Introduction to Supervised learning using scikit-learn

MAIN 2022 | July 2022

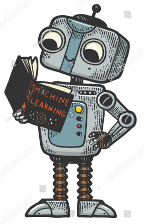
By

Nikhil Bhagwat

McGill UNIVERSITY

ORIGAMI Lab

neuro
Montreal Neurological Institute-Hospital



Objectives

- Define machine-learning nomenclature
- Describe basics of the “learning” process
- Explain model design choices and performance trade-offs
- Introduce model selection and validation frameworks
- Explain model performance metrics

Pop Quiz

Say, currently we have a population with 1% covid prevalence. We train a simple machine-learning model to identify COVID patients using their biometry.

Our model is 91% accurate! Then we also calculate,

- 90% sensitivity (i.e. probability that prediction is positive if patient has COVID)
- 91% specificity (i.e. probability that prediction is negative if patient doesn't have COVID)

What are my chances that I have COVID, if my test is positive?

A) 9 in 10 B) 1 in 2 C) 1 in 10 D) 1 in 100

Pop Quiz

Say, currently we have a population with 1% covid prevalence. We train a simple machine-learning model to identify COVID patients using their biometry.

Our model is 91% accurate! Then we also calculate,

- 90% sensitivity (i.e. probability that prediction is positive if patient has COVID)
- 91% specificity (i.e. probability that prediction is negative if patient doesn't have COVID)

What are my chances that I have COVID, if my test is positive?

A) 9 in 10 B) 1 in 2 C) 1 in 10 D) 1 in 100

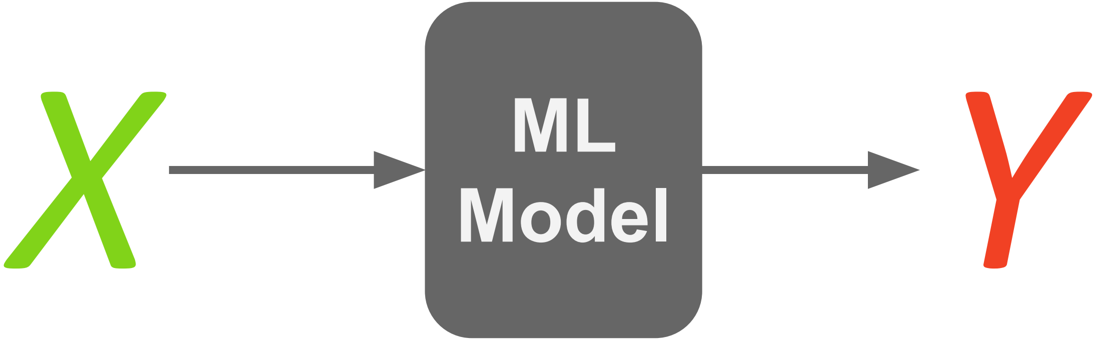
Later we train a fancy deep learning model to identify COVID patients using their chest CT! This model has accuracy of 99%! We calculate

- 80% sensitivity
- 99% specificity

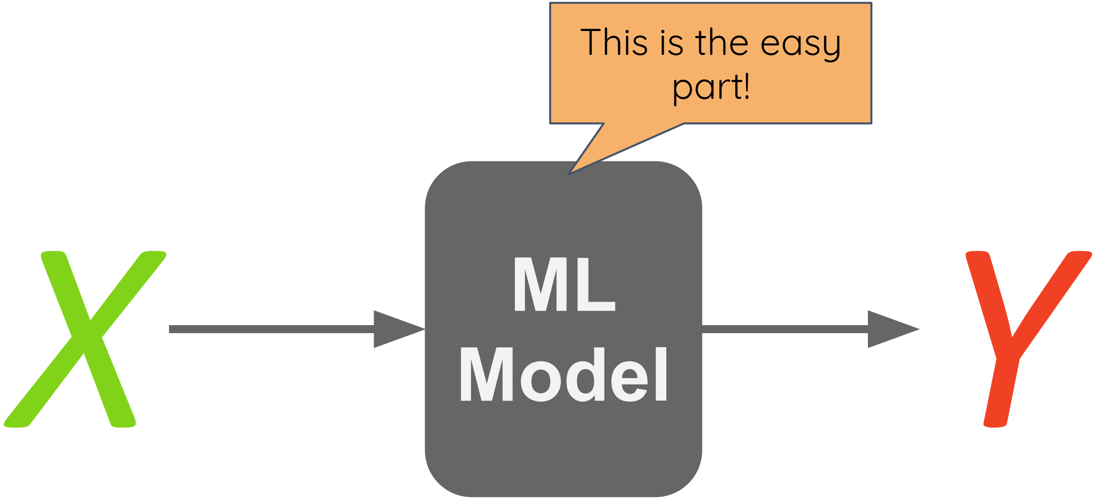
Which model is better?

