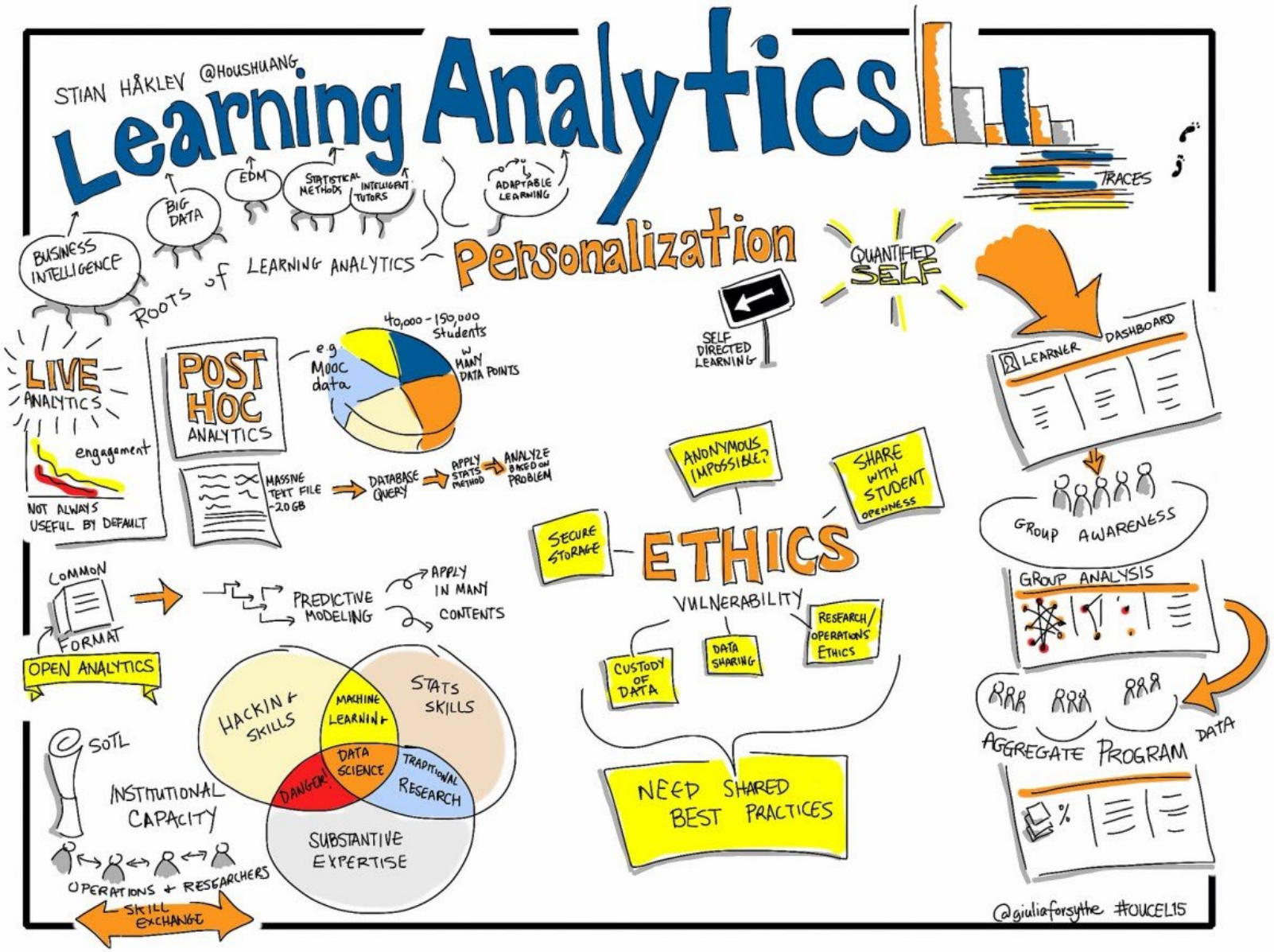
“Calculating machines are useless.
They can only give you answers.”

- Pablo Picasso, 1964

CM3015

Welcome to the World
of Machine Learning
and Neural Networks


STIAN HÅKLEV @HOUSHUANG
Learning Analytics
TRACES
BUSINESS INTELLIGENCE
BIG DATA
EDM
STATISTICAL METHODS
INTELLIGENT TUTORS
ADAPTABLE LEARNING
ROOTS of LEARNING ANALYTICS
Personalization
QUANTIFIED SELF
SELF DIRECTED LEARNING
LIVE ANALYTICS
engagement
NOT ALWAYS USEFUL BY DEFAULT
POST HOC ANALYTICS
40,000 - 150,000 students
w MANY DATA POINTS
e.g. Mooc data
MASSIVE TEXT FILE ~20GB
DATABASE QUERY
APPLY STATS METHOD
ANALYZE BASED ON PROBLEM
LEARNER DASHBOARD
GROUP AWARENESS
GROUP ANALYSIS
AGGREGATE PROGRAM DATA
COMMON FORMAT
OPEN ANALYTICS
PREDICTIVE MODELING
APPLY IN MANY CONTENTS
ANONYMOUS IMPOSSIBLE?
SHARE WITH STUDENT OPENNESS
SECURE STORAGE
ETHICS
VULNERABILITY
CUSTODY OF DATA
DATA SHARING
RESEARCH/ OPERATIONS ETHICS
NEED SHARED BEST PRACTICES
HACKING + SKILLS
STATS SKILLS
MACHINE LEARNING
DATA SCIENCE
DANGER!
TRADITIONAL RESEARCH
SUBSTANTIVE EXPERTISE
SOTL
INSTITUTIONAL CAPACITY
OPERATIONS + RESEARCHERS
SKILL EXCHANGE
@giuliaforsythe #OUCEL15

Welcome to the most fun world you will ever find. First:-

1

1. Make sure you can access Cholett's book

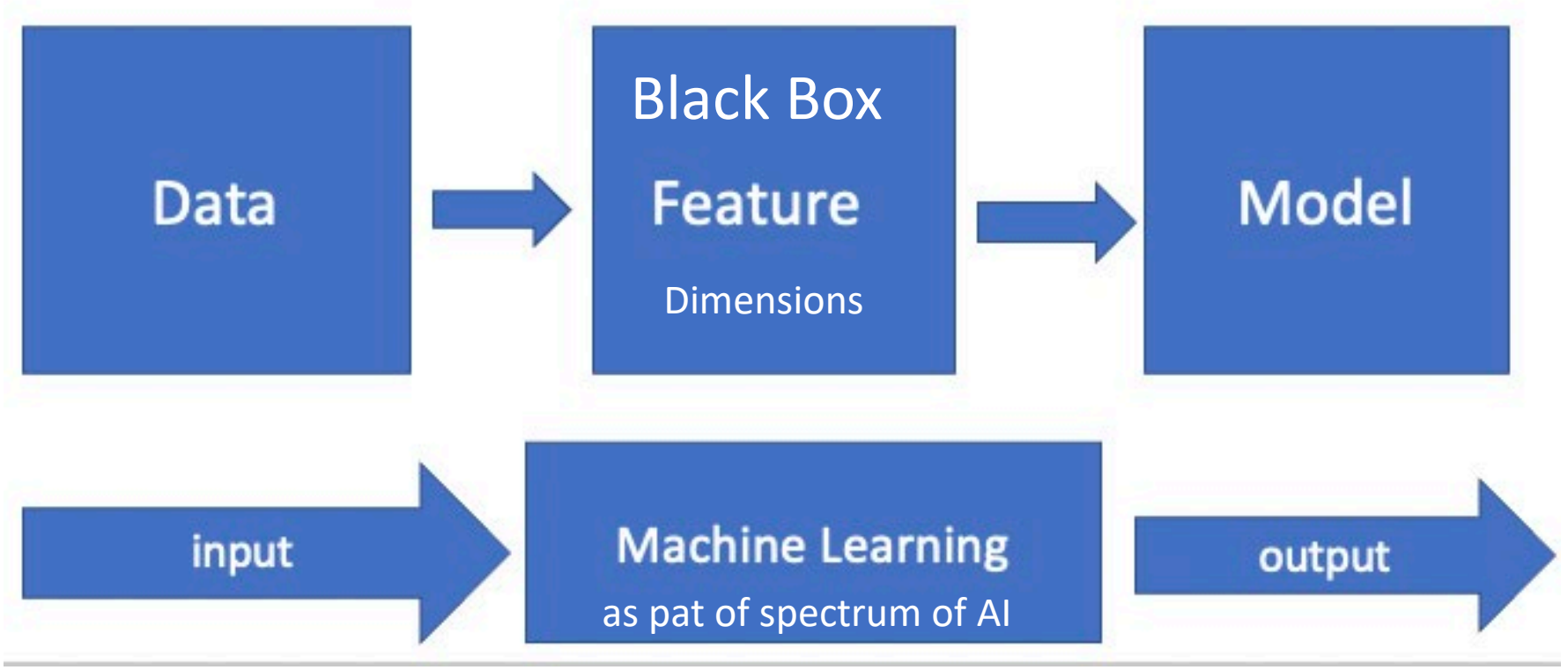
2

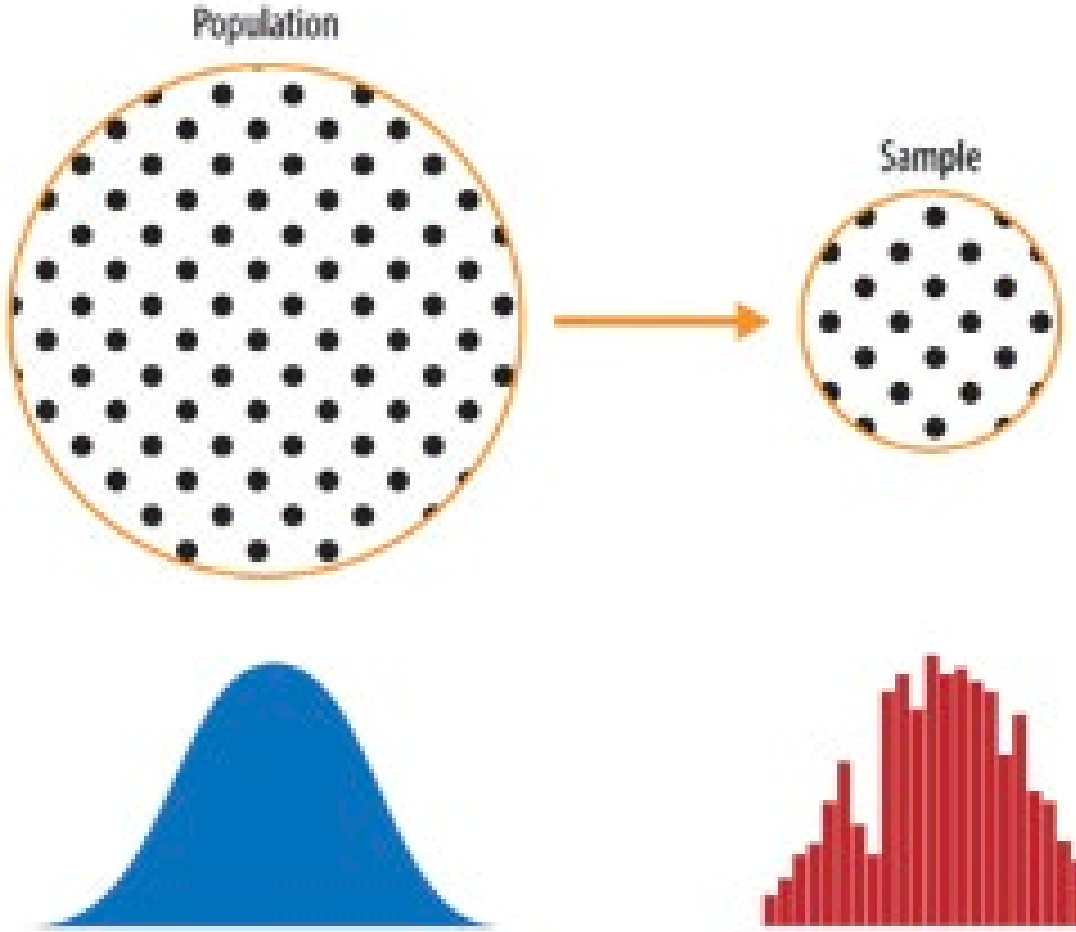
2. Make sure you can access Jupyter Notebooks

3

3. Create a very basic "Hello world" program in Jupyter

What does Data Science do?



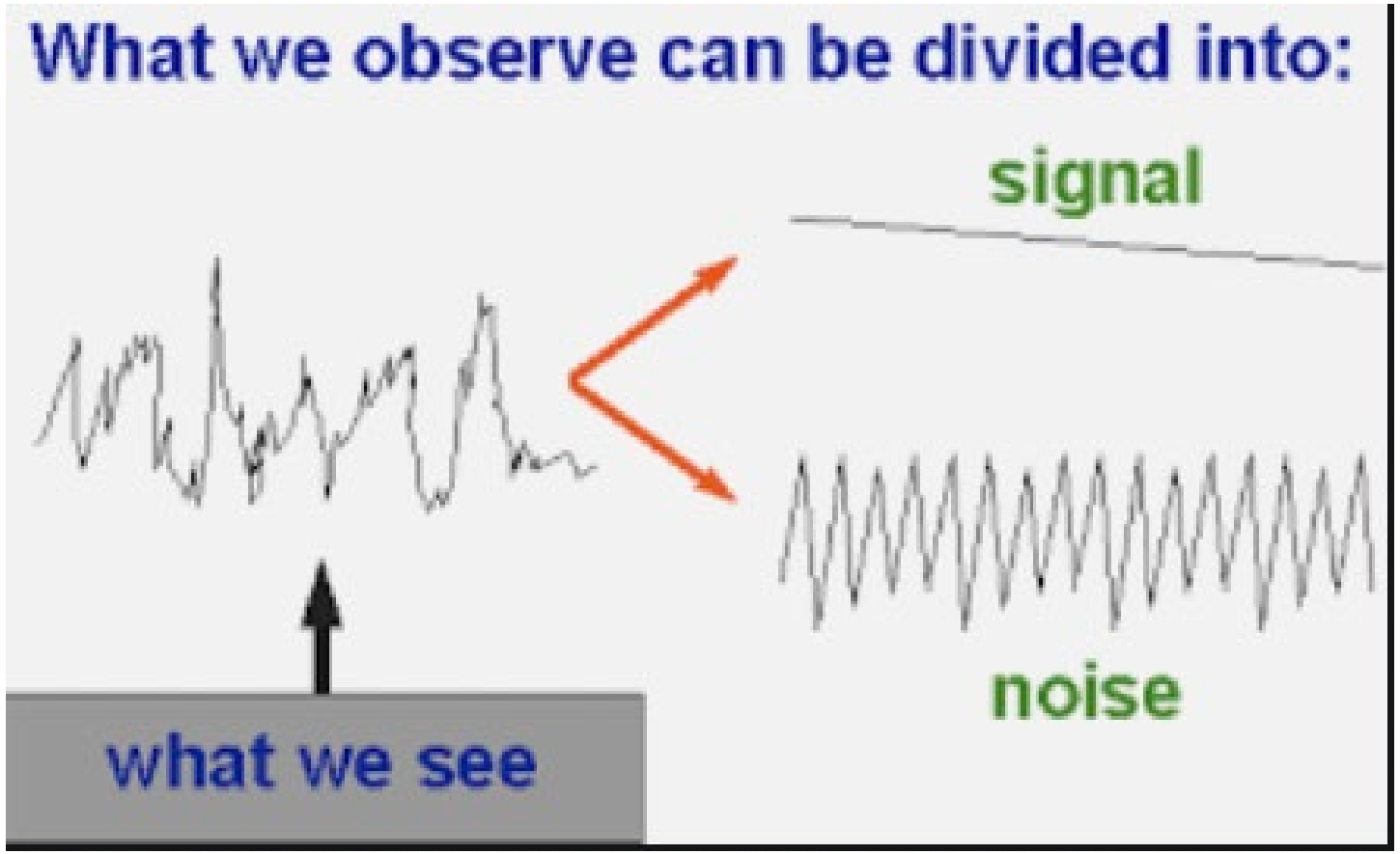


AI

Time Series

Machine Learning

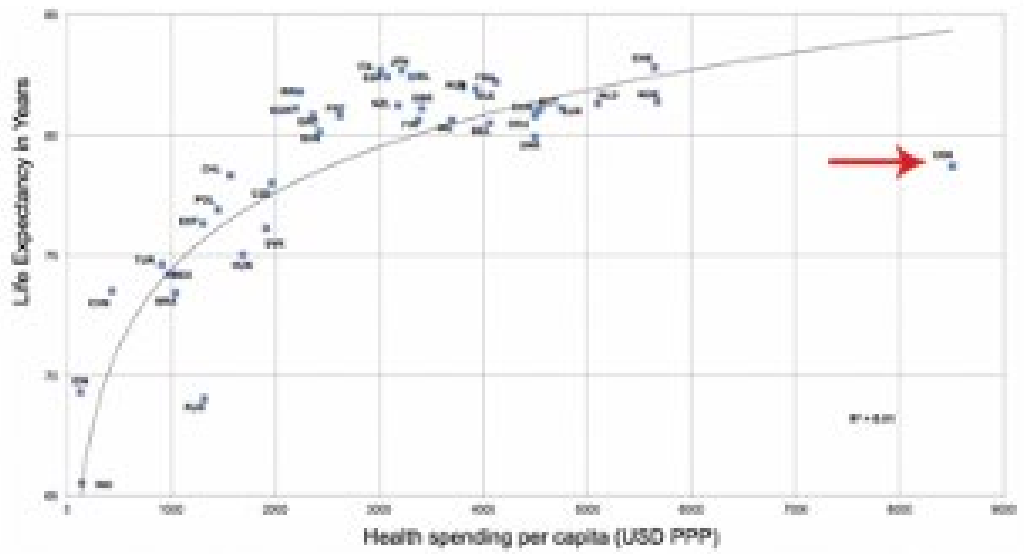
Deep Learning

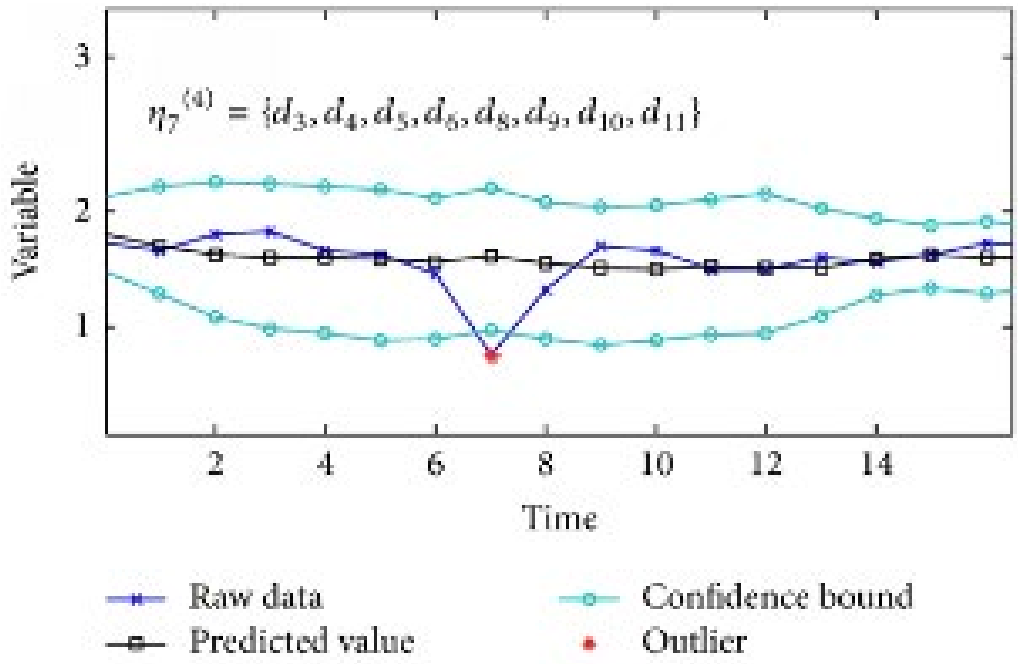


It is the goal of models to describe the signal, despite the noise.

