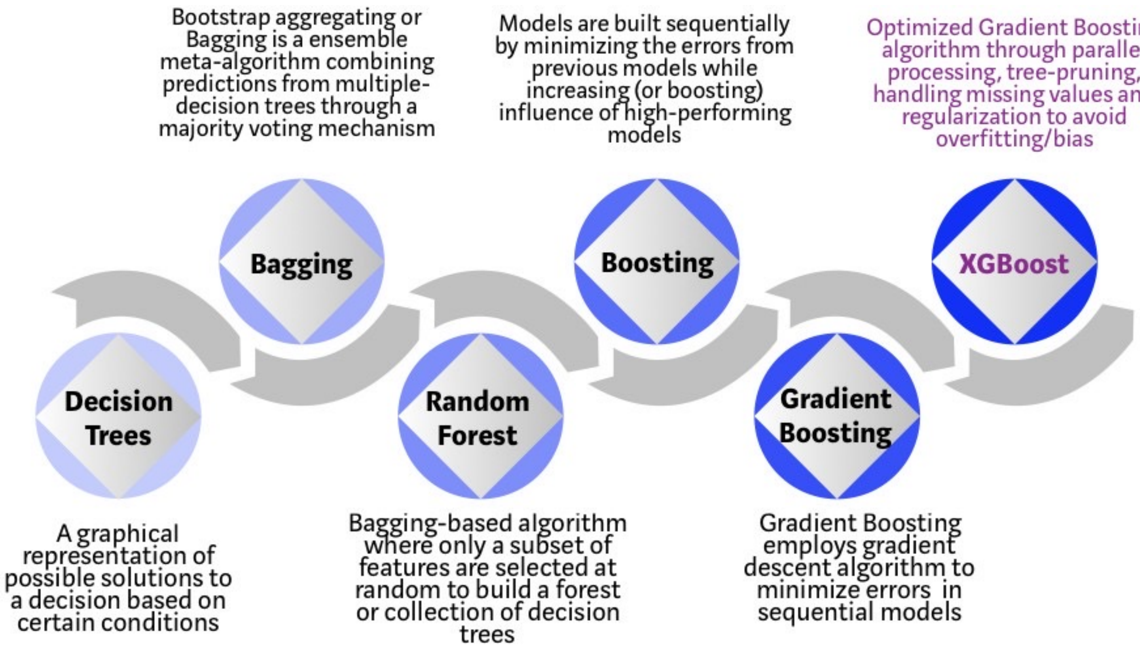
XGBoost

Tianqi Chen and Carlos Guestrin original paper from 2016 – University of Washington

XGBoost: A Scalable Tree Boosting System

Can be used for

- Classification
- Regression
- Ranking problems



- Open Source Framework
- Kaggle Competitions
- Builds upon previous

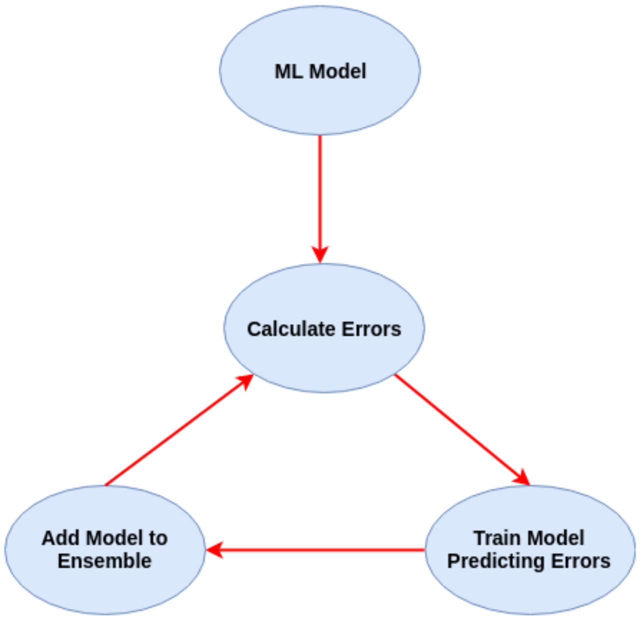
XGBoost

- Regular machine learning models, like a decision tree, simply train a single model on the dataset and use that for prediction.

- Building an ensemble, all the models are trained and applied to our data separately.

- **Boosting,** takes a more *iterative* approach. It's still technically an ensemble technique with many models are combined to perform the final prediction.

- Instead of training all the models in isolation of one another, boosting trains models in succession, with each new model being trained to correct the errors made by the previous ones.



XGBoost

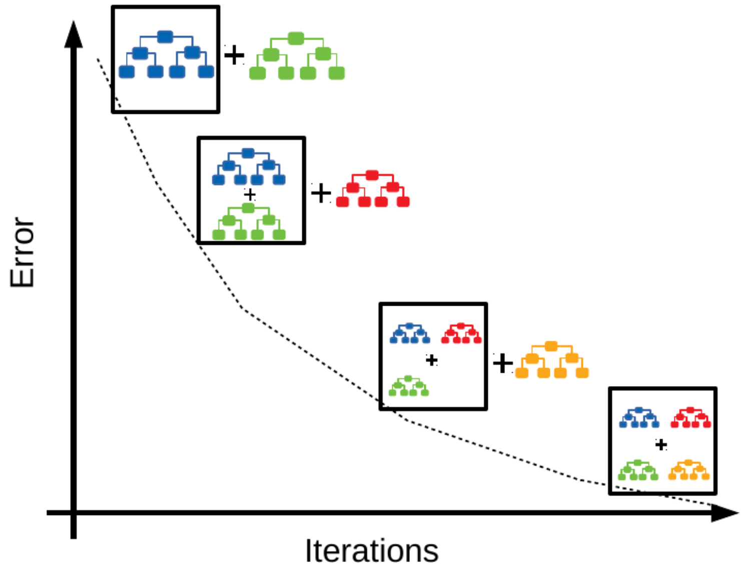
- Models are added sequentially until no further improvements can be made.

- Advantage is the new models being added are focused on correcting the mistakes which were caused by other models.