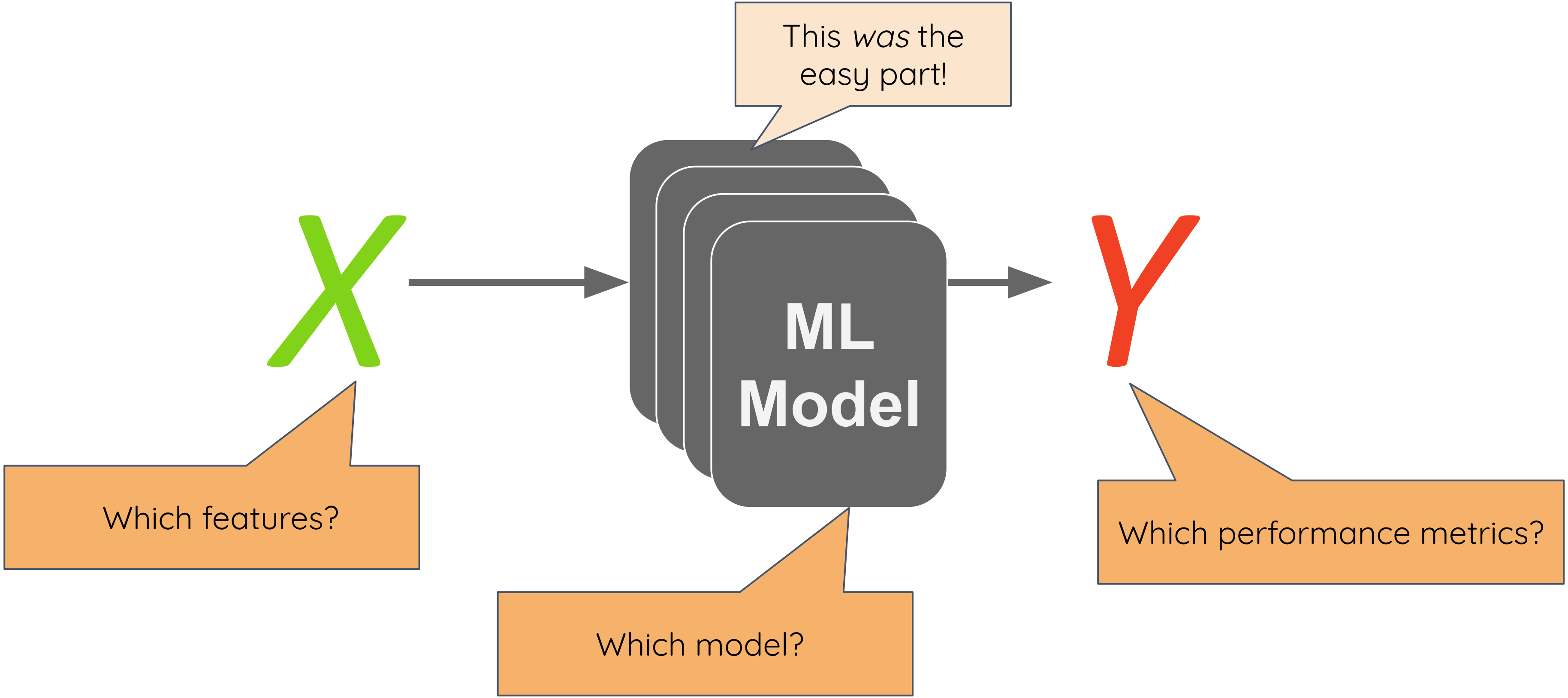
A) Simple B) Fancy

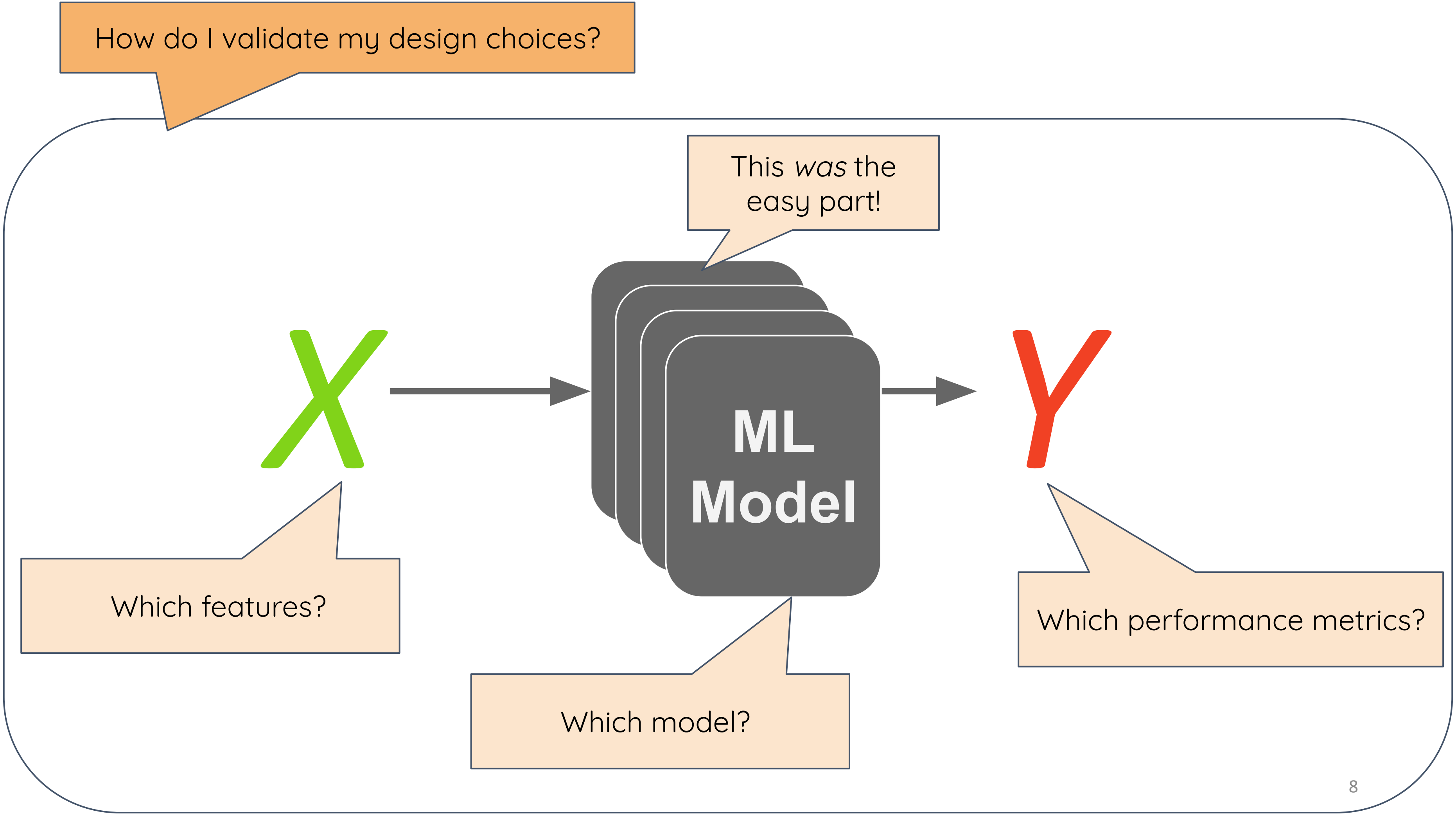
Training a machine-learning model



Training a machine-learning model

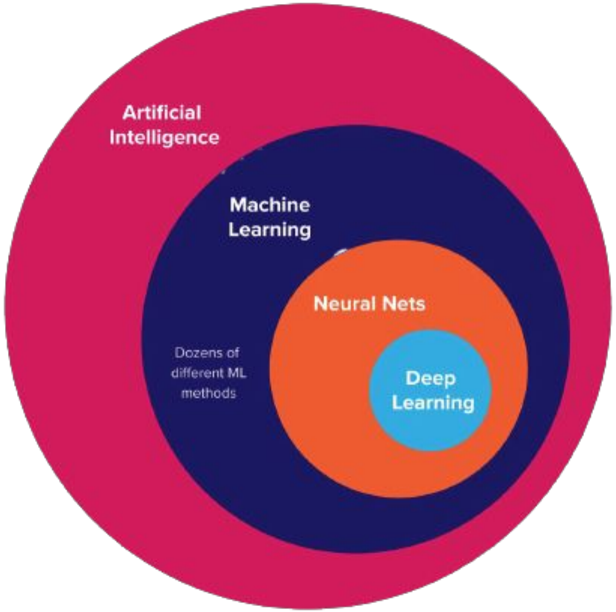



This *was* the easy part!
X
ML Model
Y
Which features?
Which model?
Which performance metrics?


How do I validate my design choices?
This *was* the easy part!
X
ML Model
Y
Which features?
Which model?
Which performance metrics?

Machine-learning - what, why, and when?

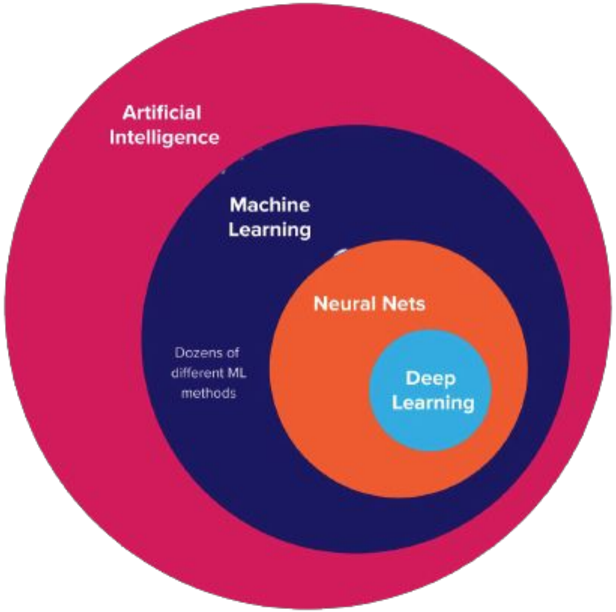
- What is Machine learning (ML)?
    - ML is the study of computer algorithms that improve automatically through experience and by the use of data.



Machine-learning - what, why, and when?

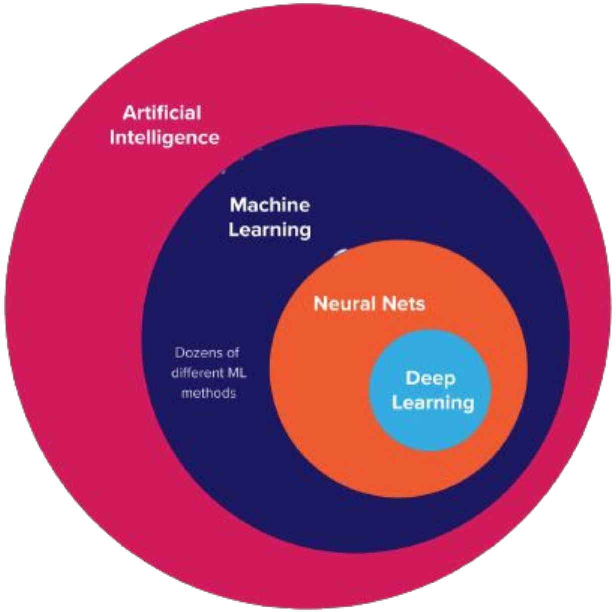
- What is Machine learning (ML)?
  - ML is the study of computer algorithms that improve automatically through experience and by the use of data.

- Why is it useful - especially in life sciences?
  - Biology, Medicine, Environmental sciences comprise phenomenons (e.g. a disease) with large number of variables.
  - We want to model complex relationships within these variables and make accurate predictions.

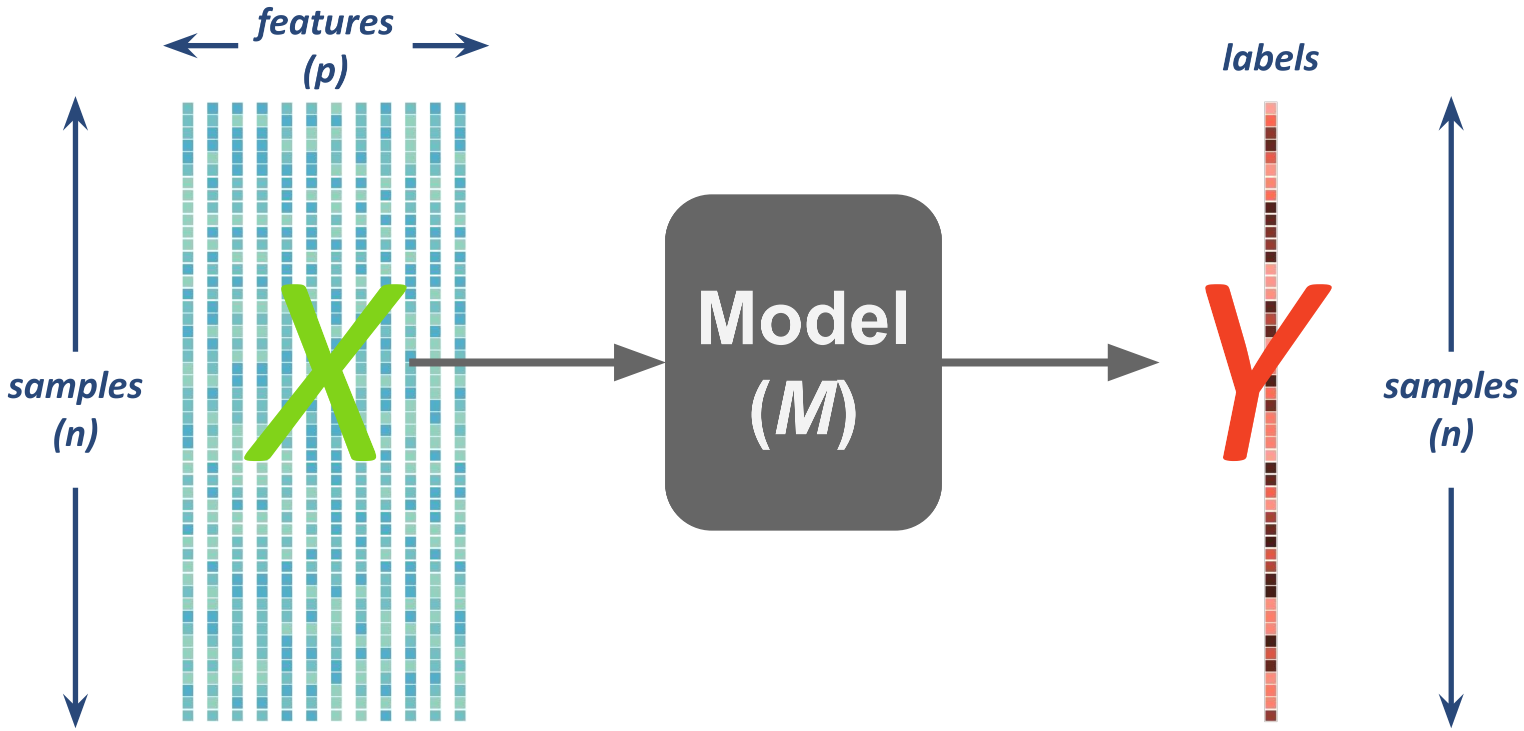


Machine-learning - what, why, and when?