A perfect model describes the signal exactly, and ignores all of the noise. If a model fails to capture all of the signal, that type of error is called **bias**. If a model captures of some of the noise, that type of error is called **variance**. Too much bias in our model means that it will perform poorly in all situations because it hasn't captured the signal well. You may also hear this called **underfitting**. This was the case

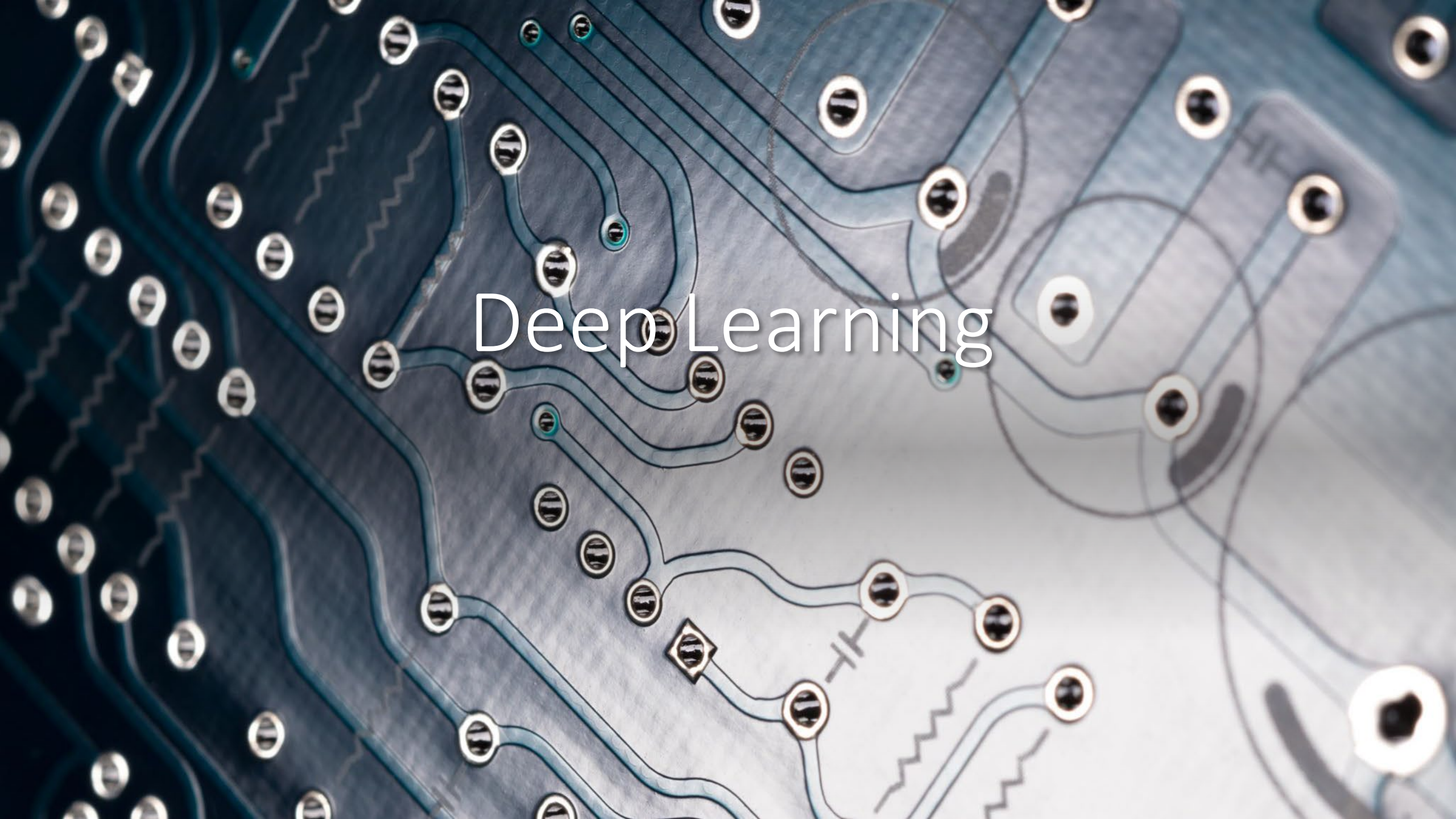
Outlier ≠ anomaly





No free lunch theorem

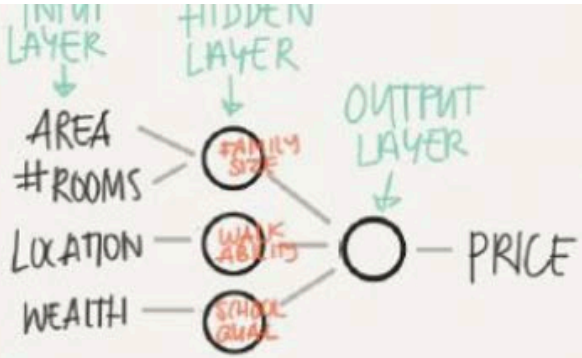
This means no single way of understanding data is superior to other methods for all possible learning problems.


Deep Learning

Deep Learning

With deep learning, the programming paradigm has moved from rules being coded as a program to be applied on data, to the rules being inferred from the data fed into the systems (Chollet, 2018, pp. 4-5).

INTRO TO DEEP LEARNING



SUPERVISED LEARNING

INPUT: X	OUTPUT: Y	NN TYPE
HOME FEATURES AD+USER INFO	PRICE WILL CLICK ON AD (0/1)	STANDARD NN
IMAGE	OBJECT (1...1000)	CONV. NN (CNN)
AUDIO ENGLISH	TEXT TRANSCRIPT CHINESE	RECURRENT NN (RNN)
IMAGE/RADAR	POS OF OTHER CARS	CUSTOM/HYBRID

NETWORK

NNs CAN DEAL WITH BOTH STRUCTURED & UNSTRUCTURED DATA

STRUCTURED

"THE QUICK BROWN FOX"

UNSTRUCTURED

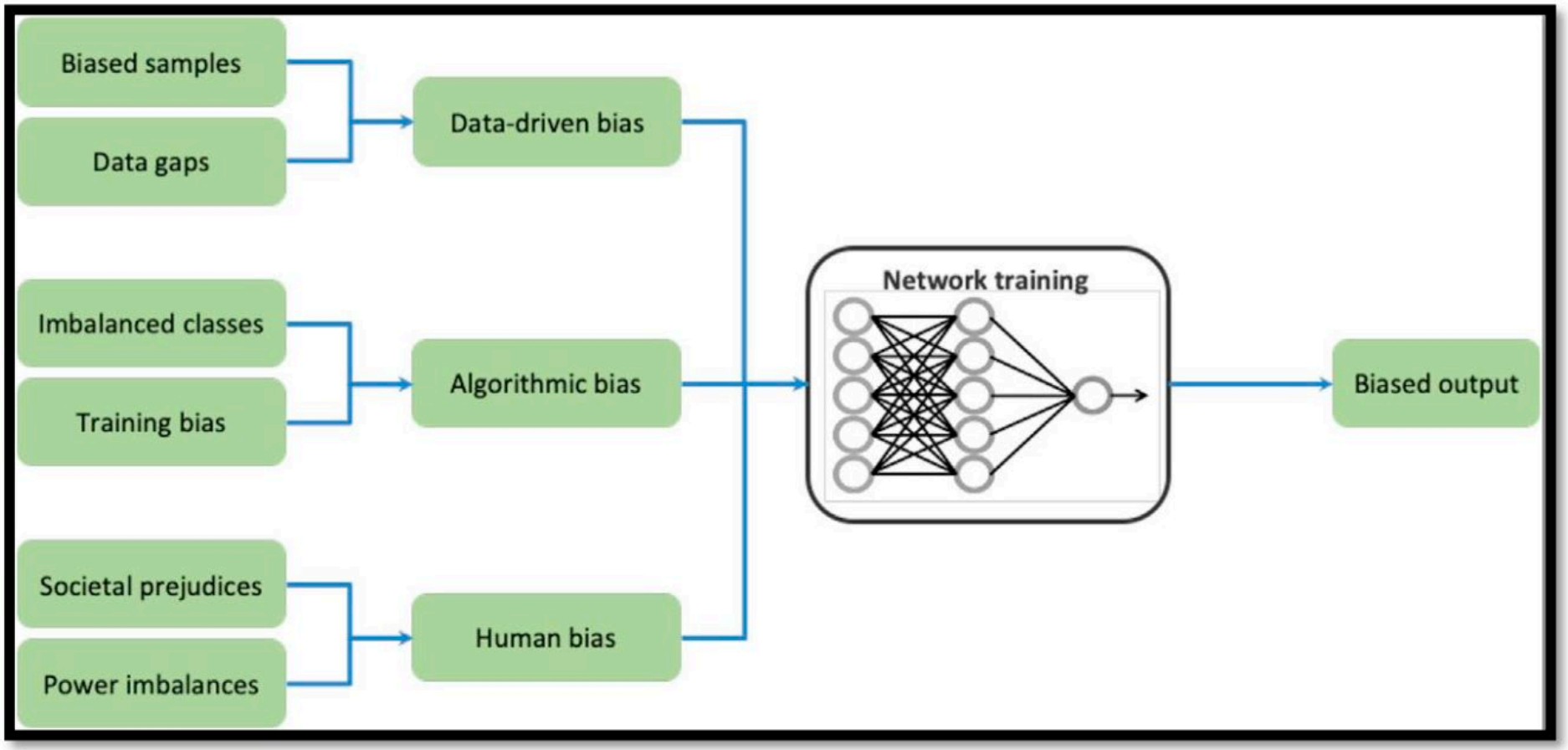
HUMANS ARE GOOD AT THIS

WHY NOW?

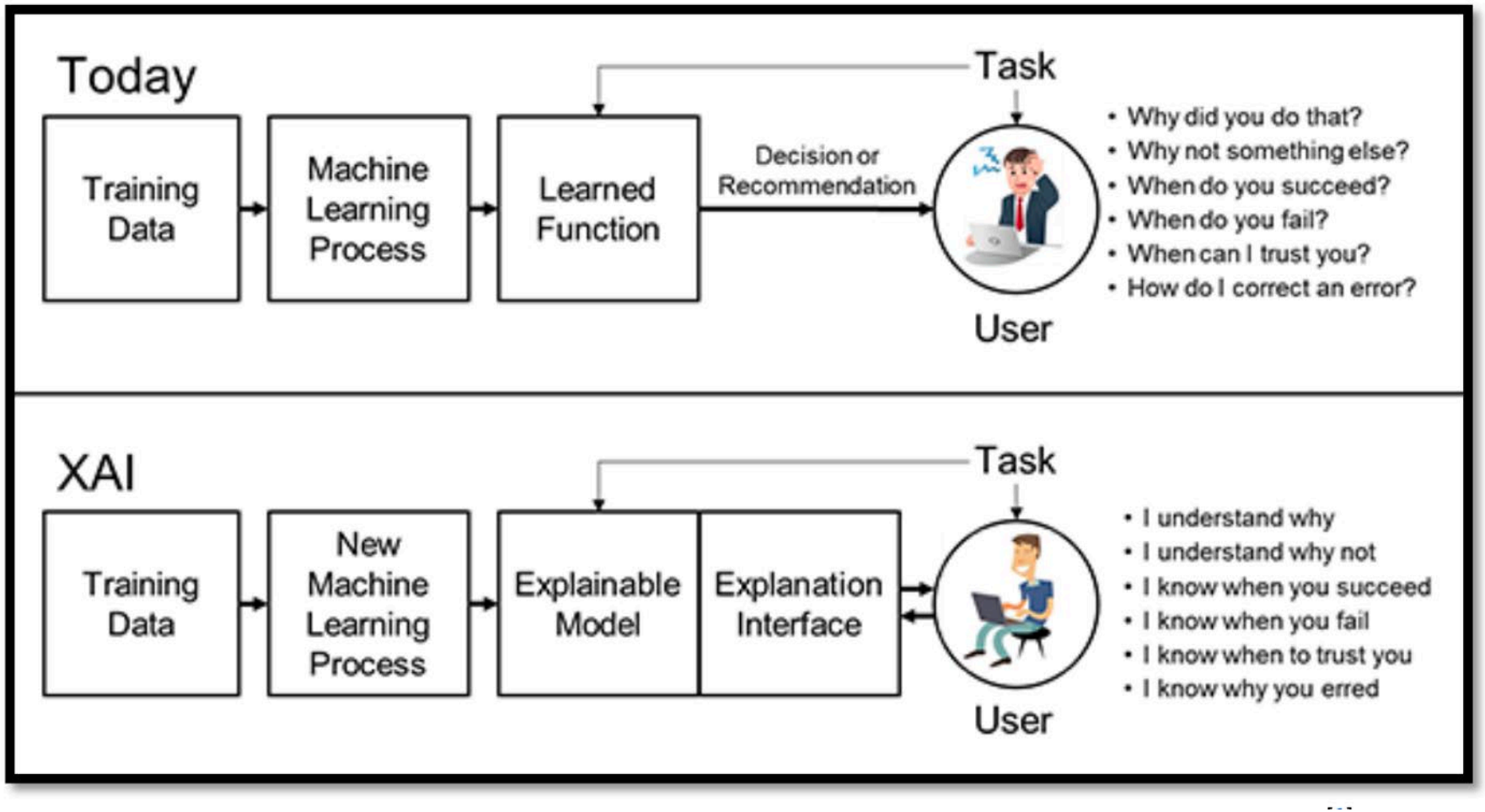
ONE OF THE BIG BREAKTHROUGHS HAS BEEN MOVING FROM SIGMOID TO RELU FOR FASTER GRADIENT DESCENT


Bias

Bias in Neural Networks



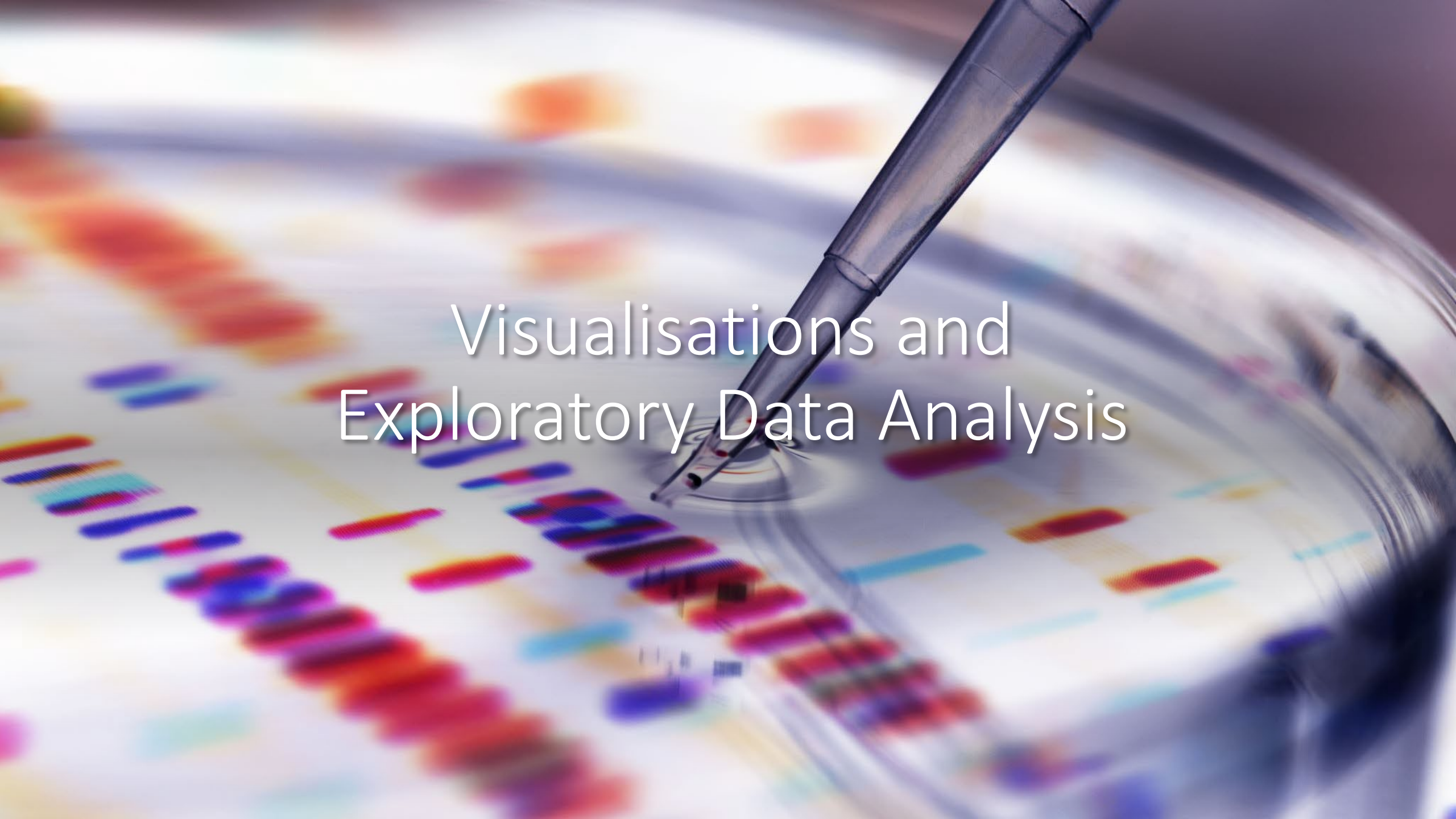
ML Today and XAI (the Future)



National Institute of Standards and Technology (NIST) 2020

The National Institute of Standards and Technology (NIST) in 2020 set out four principles they believe AI systems should follow in explaining their decisions, in order to garner public trust in these systems:-

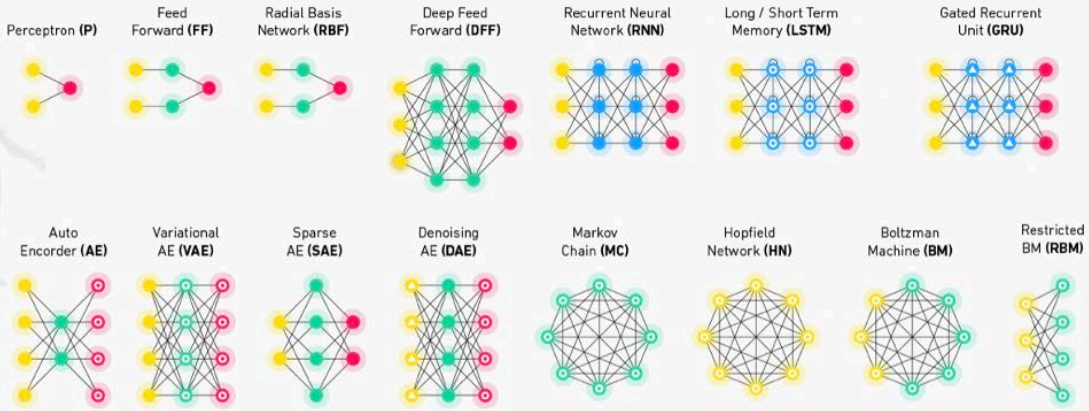
1. Provide evidence for all the outputs,
2. Give meaningful and understandable explanations to users
3. Show accurate depiction of the process used by the system to generate outputs
4. The system should only operate under the conditions it was designed for or when it has confidence in its outputs