- Each subsequent model is trained on a smaller portion of data
  - Quicker to create each subsequent model

- In regular ensemble methods models are trained in isolation, all the models might end up making the same mistakes!

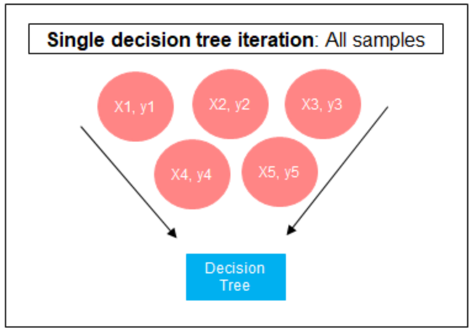
- **Gradient Boosting** specifically is an approach where new models are trained to predict the errors of prior models

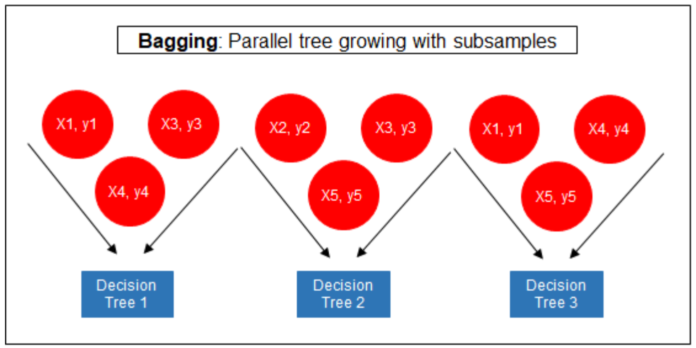


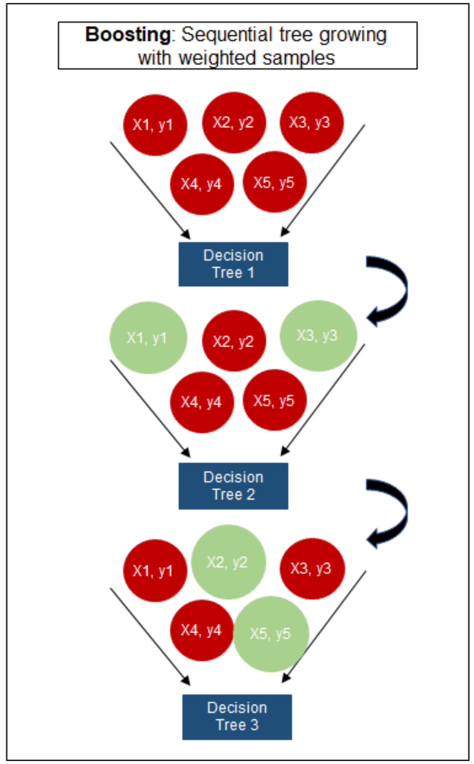
XGBoost

Optimised for

- Parallel processing
- Tree Pruning – Depth first approach
- Memory, Cache and Hardware optimised
- Fewer resources

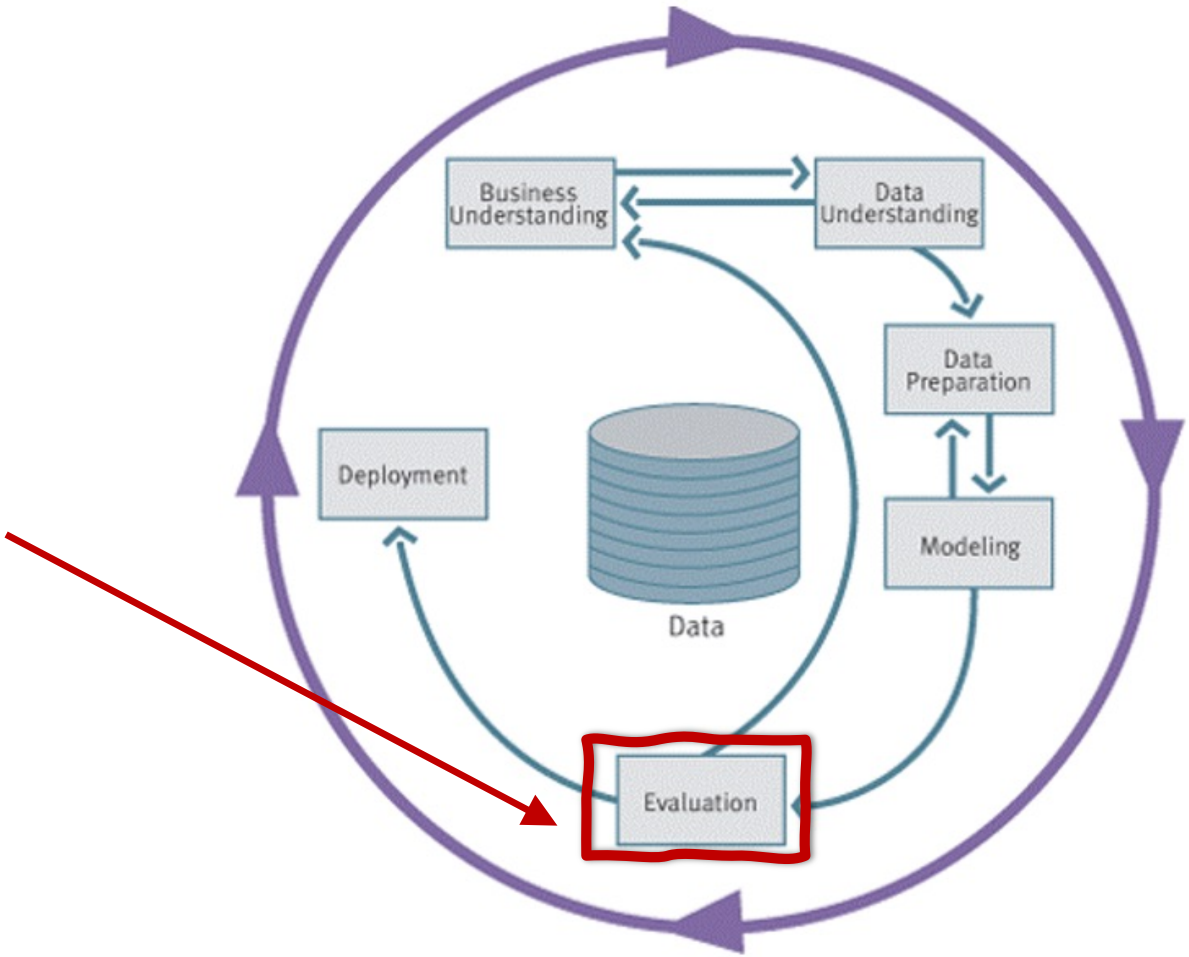






Test & Evaluate

Test & Evaluation



Why do we evaluate?