- What is Machine learning (ML)?
    - ML is the study of computer algorithms that improve automatically through experience and by the use of data.
- Why is it useful - especially in life sciences?
    - Biology, Medicine, Environmental sciences comprise phenomenons (e.g. a disease) with large number of variables.
    - We want to model complex relationships within these variables.
- When do I use it?
    - You are interested in 1) prediction tasks or 2) low-dimensional representation.
    - You have sufficient data.



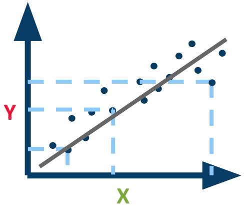
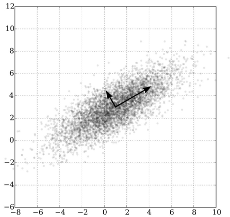
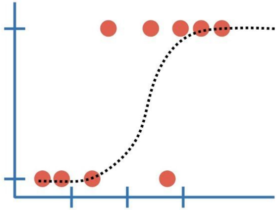
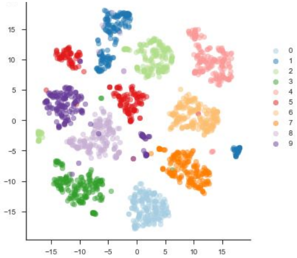
Terminology



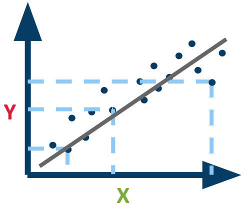
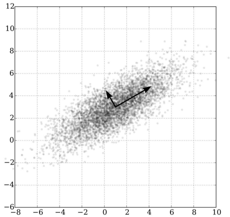
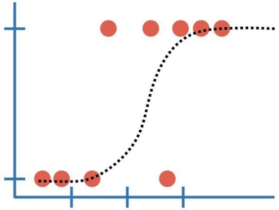
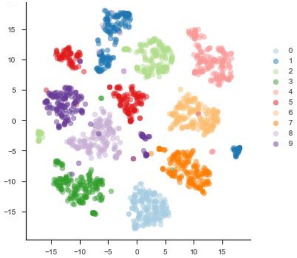
Input
Examples: 

Output Examples: Clinical
measures

Types of ML Algorithms

Outcome	Supervised Learning	Unsupervised Learning
Continuous	Regression 	Dimensionality reduction 
Categorical	Classification 	Clustering 

Types of ML Algorithms

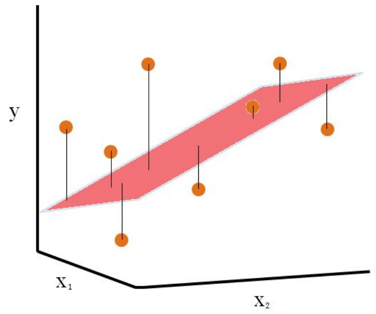
Outcome	Supervised Learning	Unsupervised Learning
Continuous	Regression 	Dimensionality reduction 
Categorical	Classification 	Clustering 

Supervised Learning: Models

- Goal: *Learn* parameters (or weights) of a model (M) that maps X to y

Supervised Learning: Models

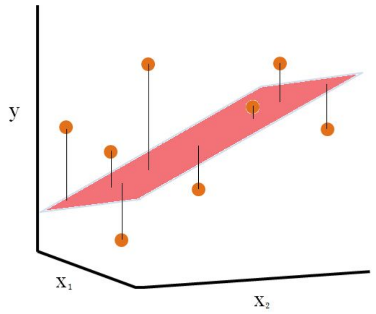
- Goal: *Learn* parameters (or weights) of a model (M) that maps X to y
- Example models:
  - Linear / Logistic regression



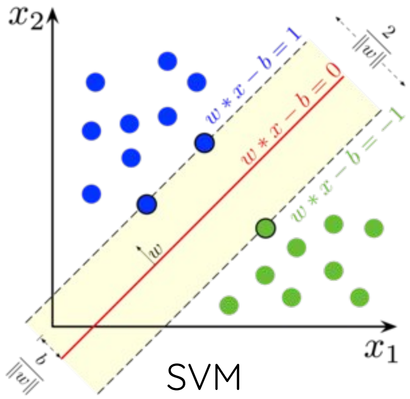
Linear Regression

Supervised Learning: Models

- Goal: *Learn* parameters (or weights) of a model (M) that maps X to y
- Example models:
  - Linear / Logistic regression
  - Support vector machines



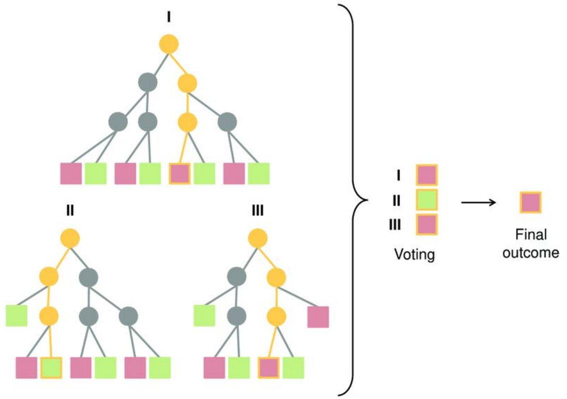
Linear Regression



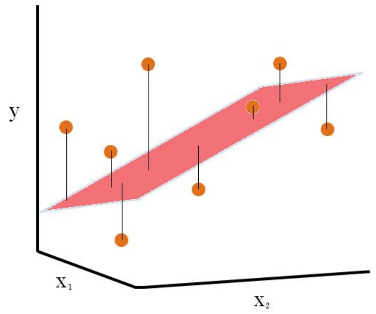
SVM

Supervised Learning: Models

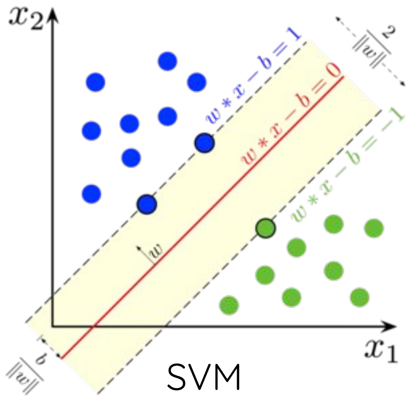
- Goal: *Learn* parameters (or weights) of a model (M) that maps X to y
- Example models:
  - Linear / Logistic regression
  - Support vector machines
  - Tree-ensembles: random forests, gradient boosting



Tree-ensembles



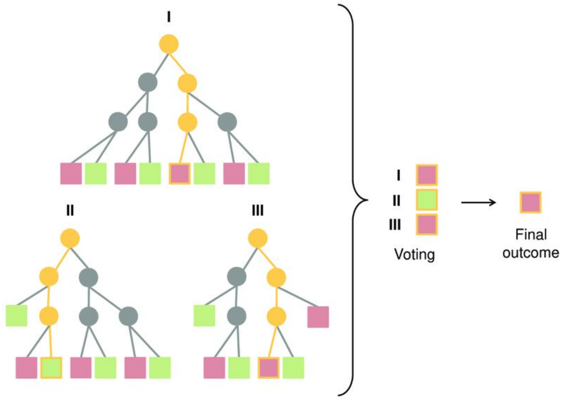
Linear Regression



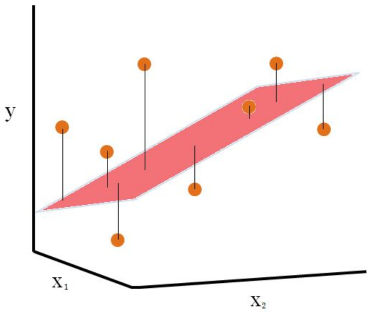
SVM

Supervised Learning: Models

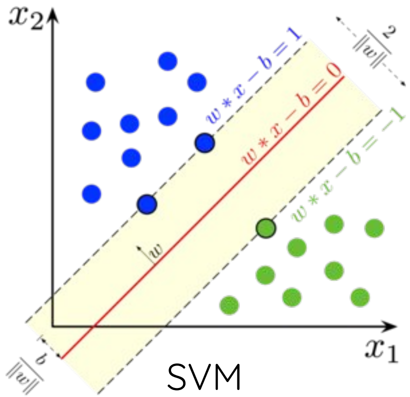
- Goal: *Learn* parameters (or weights) of a model (M) that maps X to y
- Example models:
  - Linear / Logistic regression
  - Support vector machines
  - Tree-ensembles: random forests, gradient boosting
  - Artificial Neural networks