Visualisations and Exploratory Data Analysis

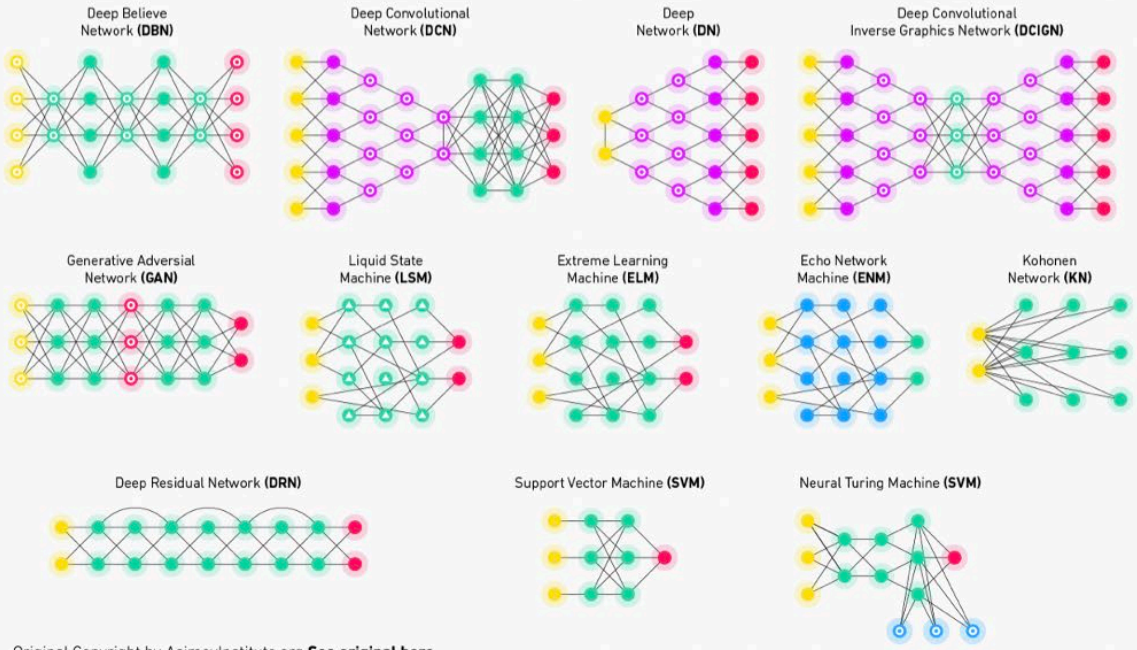
Neural Networks Basic Cheat Sheet

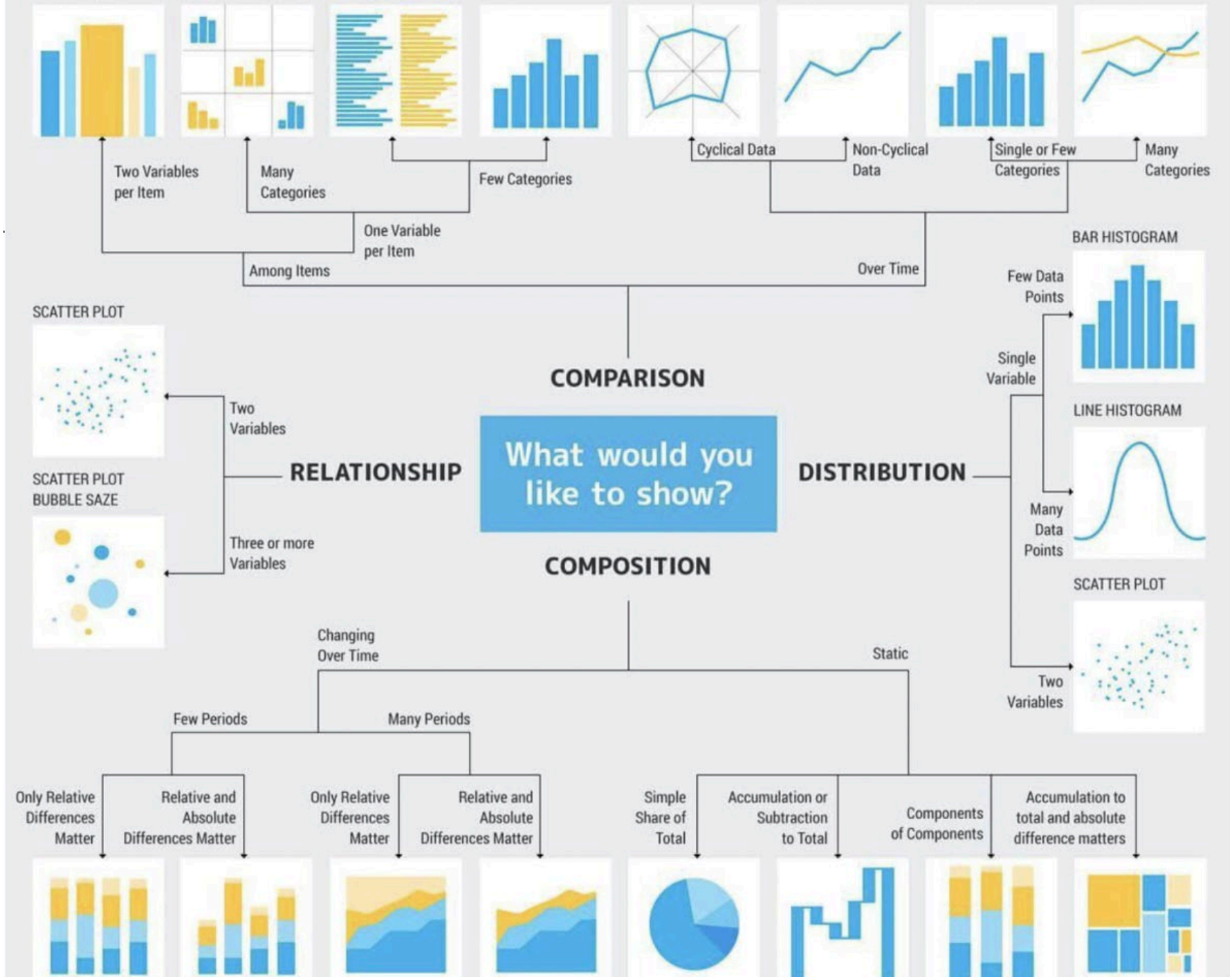
BecomingHuman.AI



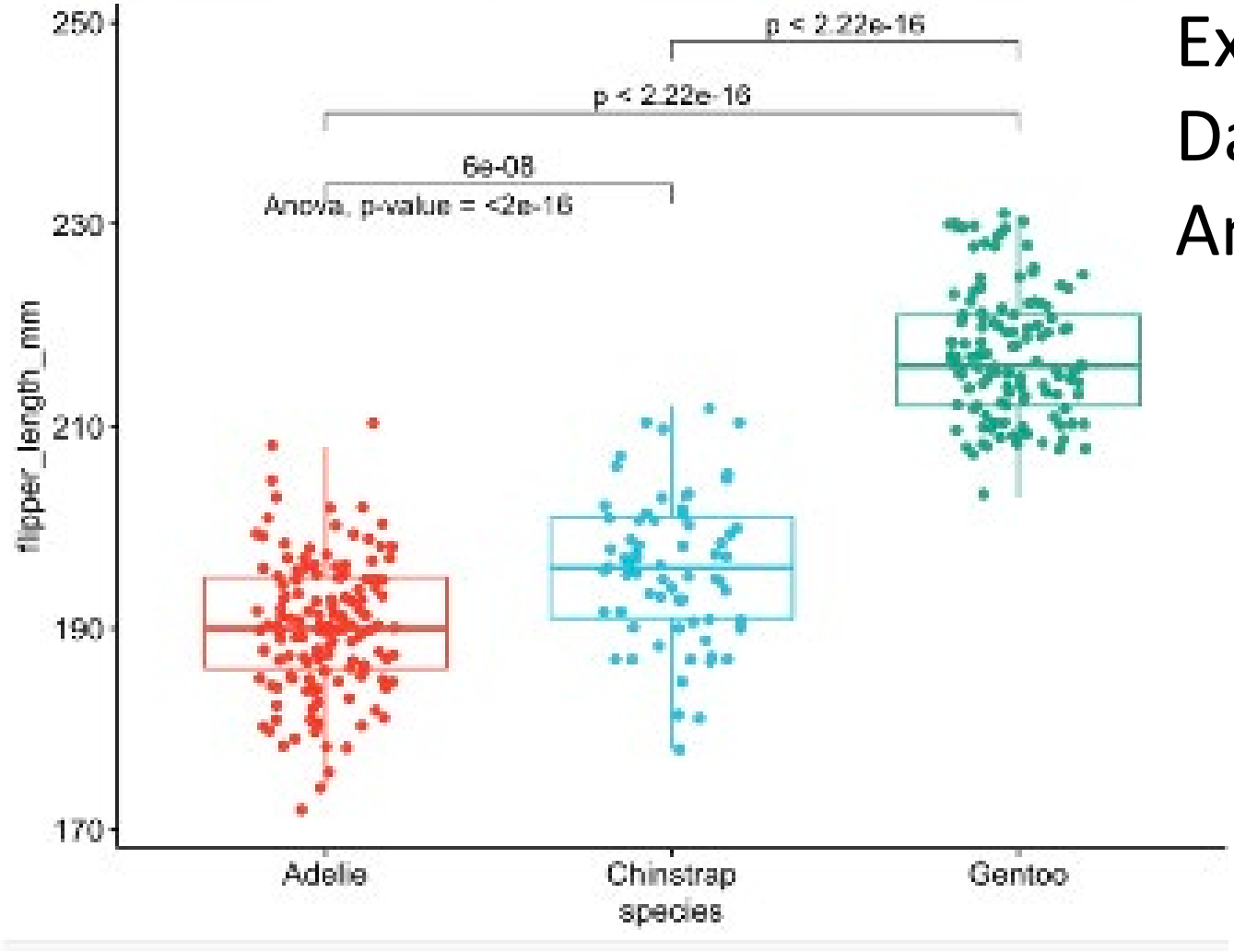
Index

- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probablisticc Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolutional or Pool

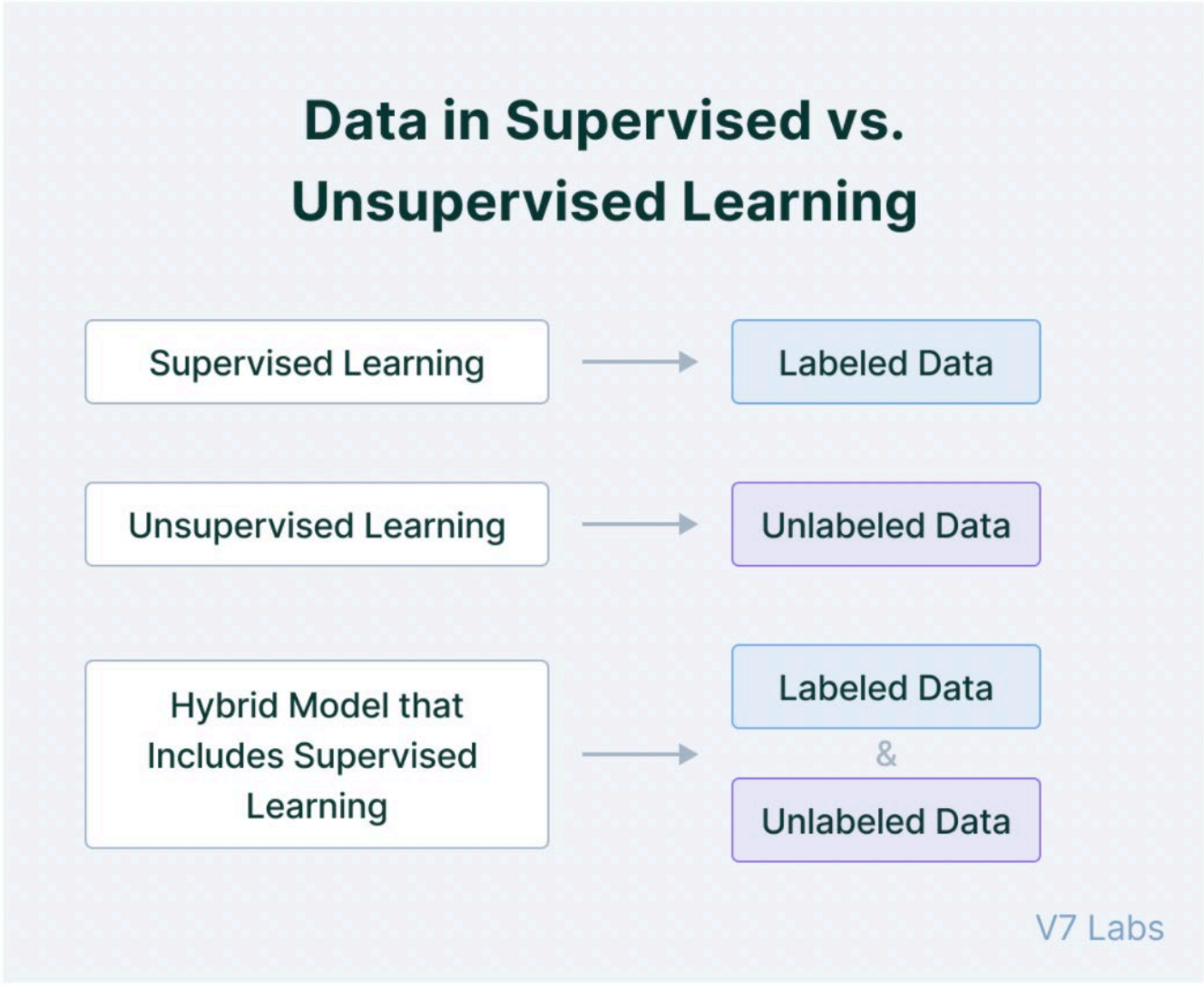



What would you like to show?
COMPARISON
RELATIONSHIP
DISTRIBUTION
COMPOSITION
Two Variables per Item
Many Categories
Few Categories
Cyclical Data
Non-Cyclical Data
Single or Few Categories
Many Categories
One Variable per Item
Among Items
Over Time
SCATTER PLOT
Two Variables
SCATTER PLOT BUBBLE SAZE
Three or more Variables
BAR HISTOGRAM
Few Data Points
Single Variable
LINE HISTOGRAM
Many Data Points
SCATTER PLOT
Two Variables
Changing Over Time
Static
Few Periods
Many Periods
Only Relative Differences Matter
Relative and Absolute Differences Matter
Only Relative Differences Matter
Relative and Absolute Differences Matter
Simple Share of Total
Accumulation or Subtraction to Total
Components of Components
Accumulation to total and absolute difference matters

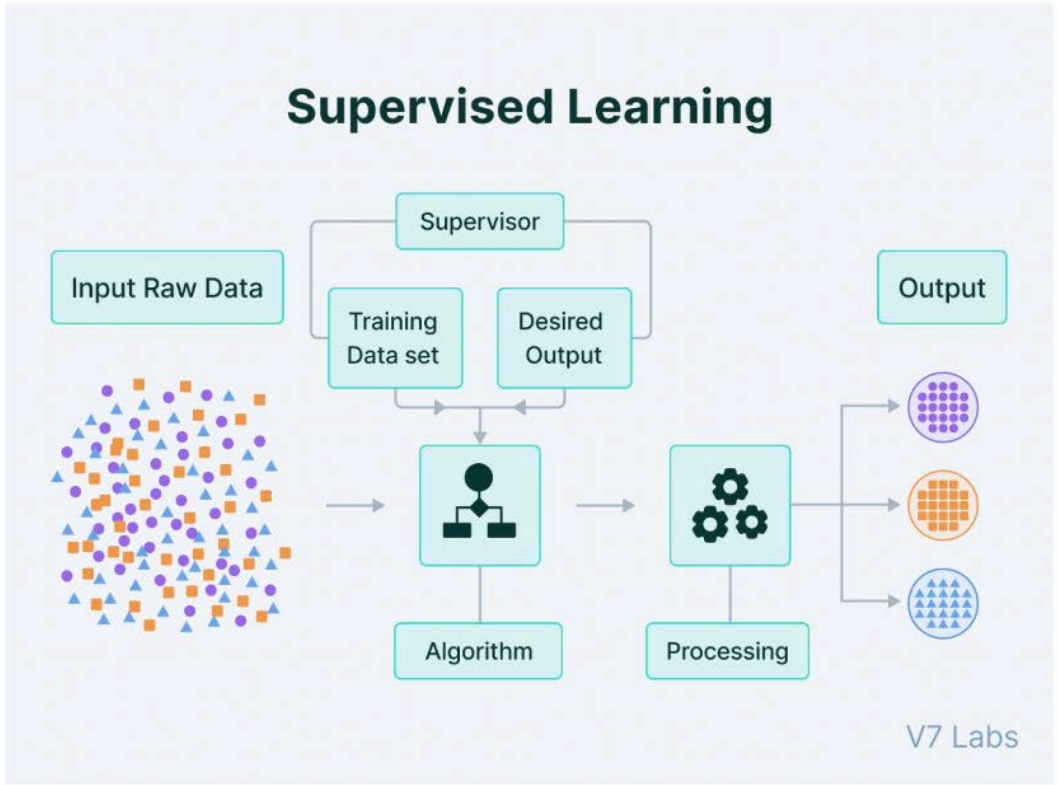
Exploratory Data Analysis



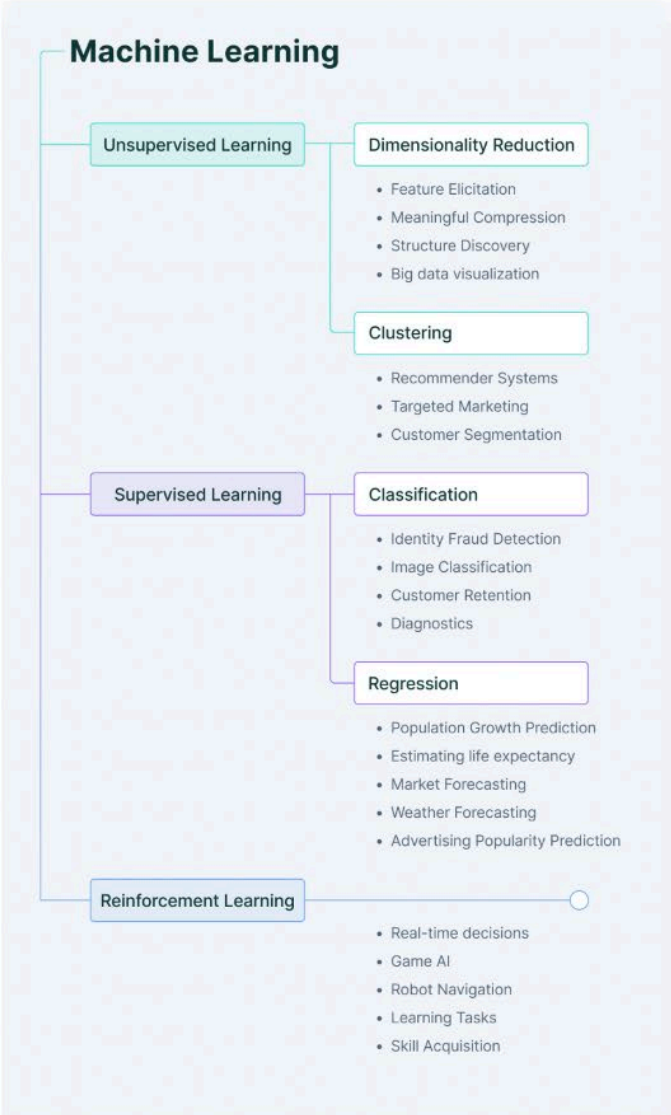
Machine Learning



Data in Supervised and Unsupervised Learning

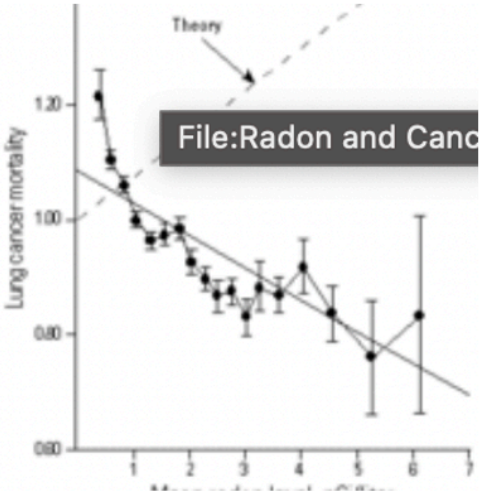


Supervised Learning process

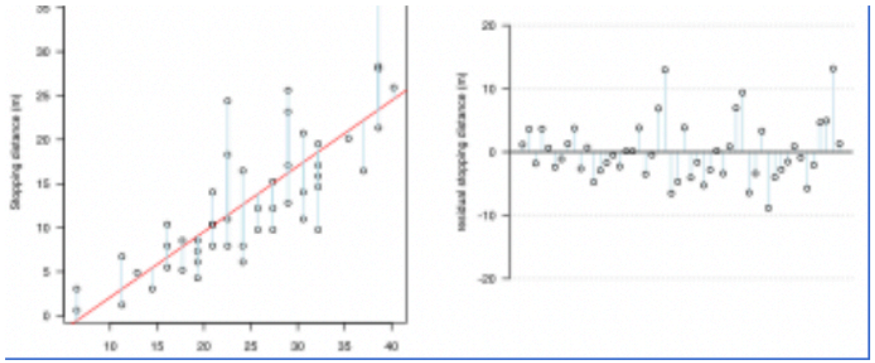


Types of learning in Machine Learning

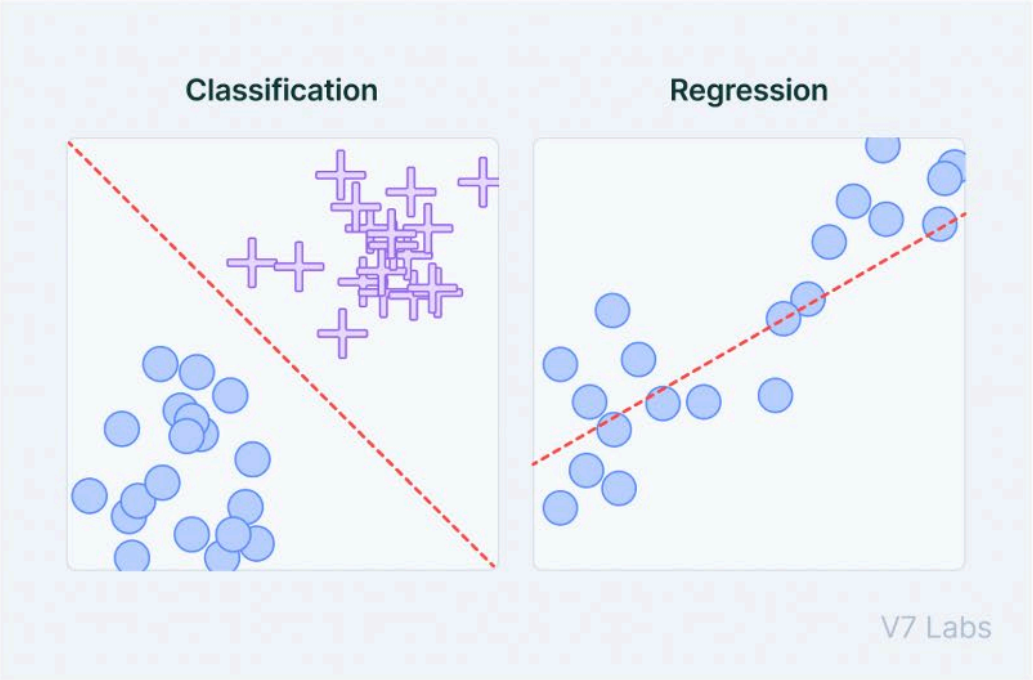
Parametric vs non-Parametric Modelled Data



- Parametric = finite dimensional model eg linear, polynomial
- Non-parametric = distribution-free model, do not assume the functional form of an underlying model eg decision tree



- Generally, linear models are treated as simple
- Non-linear models treated as complex
- Unsupervised models generally more complex than supervised



Some of the most common algorithms in Supervised Learning include Support Vector Machines (SVM), Logistic Regression, Naive Bayes, Neural Networks, K-nearest neighbor (KNN), and Random Forest.