- Why do we evaluate the models created?
  - Remember, we will create many models -> We need to find the one that works best
  - The one that works best, on our data (as it is now), for our problem, at this point in time
    - If we were to rerun all the code again with minor changes, we could get a different outcome
  - What will work best for us
    - To determine which model is the most suitable for a task
    - To communicate to (business) users on what should be used

How do we evaluate

- It isn't complicated – But many make it complicated
- It's very simple really!
- In reality it is just Counting
  - How many you correctly predicted
  - How many you incorrectly predicted

- Remember, a model will/can never be 100% correct
  - It is an approximation
- Keep It Simple!

Classifier Accuracy

- The **accuracy** of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier
  - Often also referred to as **recognition rate**
  - **Error rate** (or **misclassification rate**) is the opposite of accuracy

No Free Lunch

- In machine learning, there’s something called the “No Free Lunch” theorem. In a nutshell, it states that no one algorithm works best for every problem.
- As a result, one should **try many different algorithms for the problem**, while using a hold-out “test set” of data to evaluate performance and select the winner.

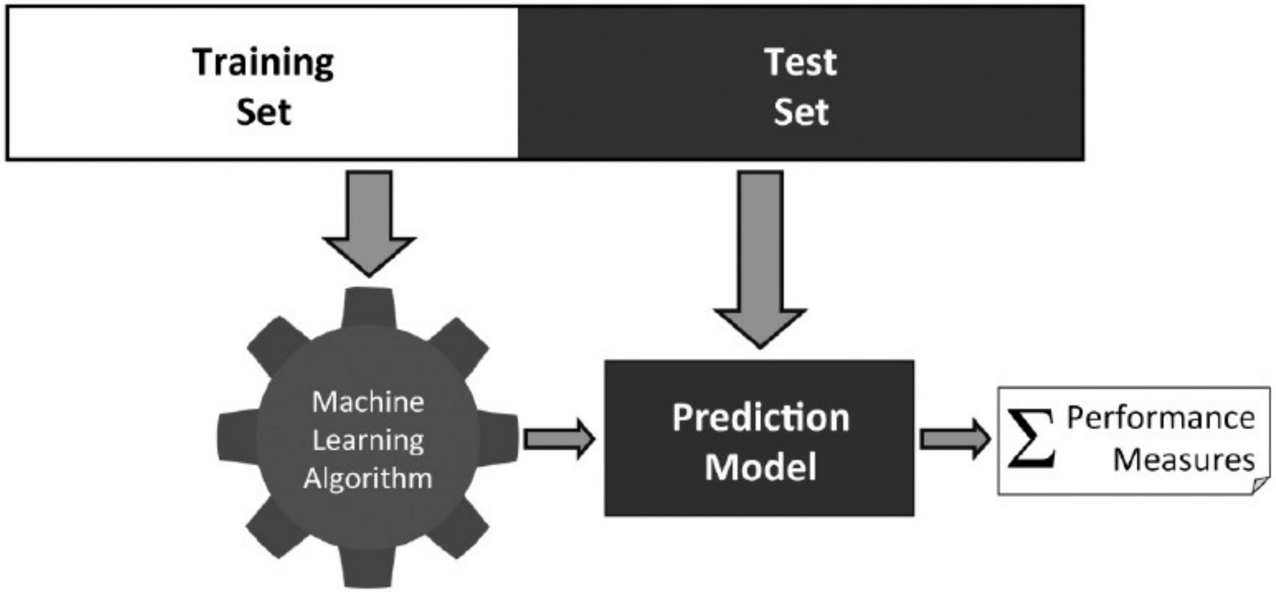
False Positives Vs False Negatives

- While it is useful to generate the simple accuracy of a classifier, sometimes we need more
- When is the classifier wrong?
  - False positives vs false negatives
  - Related to type I and type II errors in statistics

- Often there is a different cost associated with false positives and false negatives
  - Think about diagnosing diseases

Test Dataset

- We run the model against the “unseen” dataset -> Test Dataset

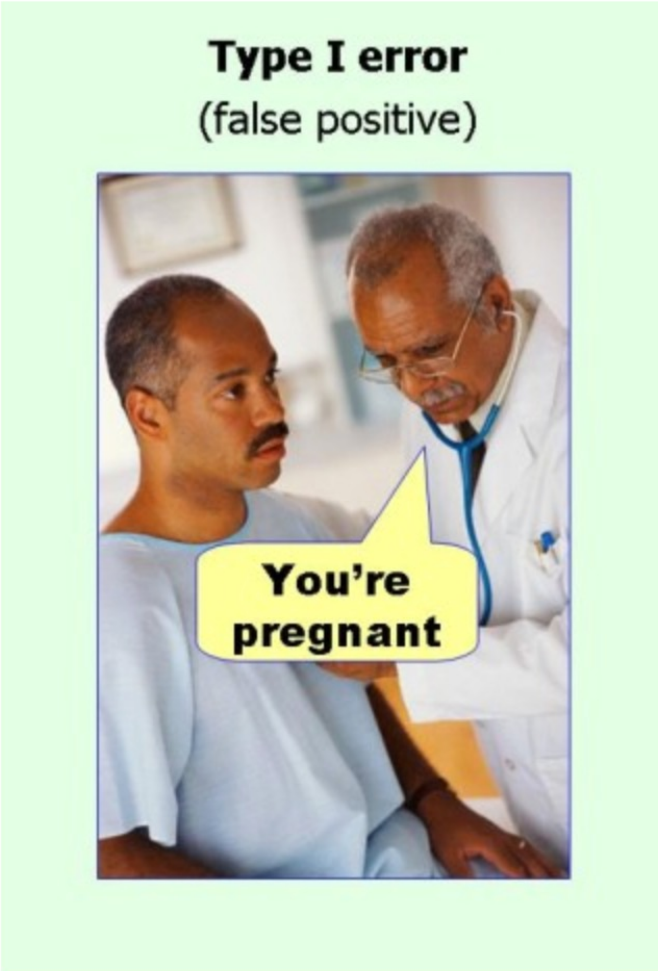


The process of building and evaluating a model using a **hold-out test set**.