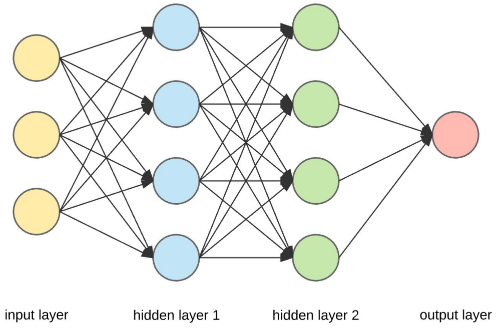
Tree-ensembles



Linear Regression



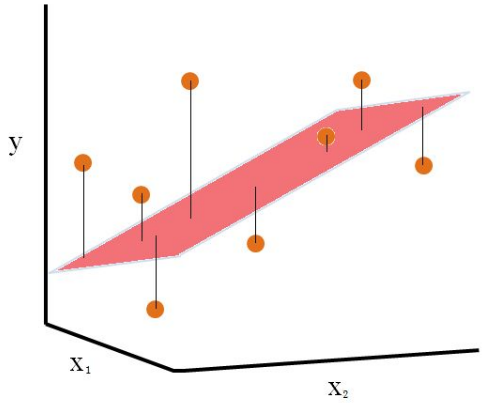
SVM

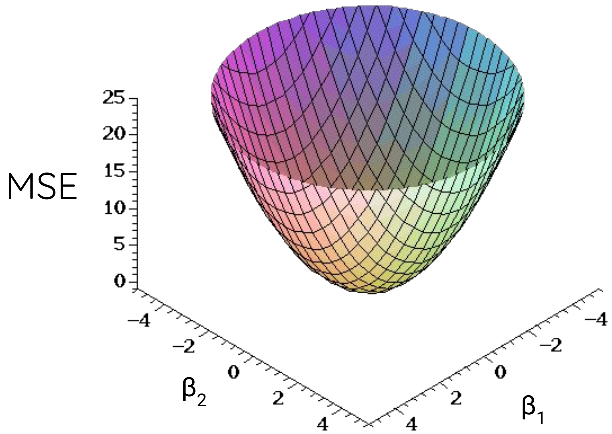


ANN

Model Fitting

- How do we learn the model weights?
    - Example: Linear regression
    - Model: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$
    - Loss function: $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
    - Optimization: Gradient descent



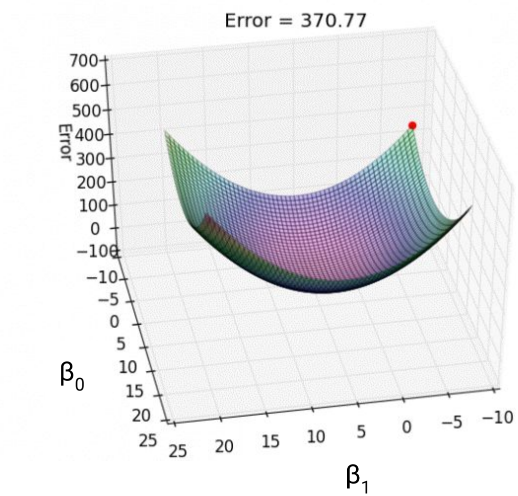


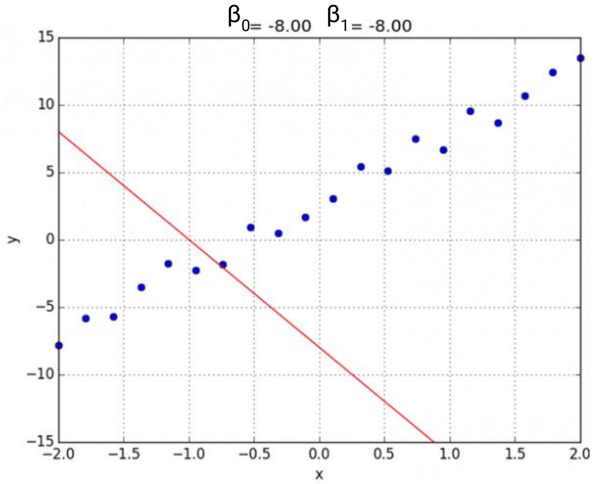
Model Fitting

- Gradient descent with a **single** input variable and **n** samples
  - Start with random weights (β_0 and β_1)
  - Compute loss (i.e. MSE)
  - Update weights based on the gradient

$$\hat{y}_i = \beta_0 + \beta_1 x_i$$

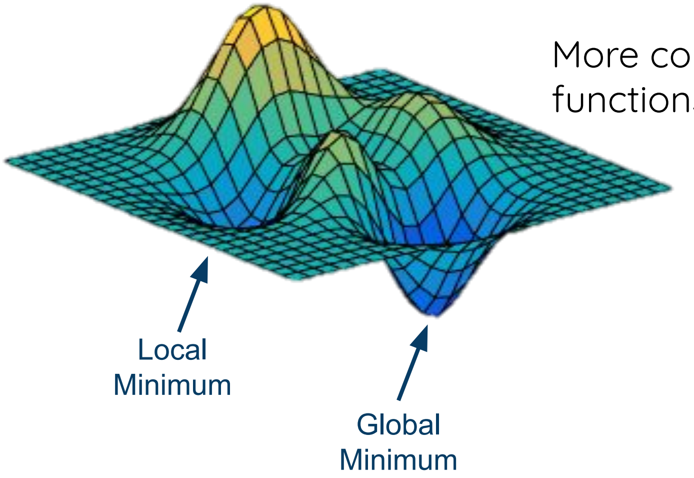
$$MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$





Model Fitting

- Gradient descent for complex models with non-convex loss functions
  - Start with random weights (β_0 and β_1)
  - Compute loss
  - Update weights based on the gradient



More complex models / loss functions (e.g. ANNs)

Model Fitting

- Can we control this fitting process to get a model with specific characteristics?

Model Fitting

- Can we control this fitting process to get a model with specific characteristics?
  - We have strong prior beliefs about what is a **plausible** model
    - e.g. I believe this symptom can be predicted with handful of genes.
  - Practical reasons
    - Prevent overfitting (n_features >> n_samples)

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_{p-1} x_{p-1} + \beta_p x_p$$

Model Fitting

- Can we control this fitting process to get a model with specific characteristics?
  - We have strong prior beliefs about what is a **plausible** model
    - e.g. I believe this symptom can be predicted with handful of genes.
  - Practical reasons
    - Prevent overfitting (n_features >> n_samples)

- **Yes! → Model regularization**

Model Fitting: Regularization

- How do we do it?
  - Modify the loss function
  - Constrain the learning process
- Examples:
  - L1 i.e. Lasso
  - L2 i.e. Ridge

1) L1/Lasso: constrains parameters to be ***sparse***

$$MSE = \sum_{i=1}^{n}(y_i - \underbrace{[\,\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j\,]}_{\hat{y}_i})^2 + \underbrace{\lambda \sum_{j=1}^{p} |\beta_j|}_{L_1}$$

2) L2/Ridge: constrains parameters to be ***small***

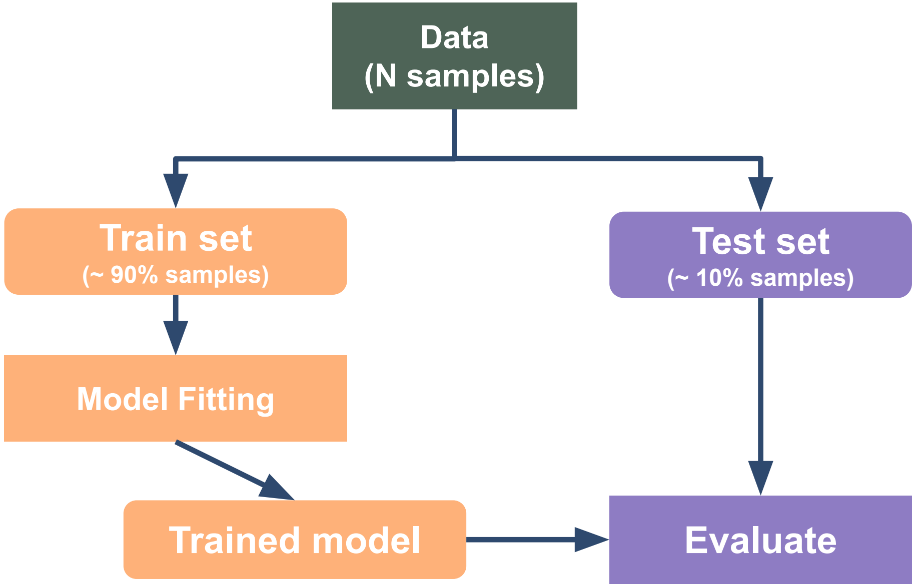
$$MSE = \sum_{i=1}^{n}(y_i - \underbrace{[\,\beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j\,]}_{\hat{y}_i})^2 + \underbrace{\lambda \sum_{j=1}^{p} \beta_j^2}_{L_2}$$