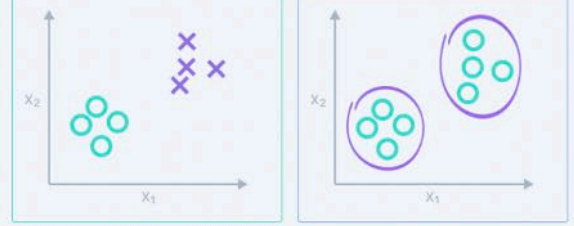
Complexity

Supervised Learning is comparatively less complex than Unsupervised Learning because the output is already known, making the training procedure much more straightforward.

In Unsupervised Learning, on the other hand, we need to work with large unclassified datasets and identify the hidden patterns in the data. The output that we are looking for is not known, which makes the training harder.

Have a look at this comparison table.

Supervised learning	Unsupervised learning
Input data is labeled	Input data is unlabeled
Has a feedback mechanism	Has no feedback mechanism
Data is classified based on the training dataset	Assigns properties of given data to classify it
Divided into Regression & Classification	Divided into Clustering & Association
Used for prediction	Used for analysis
Algorithms include: decision trees, logistic regressions, support vector machine	Algorithms include: k-means clustering, hierarchical clustering, apriori algorithm
A known number of classes	A unknown number of classes



Supervised Learning vs. Unsupervised Learning

TOP PREDICTION ALGORITHMS

TYPE	NAME	DESCRIPTION	ADVANTAGES	DISADVANTAGES
Linear	Linear regression	**The "best fit" line** through all data points. Predictions are numerical.	Easy to understand -- you clearly see what the biggest drivers of the model are.	X Sometimes too simple to capture complex relationships between variables. X Tendency for the model to "overfit".
Linear	Logistic regression	The adaptation of **linear regression** to problems of classification (e.g., yes/no questions, groups, etc.)	Also easy to understand.	X Sometimes too simple to capture complex relationships between variables. X Tendency for the model to "overfit".
Tree-based	Decision tree	A graph that uses a **branching method** to match all possible outcomes of a decision.	Easy to understand and implement.	X Not often used on its own for prediction because it's also often too simple and not powerful enough for complex data.
Tree-based	Random Forest	Takes the average of many decision trees, each of which is made with a sample of the data. Each tree is weaker than a full decision tree, but **by combining them we get better overall performance.**	A sort of "wisdom of the crowd". Tends to result in very high quality models. Fast to train.	X Can be slow to output predictions relative to other algorithms. X Not easy to understand predictions.
Tree-based	Gradient Boosting	Uses even weaker decision trees, that are increasingly **focused on "hard" examples.**	High-performing.	X A small change in the feature set or training set can create radical changes in the model. X Not easy to understand predictions.
Neural networks	Neural networks	Mimics the behavior of the brain. Neural networks are interconnected neurons that pass messages to each other. Deep learning uses several **layers of neural networks put one after the other.**	Can handle extremely complex tasks - no other algorithm comes close in image recognition.	X Very, very slow to train, because they have so many layers. Require a lot of power. X Almost impossible to understand predictions.

Most important Machine Learning methodologies

- Explanatory models
- Ensemble methods
- Clustering
- Dimensionality Reduction
- Similarity

When are they useful?

Explanatory models are useful when you want to understand "why" a decision was made or when you want to understand "how" two or more variables are related to each other.

In practice, the ability to explain what you're machine learning model does is just as important as the performance of the machine learning model itself. If you can't explain *how* a model works, no one will trust it and no one will use it.

Algorithms

Traditional explanatory models based on hypothesis testing:

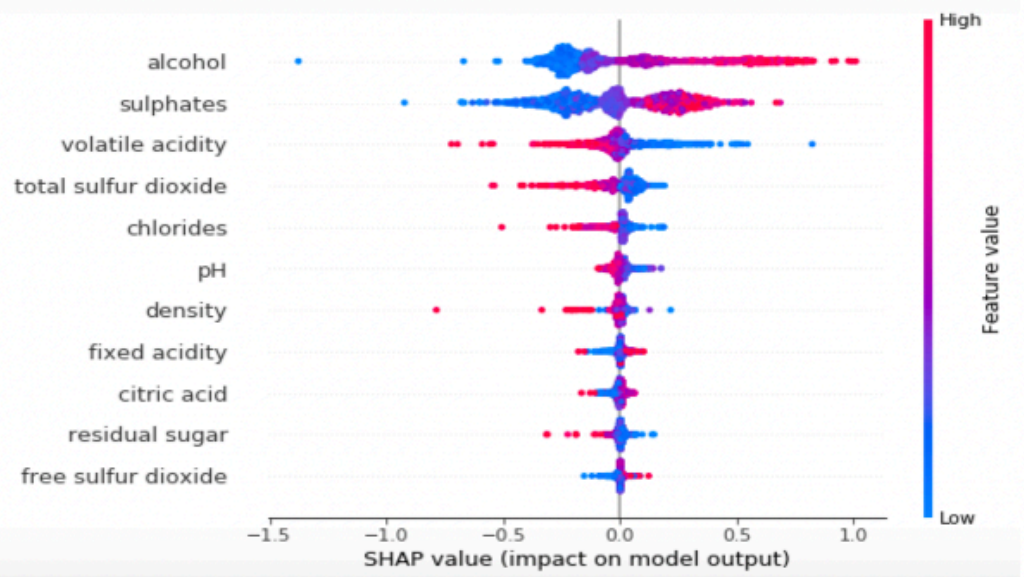
- Linear Regression
- Logistic Regression

Algorithms to explain machine learning models:

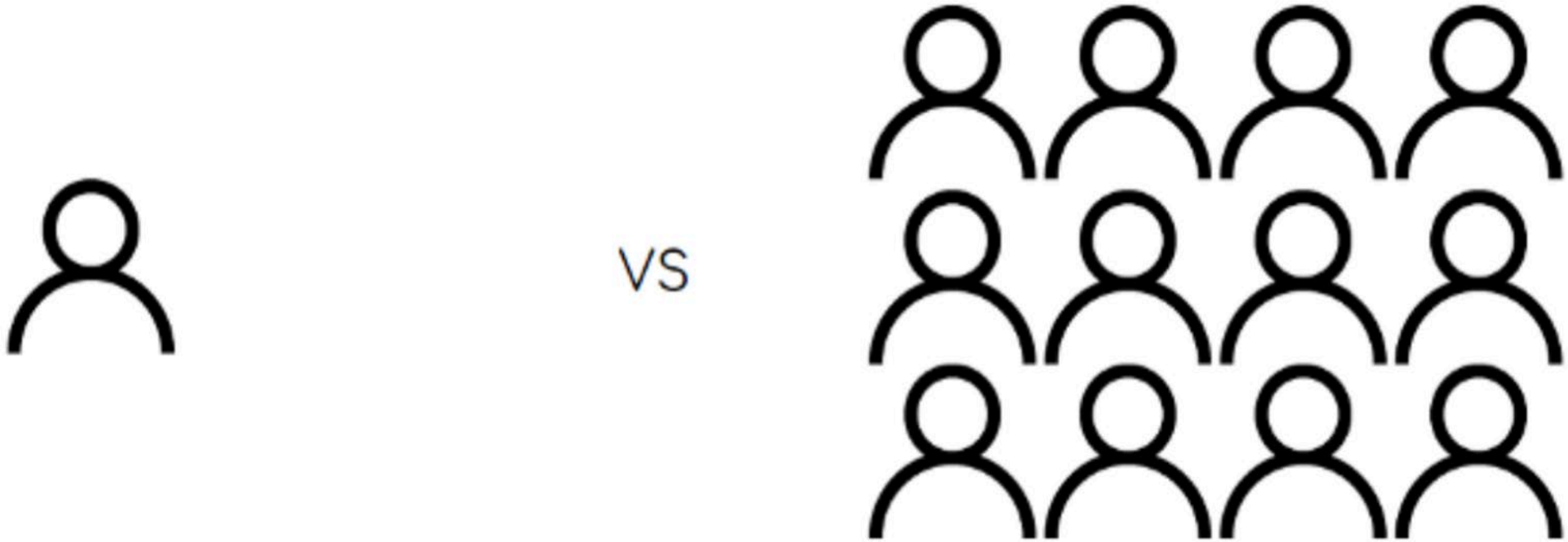
- SHAP
- LIME

Explanatory algorithms allow us to identify and understand variables that have a statistically significant relationship with the outcome. So rather than creating a model to **predict** values of the response variable, we can create explanatory models to **understand** the relationships between the variables in the model.

From a regression standpoint, there's a lot of emphasis on **statistically significant** variables. Why? Almost always, you'll be working with a sample of data, which is a subset of the entire population. In order to make any conclusions about a population given a sample, it's important to ensure that there is enough **significance** to make a confident assumption.



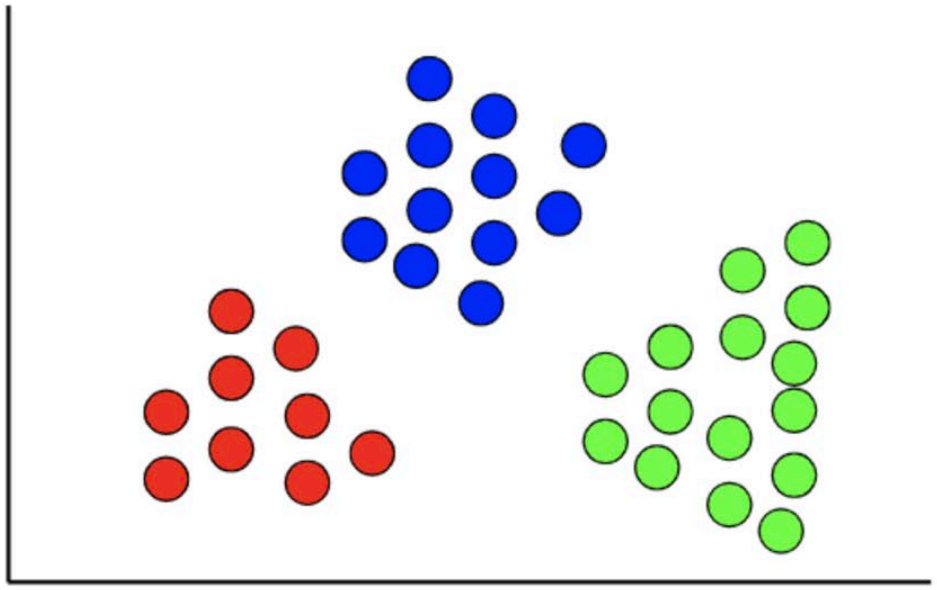
Ensemble Methods

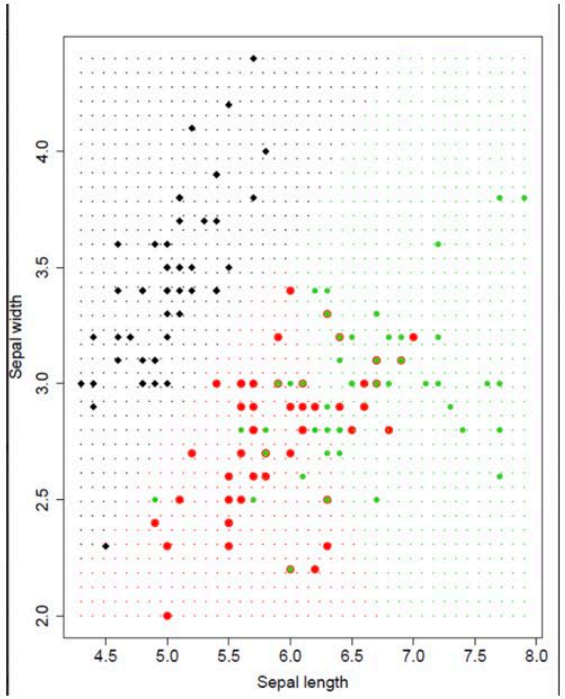


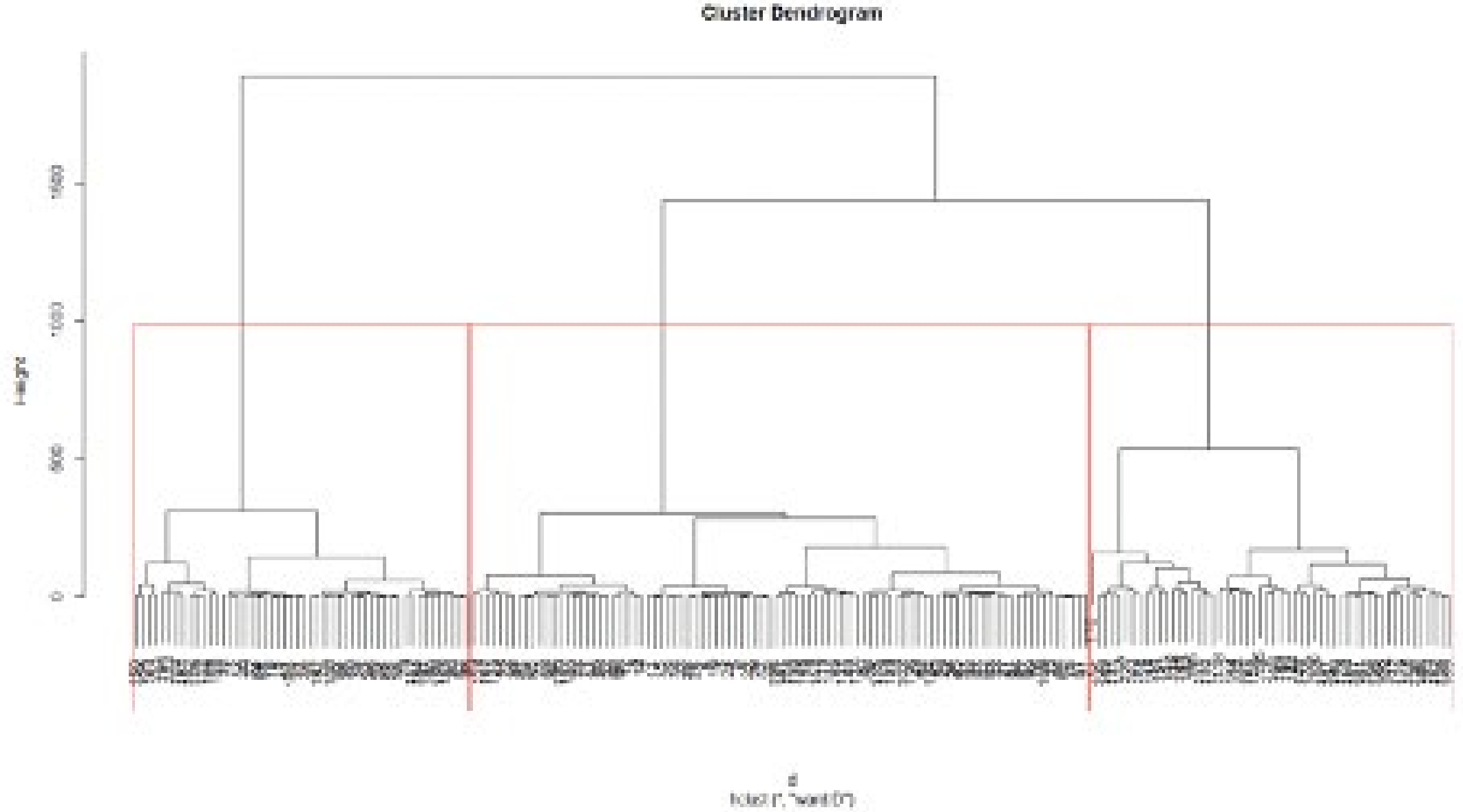
Imagine you had a intractable problem to solve. The chances of solving it if a group of you banded together. Due to their inherent structure, they tend to outperform traditional methods eg Support Vector Machines, Naïve Bayes, Decision Trees. Common ensemble methods include XGBoost, Random Forest, LightGBM, CatBoost

Clustering

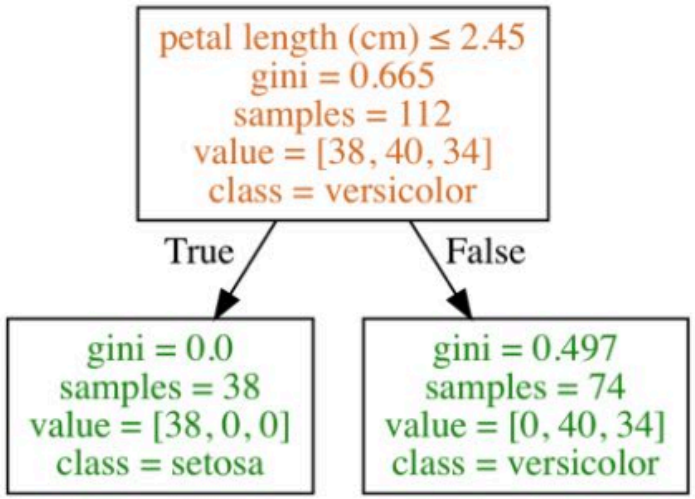
- = unsupervised algorithm
- = discover trends or patterns in data eg segmentation of customers
- = often using during EDA
- = 2 most common types are k-means and hierarchical



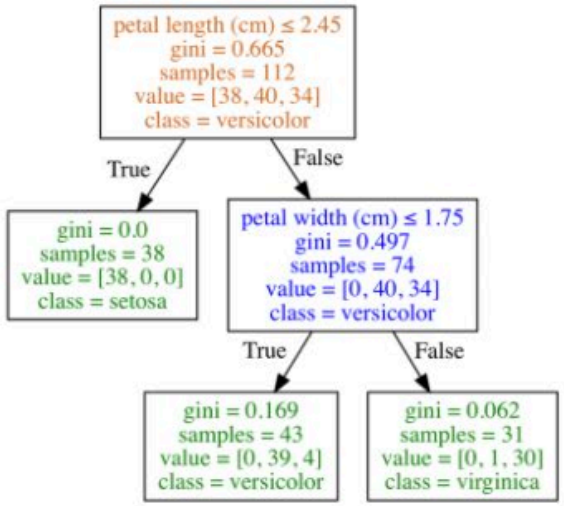



Cluster Dendrogram
Height
0
500
1000
1500
d
hclust (*, "ward.D")