Confusion Matrix

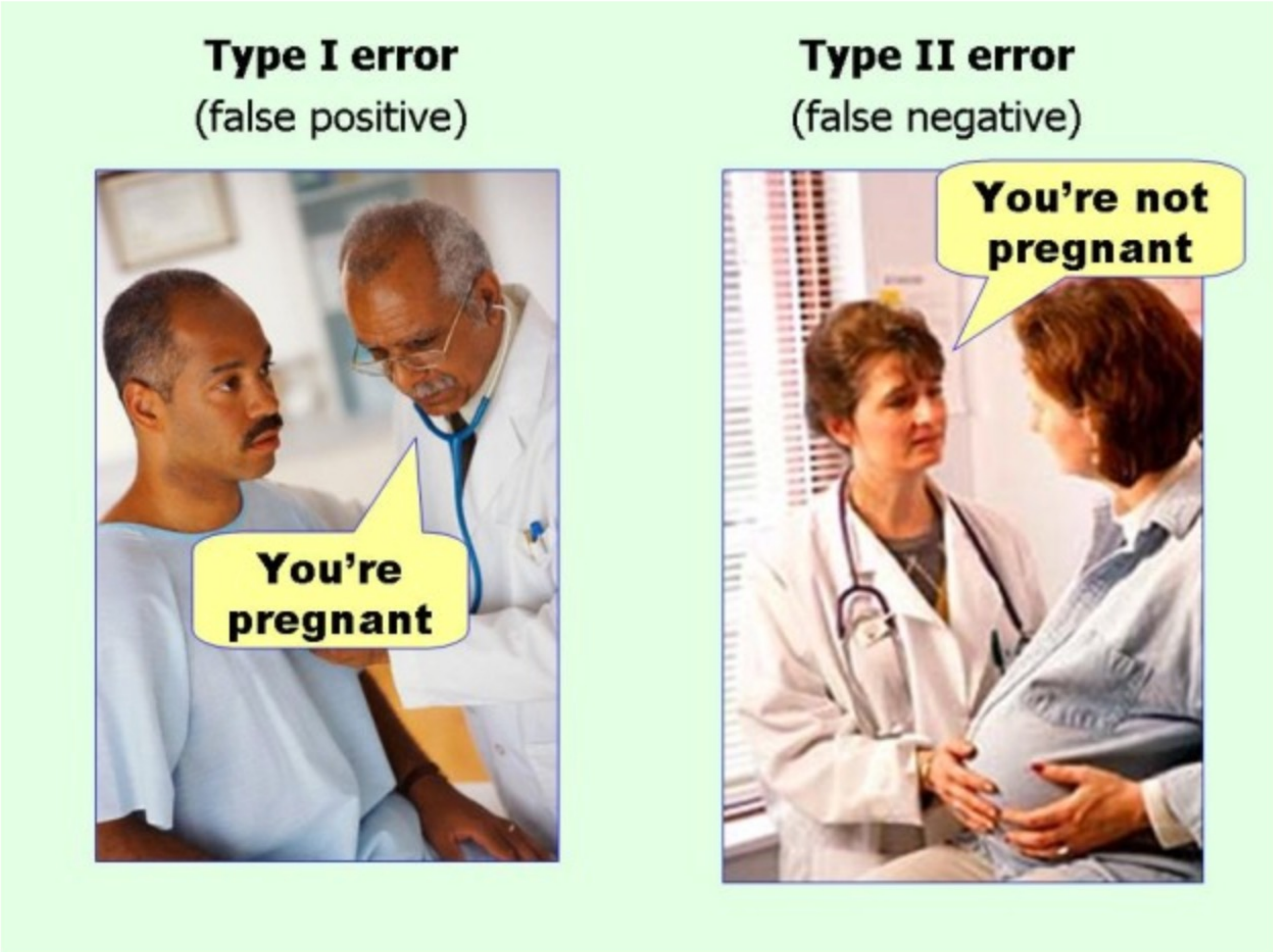
- Confusion Matrix used to illustrate how a classifier is performing in terms of false positives and false negatives
- Gives us more information than a single accuracy figure
- Allows us to think about the cost of mistakes
- Can be extended to any number of classes
  - Binary Classification
  - Multi-Class Classification

Classifier Result			
Class A (yes)	Class B (no)		
✓	fn	Class A (yes)	Expected Result
fp	✓	Class B (no)	



Type I & Type II Errors

"Type I" and "Type II" errors, names first given by Jerzy Neyman and Egon Pearson to describe rejecting a null hypothesis when it's true and accepting one when it's not, are too vague for stat newcomers (and in general). This is better. [via]