Model Fitting: Scikit-learn syntax

```
# import
from sklearn import linear_model, svm

# data
X = [[0, 0], [1, 1]]
y = [0, 1]

# pick a model
model = linear_model.Lasso(alpha=0.1)   # model = svm.SVC()

# fit the model with data
model.fit(X, y)

# predict on new data
y_pred = model.predict([[1, 0]])
```

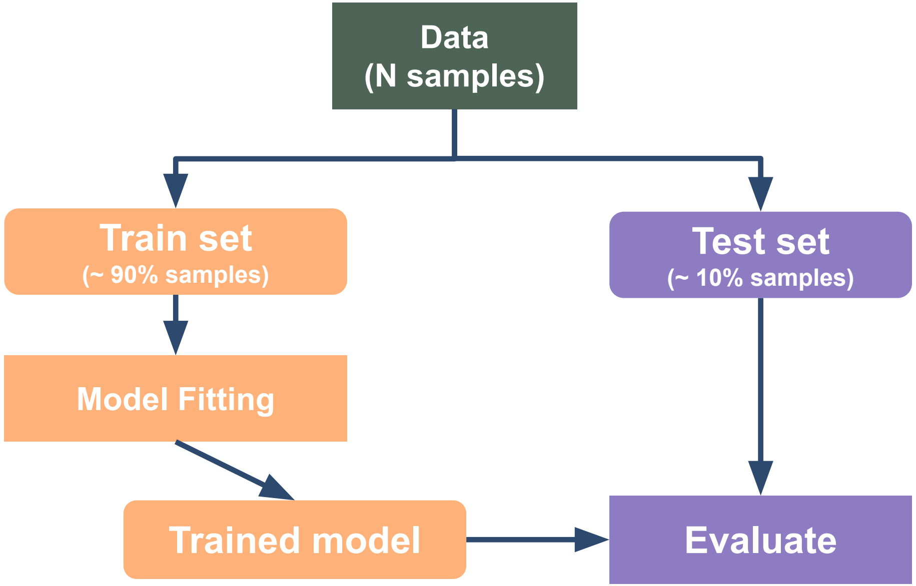
Model Evaluation

- Is the model generalizable?
- How do we sample train and test sets?
- How do we select a model?



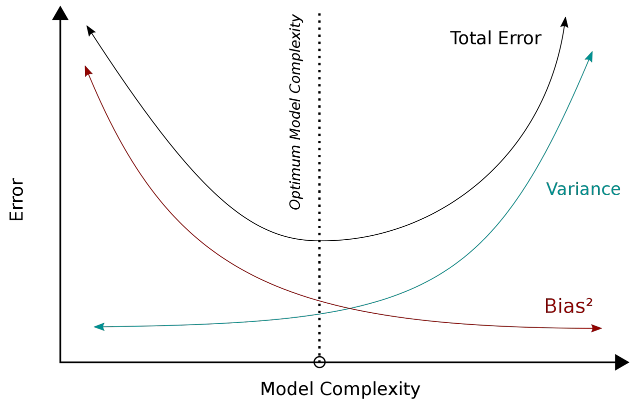
Model Evaluation

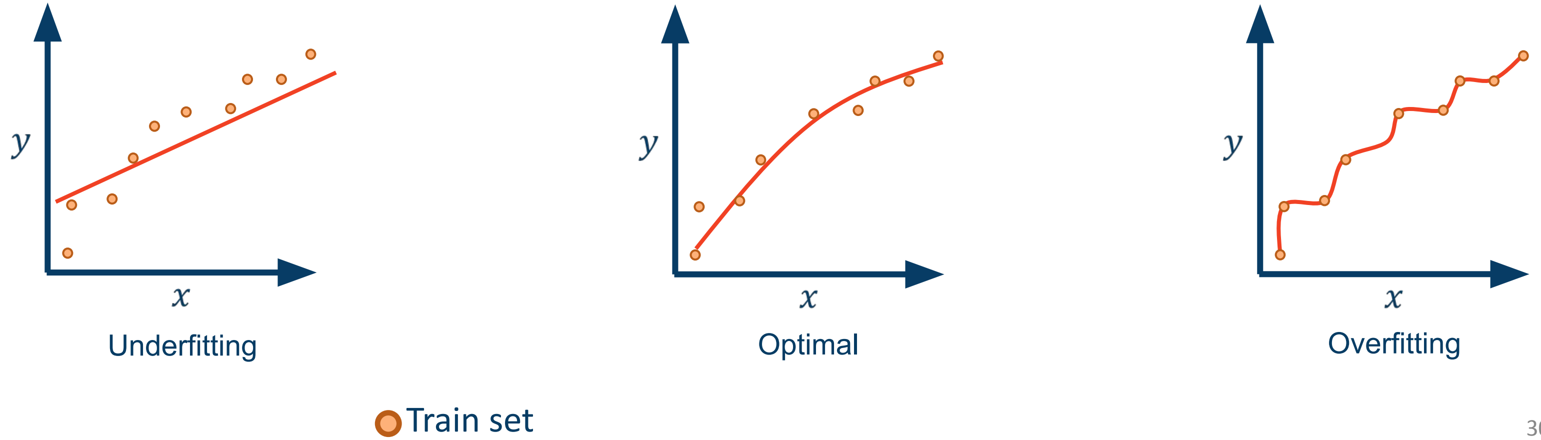
- **Is the model generalizable?**
- How do we sample train and test sets?
- How do we select a model?



Model Evaluation

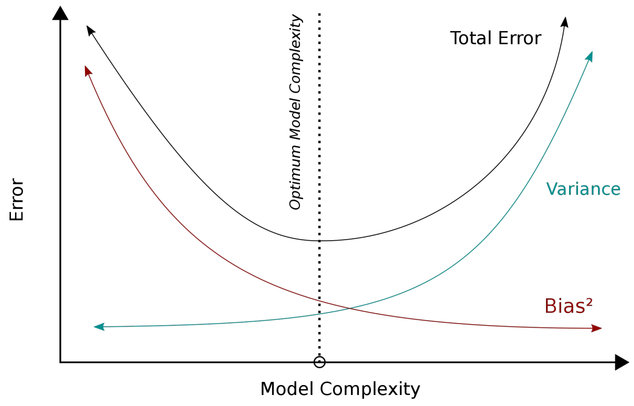
- Train performance ≠ Test performance
  - Model: Underfitting vs Overfitting
  - Errors: Bias - Variance tradeoff
  - Regression example

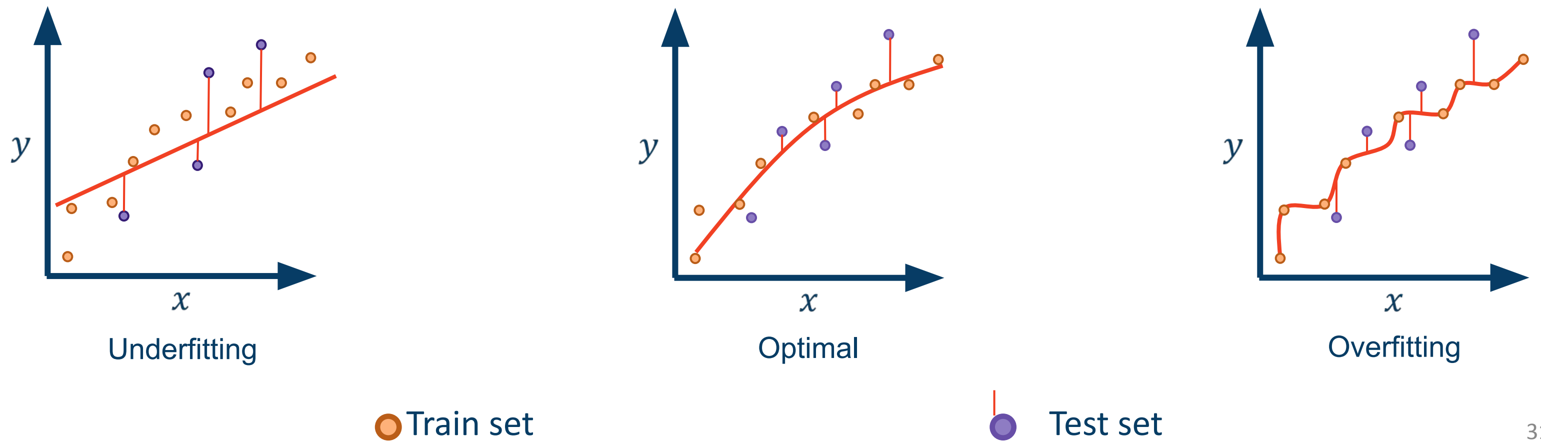




Model Evaluation

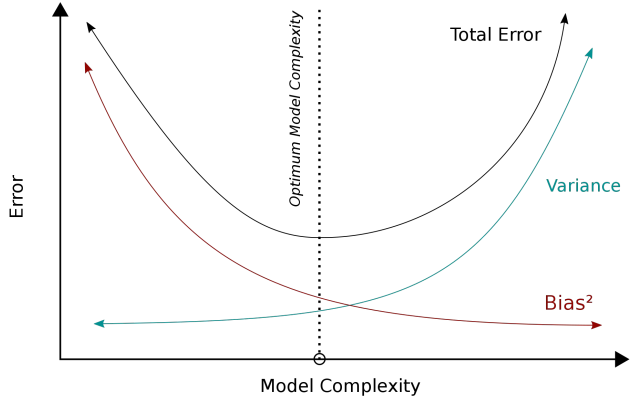
- Train performance ≠ Test performance
  - Model: Underfitting vs Overfitting
  - Errors: Bias - Variance tradeoff
  - Regression example