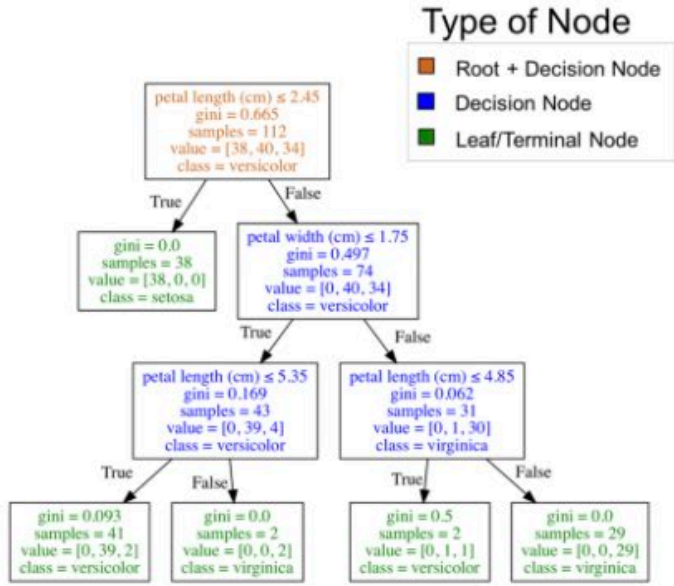
Depth of Classification Trees



Depth = 1



Depth = 2



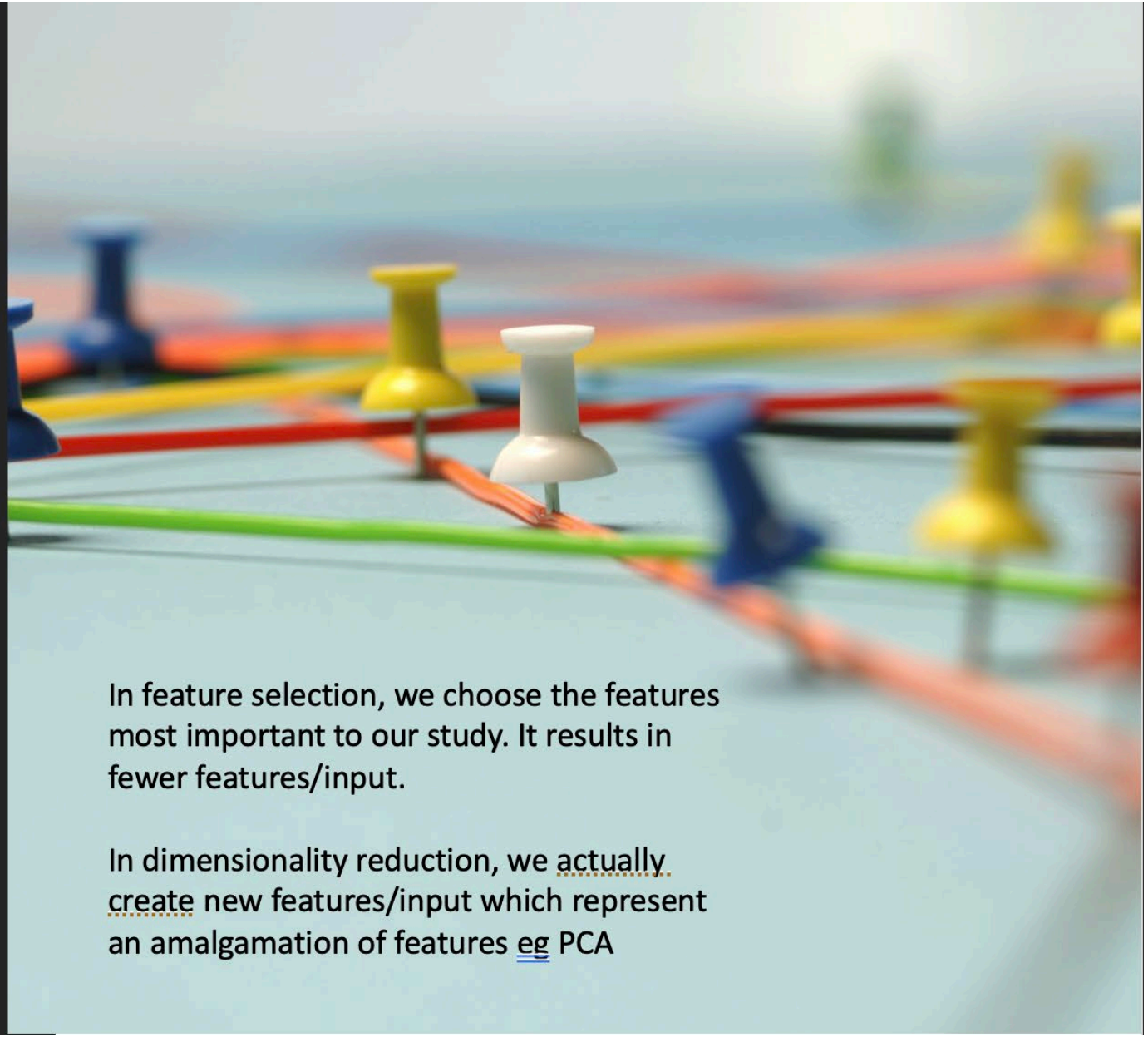
Depth = 3

Classification trees of different depths fit on the IRIS dataset.

High dimensional data can be misleading or even give erroneous results (especially with linear models) and computationally inefficient

Feature Selection ≠ Dimensionality Reduction

Dimensionality reduction is about creating new variables from a linear combination of original variables eg Principal Component Analysis



Below is a non-exhaustive list of some similarity algorithms. If you want to read about more distance algorithms, check out this article. Likewise, if you want to read about more string similarity algorithms, check out this article.

- K nearest neighbors
- Euclidean Distance
- Cosine Similarity
- Levenshtein Algorithm
- Jaro-Winkler Algorithm
- Singular Value Decomposition (SVD) (not exactly a similarity algorithm, but indirectly relates to similarity)

Similarity Algorithms

What are similarity algorithms?

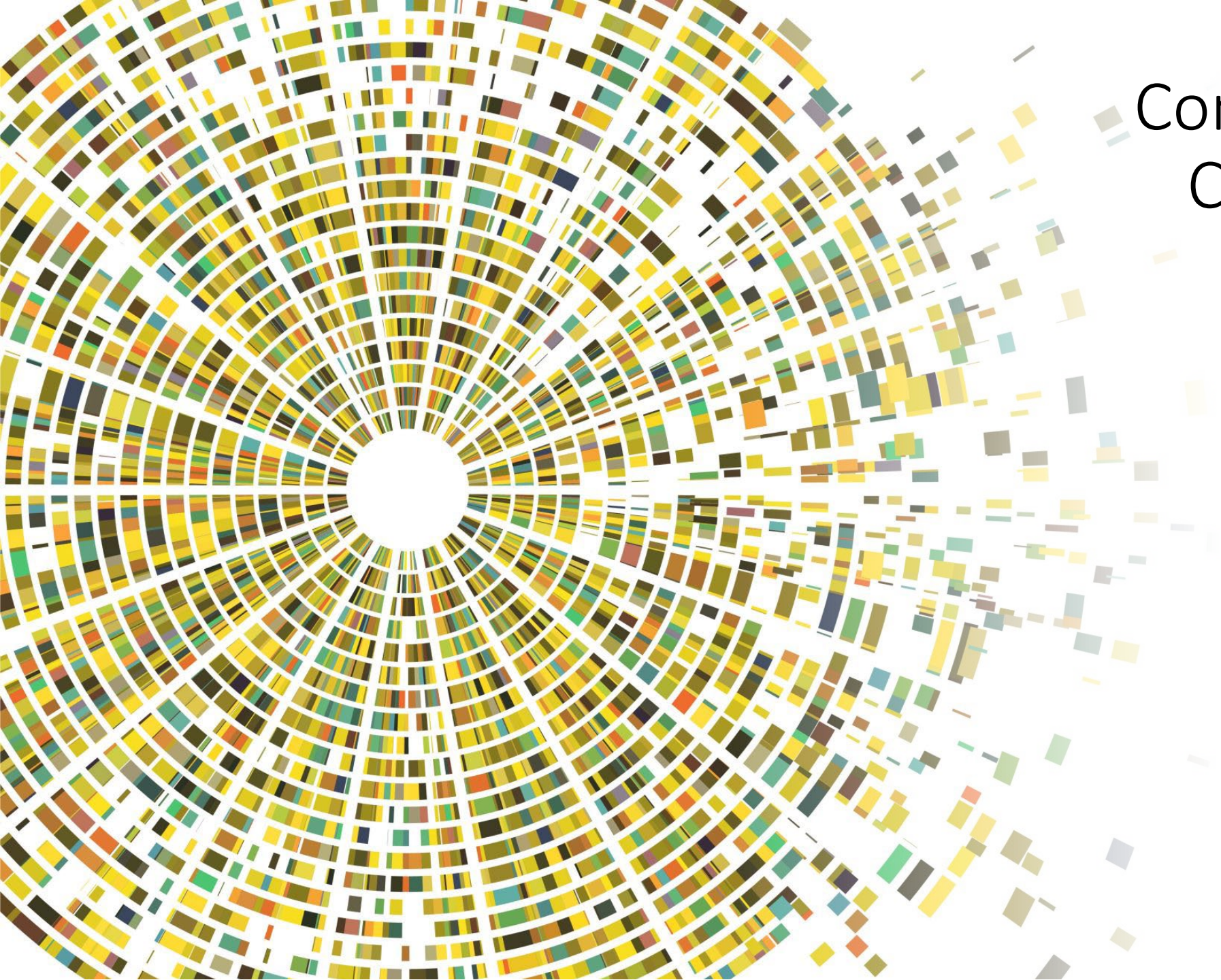
Similarity algorithms are those that compute the *similarity* of pairs of records/nodes/data points/text. There are similarity algorithms that compare the distance between two data points, like Euclidean distance, and there are also similarity algorithms that compute text similarity, like the Levenshtein Algorithm.

When are they useful?

Similarity algorithms can be used in a variety of applications, but they are particularly useful for **recommendation**.

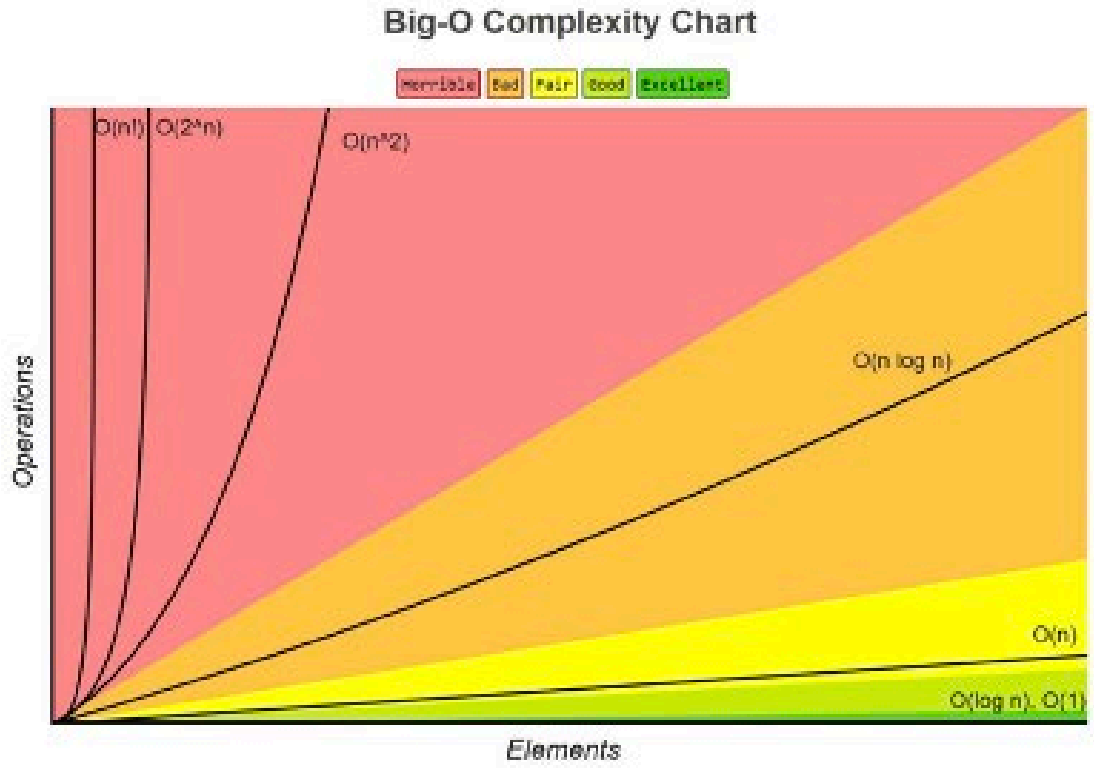
- What articles should Medium recommend to you based on what you previously read?
- What ingredients can you use as a replacement for blueberries?
- What song should Spotify recommend based on what songs you've liked already?
- What products should Amazon recommend based on your order history?

These are just a few of the many examples where similarity algorithms and recommendation is used in our everyday lives.



Computational Complexity

Time Complexity
Big O notation ranks an algorithms' efficiency



Name	Time Complexity
Constant Time	O(1)
Logarithmic Time	O(log n)
Linear Time	O(n)
Quasilinear Time	O(n log n)
Quadratic Time	O(n^2)
Exponential Time	O(2^n)
Factorial Time	O(n!)

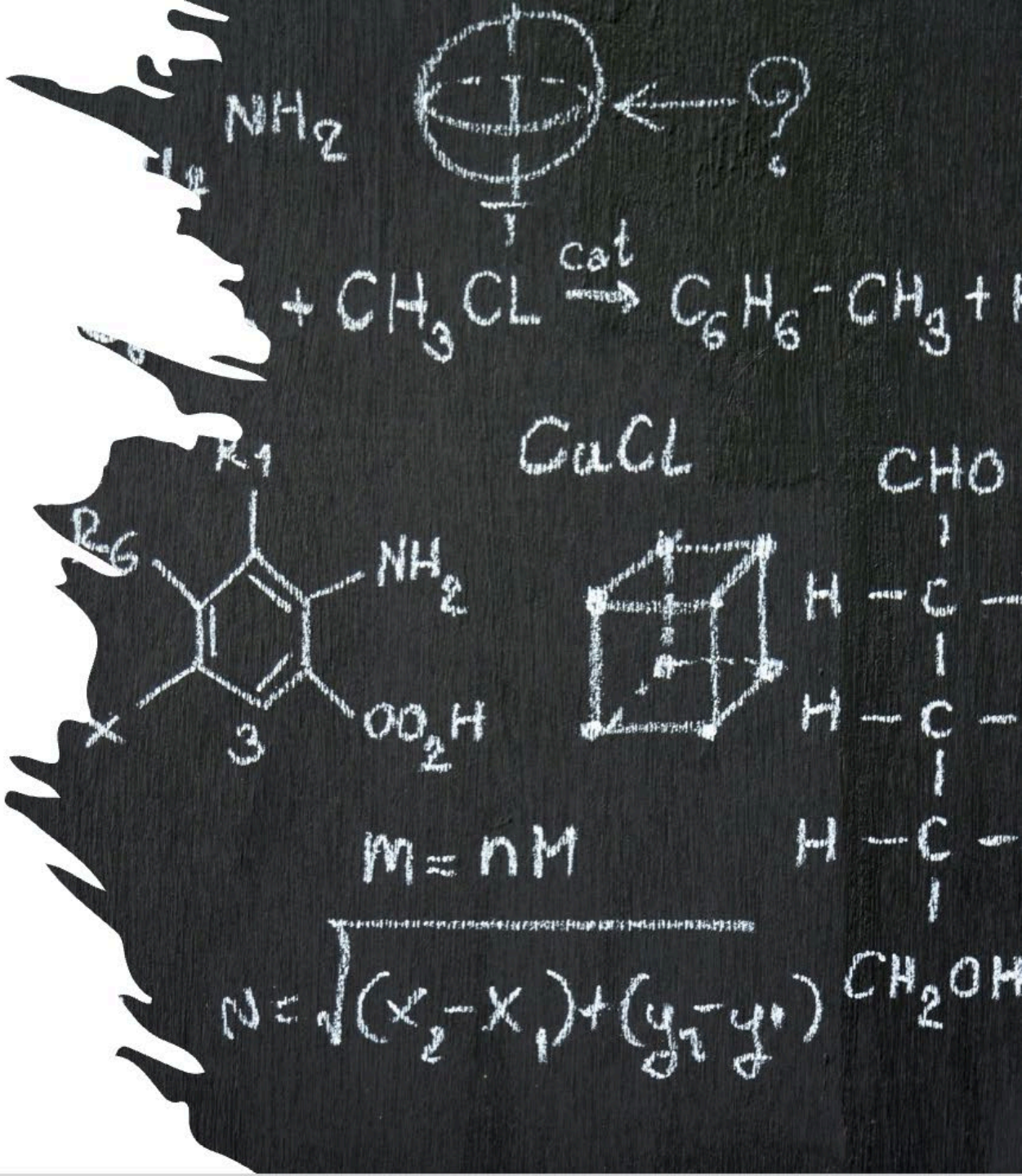
Big O notation ranks an algorithms' efficiency

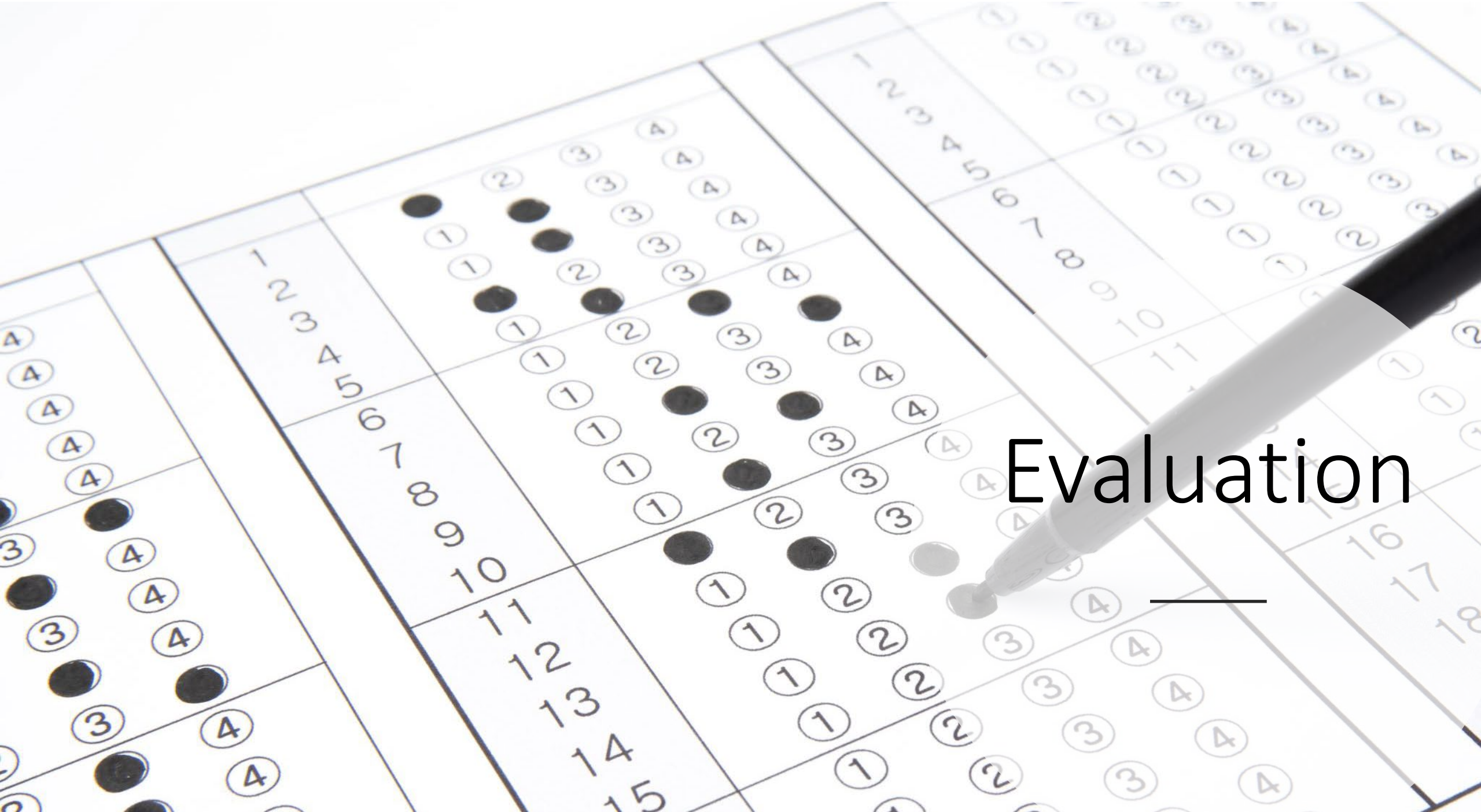
Big O notation gives us the worse case scenario for the algorithm's approximate run time