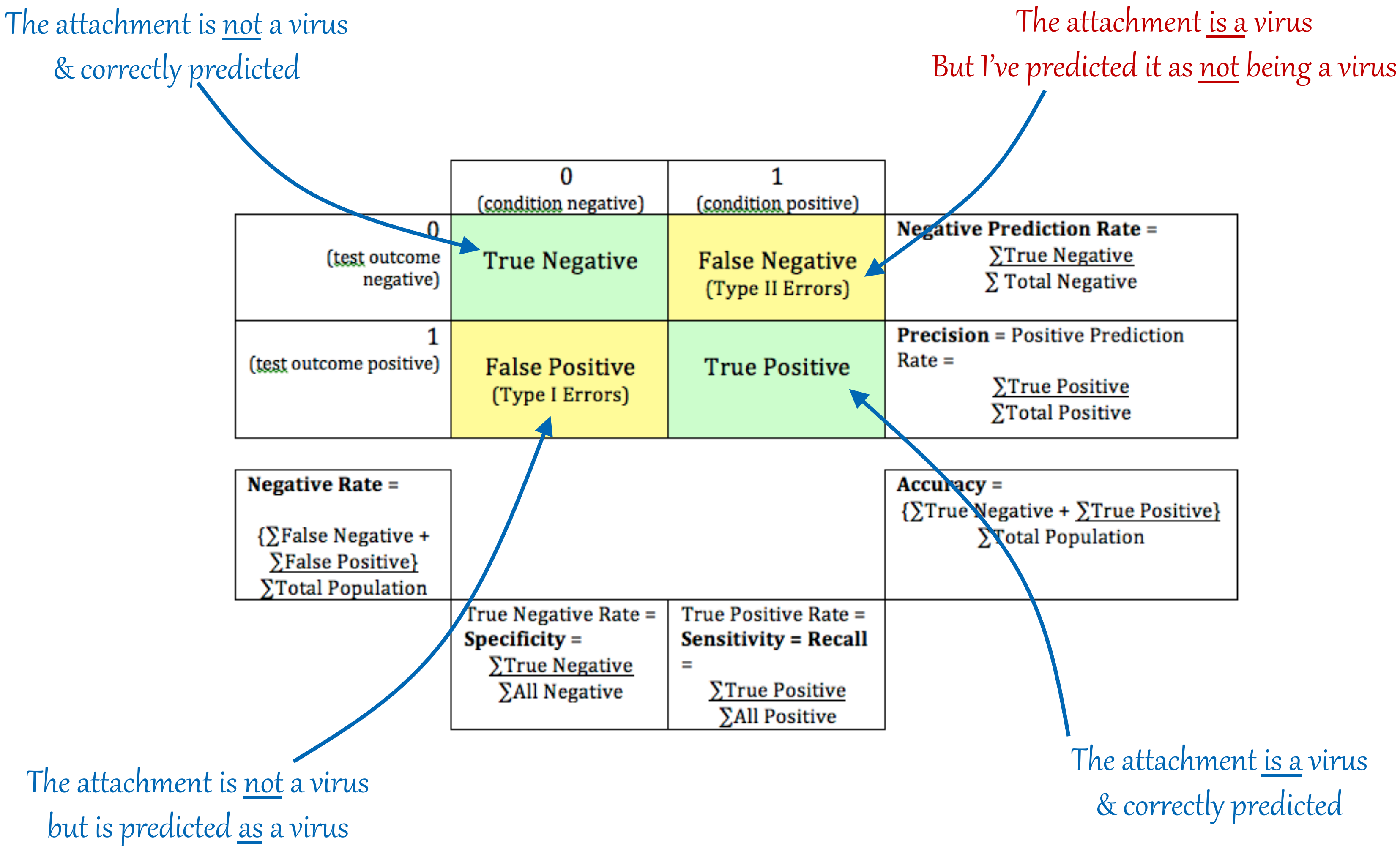


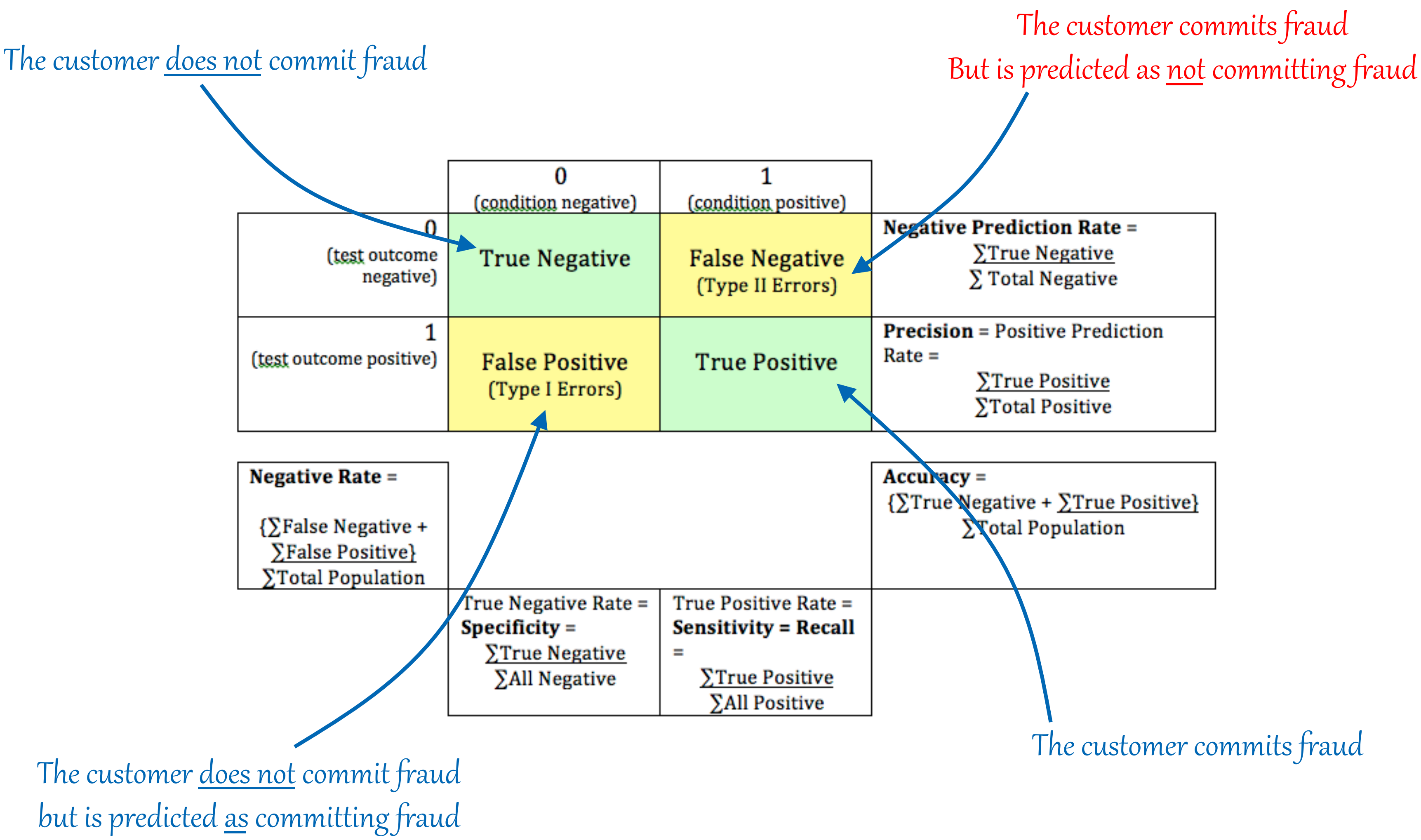
Type I & Type II Errors

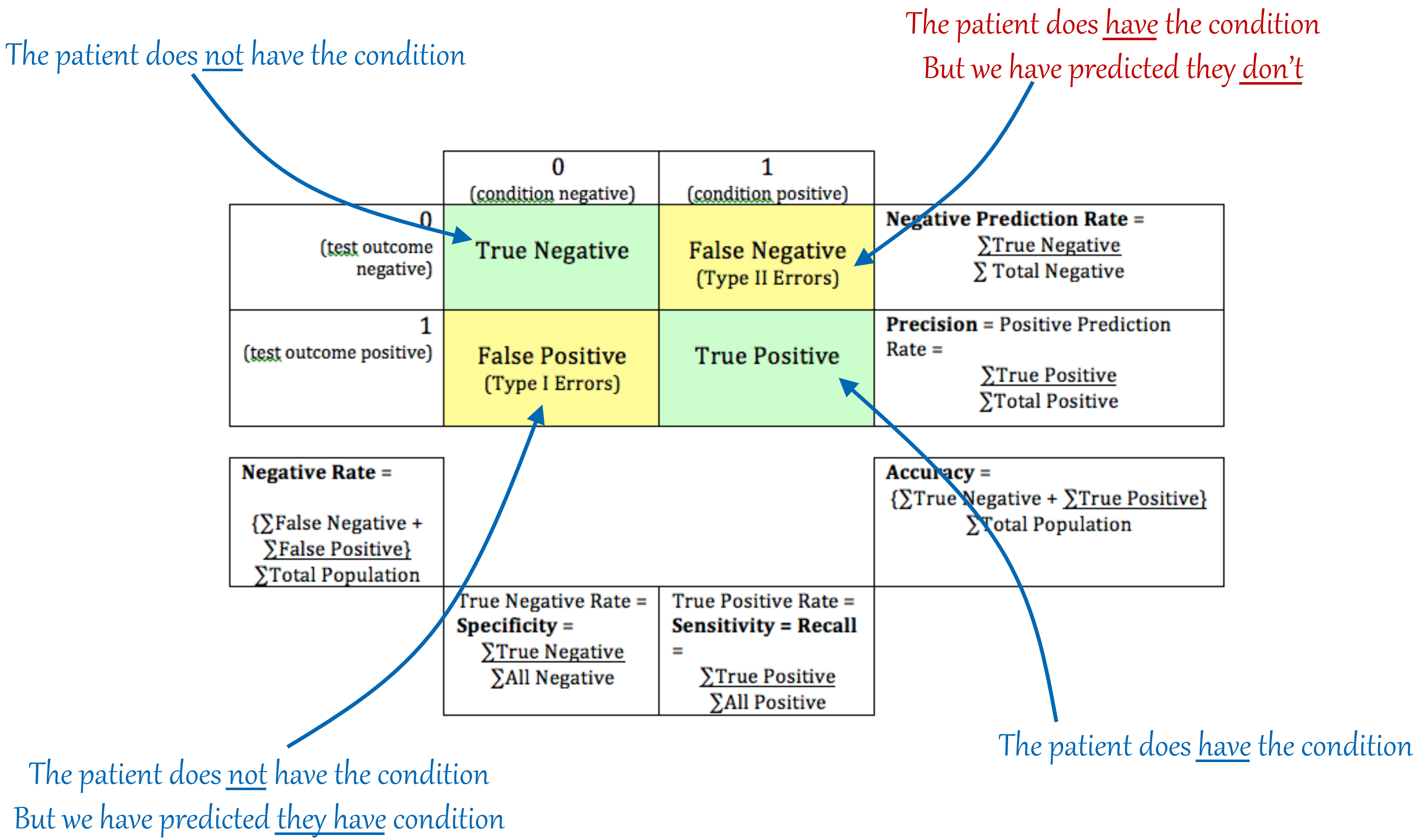
"Type I" and "Type II" errors, names first given by Jerzy Neyman and Egon Pearson to describe rejecting a null hypothesis when it's true and accepting one when it's not, are too vague for stat newcomers (and in general). This is better. [via]

Lots of different versions of this.
All are saying the same thing

	0 (condition negative)	1 (condition positive)	
0 (test outcome negative)	True Negative	False Negative (Type II Errors)	**Negative Prediction Rate** = $\frac{\sum \text{True Negative}}{\sum \text{Total Negative}}$
1 (test outcome positive)	False Positive (Type I Errors)	True Positive	**Precision** = Positive Prediction Rate = $\frac{\sum \text{True Positive}}{\sum \text{Total Positive}}$
Negative Rate = $\frac{\{\sum \text{False Negative} + \sum \text{False Positive}\}}{\sum \text{Total Population}}$			**Accuracy** = $\frac{\{\sum \text{True Negative} + \sum \text{True Positive}\}}{\sum \text{Total Population}}$
	True Negative Rate = **Specificity** = $\frac{\sum \text{True Negative}}{\sum \text{All Negative}}$	True Positive Rate = **Sensitivity = Recall** = $\frac{\sum \text{True Positive}}{\sum \text{All Positive}}$	


The attachment is not a virus
& correctly predicted
The attachment is a virus
But I've predicted it as not being a virus
0
(condition negative)
1
(condition positive)
0
(test outcome negative)
True Negative
False Negative
(Type II Errors)
Negative Prediction Rate =
∑True Negative
∑ Total Negative
1
(test outcome positive)
False Positive
(Type I Errors)
True Positive
Precision = Positive Prediction Rate =
∑True Positive
∑Total Positive
Negative Rate =
{∑False Negative + ∑False Positive}
∑Total Population
Accuracy =
{∑True Negative + ∑True Positive}
∑Total Population
True Negative Rate = Specificity =
∑True Negative
∑All Negative
True Positive Rate = Sensitivity = Recall =
∑True Positive
∑All Positive
The attachment is not a virus
but is predicted as a virus
The attachment is a virus
& correctly predicted