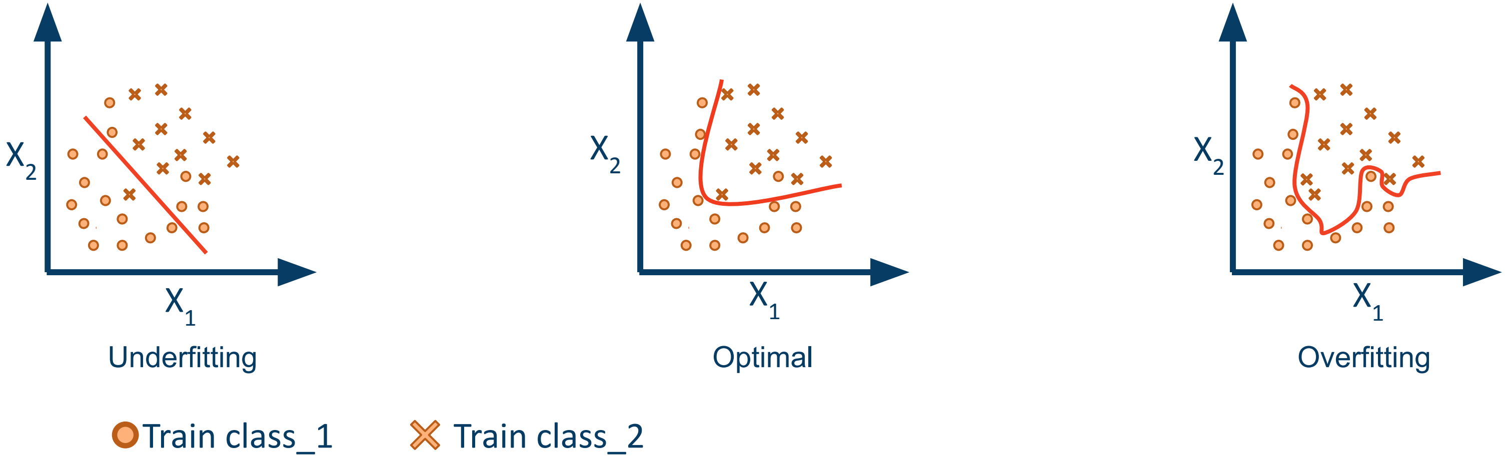




Model Evaluation

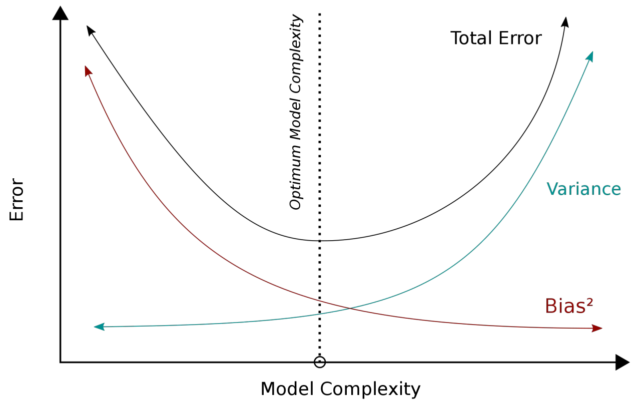
- Train performance ≠ Test performance
  - Model: Underfitting vs Overfitting
  - Errors: Bias - Variance tradeoff
  - Classification example

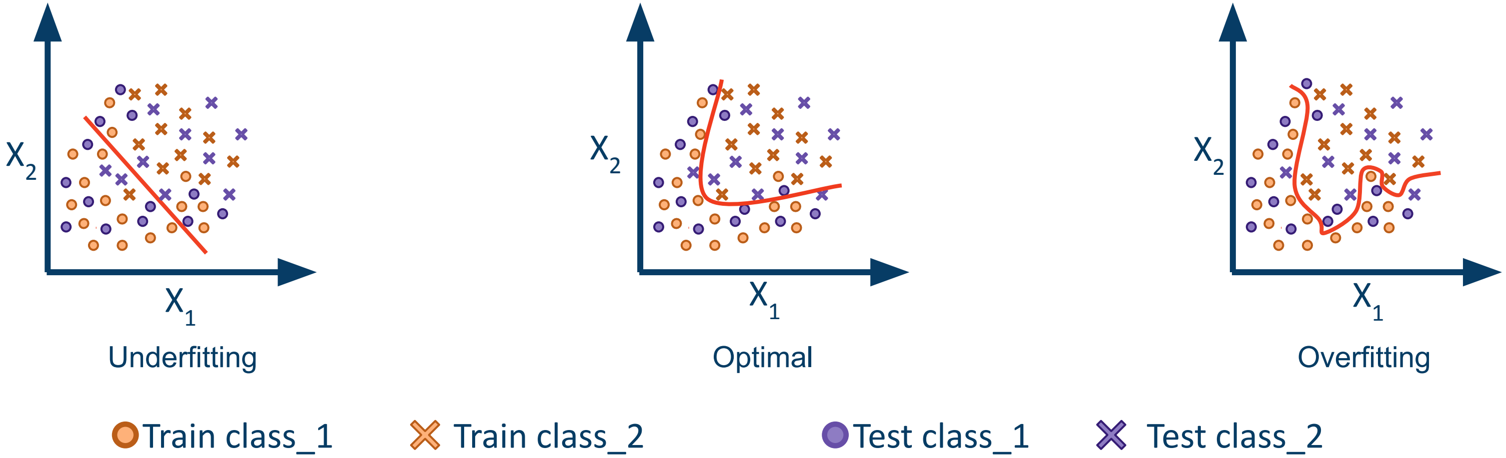




Model Evaluation

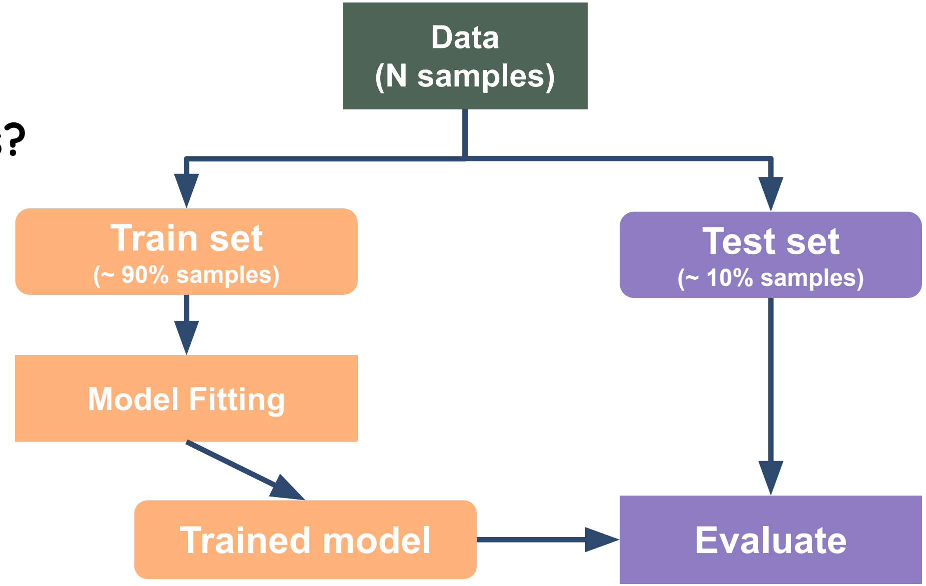
- Train performance ≠ Test performance
  - Model: Underfitting vs Overfitting
  - Errors: Bias - Variance tradeoff
  - Classification example





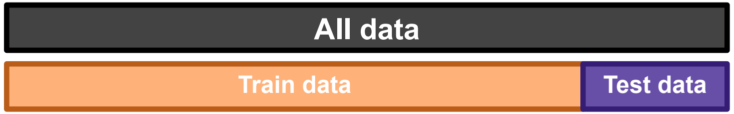
Model Evaluation

- Is the model generalizable?
- **How do we sample train and test sets?**
- How do we select a model?