It does this with regard to "O" and "n", (example: *"O(log n)").*

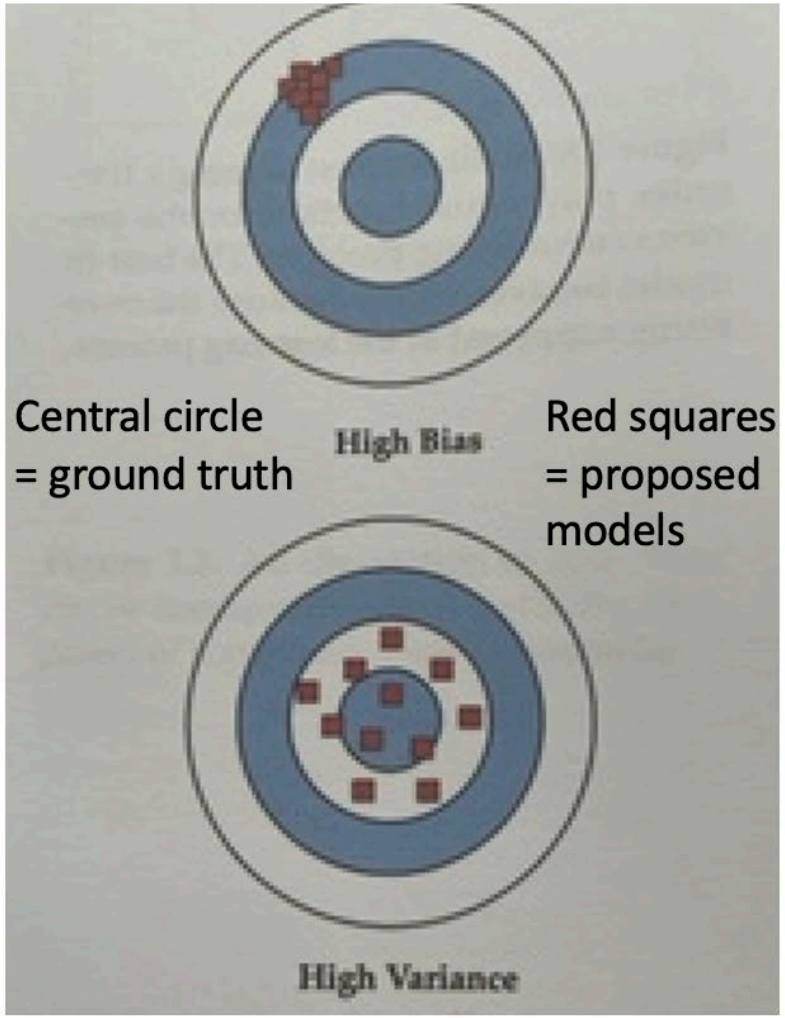
O refers to the order of the function, or its growth rate (computational complexity)

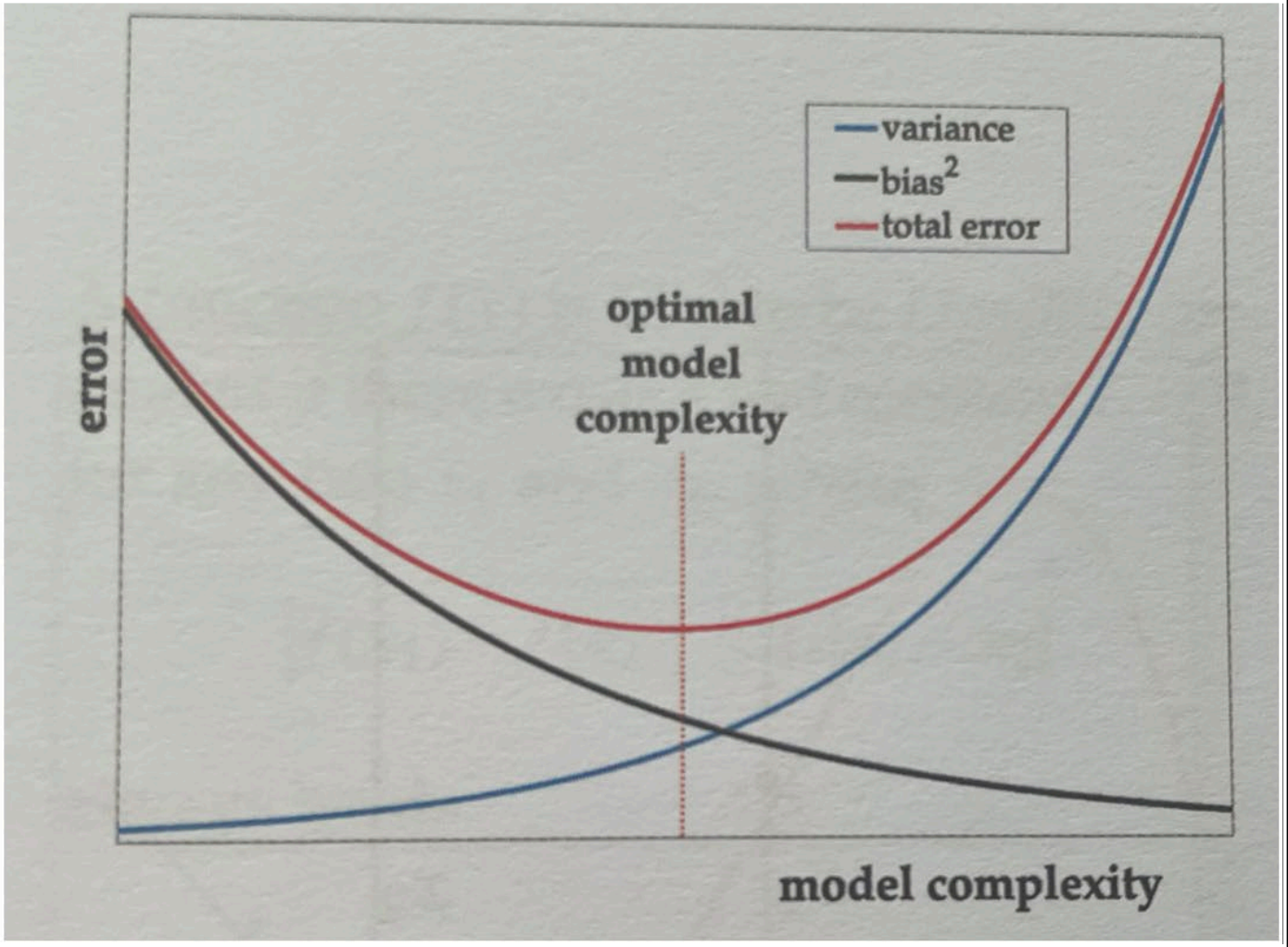
n is the length of a list or array to be sorted (basic level)





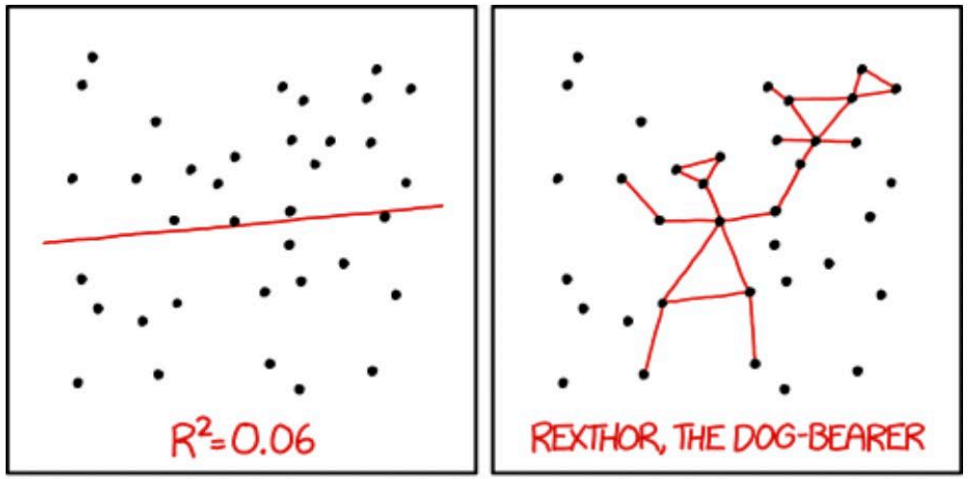
Evaluation





Balance between Bias and Variance = Balance between Overfitting and Underfitting

Too much variance in our model will also cause it to fail. It won't generalize well. Instead of capturing just the pattern we care about, it will also capture a lot of extraneous noise that we don't care about. The patterns in the noise will be different from situation to situation. When we try to generalize and apply or model to a new situation, it will have extra error. This is also called **overfitting**. The more complex our model, the greater the risk of overfitting. This was the case in the connect-the-dots



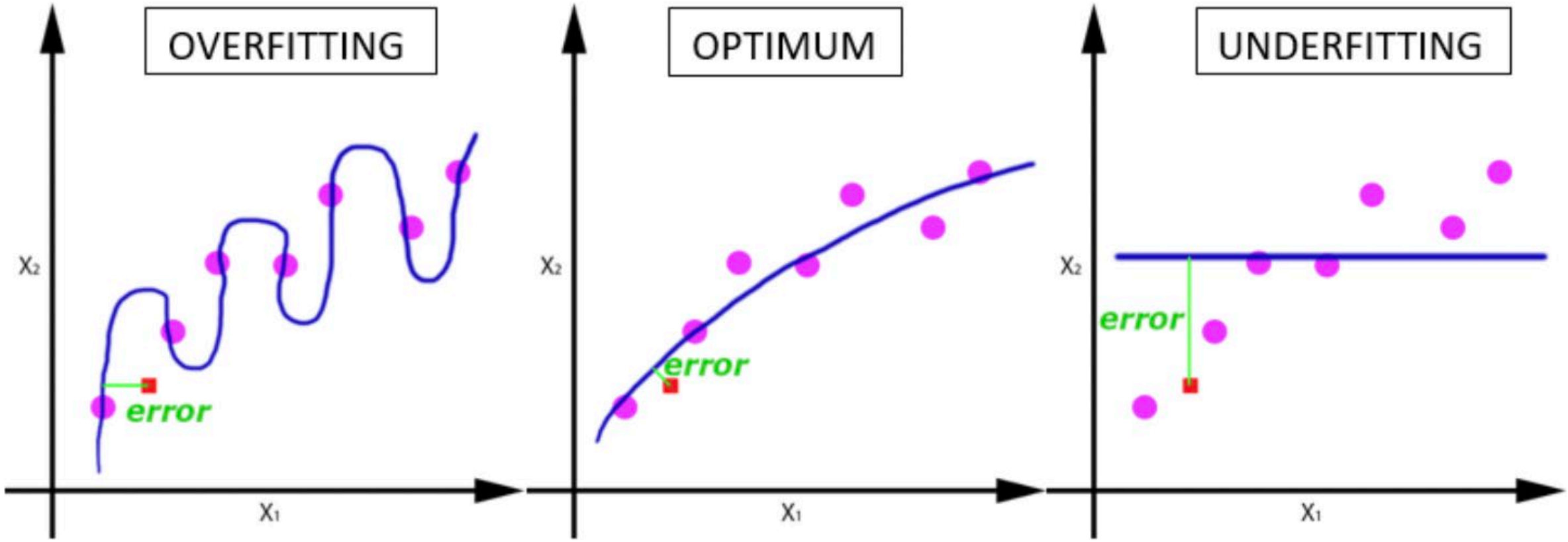
The 95% confidence interval suggests Rexthor's dog could also be a cat, or possibly a teapot.
(https://xkcd.com/1725/)

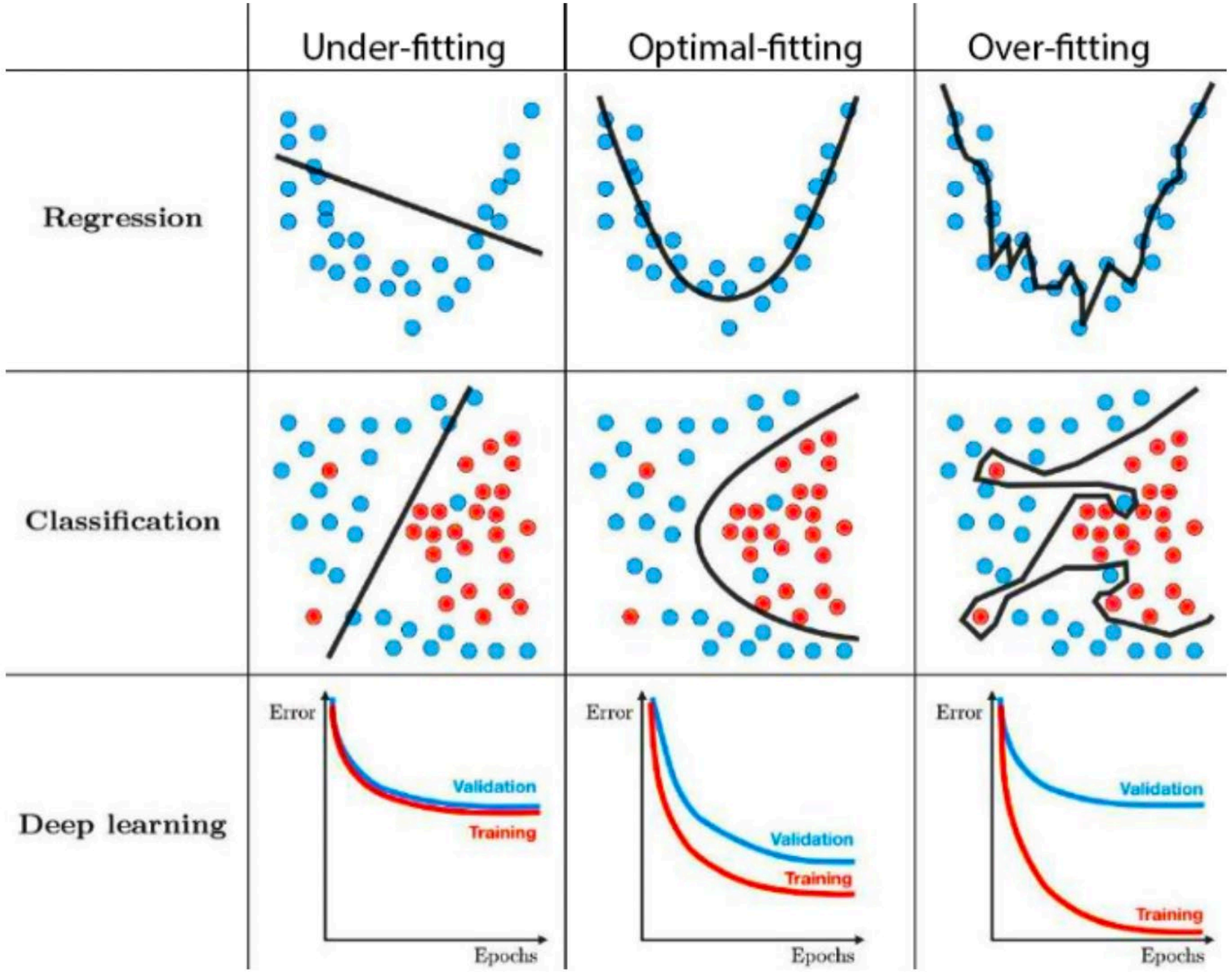
Fix against underfitting = try different models
Fix against overfitting = train/test split

The way to protect against underfitting is to try several different types of models. Each model has its own strengths and weaknesses, patterns that it finds more easily and more accurately, and other patterns that it struggles with. By trying a variety of models, we have the best chance at pulling out the patterns hidden in our data. Our best defense against bias is a rich pool of candidate models.

To protect against overfitting, we make sure to test how well each trained model generalizes. After training it on one set of observations, we then use it to predict the pattern in another set of observations. If the predictions are accurate, then we know our model is good, and our variance is low. If it makes much worse predictions on the test data set then on the training data set, then we know that the model overfitted the training data, that it captured a lot of noise, rather than just signal.

Goodness of Fit





Source: Underfitting, Optimal-fitting and Overfitting for linear regression [1]

A Good Fit in Machine Learning

Ideally, you want to select a model at the sweet spot between underfitting and overfitting.

This is the goal, but is very difficult to do in practice.

To understand this goal, we can look at the performance of a machine learning algorithm over time as it is learning a training data. We can plot both the skill on the training data and the skill on a test dataset we have held back from the training process.