The customer does not commit fraud
The customer commits fraud
But is predicted as not committing fraud
0
(condition negative)
1
(condition positive)
0
(test outcome negative)
True Negative
False Negative
(Type II Errors)
Negative Prediction Rate =
∑True Negative
∑ Total Negative
1
(test outcome positive)
False Positive
(Type I Errors)
True Positive
Precision = Positive Prediction Rate =
∑True Positive
∑Total Positive
Negative Rate =
{∑False Negative + ∑False Positive}
∑Total Population
Accuracy =
{∑True Negative + ∑True Positive}
∑Total Population
True Negative Rate = Specificity =
∑True Negative
∑All Negative
True Positive Rate = Sensitivity = Recall =
∑True Positive
∑All Positive
The customer does not commit fraud
but is predicted as committing fraud
The customer commits fraud


The patient does not have the condition
The patient does have the condition
But we have predicted they don't
0
(condition negative)
1
(condition positive)
0
(test outcome negative)
True Negative
False Negative
(Type II Errors)
Negative Prediction Rate =
∑True Negative
∑ Total Negative
1
(test outcome positive)
False Positive
(Type I Errors)
True Positive
Precision = Positive Prediction Rate =
∑True Positive
∑Total Positive
Negative Rate =
{∑False Negative + ∑False Positive}
∑Total Population
Accuracy =
{∑True Negative + ∑True Positive}
∑Total Population
True Negative Rate = Specificity =
∑True Negative
∑All Negative
True Positive Rate = Sensitivity = Recall =
∑True Positive
∑All Positive
The patient does not have the condition
But we have predicted they have condition
The patient does have the condition

Confusion Matrix - Example

A sample test set with model predictions.

ID	Target	Pred.	Outcome	ID	Target	Pred.	Outcome
1	spam	ham	FN	11	ham	ham	TN
2	spam	ham	FN	12	spam	ham	FN
3	ham	ham	TN	13	ham	ham	TN
4	spam	spam	TP	14	ham	ham	TN
5	ham	ham	TN	15	ham	ham	TN
6	spam	spam	TP	16	ham	ham	TN
7	ham	ham	TN	17	ham	spam	FP
8	spam	spam	TP	18	spam	spam	TP
9	spam	spam	TP	19	ham	ham	TN
10	spam	spam	TP	20	ham	spam	FP

		Prediction	
		'spam'	***'ham'***
Target	***'spam'***	6	3
	'ham'	2	9

Confusion Matrix - Example

	0 (condition negative)	1 (condition positive)	
0 (test outcome negative)	True Negative	False Negative (Type II Errors)	**Negative Prediction Rate** = ∑True Negative / ∑ Total Negative
1 (test outcome positive)	False Positive (Type I Errors)	True Positive	**Precision** = Positive Prediction Rate = ∑True Positive / ∑Total Positive
Negative Rate = {∑False Negative + ∑False Positive} / ∑Total Population			**Accuracy** = {∑True Negative + ∑True Positive} / ∑Total Population
	True Negative Rate = **Specificity** = ∑True Negative / ∑All Negative	True Positive Rate = **Sensitivity = Recall** = ∑True Positive / ∑All Positive	

$$\text{precision} = \frac{6}{(6+2)} = 0.75$$

$$\text{misclassification accuracy} = \frac{(2+3)}{(6+9+2+3)} = 0.25$$

$$\text{classification accuracy} = \frac{(6+9)}{(6+9+2+3)} = 0.75$$

$$\text{recall} = \frac{6}{(6+3)} = 0.667$$

Confusion Matrix – Example – What about Costs

- Not every outcome (or classification) has the same value
- A positive outcome could be worth money €
- A negative outcome could be work lots of money lost -€€€
- We can apply monetary values to the outcomes

Sample profit matrix for a credit scoring problem.

		Prediction 'good'	Prediction 'bad'
Target	'good'	140	−140
	'bad'	−700	0

It was predicted as a "Good" Risk
But in reality it turned out to be "Bad".
In this case, how much on average would such a scenario cost us

Confusion Matrix – Example – What about Costs

- Not every outcome (or classification) has the same value
- A positive outcome could be worth money €
- A negative outcome could be work lots of money lost **-€€€**
- We can apply monetary values to the outcomes

Sample profit matrix for a credit scoring problem.

How much € can we make by getting this correct

		Prediction	
		'good'	***'bad'***
Target	***'good'***	140	−140
	'bad'	−700	0

We should have approved these.
But we didn't.
Missed opportunity cost

We will do nothing with these.
So no cost/€

It was predicted as a "Good" Risk
But in reality it turned out to be "Bad".
In this case, how much on average would such a scenario cost us

Confusion Matrix – Example – What about Costs

- Not every outcome (or classification) has the same value
- A positive outcome could be worth money €
- A negative outcome could be work lots of money lost -€€€
- We can apply monetary values to the outcomes

Sample profit matrix for a credit scoring problem.

		Prediction 'good'	Prediction 'bad'
Target	'good'	140	−140
	'bad'	−700	0

How much € can we make by getting this correct

We should have approved these. But we didn't. Missed opportunity cost

We will do nothing with these. So no cost/€

It was predicted as a "Good" Risk
But in reality it turned out to be "Bad".
In this case, how much on average would such a scenario cost us

Confusion Matrix – Example – What about Costs

- Not every outcome (or classification) has the same value
- A positive outcome could be worth money €
- A negative outcome could be work lots of money lost -€€€
- We can apply monetary values to the outcomes

Sample profit matrix for a credit scoring problem.