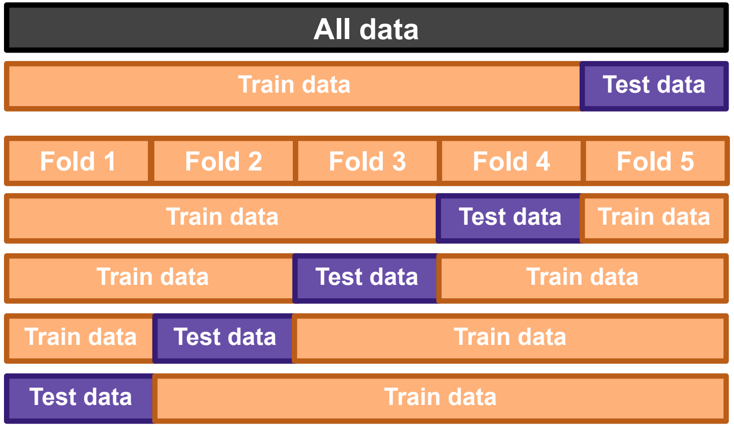
Model Evaluation: Cross-Validation (Outer loop)

- How do we sample train and test sets?
  - Train set: learn model parameters
  - Test set (a.k.a held-out sample): Evaluate model performance



Model Evaluation: Cross-Validation (Outer loop)

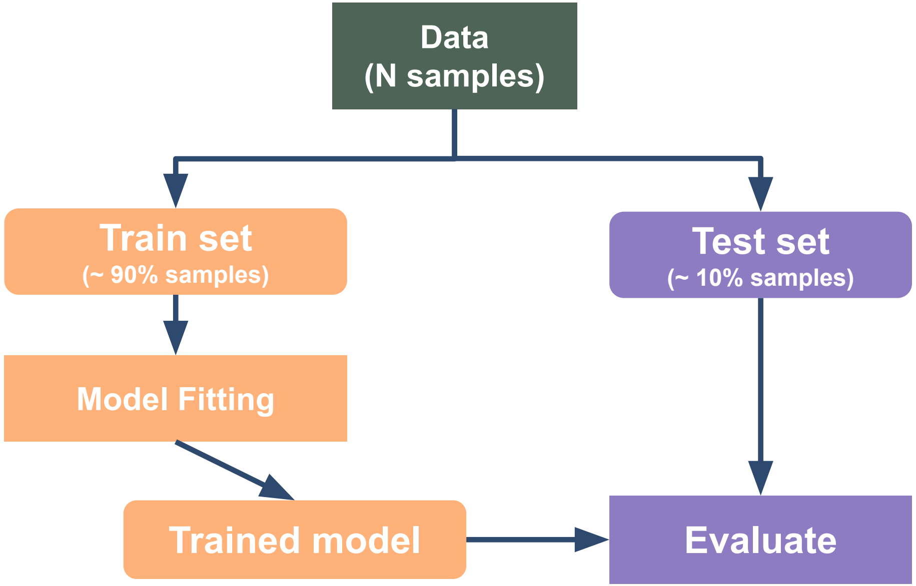
- How do we sample train and test sets?
  - Train set: learn model parameters
  - Test set (a.k.a held-out sample): Evaluate model performance
  - Repeat for different Train-Test splits
    - k-fold, shuffle-split
  - Report performance statistics over all test folds



CV outer loop

Model Evaluation

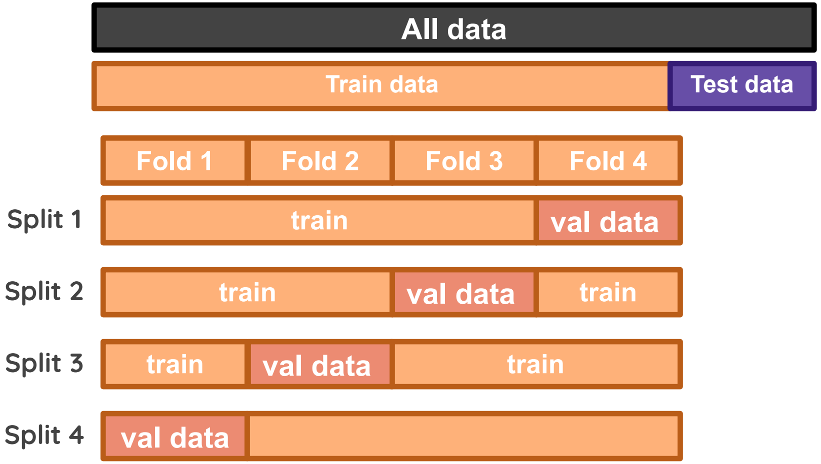
- Is the model generalizable?
- How do we sample train and test sets?
- **How do we select a model?**



Model Evaluation: Cross-Validation (Inner loop)

- How do we select a model?
  - Tune *hyper-parameters* of a model
  - Compare several different model architectures
  - Select / transform raw features

- This repeats for all train-test splits in the outer loop



CV inner loop

Model Evaluation: Hyper-parameters

- Hyper-parameter ≠ parameter (or weights)
  - Parameters are **learned**; hyper-parameters are **chosen**!

Model Evaluation: Hyper-parameters

- Hyper-parameter ≠ parameter (or weights)
  - Parameters are **learned**; hyper-parameters are **chosen**!

- Examples:
  - Degree of model (eg. linear vs quadratic)
  - Kernels
  - Number of trees
  - Number of layers, filters, batch-size, learning-rate in ANNs

Model Evaluation: Hyper-parameters

- Hyper-parameter ≠ parameter (or weights)
  - Parameters are **learned**; hyper-parameters are **chosen**!

- Examples:
  - Degree of model (eg. linear vs quadratic)
  - Kernels
  - Number of trees
  - Number of layers, filters, batch-size, learning-rate in ANNs

- How do we choose them?
  - Prior beliefs → eg. cortical thickness and age have quadratic relationship.
  - Arbitrarily → we gotta start with something!
  - Trial and error → do a computationally feasible grid-search.

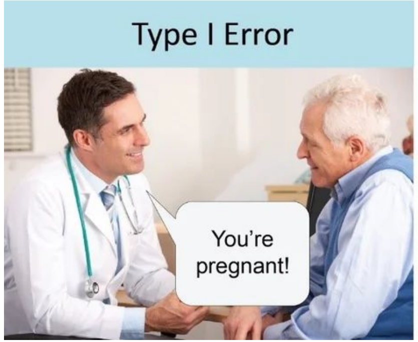
Performance Scores

- Loss functions → computationally well-suited metrics
  - May / need not completely capture performance metrics of interest

- Scores → practically useful metrics
  - Binary classification

Confusion Matrix		Ground Truth	
		POSITIVE	NEGATIVE
Prediction	POSITIVE	TP	FP
	NEGATIVE	FN	TN

False Positive



False Negative

Performance Scores

- ML model that detects Covid from chest CTs. Current Covid prevalence ~ 1%.
  - FP: model predicts *Covid* when person is *healthy*
  - FN: model predicts *healthy* when person has *Covid*
- What happens if we build model that predicts everyone as healthy?
  - i.e. zero FPs!

Performance Scores

- ML model that detects Covid from chest CTs. Current Covid prevalence ~ 1%.
  - FP: model predicts *Covid* when person is *healthy*
  - FN: model predicts *healthy* when person has *Covid*
- What happens if we build model that predicts everyone as healthy?

Score	**Formula**	**Null**	**What does it tell us?**	**When do I use it?**
Accuracy	(TP+TN) / (TP+FP+FN+TN)	0.99	How many people did we correctly predict out of all the people scanned?	FNs & FPs have similar costs
Precision (i.e. PPV)	TP/(TP+FP)	NaN	How many of those who we predicted as "covid" do actually have "covid"?	If you want to be more confident of your TPs
Recall (aka Sensitivity)	TP/(TP+FN)	0	Of all the people who have covid, how many of those did we correctly predict?	If you prefer FPs over FNs.
Specificity	TN/(TN+FP)	1	Of all the people who are healthy, how many of those did we correctly predict?	If you prefer FNs over FPs.
F1	2*(Recall * Precision) / (Recall + Precision)	NaN	Harmonic mean(average) of the precision and recall.	When you have an uneven class distribution

Pop Quiz Answers

We train a simple machine-learning model to identify COVID patients using their biometry, in a population with 1% covid prevalence. Our model is 91% accurate! Then we also calculate,

- 90% sensitivity (i.e. probability that prediction is positive if patient has COVID)
- 91% specificity (i.e. probability that prediction is negative if patient doesn't have COVID)

What are my chances that I have COVID if my test is positive?

(Imagine a sample of 1000 individuals → 10 COVID patients → 9 TP & 89 FP)

A) 9 in 10 B) 1 in 2 **C) 1 in 10** D) 1 in 100

Later we train a fancy deep Learning model to identify COVID patients using their chest CT! This model has accuracy of 99%! We calculate

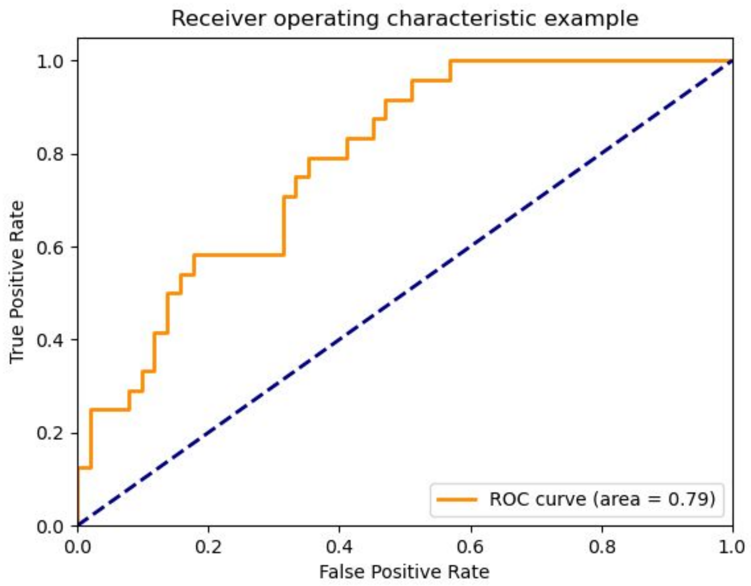
- 80% sensitivity
- 99% specificity

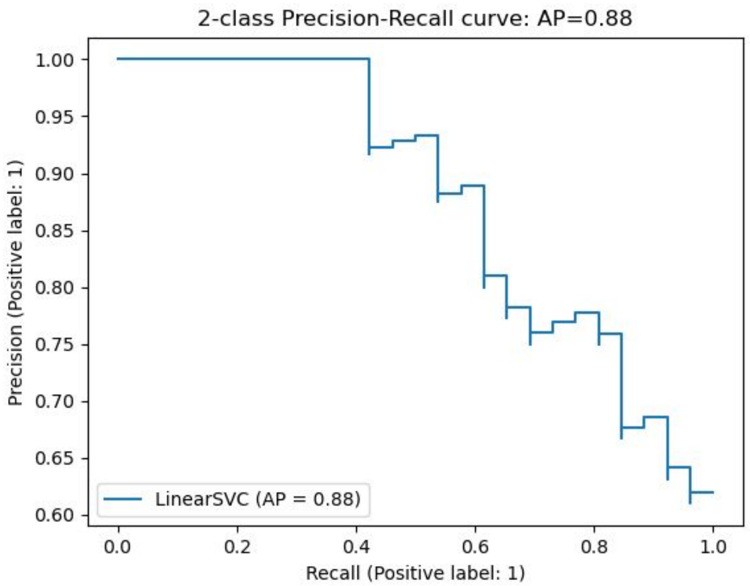
Which model is better? (We want to avoid FN to reduce the spread → we want high-sensitivity)