Then we need to evaluate the model:-

Confusion Matrix

Predicted Values	Actual Values: Positive (1)	Actual Values: Negative (0)
Positive (1)	TP	FP
Negative (0)	FN	TN

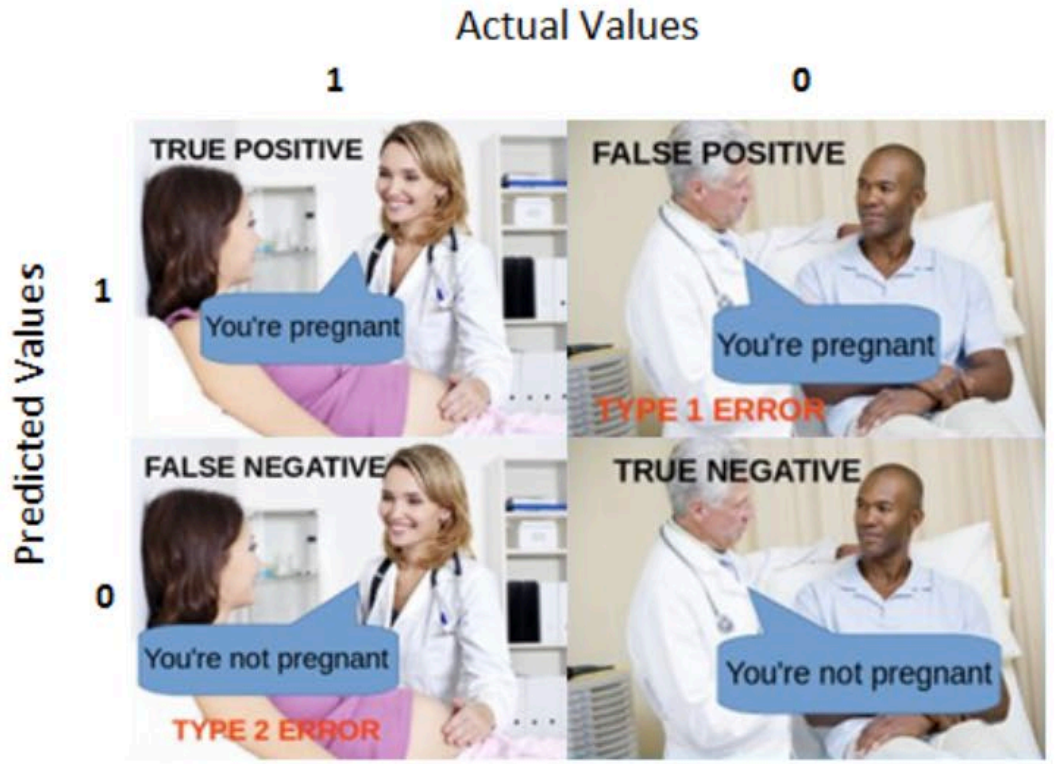


Figure 1 : Confusion Matrix

Predicted class

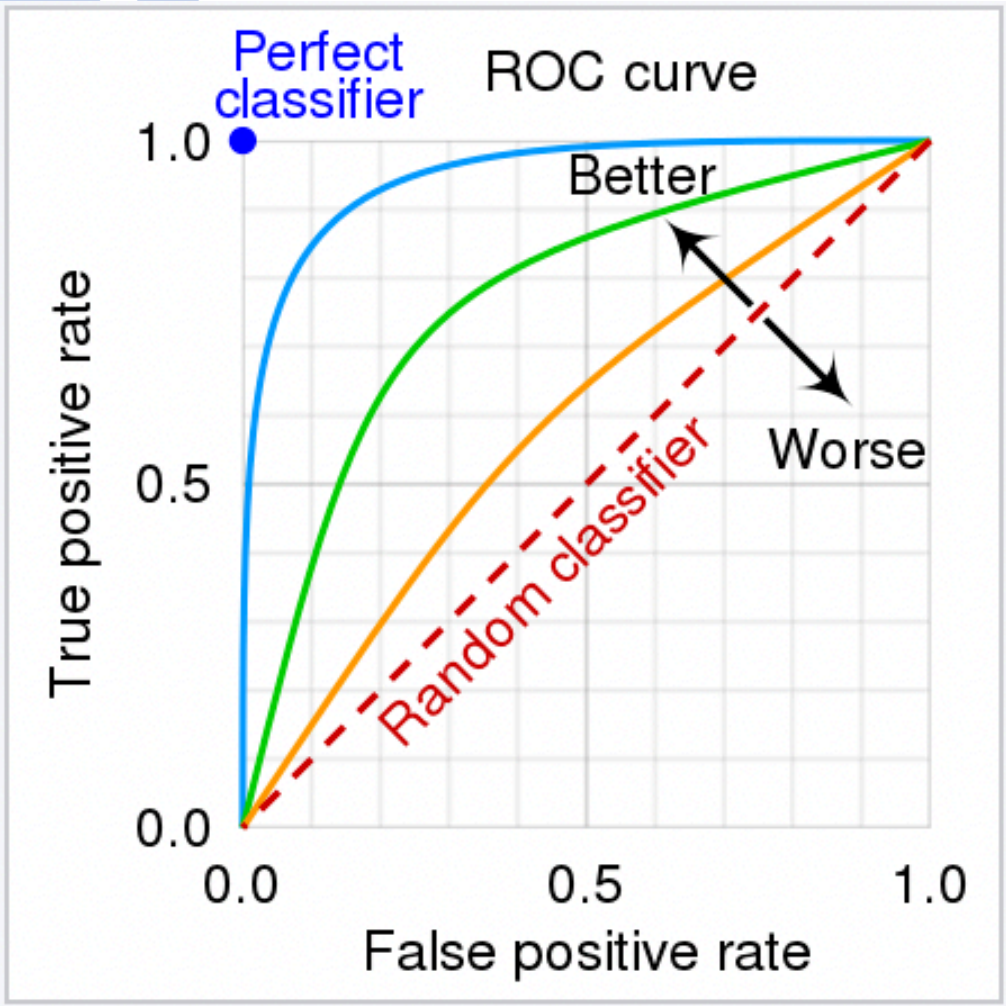
Actual Class	Positive	Negative	
Positive	**True Positive (TP)**	**False Negative (FN)** **Type II Error**	Sensitivity (i.e. Recall) $\frac{TP}{TP+FN}$
Negative	**False Positive (FP)** **Type I Error**	**True Negative (TN)**	Specificity $\frac{TN}{TN+FP}$
	Precision $\frac{TP}{TP+FP}$	Negative Predicted Value $\frac{TN}{TN+FN}$	Accuracy $\frac{TP+TN}{TP+FP+TN+FN}$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

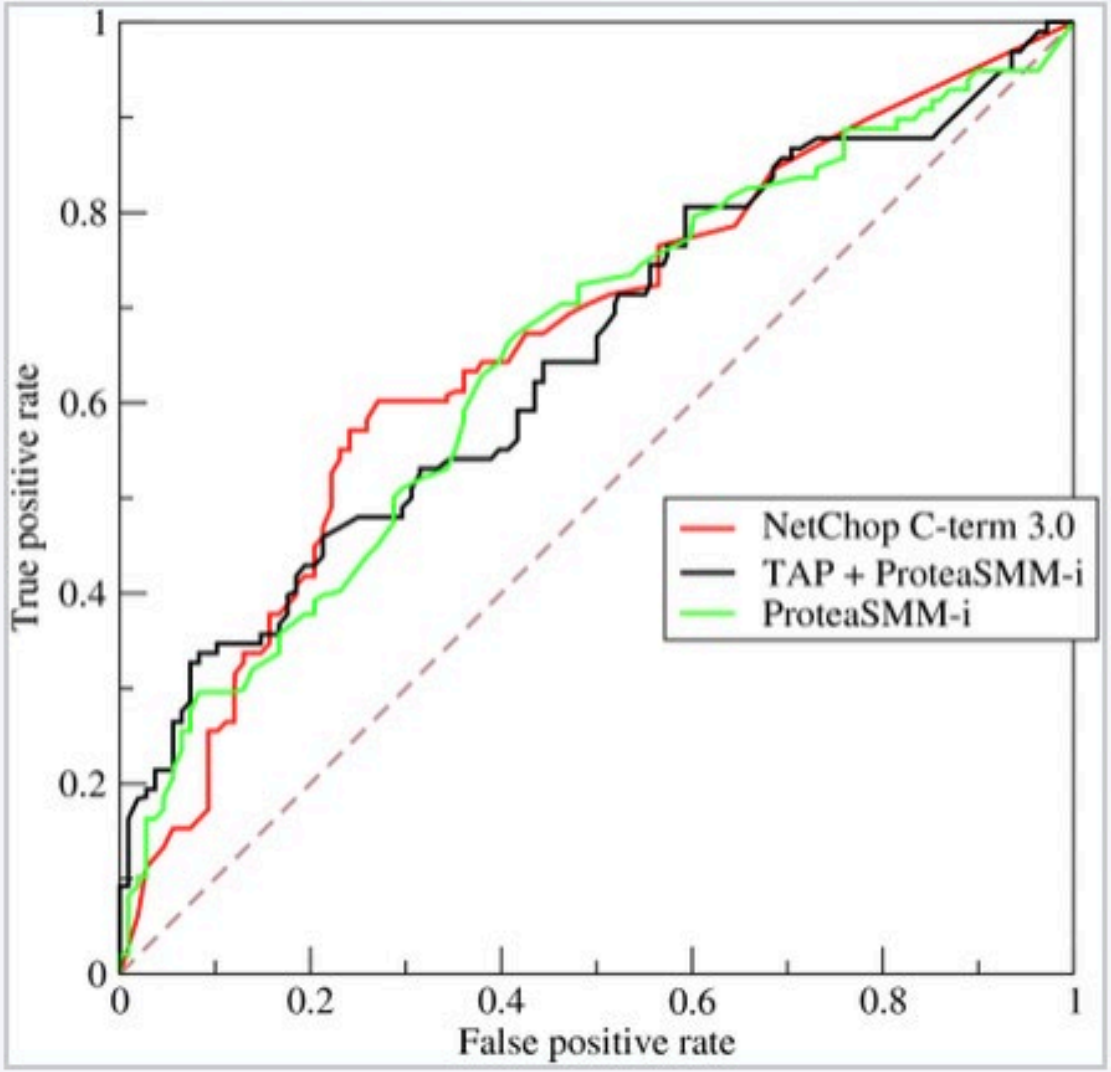
Having an imbalanced dataset can lead to a large number of True Negatives where the majority class is predicted correctly. Thus, the F1- Score is used to find the harmonic mean between Precision and Recall metrics to determine the model's overall accuracy. It is represented with the following equation:

$$F1 = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall}\right)$$

ROC Curve

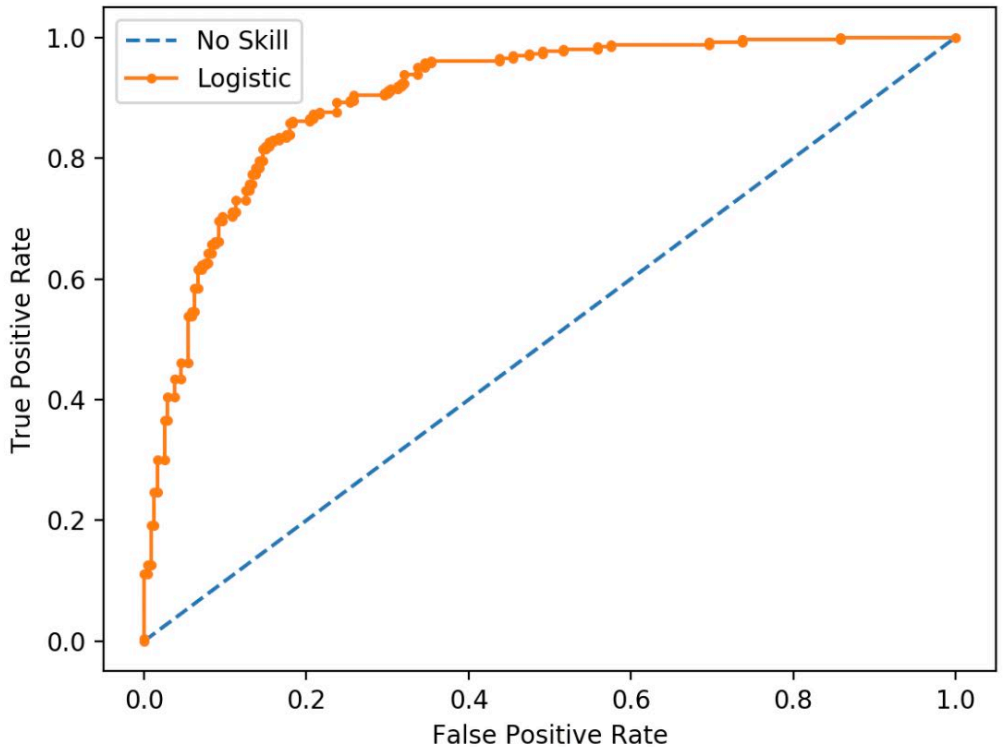


The ROC space for a "better" and "worse" classifier.

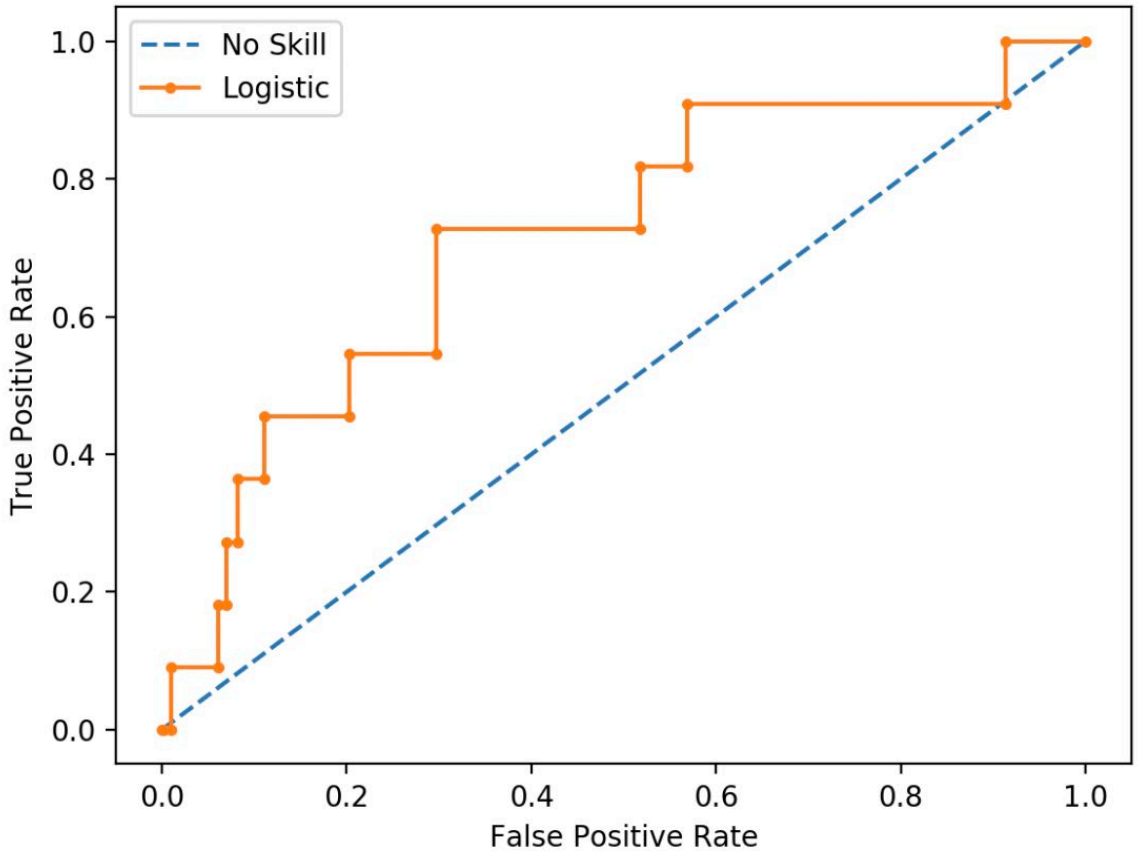


ROC curve of three predictors of peptide cleaving in the proteasome.

ROC Curve

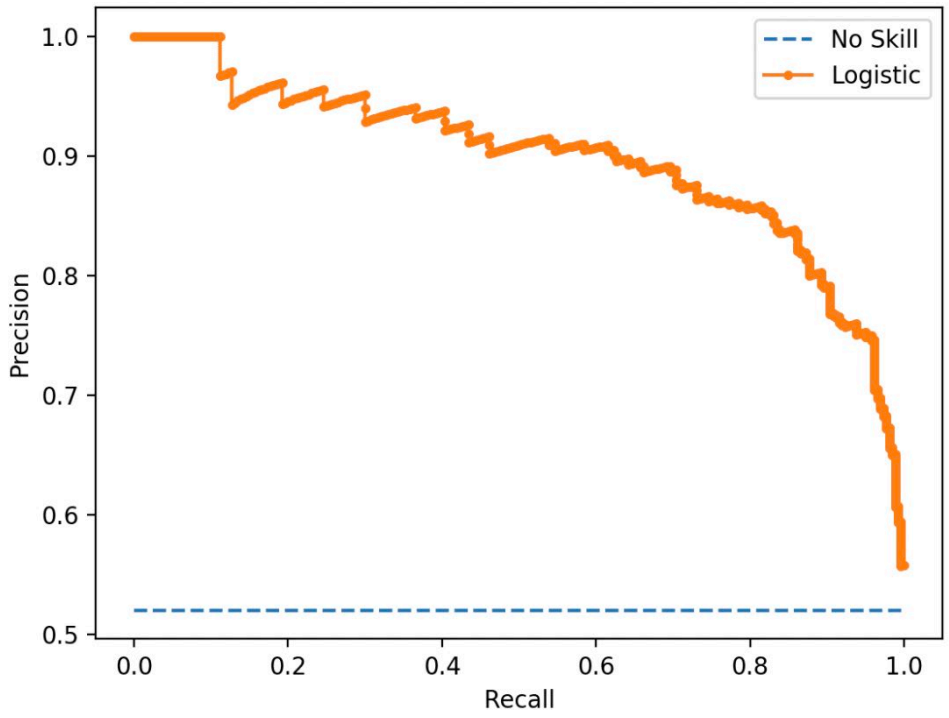


ROC Curve Plot for a No Skill Classifier and a Logistic Regression Model

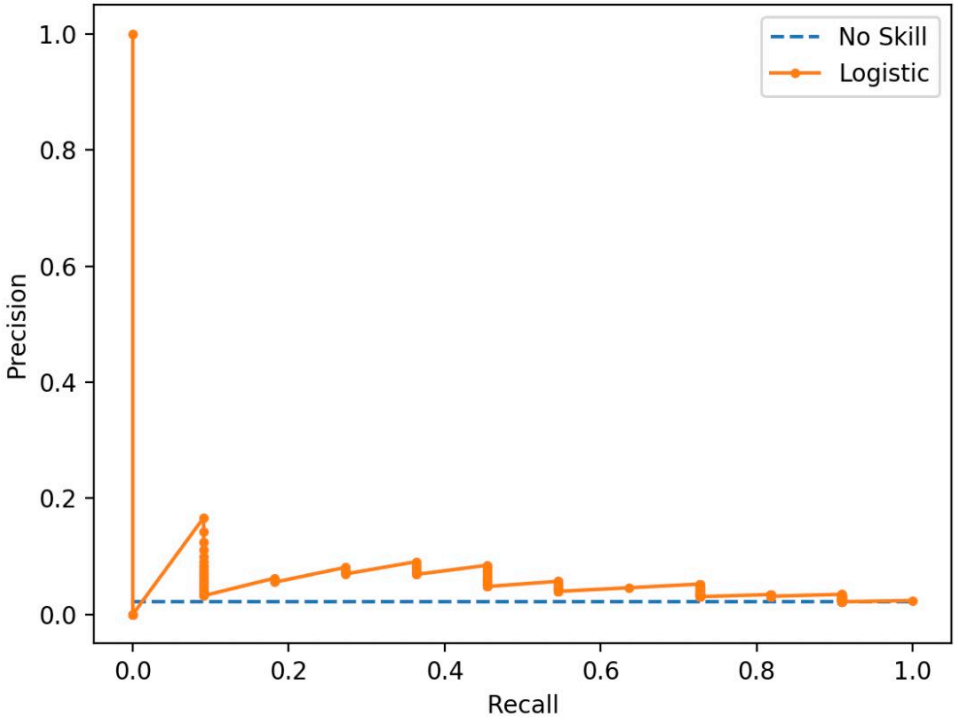


ROC Curve Plot for a No Skill Classifier and a Logistic Regression Model for an Imbalanced Dataset

PRC Curve



Precision-Recall Plot for a No Skill Classifier and a Logistic Regression Model



Precision-Recall Plot for a No Skill Classifier and a Logistic Regression Model for am Imbalanced Dataset