		Prediction 'good'	Prediction 'bad'
Target	'good'	140	−140
	'bad'	−700	0

How much € can we make by getting this correct

We should have approved these.
But we didn't.
Missed opportunity cost

We will do nothing with these.
So no cost/€

It was predicted as a "Good" Risk
But in reality it turned out to be "Bad".
In this case, how much on average would such a scenario cost us

Confusion Matrix – Example – What about Costs

- Add up the numbers from each part/box of the Confusion Matrix
    - Total = € potential value for model

- Accountants like to see these numbers
- Your managers like to see these numbers
- Bosses like to see these numbers

- It quantifies/costs the potential € for each model

- The Business will understand using € (Language of Business)
    - Vs using Numbers + Percentages + Unusual Terms (Language of Analysts, Machine Learning, etc)

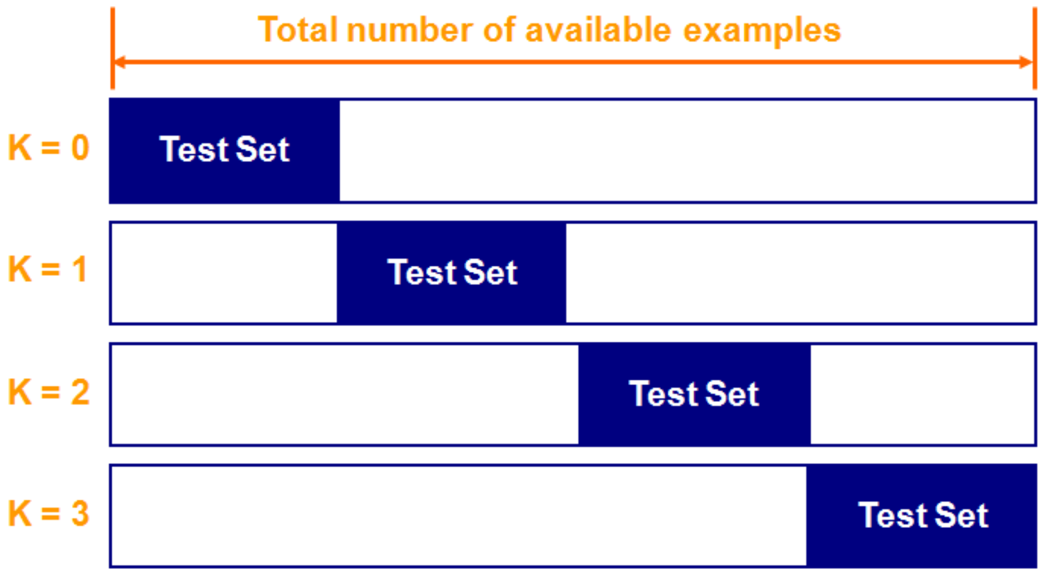
Hold-Out Testing Sets

- Split the available data into a *training set* and a *test set*

- Train the classifier in the training set and evaluate based on the test set
- A couple of drawbacks
  - We may not have enough data
  - We may happen upon an *unfortunate split*

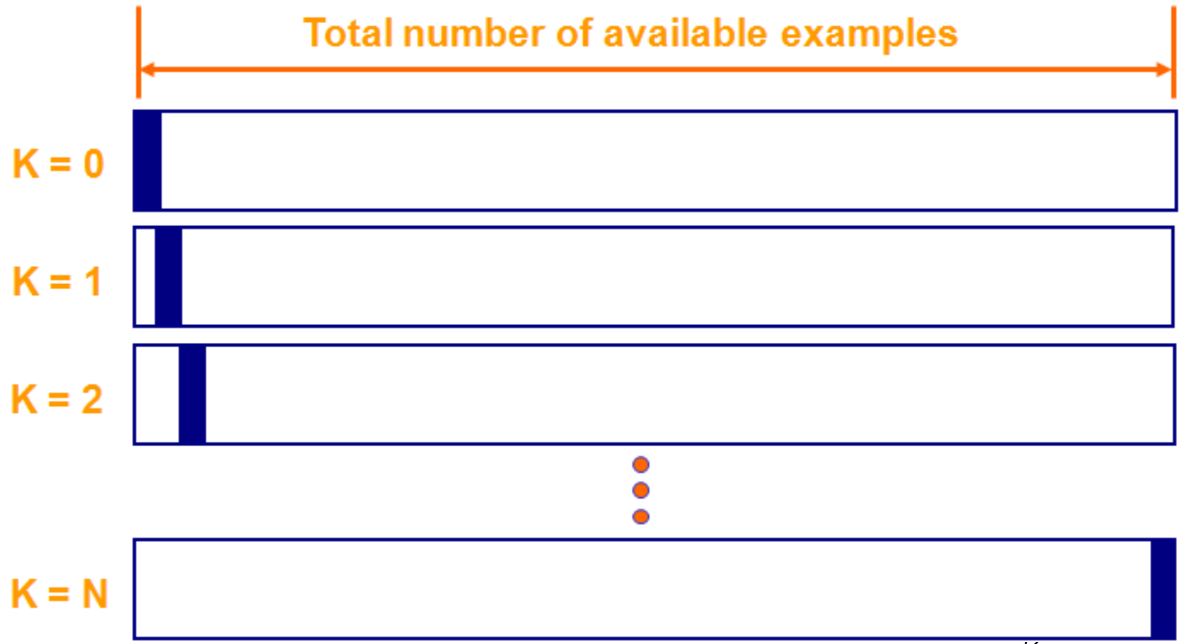
K-Fold Cross Validation

- An alternate is to divide the dataset into smaller chunks (Train & Test)
- k *folds* – where k is the number of times to divide the data
- For each of k experiments, use k^{th} fold for testing and everything else for training
- Average the results across the k folds



K-Fold Cross Validation

- The accuracy of the system is calculated as the average error across the k folds
- The main advantages of k-fold cross validation are that every example is used in testing at some stage and the problem of an *unfortunate split* is avoided
- Any value can be used for k
  - 10 is most common
  - Depends on the data set



A lot covered

- We have covered a lot in this class
- What is Classification
- Different Algorithms
- How to Evaluate

- Keep It Simple!
- Lab Work
    - Examples of the Algorithms
    - Examples of Evaluation
    - A few lines of code

- Next week we will
    - Look at a few more algorithms
    - Go over the Evaluate steps again

Time for an Example

Any Questions ?

What Now/Next ?