A) Simple B) Fancy

Performance Curves

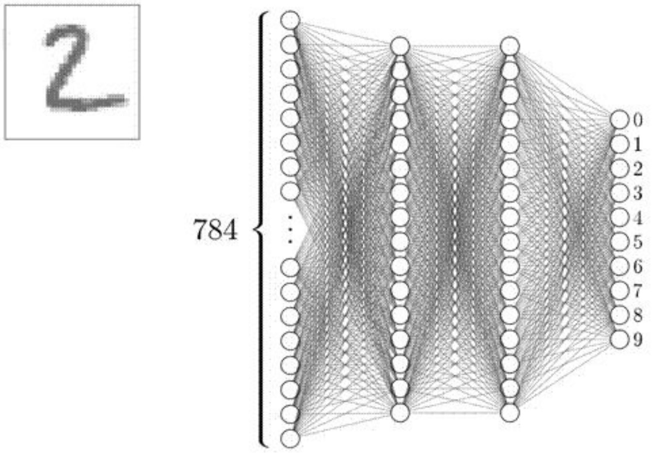
- Receiver Operating Characteristic (ROC) → Want high area-under-the-curve (AUC)
- Precision-Recall → Want high AUC or high Average precision (AP)





Deep-learning

- Why the buzz?
  - Works amazing on structured input
  - Highly flexible → universal function approximator

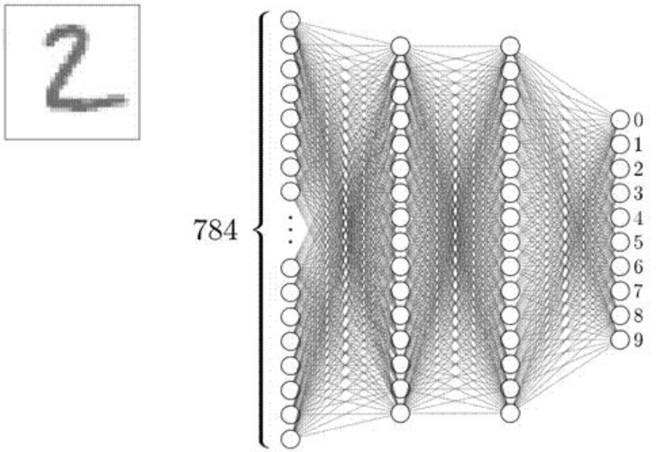


ANN for handwritten-digit images
(gif source: 3b1b)

Deep-learning

- Why the buzz?
  - Works amazing on structured input
  - Highly flexible → universal function approximator

- What are the challenges?
  - Large number of parameters → data hungry
  - Large number of hyper-parameters → difficult to train



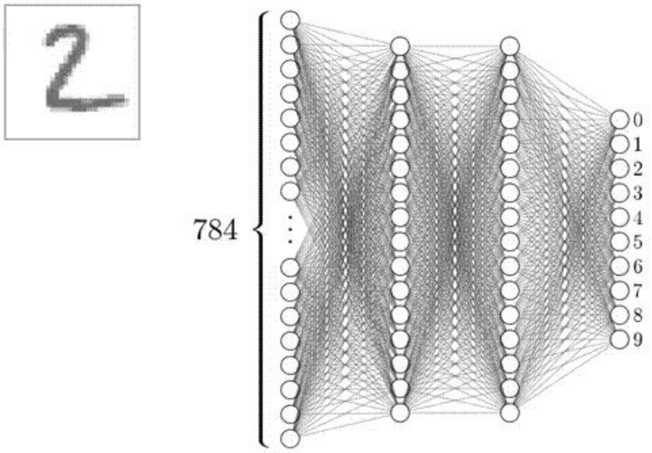
ANN for handwritten-digit images
(gif source: 3b1b)

Deep-learning

- Why the buzz?
  - Works amazing on structured input
  - Highly flexible → universal function approximator

- What are the challenges?
  - Large number of parameters → data hungry
  - Large number of hyper-parameters → difficult to train

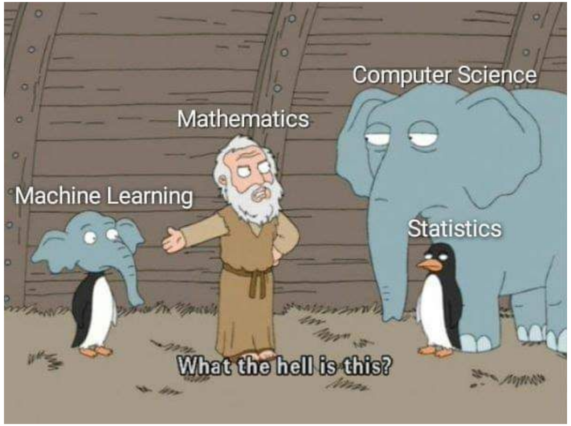
- When do I use it?
  - If you have highly-structured input, eg. medical images.
  - You have a lot of data and computational resources.



ANN for handwritten-digit images
(gif source: 3b1b)

Pitfalls and Challenges

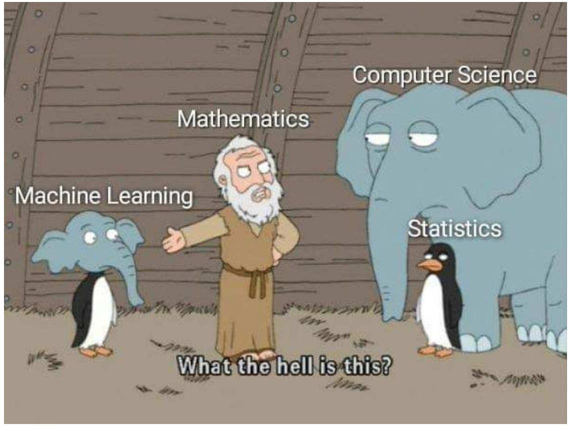
- Models do not generalize even after good CV performance
  - Implicit double-dipping
  - Dataset biases (eg. North-American demographics)
  - Noisy labels (eg. diagnosis definitions)
  - Data distribution shifts (eg. assay, scanner upgrades)



Pitfalls and Challenges

- Models do not generalize even after good CV performance
  - Implicit double-dipping
  - Dataset biases (eg. North-American demographics)
  - Noisy labels (eg. diagnosis definitions)
  - Data distribution shifts (eg. assay, scanner upgrades)

- Unnecessary complexity
  - Do I really need a giant deep-net or a simple linear model would do?



ML Novice Checklist

- Data
  - What is my n_features and n_samples?
  - Am I encoding categorical data correctly?
  - Am I using information (e.g. mean) from test set to preprocess (eg. zscore) the data?

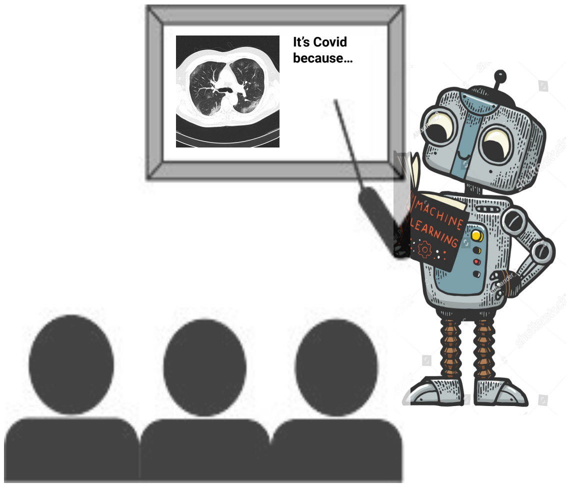
ML Novice Checklist

- Data
  - What is my n_features and n_samples?
  - Am I encoding categorical data correctly?
  - Am I using information (e.g. mean) from test set to preprocess (eg. zscore) the data?

- Model
  - Do my performance metrics capture the practical use-case of interest?
  - What is the null / dummy model performance?
    - Classification: Predict majority class all the time
    - Regression: Predict the median value all the time
  - Am I interpreting model parameters (i.e. weights) correctly?

Takeaways

- Supervised models are useful for **predictions**
  - eg. image segmentation, prognosis, drug development
- Our job is to ensure **generalizability** of these models
  - Multitude of validations
  - Understanding model biases and limitations
- Food for thought: ***engineering tools*** vs *scientific discovery*
  - Interpretability and explainability
  - Causality, reliability, fairness



Explainable AI