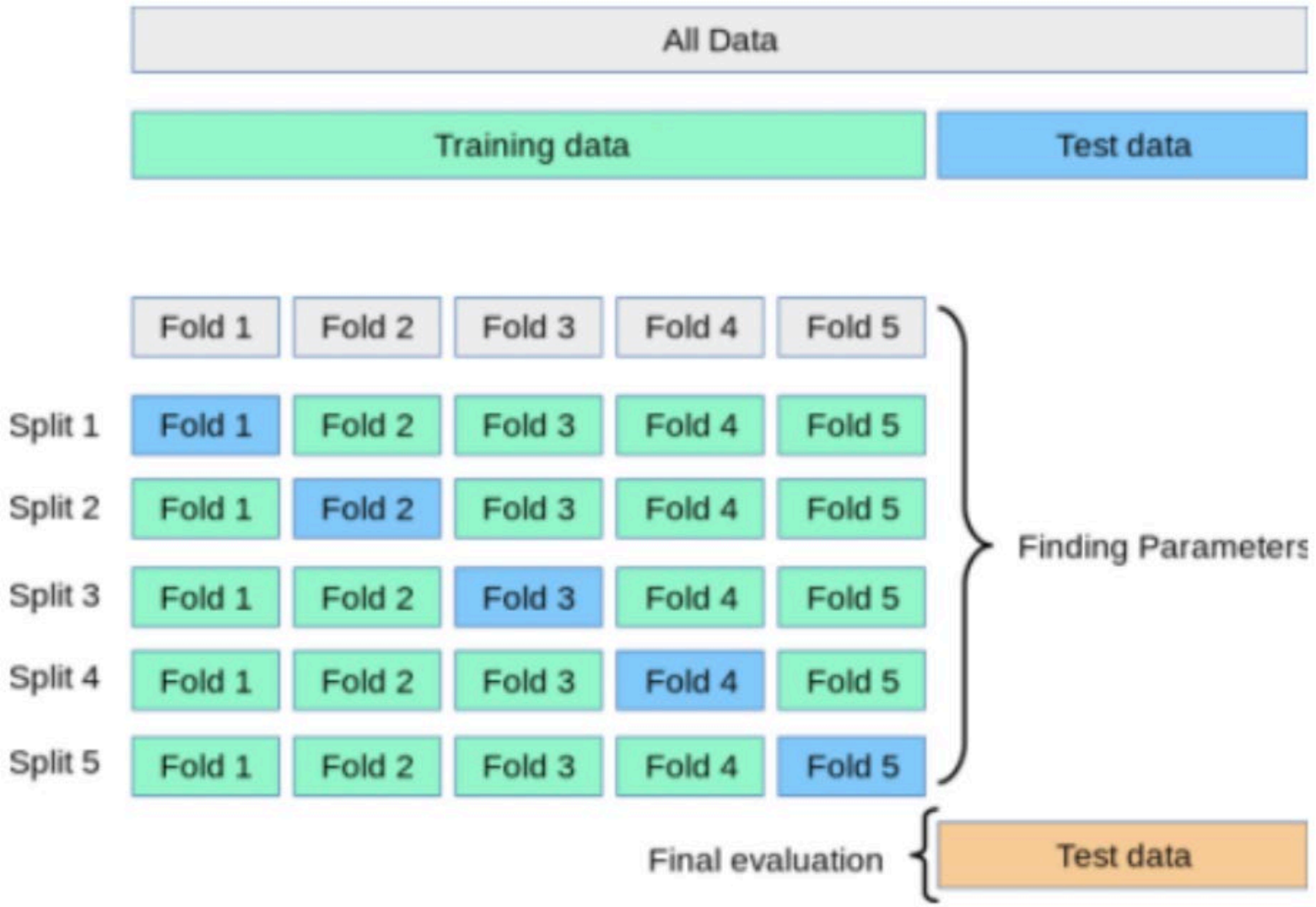
Validation

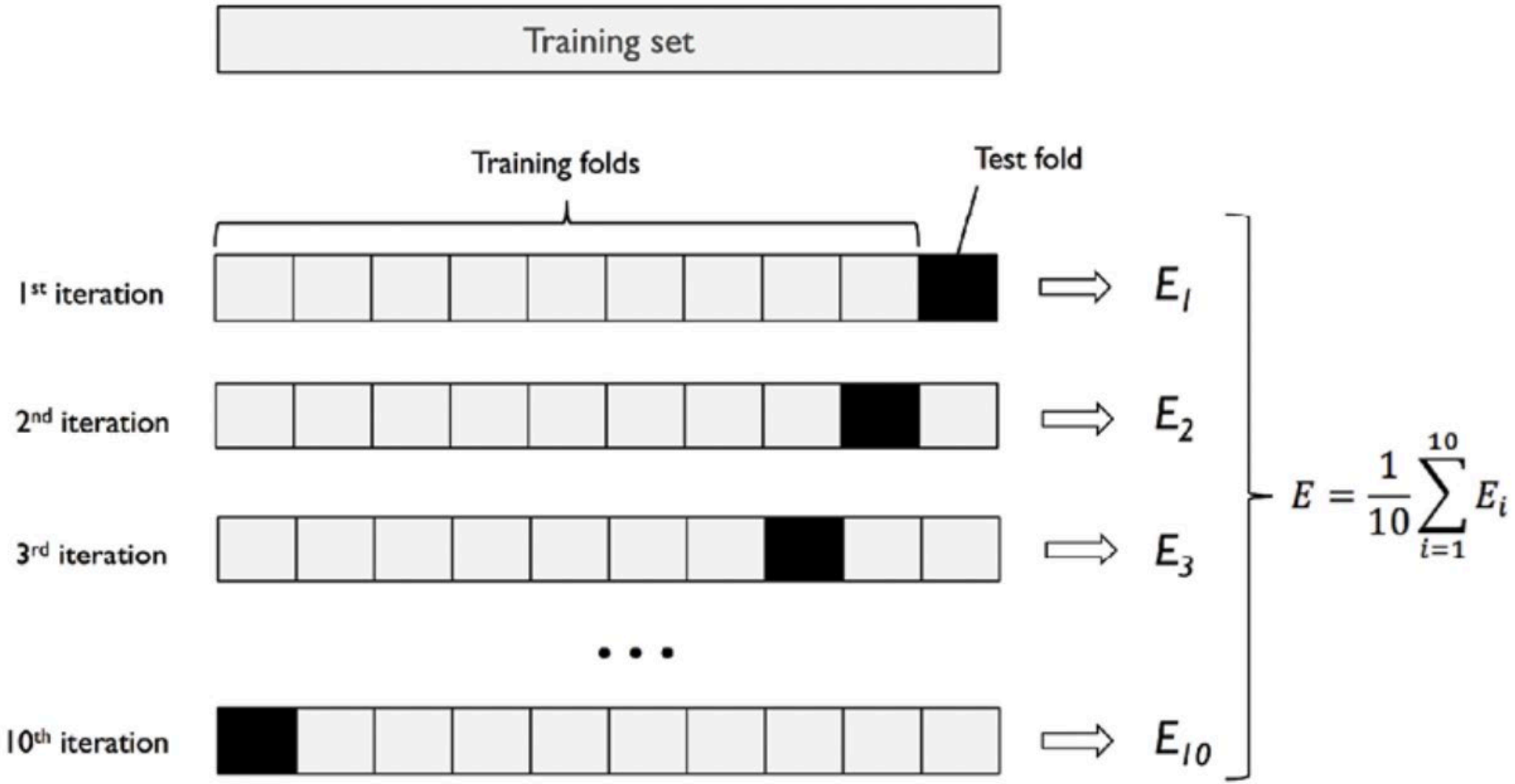
Then we can validate the model with Cross-validation:-

K-fold Cross-validation

1. The dataset is split into training and test dataset.
2. The training dataset is then split into K-folds.
3. Out of the K-folds, (K-1) fold is used for training
4. 1 fold is used for validation
5. The model with specific hyperparameters is trained with training data (K-1 folds) and validation data as 1 fold. The performance of the model is recorded.
6. The above steps (step 3, step 4 and step 5) is repeated until each of the k-fold got used for validation purpose. This is why it is called k-fold cross validation.

K-fold cross-validation




Training set
Training folds
Test fold
1st iteration
2nd iteration
3rd iteration
10th iteration
E_1
E_2
E_3
E_{10}
$E = \frac{1}{10}\sum_{i=1}^{10} E_i$

Unbalanced Data

Imbalanced data

More such example of imbalanced data is –

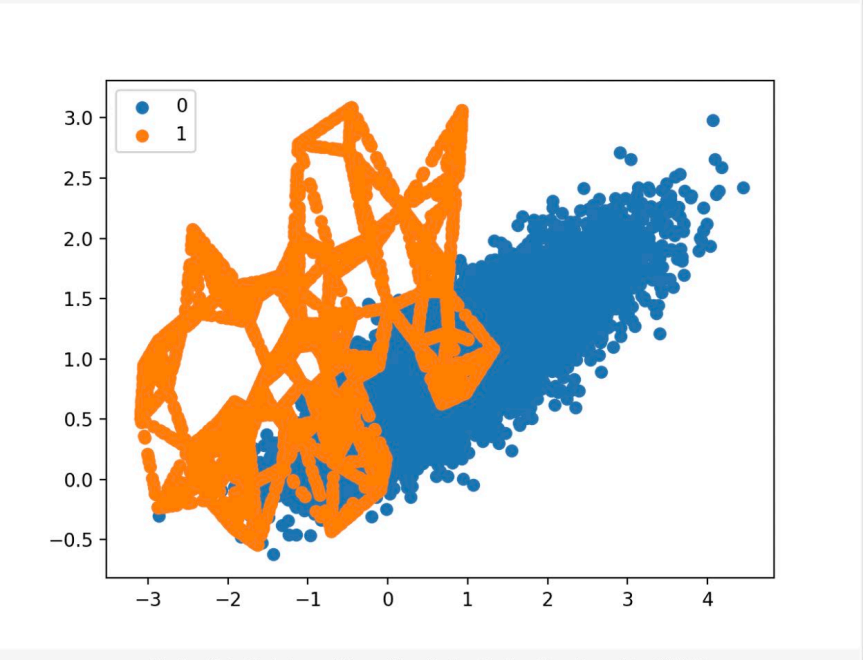
· Disease diagnosis

· Customer churn prediction

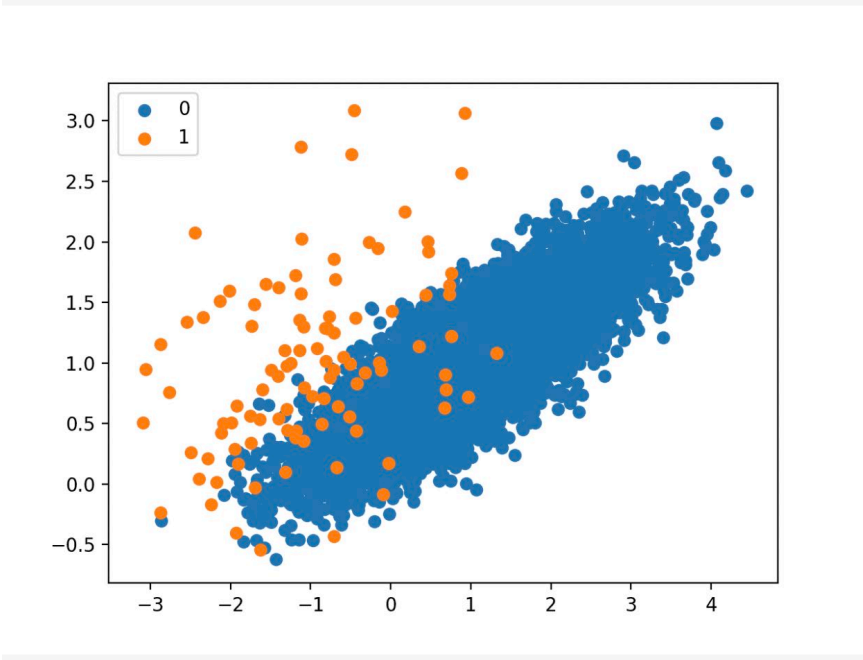
· Fraud detection

· Natural disaster

Undersampling, oversampling or hybrid model?

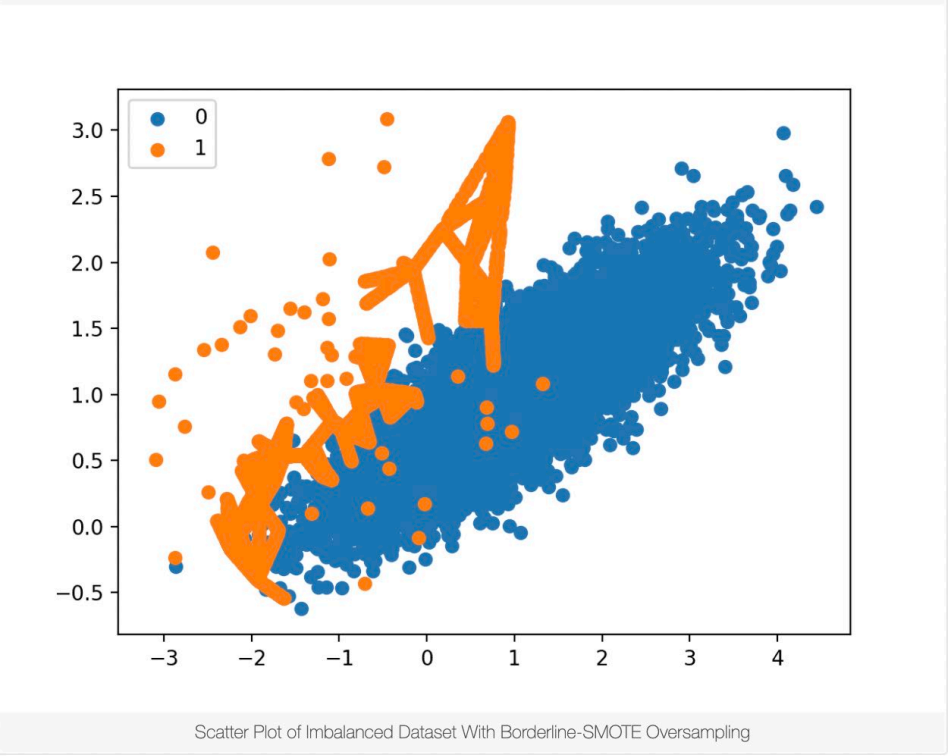


Scatter Plot of Imbalanced Binary Classification Problem Transformed by SMOTE

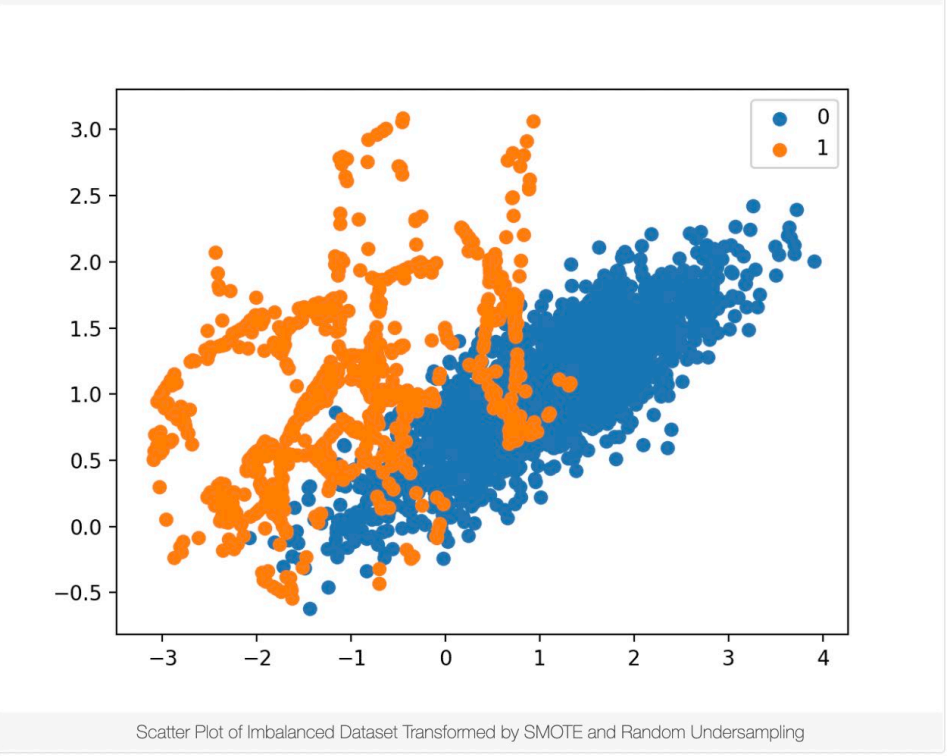


Scatter Plot of Imbalanced Binary Classification Problem

SMOTE: Synthetic Minority Oversampling Technique

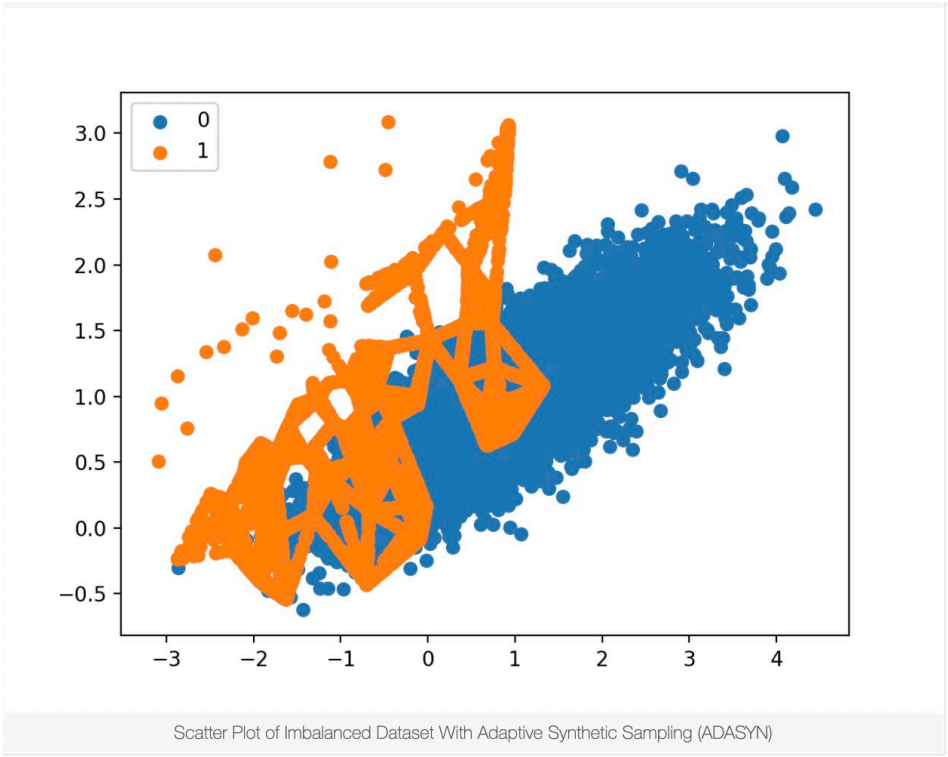


Scatter Plot of Imbalanced Dataset With Borderline-SMOTE Oversampling

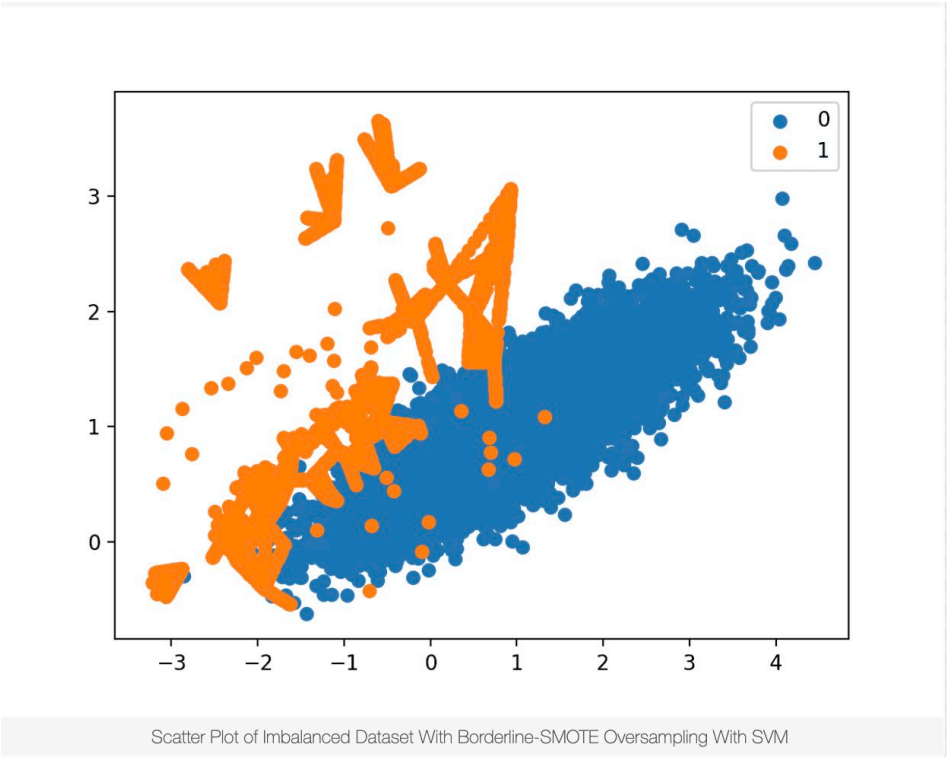


Scatter Plot of Imbalanced Dataset Transformed by SMOTE and Random Undersampling

SMOTE: Synthetic Minority Oversampling Technique



Scatter Plot of Imbalanced Dataset With Adaptive Synthetic Sampling (ADASYN)



Scatter Plot of Imbalanced Dataset With Borderline-SMOTE Oversampling With SVM

Cost Sensitive Learning

Cost-sensitive Learning

To make this concrete, we can consider a wide range of other ways we might wish to consider or measure cost when training a model on a dataset. For example, Peter Turney lists nine types of costs that might be considered in machine learning in his 2000 paper titled "Types of Cost in Inductive Concept Learning."

In summary, they are:

- Cost of misclassification errors (or prediction errors more generally).
- Cost of tests or evaluation.
- Cost of teacher or labeling.
- Cost of intervention or changing the system from which observations are drawn.
- Cost of unwanted achievements or outcomes from intervening.
- Cost of computation or computational complexity.
- Cost of cases or data collection.
- Cost of human-computer interaction or framing the problem and using software to fit and use a model.
- Cost of instability or variance known as concept drift.

Although critical to many real-world problems, the idea of costs and cost-sensitive learning is a new topic that was largely ignored up until recently.

> *In real-world applications of concept learning, there are many different types of cost involved. The majority of the machine learning literature ignores all types of cost …*